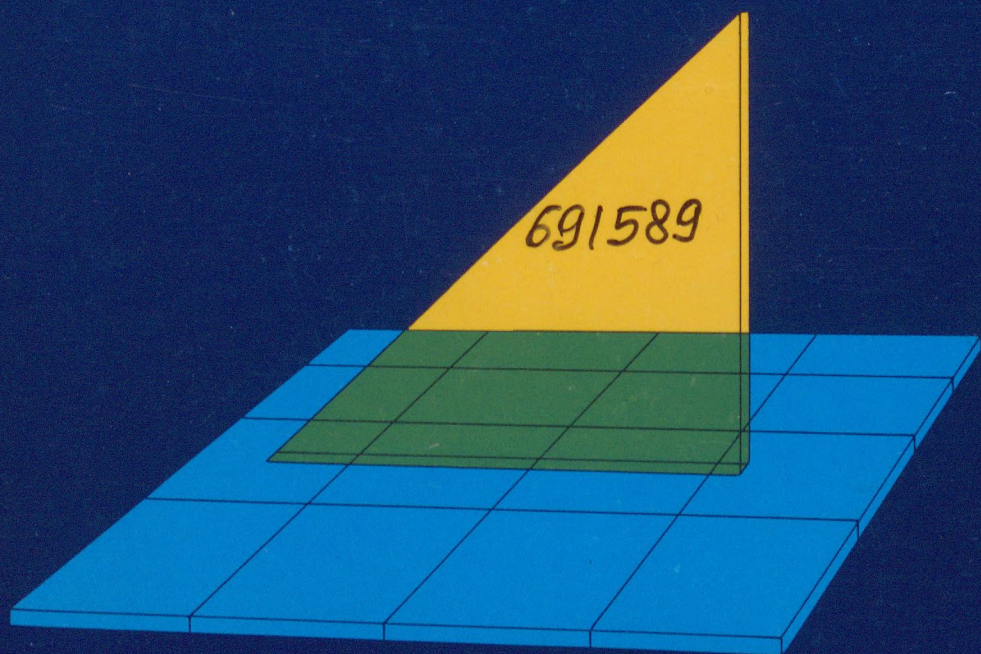


..... Popper György - Csizmás Ferenc

NUMERIKUS MÓDSZEREK

M É R N Ö K Ö K N E K



AKADÉMIAI KIADÓ · TYPOT_EX

7-18G

..... Popper György - Csizmás Ferenc

NUMERIKUS MÓDSZEREK MÉRNÖKÖKNEK

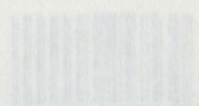
NUMERIKUS

MÓDSZEREK

M É R N Ö K Ö K N E K

AKADÉMIAI KIADÓ - TITÓRTER
BUDAPEST, 1973

MTA
KIK



feleségeinknek

Andreának és Katának

7486

..... Popper György - Csizmás Ferenc

NUMERIKUS MÓDSZEREK

M É R N Ö K Ö K N E K

AKADÉMIAI KIADÓ • TYPOTEX
BUDAPEST, 1993

MTA
KIK



00004 25753 0

691589

Szerzők:

POPPER GYÖRGY
a műszaki tudomány doktora
c. egyetemi tanár, BME

CSIZMÁS FERENC
okl. építőmérnök

Lektorálta:

STOYAN GISBERT
a matematikai tudomány doktora
egyetemi tanár, ELTE

Nyelvi szempontból ellenőrizte:

GYÖMÖRE MIHÁLYNÉ
okl. magyar-történelem szakos tanár

MAGYAR
TUDOMÁNYOS AKADÉMIA
KÖNYVTÁRA

ISBN 963 05 6454 8
ISBN 963 7546 23 5

Kiadja az Akadémiai Kiadó és a TYPOT_EX Kft., Budapest

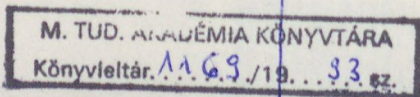
Első kiadás: 1993

A kiadvány "Az építés fejlődéséért" Alapítvány anyagi támogatásával készült.

©Dr. Popper György – Csizmás Ferenc, 1993

Minden jog fenntartva, beleértve a sokszorosítás, a nyilvános előadás, a rádió- és televízióadás, valamint a fordítás jogát az egyes fejezeteket illetően is.

Printed in Hungary



TARTALOMJEGYZÉK

ELŐSZÓ

LINEÁRIS ALGEBRAI EGYENLETRENDSZEREK

1. Bevezetés	9
2. Háromszögfelbontáson alapuló direkt módszerek	10
Főelem-kiválasztás	12
Sorok kiegyensúlyozása	14
Skaláris szorzatok kettős pontosságú számítása	15
3. Gyengén meghatározott egyenletek	
Hibaaanalízis	20
Normák	22
4. A megoldás iterációs finomítása	26
5. Pozitív definit mátrix szimmetrikus felbontása	28
6. Szimmetrikus pozitív definit szalagmátrix	34
Körvonal (sky-line) technika	36
Frontális technika	39
Csúsztatott háromszögtechnika	39
7. Iterációs módszerek	40
Az iteráció befejezésére vonatkozó feltétel	43
A konvergencia gyorsasága	44
8. Jacobi-iteráció és a Gauss–Seidel-módszer	44
9. A szukcesszív felülrelaxálás módszere (SOR)	46

VALÓS SZIMMETRIKUS MÁTRIX SAJÁTÉRTÉKEI ÉS SAJÁTVEKTORAI

1. A lineáris algebra néhány eredménye	51
2. Polinommal kapcsolatos iterációk	54
Givens-féle felezési eljárás	54
3. Vektorokkal végzett iterációk	60
Hatványiteráció	60
Inverziteráció	62
Rayleigh-hányados iteráció	64
További sajátpárok meghatározása	66
Altér iteráció	68
A Rayleigh–Ritz-módszer	71

Általánosított sajátérték-feladat	74
4. Transzformációs módszerek	82
Jacobi-diagonalizálás	82
Householder-módszer	92
A QR és a QL algoritmus	98
QL algoritmus implicit eltolásokkal	102
5. Szimmetrikus mátrix sajátérték-feladatára vonatkozó hibaanalízis	106
6. Lineáris algebrai egyenletrendszerek általánosított megoldása	112
Mátrix szinguláris értékek szerinti felbontása	112
Eljárás az SVD numerikus megoldására	114
Pseudoinverz mátrix	121
Lineáris egyenletrendszerek legkisebb négyzetes megoldása	122

NEMLINEÁRIS EGYENLETRENDSZEREK

1. Alapfogalmak	127
2. Newton-típusú módszerek	131
Newton-módszer	131
Gauss-Newton-módszer	132
Newton-Raphson-módszer	134
Gauss-Newton-iteráció egydimenziós kereséssel	136
3. Egydimenziós minimalizálás	137
Aranymetszés szerinti keresés	137
A minimum kezdő közrefogása	140
Parabolikus interpoláció és a Brent-módszer	142
4. Gradiensmódszer	145
5. Konjugált gradiensmódszerek	148
6. Kvázi-Newton-módszerek	157
7. Gauss-szelőmódszer	163

IRODALOMJEGYZÉK

ELJÁRÁSOK JEGYZÉKE

TÁRGYMUTATÓ

ELŐSZÓ

E könyv megírásával kettős célunk volt. Tankönyvet szeretnénk adni a numerikus módszerek és számítástechnika iránt érdeklődő diákok kezébe és recept-könyvként is használható programgyűjteményt nyújtani a tervező és kutató mérnökök számára.

Az elméleti rész azoknak az előadásoknak a naprakész változata, amelyeket az egyik szerző (P. Gy.) a féléves "Numerikus módszerek" tárgy keretein belül az utóbbi két évtizedben a Budapesti Műszaki Egyetem Építőmérnöki Karán tartott (nagyobb terjedelemben a matematikus szakmérnök hallgatók számára, kisebbben választható tárgyként a negyedéves hallgatóknak). Következésképpen e könyv tanulmányozásánál feltételezzük a Műszaki vagy a Közgazdasági Egyetemeken oktatott matematika alaptárgy tananyagával egyenértékű ismereteket. A programlisták megértéséhez a Pascal programozási nyelv alapszintű ismeretére van szükség.

Az algoritmusok megválasztása és a módszerek ismertetése Popper György, a programlisták elkészítése, kipróbálása (és a könyv számítógépes szedése) Csizmás Ferenc munkája.

A szerkezetek tervezésénél felmerülő kontinuumfeladatokat (pl. lemezek vagy héjak méretezése) általában finitizálással – leggyakrabban a végeselem-módszer alkalmazásával – szokás megoldani. A finitizálás eredményeként sokismeretlenes egyenletrendszert nyerünk, mely lehet lineáris algebrai vagy nemlineáris attól függően, hogy az eredeti feladatban a geometriai és fizikai nemlinearitások hatását figyelembe vesszük-e vagy sem. A stabilitási feladatok és a dinamikai hatások figyelembevétele pedig algebrai sajátérték-feladatokat eredményez. E három témakörrel foglalkozik a könyv három fejezete.

A szerzők hálás köszönettel tartoznak mindenekelőtt "Az építés fejlődéséért" Alapítvány Kuratóriumának anyagi támogatásáért, amely jelentősen elősegítette a könyv megjelentetését. Köszönetet mondunk dr. Stoyan Gisbert egyetemi tanárnak, a könyv hivatalos lektorának a kézirat gondos átnézéséért és hasznos tanácsaiért, amelyek alapján a szövegben több változtatást hajtottunk végre. Hálások vagyunk dr. Gáspár Zsolt és dr. Roller Béla professzoroknak azokért a vitákért, amelyek a kézirat jobbá tételét eredményezték. Köszönet illeti még Gyömrőre Mihálynét, aki a szöveget stílárius szempontból gondosan ellenőrizte és Fodor Attila grafikust a szép fedélterv elkészítéséért.

Budapest, 1992. szeptember

A szerzők

A könyv megírásával kedvvel és lelkesedéssel álltam hozzá. A munkám során a legújabb tudományok és a gyakorlati tapasztalatok alapján igyekeztem a lehető legteljesebb mértékben megadni a szükséges információkat. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem.

Az első fejezet a bevezető részeket tartalmazza, amelyekben megismerjük a könyv célját és szerkezetét. A további fejezetek a témakörrel kapcsolatos elméleti és gyakorlati kérdéseket tárgyalják. A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem.

Az utolsó fejezet a záró gondolatokat tartalmazza, amelyekben megismerjük a könyv célját és szerkezetét. A további fejezetek a témakörrel kapcsolatos elméleti és gyakorlati kérdéseket tárgyalják. A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem.

A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem. A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem.

A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem. A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem.

A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem.

A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem.

A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem.

A kiadás során a legújabb kutatási eredményeket és a gyakorlati tapasztalatokat figyelembe véve a szükséges módosításokat elvégeztem. A kiadás előtt a kéziratot alaposan átnéztettem, és a szükséges módosításokat elvégeztem.

LINEÁRIS ALGEBRAI EGYENLETRENDSZEREK

1. Bevezetés

Feladatunk az

$$A x = b \quad (1-1)$$

n egyenletből álló, n ismeretlenes lineáris algebrai egyenletrendszer numerikus megoldása, ahol az A mátrixról, és a b vektorról feltesszük, hogy *valósak*.¹

Jelölje $A|b$ azt az $n \times (n+1)$ típusú mátrixot, amelyet úgy kapunk, hogy a b vektort $(n+1)$ -edik oszlopként csatoljuk az A mátrix $a_1, \dots, a_n$ oszlopaihoz, és jelölje $r(M)$ egy tetszőleges M mátrix rangját, azaz az M mátrix lineárisan független oszlopainak vagy sorainak számát. Az (1-1) egyenlet megoldásának *létezését és egyértelműségét* a következő alaptétel tisztázza:

1.1. tétel. Az (1-1) egyenletrendszernek akkor és csak akkor van megoldása, ha

$$r(A) = r(A|b).$$

Ha $r(A) = n$, akkor a megoldás egyértelmű.

Ez a tétel geometriailag a következőképpen szemléltethető: Az A mátrix oszlopai a valós n dimenziós R^n térben kifeszítenek egy $r(A)$ dimenziós R^r alteret. Ezért bármely $b \in R^n$ vektor akkor és csak akkor fejezhető ki A oszlopainak

$$x_1 a_1 + \dots + x_n a_n \equiv A x$$

lineáris kombinációjaként, ha $b \in R^r$.

Ha $r(A) = n$, akkor A oszlopai az egész R^n tér bázisát alkotják, így minden b vektor egyértelműen felírható $b = x_1 a_1 + \dots + x_n a_n$ alakban.

1.1. következmény. Minthogy $r(A|0) \equiv r(A)$, a *homogén* lineáris egyenletrendszernek akkor és csak akkor van nemtriviális megoldása, ha $r(A) < n$.

Ellentétben például az n -ed fokú polinomok zérushelyeinek meghatározásával, itt nem nehéz felírni az (1-1) egyenlet analitikus megoldását. Például alkalmazhatjuk a Cramer-szabályt, ekkor azonban ki kell számítani $n+1$ számú n -ed rendű mátrix determinánsát. A számítógépekben alkalmazott korlátos pontossá-

¹ Komplex együtthatós egyenletrendszer visszavezethető $2n$ egyenletből álló, $2n$ ismeretlenes valós egyenletrendszerre.

gú aritmetika miatt felhalmozódhatnak a kerekítési hibák, és ezért – különösen nagy n értékek esetén – hatékony numerikus módszerekre van szükség. Hatékonyságon *stabilis* (a kerekítési hibák hatására nem nagyon érzékeny) és időtakarékos eljárásokat értünk.

A numerikus módszerek két csoportba sorolhatók: A *direkt* módszerekre leginkább az jellemző, hogy ha figyelmen kívül hagyhatnánk a kerekítési hibák hatását, akkor véges számú művelet elvégzése a pontos megoldást eredményezné. Az *iterációs* eljárások esetében, kiindulva a megoldás ismert közelítéseiből, egy adott összefüggéssel képezünk egy olyan vektorsorozatot, amely várhatóan tart az (1-1) egyenlet megoldásához.

2. Háromszögfelbontáson alapuló direkt módszerek

Az alkalmazásokban gyakran előforduló feladat, hogy az (1-1) egyenletrendszert több – pl. a $b_1, \dots, b_m$ – jobb oldallal kell megoldani, ugyanakkor az A együtthatómátrix nem változik. Vagyis, megoldandó az

$$AX = B \quad (2-1)$$

egyenlet, ahol A nem szinguláris n -ed rendű, valós mátrix; $B = [b_1, \dots, b_m]$ pedig $n \times m$ típusú valós mátrix.

Megjegyezzük, hogy ha $B = I$, azaz ha B az n -ed rendű egységmátrix, akkor $X = A^{-1}$, vagyis a (2-1) egyenlet magában foglalja a mátrix *invertálásának* feladatát is.

Hatékony eljárás a (2-1) egyenletrendszer megoldására, ha először az A mátrix

$$A = LU \quad (2-2)$$

alakú ún. *háromszögfelbontását* kíséreljük meg, ahol L alsó, U pedig felső háromszögmátrix (azaz L -nek minden főátló feletti eleme, U -nak minden főátló alatti eleme azonosan egyenlő zérussal).

Ha a (2-2) felbontás elvégezhető, akkor minden egyes b jobb oldalra megoldjuk az

$$Ly = b \quad (2-3a)$$

$$Ux = y \quad (2-3b)$$

egyenletrendszereket. Ez L és U háromszög alakja miatt egyszerűen megtehető.

A (2-2) felbontás egy n^2 egyenletből álló $n^2 + n$ ismeretlenes egyenletrendszer. Ha a háromszögfelbontás elvégezhető, akkor végtelen sokféleképpen végezhető el. Ugyanis, ha LU az A mátrix egyik háromszögfelbontása, akkor $(LD^{-1})(DU)$, ahol D tetszőleges nonsinguláris diagonálmátrix, az A egy másik felbontása. Minthogy az egyértelműség miatt n ismeretlent előre rögzíteni kell, a továbbiak-

ban feltételezzük, hogy $l_{11} = \dots = l_{nn} = 1$. Ekkor a (2-2) felbontást a (2-3a) megoldásával együtt *Gauss-eliminációnak*, a (2-3b) egyenlet megoldását pedig *viSSzahelyettesítésnek* nevezzük.

A (2-2) egyenlet részletesebb alakja

$$\begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & l_{32} & 1 & \\ . & . & . & . \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} & . \\ & u_{22} & u_{23} & . \\ & & u_{33} & . \\ & & & . \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & . \\ a_{21} & a_{22} & a_{23} & . \\ a_{31} & a_{32} & a_{33} & . \\ . & . & . & . \end{bmatrix} \quad (2-4)$$

és pl. a következő sorrendben oldható meg:

$$\begin{array}{llll} & & u_{11} & = a_{11} \\ l_{21}u_{11} & = a_{21}; & u_{12} & = a_{12} \\ & & l_{21}u_{12} + u_{22} & = a_{22} \\ l_{31}u_{11} & = a_{31}; & u_{13} & = a_{13} \\ l_{31}u_{12} + l_{32}u_{22} & = a_{32}; & l_{21}u_{13} + u_{23} & = a_{23} \\ & & l_{31}u_{13} + l_{32}u_{23} + u_{33} & = a_{33} \end{array} \quad (2-5)$$

és így tovább...

Könnyen ellenőrizhető, hogy a bal oldali rendszerekből az l_{ij} ismeretlenek akkor és csak akkor határozhatók meg, ha

$$u_{ii} \neq 0, \quad i = 1, 2, \dots, n-1.$$

A (2-4)-ben szereplő mátrixok determinánsát tekintve, nyilvánvaló az

$$u_{11} \dots u_{nn} = \det(A)$$

összefüggés.

Ugyancsak a (2-4) egyenletből felírható a p -ed rendű bal felső sarokminorok egyenlősége, azaz az

$$u_{11} \dots u_{pp} = \det(A_p)$$

összefüggés, ahol $\det(A_p)$ jelöli az A mátrix p -ed rendű bal felső sarokminorát. Tehát, ha $\det(A_p) \neq 0$, $p = 1, \dots, q-1$ esetén; de ha $\det(A_q) = 0$, akkor $u_{pp} \neq 0$, $p = 1, \dots, q-1$ és $u_{qq} = 0$. Ezzel bebizonyítottuk a következő állítást:

2.1. tétel. LU felbontási tétel

Egy n -ed rendű A mátrix akkor és csak akkor bontható fel egyértelműen (2-4) alakban, ha első $(n-1)$ számú bal felső sarokminora nem zérus. Ekkor $\det(A) = u_{11} \dots u_{nn}$.

Mint hogy a nem szinguláris mátrix sorai (és oszlopai) mindig átrendezhetők úgy, hogy a főminorok zérustól különbözőek legyenek, ezért – legfeljebb sorok (és

oszlopok) átrendezése után – mindig elvégezhető egy nonsinguláris mátrix (2–4) háromszögfelbontása.

Főelem-kiválasztás

A gyakorlati számításoknál – a számítógépekben alkalmazott korlátos pontosságú aritmetika miatt – nemcsak a zérussal, hanem a viszonylag kis abszolút értékű számokkal való osztást is kerülni kell, mivel ellenkező esetben jelentős kerekítési hibák halmozódhatnak fel. Az aktuális u_{ii} -vel, az ún. *főelemmel* való osztásból eredő kerekítési hibák csökkentésére ki kell keresni az e célra legmegfelelőbb elemet. Ezt az eljárást „*részleges*” vagy „*teljes főelem-kiválasztásnak*” nevezzük attól függően, hogy a főelem meghatározásához csak sorcserét vagy oszlopkeret is alkalmazunk. A numerikus tapasztalatok azt mutatják, hogy a csak sorcseréken alapuló részleges főelem-kiválasztás is biztosítja a módszer stabilitását, ugyanakkor sokkal hatékonyabban implementálható, mint a teljes főelem-kiválasztás.

A sorcserékkel kiegészített háromszögfelbontást jobban szemléltethetjük, ha L és U elemeit A megfelelő elemeinek helyére írjuk. A $(k-1)$ -edik lépés végrehajtása után (ha a mátrix rendje $n=6$ és $k=3$) az elrendezést a következő tömb szemlélteti:

$$\left[\begin{array}{cc|cccc|c} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & u_{16} & 0 \\ l_{21} & u_{22} & u_{23} & u_{24} & u_{25} & u_{26} & 0 \\ \hline l_{31} & l_{32} & a_{33} & a_{34} & a_{35} & a_{36} & s_3 \\ l_{41} & l_{42} & a_{43} & a_{44} & a_{45} & a_{46} & s_4 \\ l_{51} & l_{52} & a_{53} & a_{54} & a_{55} & a_{56} & s_5 \\ l_{61} & l_{62} & a_{63} & a_{64} & a_{65} & a_{66} & s_6 \end{array} \right], \quad (2-6)$$

ahol a_{ij} az eredeti mátrix elemei, természetesen az előző lépésekben végrehajtott esetleges sorcserék után. A k -edik lépésben kiszámítjuk az

$$s_i = a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk}, \quad i = k, \dots, n \quad (2-7)$$

értékeket, amelyeket u_{kk} főelemként nyernénk, miután felcseréltük egymással a k -edik és az i -edik sort.

Ha

$$|s_r| = \max_{i=k}^n |s_i|, \quad (2-8)$$

akkor a (2–6) tömb k -edik és r -edik sorát felcseréljük; a csere az s_k és s_r elemet is érinti. Ha a sorcsere elvégzése után a tömb i, j indexű elemét ismét l_{ij} -vel, ill. a_{ij} -vel jelöljük, akkor

$$u_{kk} = s_k; \quad l_{ik} = \frac{s_i}{u_{kk}}, \quad i = k+1, \dots, n \quad (2-9)$$

$$u_{ki} = a_{ki} - \sum_{j=1}^{k-1} l_{kj} u_{ji}, \quad i = k+1, \dots, n \quad (2-10)$$

Az r definíciójából következik, hogy

$$|l_{ik}| \leq 1. \quad (2-11)$$

Megjegyezzük, hogy a részleges főelem-kiválasztás egy, az A mátrix sorainak átrendezésével nyert, tehát PA szorzat alakban kifejezhető mátrix meghatározásával ekvivalens, ahol P egy n -ed rendű *permutációs* mátrix (tehát olyan mátrix, amelynek minden sora és oszlopa pontosan egy 1-es és $n-1$ zérust tartalmaz). A PA mátrix sorainak sorrendje olyan, hogy ha a PA mátrixot (sorcserék nélkül) felbontanánk

$$PA = LU \quad (2-12)$$

alakban, akkor az L mátrix elemei automatikusan kielégítenék a (2-11) feltételt.

Miután elvégeztük a (2-12) trianguláris felbontást, az $Ax = b$ egyenlet helyett a

$$PAx = Pb$$

permutált egyenletrendszert kell megoldani minden egyes jobb oldali b vektorra. Ez maga után vonja a (2-3) egyenletrendszerek

$$Ly = Pb, \quad Ux = y \quad (2-13)$$

szerinti módosítását. Az x megoldásvektort a (2-13) egyenletekből számíthatjuk. Ennek megfelelően

$$\left. \begin{aligned} y_i &= \tilde{b}_i - \sum_{k=1}^{i-1} l_{ik} y_k, & i &= 1, \dots, n \\ x_i &= \frac{y_i - \sum_{k=i+1}^n u_{ik} x_k}{u_{ii}}, & i &= n, \dots, 1 \end{aligned} \right\} \quad (2-14)$$

ahol $\tilde{b}_i$ jelöli a Pb vektor i -edik elemét.

Megjegyezzük, hogy ha a háromszögfelbontást a (2-7)–(2-10) képletek alkalmazásával végezzük, akkor minden egyes a_{ij} elemet pontosan egyszer kell felhasználni. Ezért az l_{ij} és az u_{ij} elemeket külön lefoglalt tömbökben nem kell tárolni; ezeket fokozatosan az A mátrix megfelelő elemeinek tárolási helyére írhatjuk (L főátlóját nem kell tárolni, hiszen tudjuk, hogy az csupa egyesből áll).

Sorok kiegyensúlyozása

Mint már említettük, a főelem-kiválasztást nem csak azért kellett bevezetni, hogy a háromszögfelbontás egyáltalában elvégezhető legyen (zérussal való osztás elkerülése), hanem azért is, hogy javítsunk a számítás pontosságán (viszonylag kis abszolút értékű osztók elkerülése). Azonban a főelem-kiválasztás ilyen formában nem minden esetben javítja a számítás pontosságát! Állításunk igazolására, tekintsük a

$$\begin{bmatrix} 1 & 1 \\ 3.0 & 10^{12} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 0.7 \cdot 10^{12} \end{bmatrix} \quad (2-15)$$

egyenletrendszert. Ha pl. 10 decimális jegyű lebegőpontos aritmetikát tételezünk fel, akkor a sorcserék nélküli háromszögfelbontás alkalmazásával azt kapjuk, hogy

$$L = \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 1 \\ 0 & 10^{12} - 3 \end{bmatrix} \cong \begin{bmatrix} 1 & 1 \\ 0 & 10^{12} \end{bmatrix}$$

ahol a $\cong$ jelet olvasd: „kerekítés után egyenlő”. A (2-3a) és (2-3b) egyenletek megoldásával

$$y = [0.9, 0.7 \cdot 10^{12} - 2.7]^T \cong [0.9, 0.7 \cdot 10^{12}]^T \quad \text{és} \quad x = [0.2, 0.7]^T,$$

ami a pontos megoldás tíz decimális jegyű közelítése. A főelem-kiválasztás ekvivalensét kapjuk, ha az egyenletek sorrendjét felcseréljük, és ismét a sorcserék nélküli LU felbontást alkalmazzuk. Ez esetben

$$L = \begin{bmatrix} 1 & 0 \\ 1/3 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 3 & 10^{12} \\ 0 & 1 - 10^{12}/3 \end{bmatrix} \cong \begin{bmatrix} 3 & 10^{12} \\ 0 & -10^{12}/3 \end{bmatrix},$$

$$y = [0.7 \cdot 10^{12}, 0.9 - 0.7 \cdot 10^{12}/3]^T \cong [0.7 \cdot 10^{12}, -0.7 \cdot 10^{12}/3]^T$$

és

$$x = [0, 0.7]^T,$$

ami a főelem-kiválasztás alkalmazása miatt létrejött hibás eredmény.

A részleges főelem-kiválasztás alkalmazása akkor ronthat a számítás pontosságán, amikor az

$$A = \begin{bmatrix} a^1 \\ \vdots \\ a^n \end{bmatrix}$$

együtthatómátrix sorai, pontosabban az a^i sorok (mint vektorok) normái¹, pl. a

$$q_i = \|a^i\|_\infty = \max_{j=1}^n |a_{ij}|, \quad i = 1, \dots, n$$

normák erősen eltérnek egymástól. Ha ez az eset áll fenn, akkor a sorokat célszerű úgy *skalázni*, hogy a sorok fent definiált normái megegyezzenek egymással. Az erre vonatkozó eljárást „*sorok kiegyensúlyozásának*” nevezzük.

A sorok kiegyensúlyozhatók pl. úgy, hogy az együtthatómátrix sorainak $q_1, \dots, q_n$ normáit előre kiszámítjuk, majd a háromszögfelbontás k -adik lépésében a részleges főelem-kiválasztásnak megfelelő (2–8) képletet az

$$\frac{|s_r|}{q_r} = \max_{i=k}^n \frac{|s_i|}{q_i} \quad (2-16)$$

képlettel helyettesítjük. Ez a képletcsere alig jár többletmunkával és a további számításokat – így a megoldást – sem terheli újabb kerekítési hibákkal. (Amennyiben az A mátrix mindegyik sorát ténylegesen osztanánk a sorhoz tartozó q_i normával, akkor még az egyenlet jobb oldalán szereplő b vektorok sorait is osztani kellene a megfelelő q_i értékekkel.)

Skaláris szorzatok kettős pontosságú számítása

A főelem-kiválasztáson túlmenően célszerű a nagyobb, pl. *kettős pontosságú műveletek* alkalmazásával védekezni a halmozódó kerekítési hibák ellen. Esetünkben ez azt jelenti, hogy a (2–7), (2–10) és (2–14) képletekben szereplő *skaláris szorzatokat* kettős pontossággal végezzük, vagyis a kétszeres pontosságú szorzatokat előbb összegezzük és csak az összeget kerekítjük egyszeres szóhosszra. Ugyancsak fontos a (2–9) típusú képletekben az osztandót kétszeres pontosságúként kezelni.

Az előbbieken részletezett algoritmust most két *Pascal eljárás* formájában foglaljuk össze.

```
procedure LUFelbont( var A: tVnXn;
                    n: integer;
                    var icsere: tIn;
                    var elojel: integer);
```

```
{ Ez az eljárás az n-ed rendű A mátrixot felváltja az A
sorainak átrendezésével nyert mátrix LU felbontásával. A és
n bemenő adatok. A kétindexű A ugyanakkor kimenő tömb is,
amelynek elrendezése a (2–6) tömb első két sorával és
oszlopával azonos. Az icsere kimenő vektor tartalmazza a
skalázott főelem-kiválasztásnak megfelelően felcserélt sorok
indexét. Az elojel kimenő változó értéke +1, ha a
```

¹ A normákról részletesebben lásd a következő pontban.

végrehajtott sorcserek száma páros; -1, ha páratlan. Ennek az eljárásnak az LUmegold eljárással való kombinálása alkalmas több jobb oldallal is rendelkező lineáris egyenletrendszer megoldására és így mátrix invertálására is.

Az LUFelbont eljárást használó programban definiálni kell a következő globális típusokat

```
tVnXn = array [1..nmax,1..nmax] of real;
tIn   = array [1..nmax] of integer;
tVn   = array [1..nmax] of real;
```

ahol az nmax konstans az n legnagyobb lehetséges értéke.

Ha az adott Pascal implementáció nem rendelkezik a double típussal, akkor definiálni kell azt a

```
double = real;
utasítással. }
```

var

```
i,j,k: integer;           {ciklusváltozók}
r: integer;               {a sorcsere indexe l. (2-16)}
s: double;                {összegzés kettős pontosságú változóban}
q: tVn;                   {az A sornormáinak reciprokát tartozó vektor}
```

```
begin elojel:=1;           {nincs még sorcsere}
  for i:=1 to n do         {ciklus a normák kiszámítására}
    begin s:=abs(A[i,1]);
      for j:=2 to n do
        if abs(A[i,j]) > s then
          s:=abs(A[i,j]);
        if s=0 then        {nincs 0-tól különböző elem}
          Hiba('0 sor az A mátrixban!'); {az i-edik sorban}
        q[i]:=1/s;         {az i-edik sor normájának reciproka}
      end;
    for k:=1 to n do       {az index inicializálása}
      icsere[k]:=k;        {a természetes sorrendnek megfelelően}
    for k:=1 to n do
      begin r:=k;
        for i:=k to n do   {ciklus r meghatározására}
          begin s:=A[i,k];
            for j:=1 to k-1 do
              s:=s-A[i,j]*A[j,k]; {l. (2-7)}
            A[i,k]:=s;
            if abs(s*q[i]) > abs(A[r,k]*q[r]) then {((2-16))}
              r:=i;
            end;
          if abs(A[r,k]*q[r]) = 0 then {A szinguláris!}
            Hiba('Az A mátrix szinguláris!');
          if r <> k then {Kell sorcsere?}
            begin elojel:=-elojel;
              j:=icsere[k]; icsere[k]:=icsere[r]; icsere[r]:=j;
              for j:=1 to n do {az r-edik sor cseréje}
                begin {a k-adik sorral}
                  s:=A[r,j]; A[r,j]:=A[k,j]; A[k,j]:=s;
                end;
              q[r]:=q[k]; {q[k]-ra már nincs szükség}
            end;
          end;
        end;
```



```

    for i:=k+1 to n do
      begin A[i,k]:=A[i,k]/A[k,k];           {1. (2-9)}
            s:=A[k,i];
            for j:=1 to k-1 do
              s:=s-A[k,j]*A[j,i];           {1. (2-10)}
            A[k,i]:=s;
          end;
    end;
end;
```

A következő eljárás a háromszögmátrixokat tartalmazó (2-13) egyenletrendszerek megoldására szolgál. Az eljárás a (2-14) képletek implementációja.

```

procedure LUmegold(      LU: tvnxn;
                       n: integer;
                       icsere: tIn;
                       var b: tvn);
```

{ Alkalmas az LU $x = b$ alakú, n lineáris egyenletből álló rendszer megoldására, ahol LU bemenő tömb, mely olyan alakban tartalmazza az együtthatómátrix háromszögfelbontásának L és U tényezőit, ahogyan az LUfelbont eljárás eredményezi. Az icsere bemenő vektor azonos az LUfelbont eljárás azonos nevű kimenő vektorával, amely a sorcserék indexeinek megőrzését szolgálja. A b vektor bemenetkor az egyenletrendszer jobb oldalát, kimenetkor pedig az x megoldást tartalmazza. Mivel ez az eljárás nem változtatja meg az LU, n és az icsere bemenő adatokat, több jobb oldali b vektor esetén is egymás után hívható anélkül, hogy újból kellene hívni az LUfelbont eljárást. A típusdefiniálásokkal kapcsolatosan l. az LUfelbont eljárást. }

```

var
  i, k: integer;           {ciklus változók}
  s: double;               {összegzés kettős pontosságú változóban}
  x: tvn;                  {segédvektor}

begin
  for i:=1 to n do          {a permutáció elvégzése}
    x[i]:=b[icsere[i]];     {az x segédvektor}
  for i:=1 to n do          {felhasználásával}
    b[i]:=x[i];
  for i:=1 to n do
    begin s:=b[i];          {a (2-14) első egyenlete}
      for k:=1 to i-1 do
        s:=s-LU[i,k]*b[k];
      b[i]:=s;
    end;
  for i:=n downto 1 do
    begin s:=b[i];          {a (2-14) második egyenlete}
      for k:=i+1 to n do
        s:=s-LU[i,k]*b[k];
      b[i]:=s/LU[i,i];
    end;
end;
```


Az LUfelbont eljárás végrehajtása kb. $n^3/3$ szorzást és összeadást igényel, ezért a megoldás kiszámításához lényegében ennyi műveletre van szükség egy (vagy néhány) jobb oldal esetén is. Az invertálás pedig – az LUmegold eljárás n -szeri hívásával együtt – összesen n^3 műveletet igényel.

Megjegyezzük, hogy az előbbi két rutin felhasználásával viszonylag egyszerűen megírható néhány, az alkalmazásokban gyakrabban előforduló feladat programja. Így például:

- a) Az $Ax = b$ lineáris algebrai egyenletrendszer megoldása nyilvánvalóan az:

LUfelbont(A, n, icsere, elojel)

LUmegold(A, n, icsere, b)

eljárások egymást követő hívásával valósítható meg. Az x megoldást a b vektor tartalmazza. Az eredeti A mátrix elemei elvesznek.

- b) *Több jobb oldal.* Ha azonos A mátrixot feltételezve, különböző b jobb oldali vektorokkal egymás után kívánunk megoldani lineáris egyenletrendszereket, akkor célszerű egymás után hívni az

LUmegold(A, n, icsere, b)

eljárást, természetesen nem az eredeti A mátrixszal, hanem az LUfelbont eljárás eredményeként nyert A és icsere tömbökkel.

- c) *Mátrix invertálása.* Célszerű, ha az előbbihez hasonló felépítésű programrészletet írunk, mivel az invertálást olyan (2-1) alakú egyenletrendszer megoldásaként lehet interpretálni, amelynek jobb oldalát az $e_1, \dots, e_n$ egységvektorok alkotják. Ezért a jobb oldal beolvasása helyett generáljuk az egységvektorokat.

```
LUfelbont(A, n, icsere, elojel);
for i := 1 to n do           {ciklus az oszlopokra}
  begin
    for j := 1 to n do       {az i-edik egységvektor}
      e[j] := 0;             {e egy tvn típusú vektor}
    e[i] := 1;
    LUmegold(A, n, icsere, e);
    for j := 1 to n do
      Ainv[j, i] := e[j];    {Ainv tvn x n típusú tömb}
  end;
```

- d) *Mátrix determinánsának kiszámítása.* Már említettük, hogy a (2-4)-ben szereplő mátrixok determinánsát tekintve, sorcserék nélküli háromszögfelbontás esetén nyilvánvaló a

$$\det(A) = u_{11} \dots u_{nn}$$

összefüggés. A főelem-kiválasztással kiegészített háromszögfelbontásnál viszont az alkalmazott sorcserék miatt (minden sorcsere megváltoztatja a determináns előjelét) az előbbi összefüggés alkalmazása hibás előjellelű determinánst eredményezhet. Ez az oka annak, hogy az LUFelbont eljárás elojel változójában megőriztük a végrehajtott sorcserék számainak páros vagy páratlan voltát. Ezért a LUFelbont eljárás hívását követően, pl. a

```
det:=elojel;           {elojel az LUFelbont eredménye}
for i:=1 to n do
  det:=det*A[i,i];     {A is az LUFelbont eredménye}
```

programrésszel számíthatjuk az A mátrix determinánsát. Az A mátrix determinánsát a det változó tartalmazza és az eredeti A elemeit elrontottuk.

Nagy rendszámú mátrixok esetén előfordulhat, hogy a determináns a számítógép lebegőpontos dinamikus tartományán kívül esik, azaz alá- vagy túlsordulás következik be. Ilyen esetekben a determináns d , e rendezett párként tárolható

$$det = d * 2^e$$

alakban, ahol e egész szám és $\frac{1}{16} \leq |d| < 1$ vagy $d = 0$. Ekkor az előbbi programrész pl. a következővel helyettesíthető:

```
d:=elojel; {elojel az LUFelbont eljárás eredménye}
e:=0;      {d real, e integer típusú változó}
for i:=1 to n do
  begin
    d:=d*A[i,i]; {A is az LUFelbont eredménye}
    while abs(d)>=1 do { |d| < 1 }
      begin d:=d*0.0625; e:=e+4; end;
    while abs(d)<0.0625 do { |d| >= 1/16 }
      begin d:=d*16; e:=e-4; end;
  end;
```


3. Gyengén meghatározott egyenletek Hibaanalízis

Feltéve, hogy az n -ed rendű A mátrix nonszinguláris, az $Ax = b$ egyenlet megoldása

$$x = A^{-1}b$$

alakban írható. Tegyük fel, hogy az A mátrix elemei úgy vannak skálázva, hogy közülük a legnagyobb abszolút értékű abszolút értéke eggyel egyenlő. Tegyük fel továbbá, hogy az $A^{-1} = [a_{ji}^{(-1)}]$ inverz mátrixnak vannak igen „nagy” elemei, melyek közül az egyik

$$a_{ji}^{(-1)} = \frac{A_{ij}}{\det(A)} = \frac{A_{ij}}{a_{i1}A_{i1} + \dots + a_{ij}A_{ij} + \dots + a_{in}A_{in}},$$

ahol A_{ij} az a_{ij} elemhez tartozó előjeles $(n-1)$ -ed rendű aldeterminánst (minort) jelenti. Az a feltevés, hogy $a_{ji}^{(-1)}$ nagy, azt jelenti, hogy A_{ij} a $\det(A)$ -hoz képest nagy. Tekintettel arra, hogy $\det(A)$ -nak az i -edik sor szerinti kifejtésében az egyik tag $a_{ij}A_{ij}$, nyilvánvaló, hogy egy kicsiny eltérés a_{ij} -ben (azaz egy kicsiny relatív hiba az egységhez viszonyítva, amelyre A elemeit skáláztuk) nagy relatív eltérést okozhat $\det(A)$ -ban (mivel A_{ij} viszonylag nagy) és ezzel együtt $a_{ji}^{(-1)}$ -ben. Ez a nagy eltérés azután az x megoldás nagymértékű torzulását idézheti elő. Ugyancsak jelentős relatív hibát okozhat a megoldásban b kismértékű megváltoztatása.

Ugyanezt a hatást létrehozhatják a számítás során fellépő és *halmozódó kerekítési hibák* is, minthogy egy számítás közben jelentkező kerekítési hiba hatása egyenértékű az A mátrix (bemenő adatok) hibás voltával. Ennek alapján a továbbiakban a hibaanalízisben csak a kezdeti adatok perturbációinak a megoldás pontosságára gyakorolt hatását vizsgáljuk.

Ha az A mátrix elemeit úgy skálázzuk, hogy a legnagyobb abszolút értékű elem abszolút értéke eggyel egyenlő, és ha az A^{-1} inverz mátrixnak viszonylag nagy elemei vannak, a mátrixot és vele együtt az egyenletrendszert *gyengén meghatározottnak* (rosszul kondicionáltnak) nevezzük. (Ha azonban az inverz mátrix legnagyobb abszolút értékű elemeinek nagyságrendje eggyel egyenlő, a mátrixot *jól meghatározottnak* (jól kondicionáltnak) nevezzük.)

Általánosabban, ha egy feladat adatainak kismértékű megváltoztatása a megoldás nagymértékű torzulását idézi elő, akkor a feladatot, ill. annak megoldását *instabilisnak* vagy *gyengén meghatározottnak* nevezzük.

A következő egyszerű példa jól szemlélteti a gyengén meghatározott egyenletrendszerekben rejlő veszélyeket. Tekintsük a

$$\begin{bmatrix} 2 & 6 \\ 2 & 6.00001 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 8 \\ 8.00001 \end{bmatrix} \quad (3-1)$$

egyenletrendszert, amelynek megoldása az $\mathbf{x} = [1, 1]^T$ vektor. Ugyanakkor a

$$\begin{bmatrix} 2 & 6 \\ 2 & 5.99999 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 8 \\ 8.00002 \end{bmatrix} \quad (3-2)$$

perturbált egyenletrendszer megoldása az $\mathbf{x} = [10, -2]^T$ vektor. Ez a meglepő eredmény azt jelenti, hogy az a_{22} elem $2 \cdot 10^{-5}$ mértékű és a b_2 elem 10^{-5} mértékű megváltoztatása a megoldás óriási megváltozását eredményezte. A (3-1) egyenletrendszer inverz mátrixának elemei 10^5 nagyságrendűek, ami arra vall, hogy az $\mathbf{A}$ mátrix gyengén meghatározott. Ez a példa azt sugalmazhatja, hogy $\det(\mathbf{A})$ kicsinysége miatt lesz $\mathbf{A}$ gyengén meghatározott. Megjegyezzük azonban, hogy vannak mátrixok, amelyeknek a determinánsa nagyon kis szám, mégis nagyon jól meghatározottak. (Egy egyszerű példa erre az esetre az a diagonálmátrix, amelynek minden főátló eleme pl. 10^{-1} -gyel egyenlő és elég nagy az n rendszáma ahhoz, hogy $\det(\mathbf{A}) = 10^{-n}$ nagyon kicsi legyen.)

Megmutatjuk, hogy az $\mathbf{A}\mathbf{x} = \mathbf{b}$ egyenlet megoldásának helyességét nem elegendő oly módon ellenőriznünk, hogy az $\tilde{\mathbf{x}}$ kiszámított megoldást behelyettesítjük az eredeti egyenletbe, és megvizsgáljuk az

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}} \quad (3-3)$$

hibavektor elemeinek a zérustól való eltérését. Egy pontos megoldás mindig kis eltéréseket eredményez ugyan, de a hibavektor kicsiny elemei nem biztosítják, hogy a megoldás tényleg pontos. Valóban, a (3-3) egyenletből közvetlenül adódó

$$\mathbf{A}^{-1}\mathbf{r} = \mathbf{A}^{-1}\mathbf{b} - \tilde{\mathbf{x}}$$

alakból már nyilvánvaló, hogy ha $\mathbf{A}^{-1}$ -nek vannak nagy elemei, akkor kicsiny $\mathbf{r}$ esetben is nagy lehet az $\mathbf{A}^{-1}\mathbf{b}$ pontos és az $\tilde{\mathbf{x}}$ kiszámított megoldás közötti különbség.

Mégis, van egy ügyes fogás, amellyel az $\mathbf{A}\mathbf{x} = \mathbf{b}$ egyenlet kiszámított megoldásának helyessége viszonylag kevés többszámítással ellenőrizhető. Az $\mathbf{A}$ mátrix

LU felbontásának ismeretében az egyenletrendszert megoldjuk a $\tilde{b}_i = \sum_{j=1}^n a_{ij}$

$i=1, \dots, n$ elemekkel meghatározott jobb oldallal, majd az így kapott megoldást összevetjük az $\tilde{\mathbf{x}} = [1, \dots, 1]^T$ pontos megoldással. (Mivel a megoldás pontossága lényegében az $\mathbf{A}$ mátrix kondíciójától függ, mindkét jobb oldal esetén hasonló pontosságú megoldás várható.)

Normák

Valamely mátrix normája, egy a mátrixhoz rendelt skaláris szám, amely bizonyos értelemben mértéke a mátrix „nagyságának”. Az $m \times n$ típusú A mátrix normájától, amelyet $||A||$ -val jelölünk, megköveteljük a következő tulajdonságok teljesülését:

$$(1) \quad ||A|| \geq 0 \quad \text{és} \quad ||A|| = 0 \quad \text{akkor és csak akkor, ha} \quad A = 0.$$

$$(2) \quad ||cA|| = |c| \cdot ||A||, \quad \text{ahol } c \text{ tetszőleges valós szám.}$$

$$(3) \quad ||A + B|| \leq ||A|| + ||B|| \quad (\text{háromszög egyenlőtlenség}).$$

Ha A $m \times n$ és B $n \times p$ típusú mátrix, akkor a mátrixnormától még a

$$(4) \quad ||AB|| \leq ||A|| \cdot ||B|| \quad (\text{Schwarz-egyenlőtlenség})$$

tulajdonságot is megköveteljük.

Az ezeket a feltételeket kielégítő $||A||$ norma sokféleképpen definiálható; ezek közül hármat említünk meg, mindegyiket tetszőleges, $m \times n$ típusú valós mátrixra értelmezzük:

$$(1) \quad ||A||_E = \left(\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2 \right)^{1/2}, \quad \text{az euklideszi vagy Frobenius-féle norma.}$$

$$(2) \quad ||A||_{\infty} = \max_{i=1}^m \sum_{j=1}^n |a_{ij}|, \quad \text{az ún. végtelen vagy maximum norma.}$$

Ha A tetszőleges, $m \times n$ típusú valós mátrix, akkor $A^T A$ valós, szimmetrikus, nemnegatív definit n -ed rendű mátrix. (Vagyis $x^T A^T A x \geq 0$, ha $x \neq 0$.) Ezért minden sajátértéke nemnegatív valós szám, tehát $\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2$ alakban írhatók, ahol $\sigma_1, \sigma_2, \dots, \sigma_n$ valós számok és nemnegatívnak választhatók.

Az $A^T A$ mátrix sajátértékeinek nemnegatív gyökeit az A mátrix szinguláris értékeinek nevezzük és ezeket $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ sorrendben jelöljük.

$$(3) \quad ||A||_S = \sigma_1, \quad \text{a spektrál norma.}$$

Ha az $\mathbf{x} = [x_1, \dots, x_m]^T$ vektort egyoszlopú mátrixnak tekintjük, akkor az euklideszi és a spektrál mátrixnorma speciális eseteként a

$$||\mathbf{x}|| = \left(\sum_{i=1}^m x_i^2 \right)^{1/2} = (\mathbf{x}^T \mathbf{x})^{1/2} \quad (3-4)$$

euklideszi vektornormát, a végtelen mátrixnorma alkalmazásával pedig a

$$||\mathbf{x}||_{\infty} = \max_{i=1}^m |x_i|$$

maximum vektornormát nyerjük.

3.1. tétel. Legyen $\mathbf{A}$ $m \times n$ típusú mátrix és $\mathbf{x}$ tetszőleges n elemű vektor. Ebben az esetben

$$\max_{\mathbf{x} \neq 0} \frac{||\mathbf{A}\mathbf{x}||}{||\mathbf{x}||} = ||\mathbf{A}||_S, \quad (3-5)$$

ahol a vektorok normáját a (3-4) euklideszi értelemben értjük.

Bizonyítás. Tetszőleges $\mathbf{x} \neq 0$ vektorra

$$\frac{||\mathbf{A}\mathbf{x}||^2}{||\mathbf{x}||^2} - ||\mathbf{A}||_S^2 = \frac{\mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x}}{\mathbf{x}^T \mathbf{x}} - \sigma_1^2 = \frac{\mathbf{x}^T (\mathbf{A}^T \mathbf{A} - \sigma_1^2 \mathbf{I}) \mathbf{x}}{\mathbf{x}^T \mathbf{x}} \leq 0, \quad (3-6)$$

ugyanis az $\mathbf{A}^T \mathbf{A} - \sigma_1^2 \mathbf{I}$ mátrix $\sigma_i^2 - \sigma_1^2$ sajátértékei nempozitívak és ennél fogva a számláló nempozitív definit. A (3-6)-ban egyenlőséget kapunk, ha $\mathbf{x}$ éppen $\mathbf{A}^T \mathbf{A}$ -nak egy σ_1^2 -hez tartozó sajátvektora. Ezzel a tételt bebizonyítottuk.

Közvetlenül a (3-5) összefüggésből következik, hogy

$$||\mathbf{A}\mathbf{x}|| \leq ||\mathbf{A}||_S ||\mathbf{x}||. \quad (3-7)$$

Tegyük fel, hogy $\mathbf{A}$ egy *nem szinguláris* (egyben négyzetes) mátrix és vizsgáljuk meg a megoldás érzékenységet a kezdeti adatok perturbációinak függvényében. Ha $\mathbf{x}$ az

$$\mathbf{A}\mathbf{x} = \mathbf{b} \quad (3-8)$$

egyenlet pontos megoldása, akkor a jobb oldal $\delta \mathbf{b}$ perturbációira fennáll az

$$\mathbf{A}(\mathbf{x} + \delta \mathbf{x}) = \mathbf{b} + \delta \mathbf{b} \text{ egyenlet.}$$

Az utóbbiból kivonva a (3-8) egyenletet az

$$\mathbf{A} \delta \mathbf{x} = \delta \mathbf{b}, \text{ ill. } \delta \mathbf{x} = \mathbf{A}^{-1} \delta \mathbf{b}$$

összefüggést kapjuk. Felhasználva a mátrixnormákra vonatkozó Schwarz-egyenlőtlenséget:

$$||b|| = ||Ax|| \leq ||A|| ||x||$$

és

$$||\delta x|| \leq ||A^{-1}|| ||\delta b||.$$

Ha a két utóbbi egyenlőtlenséget összeszorozzuk, akkor az x megoldás relatív hibája és a b jobb oldal relatív megváltoztatása közötti

$$\frac{||\delta x||}{||x||} \leq ||A|| ||A^{-1}|| \frac{||\delta b||}{||b||} \quad (3-9)$$

összefüggést nyerjük (feltéve, hogy $b \neq 0$). Ezek szerint eléggé meglepő, hogy a b jobb oldal perturbációjának hatása a megoldás pontosságára mindenekelőtt az $||A|| ||A^{-1}||$ mennyiségtől, tehát az A együtthatómátrixtól függ.

Hasonlóképpen, az A mátrix δA perturbációjának hatására is felső korlát adható az x megoldás relatív megváltozása és az A mátrix relatív megváltozása között. Valóban, az A mátrix δA perturbációjára vonatkozó

$$(A + \delta A)(x + \delta x) = b$$

egyenletből

$$Ax + A\delta x + \delta A(x + \delta x) = b$$

és $Ax = b$ miatt

$$A\delta x = -\delta A(x + \delta x)$$

következik. Az A mátrixról feltettük, hogy nem szinguláris, ezért

$$\delta x = -A^{-1}\delta A(x + \delta x)$$

és a normákra vonatkozó Schwarz-egyenlőtlenség alkalmazásával

$$||\delta x|| \leq ||A^{-1}|| ||\delta A|| ||x + \delta x||$$

adódik, ami átírható a

$$\frac{||\delta x||}{||x + \delta x||} \leq ||A|| ||A^{-1}|| \frac{||\delta A||}{||A||} \quad (3-10)$$

kívánt alakra.

Megjegyezzük, hogy az A mátrix δA perturbációja miatt az $A + \delta A$ mátrix szinguláris is lehet.

Minthogy a (3-9) és (3-10) egyenlőtlenség jobb oldalán szorzó tényezőként egyaránt a $||A^{-1}|| ||A||$ mennyiség szerepel, a megoldás pontossága e korlát függvénye. Az A skálázására vonatkozó feltevésünk egyenértékű azzal, hogy $||A||$

nagyságrendje egy, és mivel az A mátrixot akkor neveztük gyengén meghatározottnak, amikor ilyen skálázás mellett inverzének elemei nagyok voltak (vagyis amikor a $\|A^{-1}\|$ norma is viszonylag nagy), az $\|A\| \|A^{-1}\|$ mennyiséget a meghatározottság mértékének lehet tekinteni. Ezt a skaláris mennyiséget az A mátrix *kondíciós számának* nevezzük, természetesen az alkalmazott normára vonatkoztatva.

Tegyük fel, hogy A nem szinguláris mátrix. Ha $\sigma_1^2 \geq \sigma_2^2 \geq \dots \geq \sigma_n^2 > 0$ jelöli az $A^T A$ pozitív definit mátrix sajátértékeit¹, akkor $\sigma_n^{-2} \geq \sigma_{n-1}^{-2} \geq \dots \geq \sigma_1^{-2}$ jelöli az $(A^{-1})^T A^{-1}$ mátrix sajátértékeit. (Ez az $A^T A v = \sigma^2 v$ sajátérték-feladat definíciós egyenletből közvetlenül adódik: először szorozzuk ezt az egyenletet balról a nem szinguláris A -val és az $A v$ vektort jelöljük u -val. Az így kapott $A A^T u = \sigma^2 u$ egyenletet szorozzuk balról a $\sigma^{-2} (A A^T)^{-1}$ mátrixszal. Eredményként az $(A^{-1})^T A^{-1} u = \sigma^{-2} u$, $u \neq 0$ sajátérték-feladatot nyerjük, vagyis ha σ^2 az $A^T A$ mátrix sajátértéke, akkor σ^{-2} az $(A^{-1})^T A^{-1}$ mátrix sajátértéke.)

Ebből és a spektrál norma definíciójából következik, hogy az A mátrix *spektrál kondíciós száma*

$$\text{cond}_S(A) = \|A\|_S \|A^{-1}\|_S = \frac{\sigma_1}{\sigma_n} \geq 1,$$

ahol σ_1 , ill. σ_n jelöli az A mátrix legnagyobb, ill. legkisebb szinguláris értékét.

Ha az A mátrix szimmetrikus, akkor $A^T A = A^2$ és így a σ_i^2 , $i=1, \dots, n$ értékek az A mátrix λ_i sajátértékeinek négyzetei. Ezért ebben az esetben

$$\text{cond}_S(A) = \frac{|\lambda|_{\max}}{|\lambda|_{\min}},$$

ahol $|\lambda|_{\max}$, ill. $|\lambda|_{\min}$ jelöli az A mátrix legnagyobb, ill. legkisebb abszolút értékű sajátértékének az abszolút értékét.

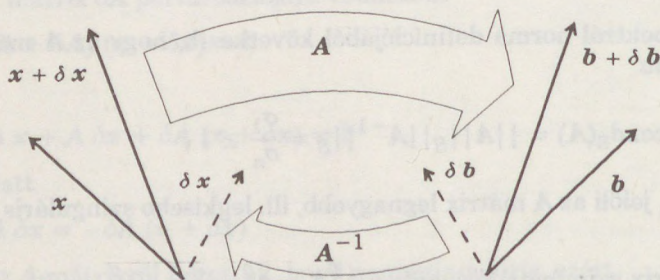
¹ A sajátértékekről részletesebben lásd a Valós szimmetrikus mátrix sajátértékei és sajátvektorai c. fejezetben.

4. A megoldás iterációs finomítása

Nyilvánvalóan nehézségbe ütközik, ha a lineáris egyenletrendszer megoldását pontosabban szeretnénk meghatározni, mint amilyen a rendelkezésre álló számítógép lebegőpontos szavának pontossága. Különösen nagy rendszámú egyenletrendszerek esetén olyan példák is előfordulnak amikor még a számítógép pontossági lehetőségeinek megfelelő eredmény sem érhető el könnyen. Direkt módszerek esetén a felhalmozott kerekítési hibák azt a látszatot kelthetik, mintha az együtthatómátrix gyengén meghatározott vagy majdnem szinguláris lenne. Két-három értékes jegyet akkor is veszíthetünk, ha a mátrix jól kondicionált (vagy annak hisszük).

Ilyen esetekben egy egyszerű fogással, ún. „*utóiteráció*” alkalmazásával megkapható a számítógép pontossági korlátjának megfelelő legnagyobb pontosságú megoldás.

Az iterációs javítás matematikai alapja nagyon egyszerű (lásd a 4.1. ábrán vázolatokat).



4.1. ábra. Az $Ax = b$ egyenlet megoldásának iterációs finomítása

Tegyük fel, hogy x az

$$Ax = b \quad (4-1)$$

egyenlet pontos, azonban ismeretlen megoldása. Ismerjük viszont a (4-1) egyenlet $x + \delta x$ közelítő megoldását, ahol δx az ismeretlen hiba. Ha a közelítő megoldást megszorozzuk A -val, akkor a jobb oldali vektornak egy $b + \delta b$ közelítését nyerjük. Vagyis

$$A(x + \delta x) = b + \delta b \quad (4-2)$$

Ha a (4-2) egyenletből kivonjuk a (4-1) egyenletet, akkor

$$A \delta x = \delta b \quad (4-3)$$

adódik. Ha a (4-2) egyenletből kifejezzük a δb vektort és behelyettesítjük a (4-3) egyenletbe, akkor azt kapjuk, hogy

$$A \delta x = A(x + \delta x) - b \quad (4-4)$$

Ennek az egyenletnek a jobb oldalát ismerjük, hiszen $x + \delta x$ a közelítő megoldás, amelyet finomítani kívánunk. Mivel két közel egyforma szám kivonása a további számítás során jelentős számjegyesztést okozhat, alapvető fontosságú, hogy a (4-4) jobb oldalát *nagyobb*; pl. *kétszeres pontossággal* végezzük! Ha a (4-4) egyenlet megoldásával nyert δx hibavektort kivonjuk az $x + \delta x$ közelítő megoldásból, akkor várható, hogy x a keresett megoldás egy jobb közelítése.

Az ábra szerint az $x + \delta x$ első közelítést megszorozzuk A -val és a $b + \delta b$ vektort kapjuk. Az ismert b vektor kivonása δb -t adja. Az A^{-1} inverz mátrix a δb vektort δx -be viszi át. Az utóbbit kivonjuk $x + \delta x$ -ből és az x javított megoldást nyerjük. Különösen előnyös, ha a megoldás kezdő közelítését a $(PA =)LU$ felbontás alkalmazásával nyertük, mert ekkor a (4-4) egyenlet megoldásához már csak a trianguláris

$$Ly = P(A(x + \delta x) - b) \text{ és } U\delta x = y$$

egyenletrendszereket kell megoldani.

Az iterációs javítás egyszerűen megvalósítható a következő Pascal eljárás alkalmazásával:

```
procedure LUjavit(      A, LU: tVnXn;
                        n: integer;
                        icsere: tIn;
                        b: tVn;
                        var x: tVn);
```

{ Ez az eljárás alkalmas az $Ax = b$ lineáris egyenletrendszer x közelítő megoldásának iterációs finomítására. Az A mátrix, a b és az icsere vektorok, valamint az n rendszám bemenő adatok. Az LU bemenő adat az A mátrix LU felbontását tartalmazó tömb. Az LU tömb és az icsere vektor az LUfelbont eljárás eredményei. Az LUjavit eljárás eredménye az x vektor, mely a közelítő megoldás finomított értékeit tartalmazza. A típusdefiníciókat l. az LUfelbont eljárásban. }

```
var
  i, j: integer;           {ciklusváltozók}
  s: double;               {összegzés kettős pontosságú változóban}
  r: tVn;                  {különbségvektor}

begin
  for i := 1 to n do       {a különbségvektor számítása}
    begin s := -b[i];      {összegzés kettős}
      for j := 1 to n do   {pontosságú változóban}
        s := s + A[i,j]*x[j];
        r[i] := s;
      end;
    LUmegold(LU,n,icsere,r); {a (4-4) megoldása}
    for i := 1 to n do     {a jobb közelítés számítása}
      x[i] := x[i] - r[i];
    end;
```


Emlékeztetünk arra, hogy az LUFelbont eljárás elrontja a bemenő mátrixot, mivel annak helyére „ráírja” az LU felbontásból származó mátrixokat. Az iterációs javítás során azonban szükség van mind az eredeti mátrixra, mind annak felbontására. Ezért az LUFelbont eljárás hívását megelőzően el kell készíteni az eredeti mátrix másolatát. Az LUMegold eljárás is elrontja a $\mathbf{b}$ vektort az $\mathbf{x}$ meghatározása során, ezért a $\mathbf{b}$ vektort is le kell másolni. Amennyiben ez a többlettárolás megoldható, az iterációs finomítást mindenképpen el kell végezni.

Az utóiteráció ugyanis csak n^2 nagyságrendű műveletet igényel (vagyis lépésenként egy mátrix vektorral való szorzását és a két trianguláris mátrixú egyenletrendszer megoldását). Ha már a megoldás kezdő közelítése is elegendően pontos, az iteráció azt nem rontja el, ugyanakkor megmentheti azt az egyébként hibás megoldást, amelyre már n^3 nagyságrendű műveletet fordítottunk.

Az LUjavít eljárás egymás után többször is hívható. Amennyiben a kezdő közelítés a pontos megoldástól nincs nagyon távol, általában egyetlen hívás is elegendő. A konvergencia ellenőrzése miatt azonban célszerű egy második hívást is beiktatni.

5. Pozitív definit mátrix szimmetrikus felbontása

Emlékeztetünk arra, hogy a szimmetrikus, valós $\mathbf{A}$ mátrixot *pozitív definitnek* nevezzük, ha

$$\mathbf{x}^T \mathbf{A} \mathbf{x} > 0 \quad \text{minden } \mathbf{x} \neq \mathbf{0} \quad (5-1)$$

vektorra.

Az alkalmazásokban $\mathbf{A}$ pozitív definit mátrix voltát gyakran az $\mathbf{x}^T \mathbf{A} \mathbf{x}$ fizikai jelentése magyarázza. (Így pl. a tartók elméletében: ha $\mathbf{x}$ az alakváltozás-vektort, és $\mathbf{A}$ a merevségi mátrixot jelöli, akkor $\frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x}$ az alakváltozási energia kifejezője.)

5.1. tétel. Cholesky-felbontás

A szimmetrikus pozitív definit $\mathbf{A}$ mátrix felbontható

$$\mathbf{A} = \mathbf{L} \mathbf{L}^T \quad (5-2)$$

alakra, ahol $\mathbf{L}$ valós, nem szinguláris alsó háromszögmátrix. Ezenkívül, ha az $\mathbf{L}$ főátlóelemeit pozitívnak választjuk, akkor a felbontás egyértelmű.

Bizonyítás. Az állítást teljes indukcióval bizonyítjuk. Tegyük fel, hogy a tétel $(n-1)$ -ed rendű szimmetrikus pozitív definit mátrixokra igaz. Az n -ed rendű $\mathbf{A}$ mátrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{n-1} & \mathbf{a} \\ \mathbf{a}^T & a_{nn} \end{bmatrix}$$

szerint particionálható, ahol A_{n-1} egy $n-1$ -ed rendű mátrix, a egy $n-1$ elemű vektor, a_{nn} pedig szám.

Tekintsük az

$$\begin{bmatrix} x' \\ 0 \end{bmatrix}$$

alakú vektorok halmazát, ahol x' tetszőleges $n-1$ elemű vektor. E vektorokkal az (5-1) egyenlőtlenség a következő összefüggésre vezet:

$$(x')^T A_{n-1} x' > 0 \quad \text{minden } x' \neq 0$$

vektorra. Vagyis bebizonyítottuk, hogy ha A pozitív definit mátrix, akkor az utolsó sorának és oszlopának elhagyásával nyert A_{n-1} mátrix szintén pozitív definit. Ezért alkalmazható az indukciós feltétel az A_{n-1} mátrixra. Ennek értelmében létezik egy olyan valós, nem szinguláris L_{n-1} mátrix, amely kielégíti az

$$L_{n-1} L_{n-1}^T = A_{n-1}$$

egyenletet. Mivel L_{n-1} valós, nem szinguláris, létezik az $n-1$ elemű valós l vektor, amelyre teljesül hogy

$$L_{n-1} l = a.$$

Ez esetben írható, hogy

$$\left[\begin{array}{c|c} L_{n-1} & 0 \\ \hline l^T & l_{nn} \end{array} \right] \left[\begin{array}{c|c} L_{n-1}^T & l \\ \hline 0^T & l_{nn} \end{array} \right] = \left[\begin{array}{c|c} A_{n-1} & a \\ \hline a^T & a_{nn} \end{array} \right] \quad (5-3)$$

feltéve, hogy

$$l^T l + l_{nn}^2 = a_{nn}. \quad (5-4)$$

Még be kell látni, hogy létezik egy olyan valós l_{nn} szám, amely kielégíti az (5-4) egyenletet. Ha (5-3) mindkét oldalának determinánsát képezzük azt kapjuk, hogy

$$[\det(L_{n-1})]^2 l_{nn}^2 = \det(A).$$

A lineáris algebrából ismeretes, hogy ha A pozitív definit mátrix, akkor minden bal felső sarokminora, így $\det(A)$ is, pozitív. Ezért l_{nn}^2 pozitív és l_{nn} pozitívnek választható.

A tétel $n = 1$ esetén nyilvánvalóan igaz és így az indukció értelmében bebizonyítottuk tekinthetjük.

Megjegyezzük, hogy az (5-3) egyenletből az is következik, hogy az L mindegyik sorában szereplő elemek négyzetösszege egyenlő A megfelelő sorának diagonálemével. Vagyis:

$$\sum_{j=1}^i l_{ij}^2 = a_{ii}, \quad i = 1, \dots, n. \quad (5-5)$$

Ha a sorok kiegyensúlyozásának megfelelően az A főátlóelemeiről feltesszük, hogy $a_{ii} = 1, i = 1, \dots, n$ (ez megtehető, hiszen $a_{ii} = \mathbf{e}_i^T A \mathbf{e}_i > 0$ miatt minden sor leosztható a főelemével), akkor (5-5)-ből következik, hogy $|l_{ij}| \leq 1$ minden i, j indexre. Ez viszont nem más, mint a (2-11) egyenlőtlenség, amely az LU felbontásban végrehajtott sorcserék eredményeként keletkezett. Mivel a (2-11) feltétel itt sorcserék végrehajtása nélkül is teljesül, kimondhatjuk, hogy pozitív definit mátrix Cholesky-felbontása főelem-kiválasztás nélkül is stabilis eljárás.

Az $A = LL^T$ Cholesky-felbontás alternatívájaként egy olyan felbontás is alkalmazható, amelynél négyzetgyökvonás nem szerepel. Mivel a Cholesky-felbontásban szereplő L alsó háromszögmátrix $l_{11}, \dots, l_{nn}$ főátlóelemei mind pozitívak (tehát osztásra alkalmasak), az L mátrix kifejezhető

$$\begin{bmatrix} 1 & & & \\ \frac{l_{21}}{l_{11}} & 1 & & \\ \frac{l_{31}}{l_{11}} & \frac{l_{32}}{l_{22}} & 1 & \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix} \begin{bmatrix} l_{11} & & & \\ & l_{22} & & \\ & & l_{33} & \\ & & & \cdot \end{bmatrix}$$

szorzat alakban. Vagyis az

$$L = \tilde{L} \langle l_{11}, \dots, l_{nn} \rangle \quad (5-6)$$

összefüggéssel egy olyan $\tilde{L}$ alsó háromszögmátrixot értelmezünk, amelynek főátlóelemeit egyesek alkotják. Ennek az összefüggésnek az (5-2)-be való behelyettesítésével

$$A = LL^T = \tilde{L} \langle l_{11}^2, \dots, l_{nn}^2 \rangle \tilde{L}^T$$

adódik, vagyis

$$A = \tilde{L} D \tilde{L}^T \quad (5-7)$$

a szimmetrikus pozitív definit mátrix egy másik felbontása, ahol D pozitív definit diagonálmátrix.

Ha az A valós mátrix szimmetrikus, de nem pozitív definit, akkor általános esetben az (5-2), (5-7) felbontások egyike sem végezhető el. Ezt már a

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

egyszerű eset is mutatja.

Megjegyezzük, hogy abból, hogy a felbontás nem létezik, még nem következik, hogy a mátrix gyengén meghatározott.

Ha A nemnegatív definit szimmetrikus mátrix (azaz ha $x^T A x \geq 0$ minden $x \neq 0$ vektorra), akkor mindkét felbontás elvégezhető valós, azonban szinguláris L, D mátrixokkal és a felbontás nem egyértelmű.

Ha az A mátrixnak legalább egy negatív sajátértéke van, akkor az $A = L L^T$ felbontás nem végezhető el valós L mátrixszal, azonban létezik olyan komplex elemű L mátrix, amellyel a Cholesky-felbontás elvégezhető. Amennyiben ilyen komplex elemű L létezik, akkor az (5-7) felbontás is elvégezhető valós $\tilde{L}, D$ mátrixokkal, azonban D főátlójában negatív elemek is lesznek.

Az (5-2) felbontás részletes felírásából következik, hogy az L mátrix elemei a

```

for i:= 1 to n do
  begin
    L[i,i] := sqrt(A[i,i] - sum_{j=1}^{i-1} L[i,j]^2);
    for k:=i+1 to n do
      L[k,i] := (A[k,i] - sum_{j=1}^{i-1} L[i,j]*L[k,j])/L[i,i];
    end;
  end;

```

képletek segítségével számíthatók.

Pozitív definit mátrix szimmetrikus felbontása esetén (amikor nincs szükség sorcserékre, amelyek a szimmetria megbontását okozhatják), elegendő a szimmetria miatt kétszer szereplő A elemeit csak egyszer tárolni. (Például az A mátrix főátló és főátló alatti a_{ij} elemei v_k , $k = (i-1)i/2 + j$ elemű vektorként tárolhatók.) Az L mátrix elemei olyan sorrendben számíthatók, hogy azok A megfelelő elemeinek helyére írhatók, amit programozási zsargonnal „in situ” számításnak szokás nevezni. A fentiek alapján hangsúlyozzuk, hogy a tárolóhely megtakarításához nem elegendő, ha az A mátrix szimmetrikus, hanem pozitív definitnek is kell lennie.

A következő két Pascal eljárás az $Ax = b$ lineáris algebrai egyenletrendszer megoldására alkalmas, ahol A pozitív definit, szimmetrikus, valós n -ed rendű mátrix és b valós n elemű vektor.

Az első, az LLTfelbont rutin az (5-2) alatti $A = LL^T$ felbontás implementációja.

```
procedure LLTfelbont(    var A: tVnxn;
                        var D: tVn;
                        n: integer;
                        var rendben: boolean);
```

{Ez az eljárás az n -ed rendű valós szimmetrikus, pozitív definit A mátrix L LT Cholesky felbontását végzi el. Az A elemeit az esetleges utóiteráció miatt megőrzi. A és n bemenő adatok. A ugyanakkor kimenő tömb is, a főátlóelemek és az e fölötti elemek A eredeti elemeit tartalmazzák, a főátló alatti elemek a felbontás során keletkezett L mátrix főátló alatti elemeit, L főátló elemeit pedig a D vektor tartalmazza. A rendben logikai változó mutatja, hogy a felbontás megtörtént-e. Ennek az eljárásnak az LLTmegold eljárással való kombinálása alkalmas lineáris egyenletrendszerek megoldására.

Az LLTfelbont eljárást használó programban definiálni kell a következő globális típusokat

```
tVn      = array [1..nmax] of real;
```

```
tVnxn    = array [1..nmax,1..nmax] of real;
```

ahol az $nmax$ konstans az n legnagyobb lehetséges értéke.

Ha az adott Pascal implementáció nem rendelkezik a double típussal, akkor definiálni kell a

```
double = real;
```

```
utasítással. }
```

```
var
```

```
  i, j, k: integer;                                {ciklusváltozók}
```

```
  s: double;                                       {összegzés kettős pontosságú változóban}
```

```
  ss: real;                                       {A j-edik főátló elem reciproka}
```

```
begin rendben:=true; j:=0;                        {rendben inicializálása}
```

```
  while rendben and (j<n) do                       {külső ciklus az oszlopokra}
```

```
    begin inc(j); s:=A[j,j];                       {l[j,j] számítása}
```

```
      for k:=1 to j-1 do
```

```
        s:=s-sqr(A[j,k]);
```

```
      if s<=0 then
```

```
        rendben:=false                             {A nem pozitív definit!}
```

```
      else
```

```
        begin D[j]:=sqrt(s); ss:=1/D[j];
```

```
          for i:=j+1 to n do {ciklus l[i,j] kiszámítására}
```

```
            begin s:=A[i,j];
```

```
              for k:=1 to j-1 do
```

```
                s:=s-A[i,k]*A[j,k];
```

```
                A[i,j]:=s*ss;
```

```
            end;
```

```
          end;
```

```
        end;
```

```
end;
```


A második, az LLTmegold eljárás pedig az

$$Ly = b \text{ és az } L^T x = y$$

háromszögmátrixszal rendelkező egyenletrendszerek egymás utáni megoldásával szolgáltatja a keresett x megoldást.

```

procedure LLTmegold(      LA: tvnxn;
                          D: tvn;
                          n: integer;
                          var b: tvn);

{ Alkalmas az  $L L^T x = b$  alakú,  $n$  lineáris egyenletből
  álló egyenletrendszer megoldására, ahol az  $L$  mátrix főátló
  alatti részét az  $LA$  bemenő tömb főátló alatti elemei
  alkotják, a főátlóelemeket a  $D$  bemenő tömb tartalmazza. Az
   $LA$  és  $D$  paraméterek az LLTfelbont eljárás eredményei. Az  $n$ 
  bemenő paraméter az egyenletek száma. A  $b$  vektor bemenetként
  az egyenletrendszer jobb oldalát, kimenetként pedig az  $x$ 
  megoldást tartalmazza. A típusdefiníciókat l. az
  LLTfelbont eljárásban. }

var
  i, j, k: integer;      {ciklusváltozók}
  s: double;             {összegzés kettős pontosságú változóban}
begin
  for i:=1 to n do      {első egyenlettől (L alsó háromszög)}
    begin s:=b[i];      {L y = b egyenletrendszer megoldása}
      for k:=1 to i-1 do
        s:=s-LA[i,k]*b[k];
        b[i]:=s/D[i];      {y[i]}
      end;
    for i:=n downto 1 do {utolsó egyenlettől}
      begin s:=b[i];    {LT x = y egyenletrendszer megoldása}
        for k:=n downto i+1 do
          s:=s-LA[k,i]*b[k];
          b[i]:=s/D[i];      {x[i]}
        end;
    end;
end;

```

Itt is célszerű a 4. pontban ismertetett utóiteráció alkalmazásával meggyőződni a megoldás pontosságáról. A következő LLTjavít eljárás az LUjavít eljárás megfelelője.


```

procedure LLTjavit(      LA: tvnxn;
                        D: tvn;
                        n: integer;
                        b: tvn;
                        var x: tvn);

{ Ez az eljárás alkalmas az  $Ax = b$  lineáris
egyenletrendszer  $x$  közelítő megoldásának iterációs
finomítására. Az LA tömb, a D és b vektor, valamint az n
rendszer bemenő adatok. Az LA és D tömb az A mátrix L LT
felbontásából származó L mátrixot, valamint az eredeti
A mátrixot tartalmazza az LLTfelbont eljárásban leírtaknak
megfelelően. Az LLTjavit eljárás eredménye az x vektor,
mely a közelítő megoldás finomított értékeit tartalmazza.
A típusdefiníciókat l. az LLTfelbont eljárásban. }

var
  i, j : integer;                                {ciklusváltozók}
  s      : double;                                {duplapontos segédváltozó}
  r      : tvn;                                   {a különbségvektort tartalmazó tömb}

begin
  for i:=1 to n do                                {a különbségvektor számítása}
    begin s:=-b[i];                                {összegzés kettős}
      for j:=1 to n do                                {pontosságú változóban}
        if i>j then s:=s+LA[j,i]*x[j]
        else s:=s+LA[i,j]*x[j];
      r[i]:=s;                                       {r = Ax - b}
    end;
  LLTmegold(LA,D,n,r);    {(4-4) egyenletrendszer megoldása}
  for i:=1 to n do                                {a jobb közelítés számítása}
    x[i]:=x[i]-r[i];
  end;

```

6. Szimmetrikus pozitív definit szalagmátrix

Ha a szimmetrikus, valós A mátrix elemeire érvényes, hogy

$$a_{ij} = 0 \quad \text{ha} \quad |i-j| \geq m, \quad (6-1)$$

akkor az A mátrixot *szalagmátrixnak*, az m pozitív egész számot pedig A *félzalagszélességének* nevezzük ($2m-1$ az A szalagszélessége).

6.1. tétel. Ha a (6-1) feltételt kielégítő A szalagmátrix *pozitív definit*, akkor az $A = LL^T$ Cholesky-felbontásában szereplő L (nem szinguláris, valós) alsó háromszögmátrix elemeire érvényes az

$$l_{ij} = 0, \quad \text{ha} \quad i-j \geq m$$

összefüggés.

A bizonyítástól eltekintünk, mert az nyilvánvaló az $A = LL^T$ egyenlet részletes felírásából.

Pozitív definit szalagmátrix esetén – a 6.1. tétel állítása miatt – elegendő, ha a szimmetrikus n -ed rendű A mátrix egésze helyett csak pl. az alsó háromszög-részének szalagon belüli elemeit tároljuk egy $n \times m$ típusú S mátrixban. A Cholesky-felbontás eredményeként nyert L mátrix elemei az A , következésképpen az S mátrix megfelelő elemeinek a helyén képezhetők.

A tárolás módja nyilvánvaló a két, A és S mátrix részletes felírásából:

$$A = \begin{bmatrix} a_{11} & & & & & \\ a_{21} & a_{22} & & & & \\ a_{31} & a_{32} & a_{33} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mm} & \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n,n-m+1} & \dots & \dots & \dots & a_{nn} \end{bmatrix}$$

szimmetrikus
rész

és

$$S = \begin{bmatrix} & & & & a_{11} \\ & & & a_{21} & a_{22} \\ & & a_{31} & a_{32} & a_{33} \\ & \vdots & \vdots & \vdots & \vdots \\ & a_{m1} & \dots & \dots & a_{mm} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n,n-m+1} & \dots & \dots & \dots & a_{nn} \end{bmatrix}$$

redundáns
elemek

Körvonal (sky-line) technika

A kontinuumfeladatok finitizálására széles körben elterjedt *végeselem-módszer* gyakran vezet olyan lineáris algebrai egyenletrendszerek megoldására, ahol az A együtthatómátrix

a) pozitív definit szalagmátrix és

b) vannak olyan sorai, amelyekre

$$a_{ij} = 0, \quad \text{ha} \quad j = 1, 2, \dots, p_i$$

érvényes, azaz A bizonyos soraiban, az első oszloptól számítva, folyamatosan sok nulla elem „nyúlik bele” a szalagrészbe.

Ekkor a szimmetrikus felbontás során nyert L alsó háromszögmátrix megfelelő elemei szintén zérussal egyenlők. Ez lehetővé teszi az ún. „körvonal” (sky-line) technika alkalmazását, amellyel az operatív tár lehetőségei még jobban kiaknázhatók, mint amikor csak a szalagmátrix-struktúrából adódó előnyöket használjuk ki. Ennek a kompakt tárolásnak az a lényege, hogy az n -ed rendű m -félszalagszélességű (példánkban $n=8$, $m=4$)

$$A = \begin{bmatrix} a_{11} & & & & & & & \\ a_{21} & a_{22} & & & & & & \\ & a_{32} & a_{33} & & & & & \\ a_{41} & 0 & a_{43} & a_{44} & & & & \\ & & & a_{54} & a_{55} & & & \\ & & & a_{63} & a_{64} & a_{65} & a_{66} & \\ & & & & & a_{76} & a_{77} & \\ & & & & & a_{85} & 0 & a_{87} & a_{88} \end{bmatrix}$$

szalagmátrix elemeit egy egydimenziós v tömbben tároljuk. Az A mátrix és a v vektor elemei közötti megfeleltetés szabálya nyilvánvaló a következő részletes felírásból:

$$\begin{aligned} v(1) &= a_{11}, & v(2) &= a_{22}, & v(4) &= a_{33}, & v(6) &= a_{44}, & v(10) &= a_{55}, \\ & & v(3) &= a_{21}, & v(5) &= a_{32}, & v(7) &= a_{43}, & v(11) &= a_{54}, \\ & & & & & & v(8) &= 0, \\ & & & & & & v(9) &= a_{41} \end{aligned}$$

és így tovább...

Megjegyezzük, hogy a szalagon belüli olyan nulla elemeket, amelyeket a saját sorukban zérustól különböző elem előz meg, tárolni kell, mert az „in situ” végrehajtott szimmetrikus felbontás során ezek a nulla elemek megváltozhatnak. (A fenti példában is ilyen az a_{42} meg az a_{86} elem.)

A körvonalatároláshoz, a v vektoron kívül, meg kell adni a v vektornak az A mátrix főátlóelemeit tartalmazó indexeit is (beleértve a „képzeletbeli” $n+1$ -edik főátlóelem indexét is). Példánk esetén ezek az 1, 2, 4, 6, 10, ... indexek.

A következő két eljárás alkalmas az $Ax = b$ lineáris algebrai egyenletrendszer megoldására, ahol A valós szimmetrikus szalagmátrix.

```
procedure LDLTfelbont( var v ;
                      iatlo: tIn1;
                      n: integer;
                      var NumPoz: integer);
```

{ Ez az eljárás az n -ed rendű valós szimmetrikus A szalagmátrix $L D L^T$ felbontására alkalmas. Ha A pozitív definit, akkor a felbontás létezik. Az A mátrixot a körvonaltechnikának megfelelően v vektorként tárolja. A típus nélküli v paraméter bármely méretű vektor átadását lehetővé teszi, mely valós elemű és 1 a kezdőindexe (l. a tv típust a továbbiakban). Az iatlo elemei indexek. A v vektor azon elemeinek indexei, amelyek az A mátrix főátlóelemeit tartalmazzák. Az iatlo $n+1$ -edik eleme a képzeletbeli $n+1$ -edik főátlóelem helyét jelöli (l. még a fenti példát). A v vektornak iatlo[n+1]-1 eleme van. A v tömb kimenetkor a bemenő sorrendnek megfelelően az L alsó háromszögmátrix főátló alatti (a főátlóelemek l-esek, ezért nem tároljuk) és a D főátló elemeit tartalmazza.

A NumPoz kimenő paraméter értéke 0, ha az A mátrix szinguláris; pozitív, ha A pozitív definit; negatív érték esetén az abszolút értéke a D negatív főátlóelemeinek a számát adja meg. (Az LDLTfelbont eljárás nem csak lineáris egyenletrendszer megoldására, hanem az Altérítérációnál részletezett ellenőrzéshez is előnyösen alkalmazható.)

Az LDLTfelbont eljárást használó programban definiálni kell a következő globális típusokat

```
tv = array [1..10000] of real;
tvn = array [1..nmax] of real;
tIn1 = array [1..nmax+1] of integer;
```

ahol az nmax konstans az n legnagyobb lehetséges értéke. A tv típust a v vektor elemeinek hozzáféréséhez használjuk.}

```
var
  i, j, k, ij, ik, jk: integer; {ciklus- és indexváltozók}
  s: double; {összegzés kettős pontosságú változóban}
```

```
begin i:=0; NumPoz:=n+1; {feltesszük, hogy pozitív definit}
  while (NumPoz<>0) and (i<n) do
    begin inc(i);
      for ij:=iatlo[i+1]-1 downto iatlo[i]+1 do
        begin s:=tv(v)[ij]; ik:=ij+1; {l[i,j] kiszámítása}
          j:=iatlo[i]+i-ij; jk:=iatlo[j]+1; k:=j-1;
          while (ik<iatlo[i+1]) and (jk<iatlo[j+1]) do
            begin s:=s-tv(v)[ik]*tv(v)[jk]*tv(v)[iatlo[k]];
              inc(ik); inc(jk); dec(k);
```



```

        end;
        tv(v)[ij]:=s/tv(v)[iatlo[j]];
    end;
    s:=tv(v)[iatlo[i]]; k:=i-1;           {d[i] kiszámítása}
    for ik:=iatlo[i]+1 to iatlo[i+1]-1 do
        begin s:=s-sqr(tv(v)[ik])*tv(v)[iatlo[k]];
            dec(k);
        end;
    tv(v)[iatlo[i]]:=s;
    if s=0 then
        NumPoz:=0                       {A szinguláris!}
    else if s<0 then
        dec(NumPoz);                     {A nem pozitív definit!}
    end;
    if (NumPoz>0) and (NumPoz<=n) then
        NumPoz:=NumPoz-n-1;
end;

```

```

procedure LDLTmegold(    var v ;
                        iatlo: TIn1;
                        n: integer;
                        var b: tvN);

```

{ Alkalmas az $L D L^T x = b$ alakú lineáris egyenletrendszer megoldására, ahol az L és D mátrixokat együtt - az LDLTfelbont eljárásban leírtak szerint - a v tömbben és az $iatlo$ segítségével tároljuk. A v tömb az LDLTfelbont eljárás eredménye. Az n az egyenletek száma. A b vektor bemenetkor az egyenletrendszer jobb oldalát, kimenetkor pedig az x megoldást tartalmazza.

A típusdefiniálásokkal kapcsolatosan 1. szintén az LDLTfelbont eljárást. }

```

var
    i, k, ik, ki: integer;               {ciklusváltozók}
    s: double;                           {összegzés kettős pontosságú változóban}
begin
    for i:=1 to n do                       {L y = b megoldása}
        begin s:=b[i]; k:=i-1;
            for ik:=iatlo[i]+1 to iatlo[i+1]-1 do
                begin s:=s-tv(v)[ik]*b[k]; dec(k); end;
            b[i]:=s;                       {y[i]=b[i]-sum(k=1,i-1,l[ik]*y[k])}
        end;
    for i:=n downto 1 do                   {D LT x = y megoldása}
        begin s:=b[i]/tv(v)[iatlo[i]];
            for k:=i+1 to n do
                begin ki:=iatlo[k]+k-i;
                    if ki<iatlo[k+1] then
                        s:=s-tv(v)[ki]*b[k];
                    end;
            b[i]:=s;                       {x[i]=y[i]/d[i]-sum(k=i+1,n,l[ki]*x[k])}
        end;
    end;
end;

```


Frontális technika

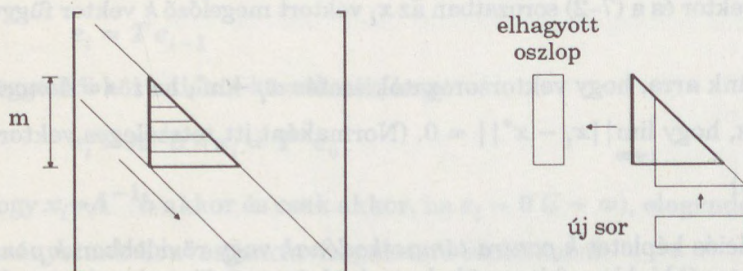
Egy másik módszer, amely kihagyja a pozitív definit szalagmátrixban a szalagon belüli zérus elemekkel feleslegesen végrehajtott műveleteket, az ún. *frontális módszer*, amely a végeselem-módszerből származtatott lineáris algebrai egyenletrendszerek megoldására szolgál. A Gauss elimináció alkalmazása során a k -adik sor kiküszöbölése az

$$\bar{a}_{ij} = a_{ij} - a_{ik} a_{kj} / a_{kk}$$

formula szerint módosítja a még fennmaradt sorok elemeit. A számítást csak a zérustól különböző a_{ik} és a_{kj} elemekkel végezzük el. Az összes ilyen elem „sűrű” háromszögbe tömöríthető, amely sok esetben lényegesen kisebb tárolási helyet igényel, mint az ezután ismertetendő technikában a „lefelé csúsztatott” háromszög.

Csúsztatott háromszögtechnika

A pozitív definit szalagmátrix $A = LL^T$ alakú Cholesky felbontását elvégezhetjük úgy is, hogy az operatív tárban egyszerre csak a következő sémán vázolt, $m(m+1)/2$ elemből álló „lefelé csúsztatott” háromszögrészt tároljuk, ahol m a mátrix félszalagszélessége.



A mátrix többi elemét, amikor szükséges, ciklikusan beolvassuk a lemezről. A felbontás során a háromszöget fokozatosan úgy „csúsztatjuk” lefelé, hogy miután a felbontással kapott egy oszlop elhagyta a „munka háromszöget”, a felszabadult helyre az A mátrix további sora kerül.

A stratégiát jól érzékelteti a következő példa ($n=3, m=2$)

$$\begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ & a_{32} & a_{33} \end{bmatrix} \rightarrow \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ & a_{32} & a_{33} \end{bmatrix} \rightarrow \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ & a_{32} & a_{33} \end{bmatrix} \rightarrow \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ & l_{32} & l_{33} \end{bmatrix}$$

7. Iterációs módszerek

Tekintsük az

$$A x = b \quad (7-1)$$

lineáris algebrai egyenletrendszert, ahol az n -ed rendű valós A mátrixról feltesszük, hogy *nem szinguláris*, b pedig n elemű valós vektor.

Ha a (7-1) egyenletrendszert iterációs eljárás alkalmazásával kívánjuk megoldani, akkor egy olyan

$$x_0, x_1, x_2, \dots, x_i, \dots \quad (7-2)$$

vektorsorozat értelmezésére törekszünk, amelyre:

$$(1) \quad x_i \rightarrow A^{-1}b, \quad \text{ha } i \rightarrow \infty \text{ és}$$

$$(2) \quad x_i = f_i(A, b, x_{i-1}, \dots, x_{i-k}), \quad i = k, k+1, \dots \quad (7-3)$$

Vagyis, a vektorsorozat tart a (7-1) egyenlet egyetlen megoldásához, és x_i az A mátrix, a b vektor és a (7-2) sorozatban az x_i vektort megelőző k vektor függvénye.

Emlékeztetünk arra, hogy vektorsorozatok esetén $x_i \rightarrow x^*$, ha $i \rightarrow \infty$ *konvergencián* azt értjük, hogy $\lim_{i \rightarrow \infty} \|x_i - x^*\| = 0$. (Normaként itt tetszőleges vektornorma használható.)

A (7-3) iterációs képletet *k pontra támaszkodónak* vagy rövidebben *k pontosnak* nevezzük. A továbbiakban feltesszük, hogy $k=1$ és az f_i függvények x_{i-1} *lineáris* függvényei. Ez esetben az iterációs képlet

$$x_i = T_i x_{i-1} + c_i$$

alakban írható, ahol $T_i = T_i(A, b)$ mátrixok és $c_i = c_i(A, b)$ vektorok. Az iterációs módszert *stacionáriusnak* nevezzük, ha $T_i \equiv T$ és $c_i \equiv c$, vagyis ha

$$x_{i+1} = T x_i + c. \quad (7-4)$$

A T mátrixot az *iteráció mátrixának* nevezzük.

Például, ha a $b - A x = 0$ egyenlet mindkét oldalához hozzáadjuk az x vektort, akkor ez az

$$x_{i+1} = (I - A) x_i + b$$

iterációs formulát sugalmazza, ahol I az n -ed rendű egységmátrix jele.

Ha a (7-4) iterációs képlettel előállított vektorsorozat konvergens és $A^{-1}b$ a határértéke, akkor a (7-4)-be való behelyettesítéssel

$$A^{-1}b = TA^{-1}b + c$$

következik, vagyis ekkor teljesül a

$$c = (I - T)A^{-1}b \quad (7-5)$$

ún. *konzisztenciafeltétel*.

A (7-4) iterációról a továbbiakban mindig feltételezzük, hogy kielégíti a (7-5) konzisztenciafeltételt, melynek értelmében $A^{-1}b$ megoldása az $x = Tx + c$ egyenletnek.

Határozzuk meg a megoldás i -edik közelítéséhez tartozó

$$e_i = A^{-1}b - x_i$$

hibavektort. A (7-4), majd a (7-5) összefüggés felhasználásával adódik:

$$e_i = A^{-1}b - Tx_{i-1} - c = A^{-1}b - Tx_{i-1} - (I - T)A^{-1}b = T(A^{-1}b - x_{i-1}) = Te_{i-1}.$$

Az

$$e_i = Te_{i-1} \quad (7-6)$$

összefüggésből közvetlenül következik, hogy

$$e_i = A^{-1}b - x_i = T^i e_0. \quad (7-7)$$

Mínthogy $x_i \rightarrow A^{-1}b$ akkor és csak akkor, ha $e_i \rightarrow 0$ ($i \rightarrow \infty$), elegendő a T mátrix T^i hatványainak konvergenciavizsgálatára szorítkozni.

7.1. tétel. A konvergencia szükséges és elégséges feltétele

Tetszőleges x_0 kezdővektorból kiindulva, az $x_{i+1} = Tx_i + c$ iteráció akkor és csak akkor konvergens és $A^{-1}b$ a határértéke (vagyis $i \rightarrow \infty$ esetén $T^i e_0 \rightarrow 0$), ha

$$\rho(T) = \max_k |\lambda_k(T)| < 1, \quad (7-8)$$

ahol $\lambda_k(T)$ a T mátrix sajátértékei, a $\rho(T)$ számot pedig a T mátrix *spektrálsugarának* nevezzük.

Bizonyítás. A bizonyításnak csak az alapötletét ismertetjük. Ha $\lambda_1, \dots, \lambda_n$ jelöli a T mátrix sajátértékeit, akkor T kifejezhető

$$T = SJS^{-1}$$

alakban, ahol S egy nonszinguláris mátrix és J az ún. *Jordan-mátrix*, vagyis olyan blokkszerű diagonálmátrix, amelyben a diagonális blokkok

$$J_k = \begin{bmatrix} \lambda_k & 1 & & 0 \\ & \lambda_k & & \\ & & \ddots & \\ & & & \ddots & \\ 0 & & & & 1 \\ & & & & \lambda_k \end{bmatrix}$$

alakúak. Ekkor

$$T^i = S J^i S^{-1}.$$

Ha a (7-8) egyenlőtlenség teljesül, akkor $i \rightarrow \infty$ esetén J^i tart a zérus mátrixhoz és így $T^i e_0 \rightarrow 0$ bármely e_0 vektor esetén. Megfordítva: az $i \rightarrow \infty$ esetén $T^i \rightarrow 0$ feltételből $J^i \rightarrow 0$ következik, és ez csak akkor lehetséges, ha valamennyi λ_k sajátértékre fennáll a $|\lambda_k| < 1$ összefüggés.

A (7-6) összefüggés szerint e_i az e_{i-1} lineáris függvénye. Ezért a megfelelő iterációs eljárás konvergenciáját sokszor *lineárisnak* nevezik.

7.2. tétel. A konvergencia elégséges feltétele

Ha teljesül a

$$||T|| < 1$$

(7-9)

feltétel, akkor az $x_{i+1} = T x_i + c$ iteráció tetszőleges x_0 kezdővektorból kiindulva konvergens és határértéke az $A^{-1}b$ megoldás.

Bizonyítás. Ha a (7-7) összefüggésre alkalmazzuk a Schwarz-egyenlőtlenséget, akkor

$$||e_i|| = ||A^{-1}b - x_i|| = ||T^i e_0|| \leq ||T||^i ||e_0||.$$

Tekintettel a $||T|| < 1$ feltételre, az egyenlőtlenségből $i \rightarrow \infty$ esetén $||e_i|| \rightarrow 0$ és ezért (a konvergencia definíciója miatt) $e_i = T^i e_0 \rightarrow 0$ bármely e_0 vektorra következik.

Az iteráció befejezésére vonatkozó feltétel

Az iterációt általában akkor szokás befejezni, amikor két egymást követő iterált vektorra már teljesül a

$$||x_i - x_{i-1}|| \leq \delta$$

feltétel, ahol δ egy, a felhasználó által megadott (elegendően kis) pozitív szám.

Megmutatjuk, hogy ekkor hogyan becsülhető az $A^{-1}b - x_i$ hiba, azaz határozható meg a

$$||A^{-1}b - x_i|| \leq \varepsilon$$

egyenlőtlenségben szereplő ε mint δ függvénye. A (7-6) alapján az i -edik közelítés hibája

$$A^{-1}b - x_i = T(A^{-1}b - x_{i-1}) = T(A^{-1}b - x_i) + T(x_i - x_{i-1}).$$

Ebből

$$||A^{-1}b - x_i|| \leq ||T|| ||A^{-1}b - x_i|| + ||T|| ||x_i - x_{i-1}||,$$

vagyis

$$(1 - ||T||) ||A^{-1}b - x_i|| \leq ||T|| ||x_i - x_{i-1}||.$$

Ha teljesül a (7-9) feltétel, akkor $1 - ||T|| > 0$ és így

$$||A^{-1}b - x_i|| \leq \frac{||T||}{1 - ||T||} ||x_i - x_{i-1}||$$

adódik.

Ha $||x_i - x_{i-1}|| \leq \delta$, akkor

$$||A^{-1}b - x_i|| \leq \frac{||T||}{1 - ||T||} \delta = \varepsilon$$

a kívánt hibabecslésre alkalmas formula.

A konvergencia gyorsasága

A (7–8), illetve a (7–9) feltétel biztosítja ugyan az iterációs eljárás konvergenciáját, azonban nem ad információt arra vonatkozóan, hogy hány iterációs lépést kell végrehajtani ahhoz, hogy a megoldás kívánt pontosságú közelítését megkapjuk.

A (7–6) és (7–7) összefüggésekből

$$||e_i|| \leq ||T|| ||e_{i-1}|| \leq ||T||^i ||e_0||$$

következik. Ebből nyilvánvaló, ha $||T||$ közel van az egyhez, akkor kis $||e_i||$ érték eléréséhez sok iterációs lépésre van szükség.

Például, ha $||e_{i-1}|| = 10^{-2}$ és $||e_i|| = 10^{-4}$, azaz ha egyetlen iterációs lépéssel két értékes tizedesjegyet nyertünk, akkor (nagyobb i esetén) célszerű a (lineáris) konvergencia gyorsaságát kifejezni az

$$r = \log \left(\frac{||e_{i-1}||}{||e_i||} \right)$$

számmal, amely az iterációs lépésenként kapott értékes tizedesjegyek számának kifejezője.

8. Jacobi-iteráció és a Gauss–Seidel-módszer

Az $Ax = b$ egyenlet együtthatómátrixát írjuk

$$A = L + D + U \quad (8-1)$$

alakban, ahol L tartalmazza azokat az a_{kj} elemeket, amelyekre $k > j$, D azokat, amelyekre $k = j$ és U azokat, amelyekre $k < j$. Az L , D és U többi eleme zérus. Tegyük fel, hogy $a_{kk} \neq 0$, $k = 1, \dots, n$. Az $Ax = b$ egyenletből (8-1) felhasználásával kapjuk

$$Dx = -Lx - Ux + b \quad (8-2)$$

ahonnan

$$x = -D^{-1}(L + U)x + D^{-1}b \quad (8-3)$$

következik. Ez az egyenlőség az

$$x_{i+1} = -D^{-1}(L + U)x_i + D^{-1}b \quad (8-4)$$

formulát sugallja, amelyet Jacobi-iterációnak vagy az egyidejű változtatás módszerének is nevezünk. A (8-4) iterációs formulából kiolvasható, hogy $T = -D^{-1}(L + U)$ és így az iteráció nyilván konvergens, ha

$$||D^{-1}(L + U)||_E < 1. \quad (8-5)$$

Bebizonyítható, hogy ha az A mátrix főátlójának elemei szigorúan dominálnak, azaz ha teljesülnek az

$$|a_{kk}| > \sum_{\substack{j=1 \\ j \neq k}}^n |a_{kj}|, \quad k = 1, \dots, n$$

egyenlőtlenségek, akkor a Jacobi-módszer és az alábbiakban ismertetendő Gauss-Seidel-eljárás, tetszőleges x_0 kezdővektor esetén konvergens.

A Gauss-Seidel-eljárás, vagy más néven a *folyamatos változtatások módszere* ismét a (8-3) összefüggés felhasználásával fogalmazható meg:

$$x_{i+1} = -D^{-1}(Lx_{i+1} + Ux_i) + D^{-1}b. \quad (8-6)$$

A (8-6) és a (8-4) iterációs formulák összehasonlításából kitűnik, hogy a Gauss-Seidel-eljárás annyiban tér el a Jacobi-iterációtól, hogy mihamarabb kiszámítottuk az x_{i+1} vektor egy újabb elemét, ezt azonnal fel is használjuk az iterációban.

Ha a (8-6) egyenletből kifejezzük az x_{i+1} vektort, akkor

$$x_{i+1} = -(D + L)^{-1}Ux_i + (D + L)^{-1}b \quad (8-7)$$

adódik. Ez a formula ugyan nem előnyös számítások végzésére, de közvetlenül szolgáltatja az iteráció T mátrixát. Következésképpen a Gauss-Seidel-eljárás konvergens, ha

$$\|(D+L)^{-1}U\|_E < 1.$$

A gyakorlatban jobban használható a következő:

8.1. tétel. Ha az A együtthatómátrix szimmetrikus és pozitív definit, akkor a (8-6) Gauss-Seidel-iteráció a kezdeti vektortól függetlenül konvergens és $A^{-1}b$ a határértéke.

Minthogy a 8.1. tétel a továbbiakban szereplő és bebizonyított 9.1. tétel speciális esete (amikor $\omega = 1$), a 8.1. tétel külön bizonyításától eltekintünk.

9. A szukcesszív felülrelaxálás módszere (SOR¹)

Vannak mátrixok, ilyen pl. a valós, szimmetrikus pozitív definit mátrix esete, amikor a Gauss–Seidel-eljárás (lineáris) *konvergenciája meggyorsítható*. A Gauss–Seidel-eljárás (8–6) képletét írjuk át

$$x_{i+1} = x_i - D^{-1} [L x_{i+1} + (D + U) x_i - b] \quad (9-1)$$

alakra.

Az SOR eljárásnál a (9-1) jobb oldalának második tagját megszorozzuk egy ω , ún. *relaxációs tényezővel*. Így kapjuk, hogy

$$x_{i+1} = x_i - \omega D^{-1} [L x_{i+1} + (D + U) x_i - b], \quad (9-2)$$

ami tulajdonképpen a (9-1) összefüggésben az $x_{i+1} - x_i$ különbség súlyozása. Ha a (9-2) egyenletből kifejezzük az x_{i+1} vektort, akkor

$$x_{i+1} = -(D + \omega L)^{-1} [\omega U + (\omega - 1) D] x_i + \omega (D + \omega L)^{-1} b$$

adódik, ahonnan a relaxáció mátrixa:

$$T_\omega = -(D + \omega L)^{-1} [\omega U + (\omega - 1) D]. \quad (9-3)$$

9.1. tétel. Tegyük fel, hogy A valós, szimmetrikus pozitív definit mátrix. Ha teljesül a

$$0 < \omega < 2 \quad (9-4)$$

feltétel, akkor a (9-2) iteráció tetszőleges x_0 kezdővektor esetén konvergens és $A^{-1}b$ a határértéke.

Bizonyítás. Az $A = L + D + U$ csoportosításban a tétel feltételei miatt $U = L^T$ és $y^T A y > 0$ minden $y \neq 0$ vektorra, továbbá

$$y^T (L - U) y = y^T L y - y^T L^T y = y^T L y - (y^T L y)^T = 0, \quad (9-5)$$

hiszen a skalár transzponáltja önmaga.

Jelölje λ a (9-3) relaxációmátrix tetszőleges sajátértékét és $y \neq 0$ egy hozzá tartozó sajátvektort. A $T_\omega y = \lambda y$, azaz a

$$-(D + \omega L)^{-1} [\omega U + (\omega - 1) D] y = \lambda y$$

egyenlet átírható

$$[(1 - \omega) D - \omega U] y = \lambda (D + \omega L) y \quad (9-6)$$

alakra.

¹ Az angol *Successive Overrelaxation* rövidítése

A 7.1. tétel értelmében elegendő bebizonyítani, hogy ha $0 < \omega < 2$ akkor $|\lambda| < 1$. Felhasználva az $A = L + D + U$ összefüggést, (9-6) átírható

$$\left[(2 - \omega)D - \omega A + \omega (L - U) \right] y = \lambda \left[(2 - \omega)D + \omega (A - U + L) \right] y \quad (9-7)$$

alakra. (A (9-7) a legegyszerűbben úgy ellenőrizhető, hogy az egyenlet mindkét oldalán elvégezzük a számítási műveleteket, s ekkor megkapjuk a (9-6) kétszerezését.) A (9-7) egyenlet mindkét oldalát balról szorozzuk meg az y^T vektorral. Ha alkalmazzuk az

$$y^T A y = a, \quad y^T D y = d$$

jelöléseket és (9-5) alapján figyelembe vesszük, hogy $y^T (L - U) y = 0$, akkor

$$(2 - \omega) d - \omega a = \lambda \left[(2 - \omega) d + \omega a \right]$$

adódik, ahol $a > 0$ és $(e_i^T A e_i = a_{ii} > 0$ miatt) $d = \sum_{i=1}^n a_{ii} y_i^2 > 0$.

Ha $0 < \omega < 2$, akkor

$$|(2 - \omega) d - \omega a| < |(2 - \omega) d + \omega a|$$

és ezzel bebizonyítottuk, hogy $|\lambda| < 1$.

Pontosabban, *felülrelaxálásról* akkor beszélünk, ha a (9-2) relaxációs eljárásnál az ω tényezőt úgy választjuk meg, hogy $1 < \omega < 2$ legyen. Ha $0 < \omega < 1$, akkor *alülrelaxálásról* beszélünk. Ha $\omega = 1$, akkor a (9-2) relaxációs eljárás speciális eseteként a (8-6) Gauss-Seidel-eljárást nyerjük.

Az ω relaxációtényező leggyorsabb konvergenciát biztosító ω_{opt} értéke a következő megfontolással határozható meg.

Amint már említettük, az a feltétel, hogy az $x_{i+1} = T x_i + c$ iteráció T mátrixának $\rho(T)$ spektrálsugara egynél kisebb legyen, biztosítja ugyan a konvergenciát, de ez nagyon lassú is lehet. A konvergencia annál gyorsabb, minél kisebb a spektrálsugár. Minthogy relaxáció esetében az iteráció T_ω mátrixa és így a spektrálsugár is függvénye ω -nak, meg kell határozni azt az $\omega = \omega_{opt}$ értéket, amelyre a T_ω mátrix $\rho(T_\omega)$ spektrálsugara a legkisebb, vagyis meg kell oldani a

$$\rho(T_{\omega_{opt}}) = \min_{0 < \omega < 2} \rho(T_\omega) = \min_{0 < \omega < 2} \max_k |\lambda_k(T_\omega)|$$

ún. minimax feladatot.

Befejezésül röviden megemlítünk még az SOR módszer témaköréhez tartozó néhány érdekes eredményt.

A Gauss-Seidel-módszer és így az SOR is függ az egyenletek sorrendjétől.

A (8-1) alatti $A = L + D + U$ csoportosítású n -ed rendű valós mátrix *konzisztensen rendezett*, ha az

$$\frac{1}{\alpha} D^{-1} L + \alpha D^{-1} U \quad (9-8)$$

mátrix sajátértékei nem függenek α -tól bármely $\alpha \neq 0$ esetén.

9.1. példa. Az

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \quad (9-9)$$

mátrix konzisztensen rendezett. Ugyanis az

$$\frac{1}{\alpha} D^{-1} L + \alpha D^{-1} U = -\frac{1}{2\alpha} \begin{bmatrix} 0 & \alpha^2 & 0 \\ 1 & 0 & \alpha^2 \\ 0 & 1 & 0 \end{bmatrix}$$

mátrix karakterisztikus egyenlete

$$\det \left(\frac{1}{\alpha} D^{-1} L + \alpha D^{-1} U - \lambda I \right) \equiv -\lambda \left(\lambda^2 - \frac{1}{2} \right) = 0,$$

tehát együtthatói nem függenek α -tól.

Az n -ed rendű valós A mátrix (A) tulajdonságú, ha van olyan P permutációmátrix, amellyel

$$P A P^T = \begin{bmatrix} D_1 & F \\ G & D_2 \end{bmatrix}$$

ahol D_1 és D_2 diagonálmátrixok.

9.2. példa. A (9-9) alatti mátrix (A) tulajdonságú, mert

$$P A P^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

9.2. tétel. Tegyük fel, hogy az n -ed rendű valós A mátrix konzisztensen rendezett, (A) tulajdonságú és főátlójának nincs zérus eleme. Ha a (8-4) Jacobi-iteráció $-D^{-1}(L + U)$ mátrixának sajátértékei valósak és μ spektrálsugara kisebb mint egy, akkor a (9-3) relaxációmátrix spektrálsugarára érvényes:

$$\rho(T_\omega) < 1, \quad \text{ha } 0 < \omega < 2,$$

továbbá létezik egyetlen

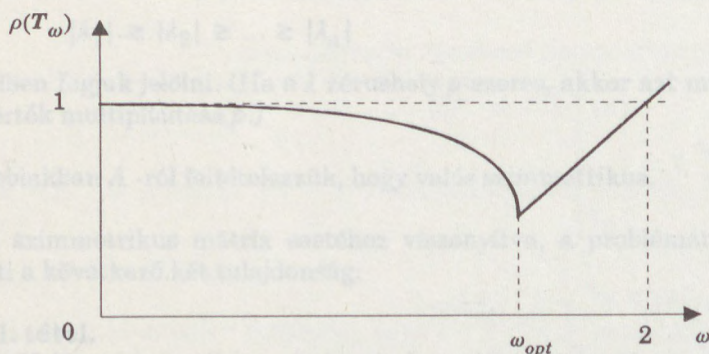
$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \mu^2}}$$

érték, amelyre teljesül hogy

$$\rho(T_{\omega_{opt}}) = \min_{0 < \omega < 2} \rho(T_\omega).$$

A bizonyítás megtalálható pl. Ortega[7] könyvének 7.2. pontjában.

A relaxációmátrix spektrálsugarának változását ω függvényében a 9.1. ábrán vázoltuk. Belátható, hogy ha



9.1. ábra. A spektrálsugár változása ω függvényében

ω balról tart az ω_{opt} értékhez, akkor a derivált tart a végtelenhez, ha ω jobbról tart az ω_{opt} -hoz, akkor a derivált értéke egy. Ezért jobb, ha az ω_{opt} értéket felülről közelítjük, mint alulról.

VALÓS SZIMMETRIKUS MÁTRIX SAJÁTÉRTÉKEI ÉS SAJÁTVEKTORAI

1. A lineáris algebra néhány eredménye

Tekintsük az

$$A v = \lambda v \quad (1-1)$$

egyenlettel értelmezett sajátérték-feladatot, ahol A n -ed rendű mátrix. Az A mátrix az n dimenziós vektortérben egy lineáris transzformáció reprezentációja, feltéve, hogy e térben bázist értelmeltünk. Keressük azokat a $v \neq 0$ vektorokat, amelyek irányát a $v \rightarrow A v$ leképezés nem változtatja meg. Ezek az A sajátvektorai.

Mivel az (1-1) lineáris algebrai egyenletrendszer homogén, $v \neq 0$ nemtriviális megoldása akkor és csak akkor létezik, ha együtthatómátrixa szinguláris, azaz ha

$$\det(A - \lambda I) = 0.$$

Kifejtéssel belátható, hogy ez a determináns a λ változó n -ed fokú polinomja. Ezért létezik n zérushelye, ezek az A sajátértékei, amelyeket általában a

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$$

sorrendben fogjuk jelölni. (Ha a λ zérushely p -szeres, akkor azt mondjuk, hogy a λ sajátérték multiplicitása p .)

A továbbiakban A -ról feltételezzük, hogy valós szimmetrikus.

A nem szimmetrikus mátrix esetéhez viszonyítva, a problémát nagyon megkönnyíti a következő két tulajdonság:

1.1. tétel.

(1) Valós szimmetrikus mátrix minden sajátértéke valós.

(2) Létezik n páronként ortogonális $v_1, \dots, v_n$ sajátvektor, amelyek úgy normálhatók, hogy teljesüljön a

$$(v_i, v_j) \equiv v_i^T v_j = \begin{cases} 1, & \text{ha } i = j \\ 0, & \text{ha } i \neq j \end{cases}$$

összefüggés.

A sajátértékekhez tartozó sajátvektorok lineáris alteret feszítenek ki az n dimenziós vektortérben, vagyis e sajátvektorok bázist alkotnak ebben az ún. sajátaltérben. A sajátaltér dimenziója egyenlő a megfelelő sajátérték multiplicitásával.

Egynél több dimenziós sajátaltér esetében ez a bázis nem egyértelmű. Valóban, ha például a v_1, v_2 vektorok az A mátrix $\lambda = \lambda_1 = \lambda_2$ kétszeres sajátértékéhez tartozó sajátaltér bázisát képezik, akkor

$$A(c_1 v_1 + c_2 v_2) = c_1 A v_1 + c_2 A v_2 = \lambda(c_1 v_1 + c_2 v_2)$$

miatt a $c_1 v_1 + c_2 v_2$ lineáris kombináció ugyancsak A -nak a λ -hoz tartozó sajátvektora.

Ebből az összefüggésből az is kitűnik, hogy ha az A operátort a sajátaltér vektora-ira alkalmazzuk, akkor ez az operáció a skalárral való szorzásra egyszerűsödik.

Az (1-1) egyenlet n megoldása kifejezhető

$$A \begin{bmatrix} v_1, \dots, v_n \end{bmatrix} = \begin{bmatrix} v_1, \dots, v_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$$

alakban, vagyis

$$A V = V \Lambda \quad (1-2)$$

egyetlen egyenletként is, ahol Λ az A mátrix sajátértékeiből alkotott diagonálmátrix, V oszlopai pedig a λ -khoz tartozó sajátvektorok.

A sajátvektorok ortonormált rendszert alkotnak, azaz

$$V^T V = \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix} \begin{bmatrix} v_1, \dots, v_n \end{bmatrix} = I_n$$

és $V^T = V^{-1}$ miatt $V V^T = I_n$, úgyhogy

$$V^T V = V V^T = I_n$$

is írható, ahol I_n az n -ed rendű egységmátrixot jelöli. Tehát a V mátrix ortogonális és ezért az (1-2) egyenletből következik, hogy

$$V^T A V = \Lambda \quad (1-3)$$

és

$$A = V \Lambda V^T. \quad (1-4)$$

Az (1-4) egyenletből kitűnik, hogy a szimmetrikus A mátrixot sajátértékei és ortonormált sajátvektorai egyértelműen meghatározzák.

Az (1-3) egyenlet azt fejezi ki, hogy ortogonális hasonlósági transzformáció alkalmazásával megváltoztattuk a bázist. A Λ diagonálmátrix és az A ugyanannak a

lineáris transzformációnak a reprezentációja, Λ azonban a sajátvektorok által alkotott bázisban van kifejezve.

Valóban, ha az n -dimenziós tér egy és ugyanazon vektorát előbb az egységmátrix oszlopai által alkotott bázisban, majd a sajátvektorok bázisában is kifejezzük, akkor írható, hogy

$$x_1 e_1 + \dots + x_n e_n = x'_1 v_1 + \dots + x'_n v_n$$

vagy tömörebben

$$I x = V x'.$$

Ha ugyanezt az ortogonális bázisváltoztatást az $y = A x$ képvektorral is végrehajtjuk, akkor $y = V y' = A V x'$ adódik. A V mátrix ortogonalitása és az (1-3) miatt $y' = V^T A V x' = \Lambda x'$ következik, s ezzel az (1-3) egyenletre vonatkozó állítást beláttuk.

1.2. tétel. Az A mátrixra alkalmazott bármely $P^{-1} A P$ hasonlósági transzformáció változatlanul hagyja a sajátértékeket.

Bizonyítás. Jelölje λ az A mátrix sajátértékét és v a hozzá tartozó sajátvektort. Vagyis

$$A v = \lambda v$$

és így tetszőleges nem szinguláris P mátrixszal

$$P^{-1} A v = \lambda P^{-1} v. \quad (1-5)$$

Ekkor $u = P^{-1} v$ jelöléssel a $v = P u$ transzformációt nyerjük. Behelyettesítve (1-5)-be a

$$P^{-1} A P u = \lambda u \quad (1-6)$$

adódik, ami éppen annak a kifejezője, hogy λ a $P^{-1} A P$ -nek is sajátértéke és u a hozzá tartozó sajátvektor.

Az alkalmazásokban a sajátértékek és a sajátvektorok iránti igény különböző. Szükségünk lehet csak néhány, vagy az összes sajátértékre, és hasonlóképpen csak néhány vagy az összes sajátvektorra.

Ami a megoldási módszereket illeti, a sajátértékek és sajátvektorok meghatározására alkalmas algoritmusok mind végtelen iterációk. Ez abból következik, hogy egy 4-nél magasabb fokú polinom zérushelyeit általános esetben csak iterációval lehet meghatározni.

A számítási technikák a következő három csoportba sorolhatók:

- Polinommal kapcsolatos iterációk
- Vektorokkal, ill. alterekkel végzett iterációk
- Transzformációs módszerek

2. Polinommal kapcsolatos iterációk

Az n -ed fokú algebrai egyenlet gyökeinek meghatározására szolgáló eljárások mindegyikétől akkor kínálkoznak az A mátrix *sajátértékeinek* kiszámítására, ha a

$$p_A(\lambda) = \det(A - \lambda I) \quad (2-1)$$

karakterisztikus polinom együtthatóit már előállítottuk. A karakterisztikus polinom együtthatóinak kiszámítására több eljárás is ismeretes, ilyen pl. a Krilov-, a Le-Verrier- vagy a Danyiljevskij-módszer (lásd pl. Ralston [12]). Ez a stratégia mégsem javasolható, mert *instabilis* abban az értelemben, hogy a polinom együtthatóinak kicsiny hibái általában a sajátértékek nagy torzulását okozzák.

Ez az instabilitás kiküszöbölhető azáltal, hogy az n -ed fokú algebrai egyenlet megoldására alkalmas eljárásokat közvetlenül (a polinom együtthatóinak kiszámítása nélkül) alkalmazzuk a polinom (2-1) alatti implicit alakjára. Adott λ értékhez a $p_A(\lambda)$ érték pl. úgy számítható ki, hogy miután elvégeztük az $A - \lambda I$ mátrix LU alakú trianguláris felbontását, felírható, hogy $p_A(\lambda) = (-1)^k u_{11} \dots u_{nn}$, ahol $u_{11}, \dots, u_{nn}$ az U mátrix főátlóelemei, k pedig a felbontás során végrehajtott sorcserék száma. (Megjegyezzük, hogy a Cholesky-felbontást és annak LDL^T alakú alternatíváját óvatosan kell kezelni, mert az $A - \lambda I$ mátrix pozitív definitása és így a felbontás sorcserék nélküli stabilitása nem garantálható.)

Givens-féle felezési eljárás

Tipikus példája az ilyen stabilis módszereknek a *Givens* által javasolt *felezési eljárás*, amely szimmetrikus háromátlós mátrix néhány vagy összes sajátértékének meghatározására alkalmas. Mivel egy valós szimmetrikus mátrix véges számú ortogonális hasonlósági transzformáció alkalmazásával stabilisan redukálható háromátlós alakra (lásd a 3. pontban szereplő Householder-módszert), ez a módszer bármely szimmetrikus mátrix sajátértékeinek kiszámítására is alkalmas.

Tekintsük a

$$T = \begin{bmatrix} c_1 & b_1 & & & & \\ b_1 & c_2 & b_2 & & & \\ & b_2 & c_3 & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & & c_{n-1} & b_{n-1} \\ & & & & & b_{n-1} & c_n \end{bmatrix}$$

szimmetrikus háromátlós mátrixot és tegyük fel, hogy $b_i \neq 0$, $i = 1, \dots, n-1$. Ha valamelyik $b_i = 0$, akkor a feladat alacsonyabb rendű háromátlós mátrixok sajátértékeinek meghatározására bontható.

Képezzük most a $T - \lambda I$ karakterisztikus mátrix bal felső sarokminorainak sorozatát. Jelölje $p_i(\lambda)$ az i -ed rendű bal felső sarokminort. Könnyen belátható, hogy a $p_i(\lambda)$ polinomok a következő rekurziós képlet alapján számíthatók:

$$\left. \begin{aligned} p_0(\lambda) &\equiv 1 \\ p_1(\lambda) &= c_1 - \lambda \\ p_i(\lambda) &= (c_i - \lambda)p_{i-1}(\lambda) - b_{i-1}^2 p_{i-2}(\lambda), \quad i = 2, \dots, n \end{aligned} \right\} \quad (2-2)$$

Az n -re adódó

$$p_n(\lambda) \equiv \det(T - \lambda I)$$

polinom a T mátrix karakterisztikus polinomja.

Minthogy a T főminormátrixai mind szimmetrikus mátrixok, a $p_1, \dots, p_n$ polinomok zérushelyei kizárólag valósak.

Sturm-sorozatnak nevezzük a

$$f_n(x), f_{n-1}(x), \dots, f_1(x), f_0(x)$$

polinomsorozatot, ahol $f_k(x)$ k -ad fokú polinom és $f_0(x) \not\equiv 0$. Az f_k bármely x_0 zérushelyén, azaz $f_k(x_0) = 0$ esetén

$$f_{k-1}(x_0)f_{k+1}(x_0) < 0, \quad k = 1, \dots, n-1,$$

ha tehát a sorozat bármely (nem szélső) polinomjának a zérushelyén a szomszédos polinomok ellenkező előjelűek.

Most bebizonyítjuk a $p_i(\lambda)$ polinomokra vonatkozó következő tételt.

2.1. tétel. A (2-2) képletekkel megadott $p_i(\lambda)$, $i = 1, \dots, n-1$ polinomokra fennáll, hogy ha $\lambda = \lambda_0$ esetén

$$p_i(\lambda_0) = 0,$$

akkor

$$p_{i+1}(\lambda_0)p_{i-1}(\lambda_0) < 0, \quad i = 1, \dots, n-1, \quad (2-3)$$

tehát a $p_0(\lambda), p_1(\lambda), \dots, p_n(\lambda)$ polinomok Sturm-sorozatot alkotnak.

Bizonyítás. Teljes indukcióval:

Először is $p_1(\lambda_0) = 0$ miatt

$$p_0(\lambda_0) p_2(\lambda_0) = p_2(\lambda_0) = (c_2 - \lambda_0) p_1(\lambda_0) - b_1^2 p_0(\lambda_0) = -b_1^2 < 0,$$

hiszen feltettük, hogy $b_i \neq 0$, $i = 1, \dots, n-1$.

Tegyük fel, hogy $i = k$ esetén az állítás igaz, vagyis hogy $p_k(\lambda_0) = 0$ -ból $p_{k+1}(\lambda_0) p_{k-1}(\lambda_0) < 0$ következik. Ebből viszont az következik, hogy ha $p_{k+1}(\lambda_0) = 0$, akkor $p_k(\lambda_0) \neq 0$ és így

$$p_{k+2}(\lambda_0) = (c_{k+2} - \lambda_0) p_{k+1}(\lambda_0) - b_{k+1}^2 p_k(\lambda_0) \neq 0$$

miatt

$$p_k(\lambda_0) p_{k+2}(\lambda_0) = -b_{k+1}^2 p_k^2(\lambda_0) < 0,$$

s ezzel a tételt bebizonyítottuk.

A továbbiakban szükség lesz még a

$$p_i(\lambda) > 0, \text{ ha } \lambda \rightarrow -\infty, \quad i = 1, \dots, n \quad (2-4)$$

összefüggésre, amely a (2-2) képletekből szintén teljes indukcióval közvetlenül adódik. (Valóban, ha feltesszük, hogy (2-4) teljesül $i = 1, \dots, n-1$ esetén, akkor az állítás nyilvánvalóan következik a (2-2) képlet $\frac{p_i}{p_{i-2}} = (c_i - \lambda) \frac{p_{i-1}}{p_{i-2}} - b_{i-1}^2$ alakjából.)

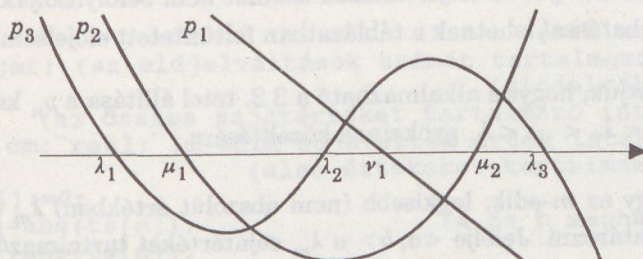
Vezessük be az $\{1, p_1(\lambda), \dots, p_n(\lambda)\}$ polinomsorozat λ helyhez tartozó *előjelváltásainak* számára a $V(\lambda)$ jelölést. Ha valamely λ_0 helyen $p_i(\lambda_0) = 0$, akkor $p_i(\lambda_0)$ előjeleként $p_{i-1}(\lambda_0)$ előjelét tekintjük. (A (2-3) összefüggésből következik, hogy ugyanabban a λ pontban két egymást követő p_i érték nem lehet zérus.)

A következő tételben megfogalmazzuk $V(\lambda)$ -nak azt a tulajdonságát, amely felhasználható a T mátrix néhány vagy összes sajátértékének nagyon pontos meghatározására.

2.2. tétel. A (2-2) polinomsorozat λ helyhez tartozó előjelváltásainak $V(\lambda)$ száma megadja a T szimmetrikus háromatlós mátrix $p_n(\lambda) \equiv \det(T - \lambda I)$ karakterisztikus polinomjában a λ -nál kisebb zérushelyek számát.

Bizonyítás. A bizonyítást $n = 3$ esetre végezzük, mivel ez (újabb ötlet nélkül) indukcióval kiterjeszthető az általános esetre.

Jelölje $\lambda_1 \leq \lambda_2 \leq \lambda_3$ a p_3 , $\mu_1 \leq \mu_2$ a p_2 , valamint ν_1 a p_1 polinom gyökét (lásd a 2.1. ábrát).



2.1. ábra. A zérushelyek elválasztása

Először bebizonyítjuk a

$$\mu_1 < \nu_1 < \mu_2 \quad \text{és} \quad \lambda_1 < \mu_1 < \lambda_2 < \mu_2 < \lambda_3 \quad (2-5)$$

egyenlőtlenségek érvényességét, vagyis azt, hogy a p_i polinom gyökei elválasztják a p_{i+1} polinom gyökeit, ami a Sturm-sorozatok jellemző tulajdonsága.

A (2-5) egyenlőtlenségből következik, hogy a p_3 polinom gyökei egyszeresek, ami a $b_i \neq 0$, $i = 1, \dots, n-1$ feltétel következménye. (Ugyanis bebizonyítható, hogy ha a T szimmetrikus háromatlós mátrix valamelyik sajátértéke r -szeres, akkor a b_i elemek közül legalább $r-1$ elem egyenlő zérussal.)

A p_2 polinom másodfokú, ezen kívül (2-4) alapján $p_2(\lambda) > 0$, ha $|\lambda| \rightarrow \infty$. Továbbá $p_1(\nu_1) = 0$ és (2-3) alapján $p_2(\nu_1) p_0 = p_2(\nu_1) < 0$, amiből $\mu_1 < \nu_1 < \mu_2$ következik (lásd a 2.1. ábrát). Hasonlóképpen, a p_3 polinom harmadfokú, $p_3(\lambda) > 0$, ha $\lambda \rightarrow -\infty$; $p_1(\mu_1) > 0$ míg $p_1(\mu_2) < 0$, ismét (2-3) alapján $p_2(\mu_1) = 0$ miatt $p_3(\mu_1) p_1(\mu_1) < 0$, amiből $p_3(\mu_1) < 0$ és $p_2(\mu_2) = 0$ miatt $p_3(\mu_2) p_1(\mu_2) < 0$, amiből $p_3(\mu_2) > 0$ következik. Ez viszont azt jelenti, hogy a p_3 polinomnak a $(-\infty, \mu_1)$, (μ_1, μ_2) és $(\mu_2, +\infty)$ szakaszok mindegyikében létezik gyöke, amivel a (2-5) egyenlőtlenségeket bebizonyítottuk. Ennek következtében a p_1 , p_2 és p_3 polinomok viselkedésére jellemző a 2.1. ábra (a λ_2 és ν_1 gyökök egymáshoz viszonyított helyzete közömbös), melynek segítségével összeállíthatjuk a p_i polinomok előjeleire vonatkozó következő táblázatot:

	$(-\infty, \lambda_1 >$	$(\lambda_1, \mu_1 >$	$(\mu_1, \lambda_2 >$	$(\lambda_2, \nu_1 >$	$(\nu_1, \mu_2 >$	$(\mu_2, \lambda_3 >$	$(\lambda_3, +\infty)$
p_0	+	+	+	+	+	+	+
p_1	+	+	+	+	-	-	-
p_2	+	+	-	-	-	+	+
p_3	+	-	-	+	+	+	-
$V(\lambda)$	0	1	1	2	2	2	3

Vegyük észre, hogy a $V(\lambda)$ előjelváltások száma csak akkor változik, ha λ olyan intervallumvégponton halad keresztül, amely zérushelye a p_3 polinomnak; a p_1 és p_2 polinomok zérushelyei az előjelváltások számát nem befolyásolják annak ellenére sem, hogy hatással lehetnek a táblázatban feltüntetett előjelzésára.

Végül megmutatjuk, hogyan alkalmazható a 2.2. tétel állítása a p_n karakterisztikus polinom $\lambda_1 < \lambda_2 < \dots < \lambda_n$ gyökeinek közelítésére.

Tegyük fel, hogy az m -edik, legkisebb (nem abszolút értékben) λ_m sajátértéket kívánjuk meghatározni. Jelölje $\langle a, b \rangle$ a λ_m sajátértéket tartalmazó intervallumot. (Mint hogy négyzetes M mátrix sajátértékeire érvényes a $|\lambda| \leq \|M\|$ egyenlőtlenség, pl. az $a = -\|T\|_\infty$, $b = \|T\|_\infty$ választással biztosítható, hogy a $\langle a, b \rangle$ intervallum a T mátrix összes, így a λ_m sajátértéket is tartalmazza.) Számítsuk ki az előjelváltások számát az intervallum $c = (a + b) / 2$ középpontjában. Ha $V(c) < m$, akkor $\lambda_m \in \langle c, b \rangle$, míg ha $V(c) \geq m$, $\lambda_m \in \langle a, c \rangle$. Ha ezt a felezési eljárást úgy folytatjuk, hogy mindig a λ_m zérushelyet tartalmazó intervallumra alkalmazzuk, akkor k lépés végrehajtása után a keresett λ_m sajátértéket tartalmazó intervallum $(b - a) / 2^k$ hosszúságú lesz. Elvileg ez az intervallumhossz tetszőlegesen szűkíthető a k lépésszám megfelelő növelésével.

A Felez eljárás lényegét a 2.2. tétel fogalmazza meg, és alkalmas az n -ed rendű szimmetrikus háromátlós mátrix $\lambda_{m1}, \lambda_{m1+1}, \dots, \lambda_{m2}$ ($\lambda_i \leq \lambda_{i+1}$) sajátértékeinek meghatározására.

```
procedure Felez(
    tf, ts: tvn;
    n, m1, m2: integer;
    eps1, eps: real;
    var eps2: real;
    var NumFelez: integer;
    var lambda: tvn);
```

{ Az eljárás alkalmas az n -ed rendű szimmetrikus háromátlós T mátrix $\lambda_{m1}, \dots, \lambda_{m2}$ ($\lambda_{i+1} \geq \lambda_i$) sajátértékeinek meghatározására. A T mátrixot az n elemű tf és ts vektorokban tároljuk úgy, hogy tf a mátrix főátlóelemeit, ts a főátlóval szomszédos elemeit tartalmazza. $ts[1]$ tetszőleges lehet, az eljárás nullává teszi. $Eps1$ és eps a számítás pontosságát befolyásoló értékek. $Eps1$ a kívánt pontosságot (nem lehet nulla az értéke), eps azt a számítógép aritmetikájától függő legkisebb értéket tartalmazza, melyre $1+eps > 1$ még teljesül. Az eljárás addig folytatja a felezéseket, ameddig a sajátértékek felső és alsó határának különbsége még nagyobb, mint $eps1$. Az eljárás eredményeként a λ vektor az $m1$. elemtől az $m2$. elemig a megfelelő sajátértékeket tartalmazza. A $NumFelez$ a szükséges felezések számát, $eps2$ pedig a sajátértékek hibáinak felső korlátját jelzi.

A Felez eljárást használó programban definiálni kell a

következő globális típust

```

tvn      = array [1..nmax] of real;
ahol az nmax konstans az n legnagyobb lehetséges értéke.}
var
  i, m,                                     {ciklusváltozók}
  V: integer; {az előjelváltások számát tartalmazó változó}
  qi,                                           {előjelváltásokhoz}
  a, b,     {az összes sajátértéket tartalmazó intervallum}
  am, bm, cm: real; {m-edik sajátérték érték intervallumai}
  ai: tvn;      {alsó értékeket tartalmazó vektor}
begin ts[1]:=0;
a:=tf[n]-abs(ts[n]);                               {a és b meghatározása}
b:=tf[n]+abs(ts[n]);
for i:=n-1 downto 1 do
  begin cm:=abs(ts[i])+abs(ts[i+1]);
    if tf[i]+cm>b then b:=tf[i]+cm;
    if tf[i]-cm<a then a:=tf[i]-cm;
  end;
  if (a+b)>0 then eps2:=eps*b                       {b pozitív és b > |a|}
  else eps2:=-eps*a;                               {a negatív és |a| > b}
  if eps1<=0 then eps1:=eps2;
  eps2:=0.5*eps1+7*eps2;
  for i:=m1 to m2 do
    begin lambda[i]:=b; ai[i]:=a; end;
  NumFelez:=0; bm:=b;
  for m:=m2 downto m1 do
    begin am:=a; i:=m;                             {egy jobb alsó érték keresése}
    while (i>=m1) and (am>=ai[i]) do dec(i);
    if i>=m1 then am:=ai[i];
    if bm>lambda[m] then
      bm:=lambda[m];                               {egy jobb felső érték}
    while (bm-am)>(2*eps*(abs(b)+abs(a))+eps1) do
      begin inc(NumFelez); cm:=(am+bm)/2; V:=0; qi:=1;
        for i:=1 to n do {az előjelváltások számítása}
          begin {qi = pi / pi-1}
            if qi<>0 then qi:=tf[i]-cm-sqr(ts[i])/qi
            else qi:=ts[i]-cm-abs(ts[i])/eps;
            if qi<0 then inc(V); {előjelváltás történt}
          end;
        if V<m then
          begin am:=cm; {a sajátérték a <cm,bm>-ban van
            if V<m1 then
              ai[m1]:=cm {m1. alsó értéknek még jó lesz}
            else
              begin ai[V+1]:=cm; {V+1-edik alsó értéknek}
                if lambda[V]>cm then
                  lambda[V]:=cm; {V-edik felső értéknek}
                end;
              end
            end
          else
            bm:=cm; {a sajátérték az <am,cm>-ban van}
          end;
        end;
      lambda[m]:=(am+bm)/2;
    end;
  end;
end;

```


3. Vektorokkal végzett iterációk

Hatványiteráció

Jelölje

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$$

a (valós szimmetrikus) A mátrix sajátértékeit és

$$v_1, \dots, v_n$$

a megfelelő (páronként ortogonális) sajátvektorokat.

Az x_1 vektorral indított

$$\begin{aligned} x_{k+1} &= A x_k, \quad k = 1, 2, \dots \\ &= A^k x_1 \end{aligned} \quad (3-1)$$

iteráció a v_1 ún. *domináló sajátvektorhoz* konvergál, ha

$$(1) \quad |\lambda_1| > |\lambda_2|, \quad (3-2)$$

azaz ha A -nak egyetlen legnagyobb abszolút értékű, ún. *domináló sajátértéke* van, és ha

$$(2) \quad x_1 = c_1 v_1 + \dots + c_n v_n,$$

azaz ha a kezdővektor nem merőleges a v_1 sajátvektorra, vagyis ha $c_1 \neq 0$.

Valóban, ezzel a kezdővektorral (3-1) átírható az

$$\begin{aligned} A^k x_1 &= c_1 \lambda_1^k v_1 + \dots + c_n \lambda_n^k v_n \\ &= \lambda_1^k \left[c_1 v_1 + \sum_{i=2}^n \left(\frac{\lambda_i}{\lambda_1} \right)^k c_i v_i \right] \end{aligned} \quad (3-3)$$

alakra és a (3-2) egyenlőtlenség, azaz a

$$\max_{i=2}^n \frac{|\lambda_i|}{|\lambda_1|} = \frac{|\lambda_2|}{|\lambda_1|} < 1$$

feltétel miatt (3-3)-ból már következik, hogy

$$\frac{1}{\lambda_1^k} A^k x_1 \rightarrow c_1 v_1, \quad \text{amint} \quad k \rightarrow \infty.$$

A hatványiteráció a $|\lambda_2|/|\lambda_1|$ tényezővel lineárisan konvergál a v_1 sajátvektorhoz. (Lineáris konvergencián itt azt értjük, hogy a v_1 sajátvektort közelítő $A^k x_1$ iterált vektorok lineárisan tartalmazzák a „leépítendő” $v_2, \dots, v_n$ sajátvektorokat.)

A gyakorlatban, hogy elkerüljük a $|\lambda_1| > 1$ esetben nagyon nagy és a $|\lambda_1| < 1$ esetben nagyon kicsi mennyiségekkel való számolást, a (3-1) iteráció minden lépéséhez az x_k vektort egységnyi hosszúságúra normáljuk. Ez a konvergenciát nem befolyásolja, hiszen a sajátvektort csak iránya definiálja.

Ha $|\lambda_2|$ csak kevéssel kisebb, mint $|\lambda_1|$, akkor a konvergencia lassú. Ha $\lambda_1 = \lambda_2$, azaz ha a domináló sajátérték kétszeres (esetleg többszörös), az eljárás folyamán látszólag nem következik be változás. A (3-3) összefüggésből következik, hogy

$$\frac{1}{\lambda_1^k} A^k x_1 \rightarrow c_1 v_1 + c_2 v_2, \quad \text{ha } k \rightarrow \infty.$$

Az iteráció tehát a v_1 és a v_2 sajátvektorok által kifeszített altér valamely vektorához tart. Ez az eset abból ismerhető fel, hogy különböző kezdővektorokkal az iteráció különböző sajátvektorokhoz konvergál.

Ha azonban $|\lambda_1| = |\lambda_2|$ és $\lambda_1 \neq \lambda_2$, tehát ha $\lambda_2 = -\lambda_1$, akkor az iteráció nem konvergens. Ekkor ugyanis a (3-3) összefüggésből az következik, hogy

$$\frac{1}{\lambda_1^k} A^k x_1 \rightarrow c_1 v_1 + (-1)^k c_2 v_2, \quad \text{amint } k \rightarrow \infty,$$

tehát a páros és páratlan iterált vektorok más-más határértékhez tartanak.

Ha x_k már jó közelítése a v_1 sajátvektornak, akkor (3-1)-ből

$$x_{k+1} = A x_k \approx \lambda_1 x_k$$

következik. Ha ezt az összefüggést skalárisan megszorozzuk az x_k vektorral és figyelembe vesszük a lépésenkénti $||x_k||^2 = (x_k, x_k) = 1$ normálást, akkor

$$\lambda_1 \approx (x_k, x_{k+1})$$

adódik, vagyis formulát nyerünk a λ_1 sajátérték közelítésére.

Végül megjegyezzük, hogy a

$$c_1 = (x_1, v_1) = 0$$

eset előfordulása sem akadályozza meg a v_1 sajátvektorhoz való konvergenciát, mivel a számítás közben fellépő kerekítési hibák „becsempészik” az $A^k x_1$ iterált vektorokba a v_1 irányú komponenst.

Inverziteráció

Nem szinguláris mátrixot feltételezve, ha az

$$A v = \lambda v$$

alapegyenletet balról megszorozzuk az $(1/\lambda) A^{-1}$ -nel, akkor

$$A^{-1} v = \frac{1}{\lambda} v$$

adódik. Ez azt jelenti, hogy ha v_i a nem szinguláris A mátrix λ_i sajátértékéhez tartozó sajátvektor, akkor v_i az A^{-1} mátrixnak is az $1/\lambda_i$ sajátértékéhez tartozó sajátvektora.

Tegyük fel, hogy a (nem szinguláris, szimmetrikus) A mátrixnak egyetlen *legkisebb* abszolút értékű sajátértéke van, tehát fennállnak a

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_{n-1}| > |\lambda_n| > 0$$

egyenlőtlenségek, amelyek átírhatók

$$\frac{1}{|\lambda_n|} > \frac{1}{|\lambda_{n-1}|} \geq \dots \geq \frac{1}{|\lambda_2|} \geq \frac{1}{|\lambda_1|}$$

alakra is. Következésképpen $1/\lambda_n$ az A^{-1} inverz mátrix legnagyobb abszolút értékű sajátértéke. Így a hozzá tartozó v_n sajátvektor meghatározására alkalmazható az

$$x_{k+1} = A^{-1} x_k, \quad k = 1, 2, \dots$$

hatványiteráció. Ez tetszőleges (a v_n sajátvektorra nem merőleges) x_1 kezdővektorral és

$$\frac{1}{|\lambda_{n-1}|} = \frac{|\lambda_n|}{|\lambda_{n-1}|} < 1 \quad (3-4)$$

konvergenciátényezővel lineárisan tart az A mátrix legkisebb abszolút értékű λ_n sajátértékéhez tartozó v_n sajátvektorhoz.

Az A^{-1} mátrixra alkalmazott hatványiteráció kifejezhető

$$A x_{k+1} = x_k, \quad k = 1, 2, \dots \quad (3-5)$$

vagy

$$x_{k+1} = A^{-k} x_1$$

alakban is. Az eljárást *inverziterációnak* nevezzük.

Megjegyezzük, hogy a (3–5) inverziteráció tipikus példa az olyan lineáris algebrai egyenletrendszerek egymás utáni megoldására, amelyeknél ugyanaz az együtthatómátrix, csak a jobb oldal változik esetenként.

A konvergencia a konvergenciatényező csökkentésével gyorsítható. E célra megfelelő, ún. *spektráleltolások* alkalmazhatók, amelyeknek az az elvi alapja, hogy ha az $A v = \lambda v$ alapegyenlet mindkét oldalából kivonjuk a v sajátvektor μ -szeresét, akkor a

$$(A - \mu I) v = (\lambda - \mu) v$$

egyenletet nyerjük. Ez azt fejezi ki, hogy ha az A mátrix sajátértékei rendre $\lambda_1, \dots, \lambda_n$ és a hozzá tartozó sajátvektorok $v_1, \dots, v_n$, akkor az $A - \mu I$ mátrixnak $\lambda_1 - \mu, \dots, \lambda_n - \mu$ a sajátértékei, a sajátvektorai pedig változatlanul az A mátrix $v_1, \dots, v_n$ sajátvektorai.

Vezessük be az $A - \mu I$ mátrix sajátértékeire vonatkozóan a

$$|\lambda_q - \mu| = \min_{i=1, \dots, n} |\lambda_i - \mu|, \quad |\lambda_p - \mu| = \max_{i=1, \dots, n} |\lambda_i - \mu|$$

jelöléseket. Ekkor az $A - \mu I$ mátrixra alkalmazott hatványiteráció konvergenciatényezője kifejezhető

$$\frac{\max_{i \neq p} |\lambda_i - \mu|}{|\lambda_p - \mu|}$$

alakban, az $A - \mu I$ mátrixszal végzett inverziteráció konvergenciatényezője pedig a (3–4) analógiájaként nem más mint

$$\frac{|\lambda_q - \mu|}{\min_{i \neq q} |\lambda_i - \mu|}.$$

Az $A - \mu I$ mátrixra alkalmazott inverziteráció annál gyorsabban tart a v_q sajátvektorhoz minél jobb közelítése μ a λ_q sajátértéknek!

A hatványiteráció konvergenciatényezője ugyanakkor nem csökkenthető korlát nélkül. Így például, ha a sajátértékekre teljesül, hogy $\lambda_1 > \dots > \lambda_n > 0$, akkor a λ_1 sajátértékhez tartozó v_1 sajátvektorhoz a leggyorsabb konvergenciát a $\mu = (\lambda_2 + \lambda_n) / 2$ középpértékkel való eltolás adja (lásd pl. Bahvalov [1]).

Rayleigh-hányados iteráció

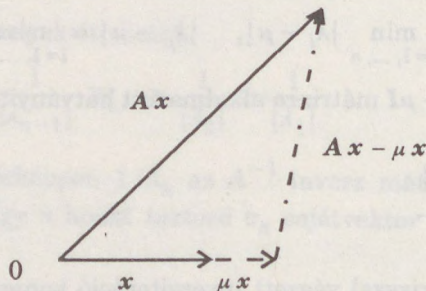
Az inverziteráció konvergenciája nagyon meggyorsítható, ha *eltolásként* a Rayleigh-hányadost választjuk.

Valamely zérustól különböző x vektor (A mátrixszal képezett) *Rayleigh-hányadosán* az

$$R(x) = \frac{(x, Ax)}{(x, x)} \quad (3-6)$$

törtet értjük.

Geometriai szemlélettel, rögzített x vektort feltételezve, tekintsük az Ax és a μx vektorok végpontjainak $\|Ax - \mu x\|_2$ távolságát (lásd a 3.1. ábrát).



3.1. ábra. A Rayleigh-hányados szemléltetése

Ez a távolság pontosan akkor *minimális*, ha az $Ax - \mu x$ vektor *merőleges* az x vektorra, azaz ha $(Ax - \mu x, x) = 0$, vagyis ha $\mu = R(x)$.

Vegyük észre, hogy ha v az A mátrix sajátvektora, akkor $R(v) = \lambda$ a hozzá tartozó sajátérték és $\|Av - \lambda v\| = 0$.

Bebizonyítjuk a következő állítást:

Ha az x vektor a v_q sajátvektor *elsőrendű* közelítése, akkor $R(x)$ a megfelelő λ_q sajátérték *másodrendű* közelítése.

Valóban, ha x a v_q sajátvektor elsőrendű közelítése, ezen azt értjük, hogy az

$$x = c_1 v_1 + \dots + c_n v_n = c_q \left(v_q + \sum_{\substack{i=1 \\ i \neq q}}^n \frac{c_i}{c_q} v_i \right)$$

felírásban $\left| \frac{c_i}{c_q} \right| < 1$ érvényes, ha $i \neq q$. Az $\varepsilon_i = \frac{c_i}{c_q}$, $\sum_{i=1, i \neq q}^n$ jelölések alkalmazásával $R(x)$ kifejezhető

$$R(x) = \frac{(x, Ax)}{(x, x)} = \frac{c_q^2 \left(v_q + \sum \varepsilon_i v_i, \lambda_q v_q + \sum \varepsilon_i \lambda_i v_i \right)}{c_q^2 \left(v_q + \sum \varepsilon_i v_i, v_q + \sum \varepsilon_i v_i \right)}$$

alakban és a sajátvektorok ortonormáltsága miatt

$$\begin{aligned} R(x) &= \frac{\lambda_q + \sum \varepsilon_i^2 \lambda_i}{1 + \sum \varepsilon_i^2} = \left(\lambda_q + \sum \varepsilon_i^2 \lambda_i \right) \left(1 - \sum \varepsilon_i^2 + \left(\sum \varepsilon_i^2 \right)^2 - \dots \right) \\ &\approx \lambda_q + \sum \varepsilon_i^2 \lambda_i, \end{aligned}$$

ami annak a kifejezője, hogy $R(x)$ a λ_q sajátérték másodrendű közelítése és ezzel a bizonyítást befejeztük.

Az inverziteráció alkalmazásánál nemcsak egy stratégia követhető. A Rayleigh-hányadossal való eltolást vagy minden lépésben el kell végezni, vagy esetenként nem. Ha ezt az eltolást az iteráció minden lépésében végrehajtjuk, akkor a

$$(A - R(x_k)I)x_{k+1} = x_k l_k \quad (3-7)$$

összefüggéssel értelmezett Rayleigh-hányados iterációt nyerjük, ahol az l_k számot úgy határozzuk meg, hogy $||x_{k+1}||_2 = 1$.

A (3-7) formula (az l_k szorzótényező szerepe) jobban megérthető, ha az iterációt a következő két lépésben fejezzük ki

$$(A - R(x_k)I)y_{k+1} = x_k \quad \text{és} \quad x_{k+1} = \frac{y_{k+1}}{||y_{k+1}||_2} \equiv y_{k+1} l_k.$$

A gyakorlatban a Rayleigh-hányados iterációt célszerű a

$$(A - R(\bar{x}_k)I)\bar{x}_{k+1} = x_k, \quad (3-8)$$

$$R(\bar{x}_{k+1}) = \frac{\bar{x}_{k+1}^T x_k}{\bar{x}_{k+1}^T \bar{x}_{k+1}} + R(\bar{x}_k), \quad (3-9)$$

$$x_{k+1} = \frac{\bar{x}_{k+1}}{||\bar{x}_{k+1}||_2} \quad (3-10)$$

képletek alapján implementálni, ahol az x_1 kezdővektorra $||x_1||_2 = 1$ érvényes és az $R(\bar{x}_1)$ kezdő eltolást általában zérusnak választjuk. (A (3-9) összefüggést

úgy nyerjük (3-8)-ból, hogy azt balról megszorozzuk a $\bar{x}_{k+1}^T / (\bar{x}_{k+1}^T \bar{x}_{k+1})$ vektorral és figyelembe vesszük a $R(\bar{x}_{k+1}) = \bar{x}_{k+1}^T A \bar{x}_{k+1} / (\bar{x}_{k+1}^T \bar{x}_{k+1})$ definíciót.)

Ekkor $k \rightarrow \infty$ esetén

$$x_{k+1} \rightarrow v_r \text{ és } R(\bar{x}_k) \rightarrow \lambda_r$$

írható, ahol a λ_r sajátérték és a v_r sajátvektor függ az iteráció x_1 kezdővektorától és az $R(\bar{x}_1)$ kezdő eltolástól. Ha az x_1 vektor nagy súlyszámmal tartalmazza pl. a v_q sajátvektort és $R(\bar{x}_1)$ elegendően közel van a v_q -hoz tartozó λ_q sajátértékhez, akkor az $\{R(x_k), x_k\}$ sorozat a (λ_q, v_q) sajátpárhoz konvergál. Egyébként az iteráció egy másik sajátpárhoz tart. A konvergencia harmadrendű mind a sajátérték, mind a sajátvektor vonatkozásában (lásd pl. Bathe-Wilson [2]). Ez lényegében azért van így, mert a (3-5) inverziteráció ugyan mindössze lineárisan tart a sajátvektorhoz, de az iteráció minden lépésében az iterált vektorral képezzük a sajátérték másodrendű közelítését és ezzel spektrál eltolást hajtunk végre.

Megjegyezzük, hogy amint $R(x_k)$ tart az A sajátértékéhez, $A - R(x_k)I$ tart a szinguláris mátrixhoz. A Rayleigh-hányados iteráció mégis a megfelelő sajátvektorhoz konvergál.

További sajátpárok meghatározása

Tegyük fel, hogy az A mátrix λ_r sajátértékét és a hozzá tartozó v_r sajátvektort már meghatároztuk. Hogy az iteráció további sajátpár meghatározására is alkalmas legyen, vagy a mátrixot, vagy az iterált vektorokat ún. „deflációnak” kell alávetni.

Stabilis mátrixdefláció elvégezhető egy olyan ortogonális $P = [v_r, Q]$ mátrix előállításával, amelynek első oszlopa a már kiszámított v_r sajátvektor. Ekkor ugyanis írható, hogy

$$P^T A P = \begin{bmatrix} v_r^T \\ Q^T \end{bmatrix} \begin{bmatrix} A v_r, A Q \end{bmatrix} = \begin{bmatrix} \lambda_r v_r^T v_r, & v_r^T A Q \\ Q^T A v_r, & Q^T A Q \end{bmatrix} = \begin{bmatrix} \lambda_r, & 0^T \\ 0, & A_{n-1} \end{bmatrix}, \quad (3-11)$$

hiszen $v_r^T v_r = 1$ és az ortogonalitás miatt $Q^T A v_r = \lambda_r$, $Q^T v_r = 0$ és $v_r^T A Q = v_r^T A^T Q = (A v_r)^T Q = \lambda_r v_r^T Q = 0^T$.

A $P^T A P (= P^{-1} A P)$ ortogonális hasonlósági transzformáció az 1.2. tétel miatt nem változtatja meg az A sajátértékeit. Ezért (3-11)-ből következik, hogy az $A_{n-1} = Q^T A Q$ mátrixnak ugyanazok a sajátértékei mint A -nak, kivéve a λ_r sajátértéket.

Megjegyezzük, hogy ha meghatároztuk az A_{n-1} mátrix λ_s sajátértékéhez tartozó x sajátvektorát, akkor a $Q^T A Q x = \lambda_s x$ egyenletből következik, hogy $Q x$ az A mátrix λ_s sajátértékéhez tartozó sajátvektor.

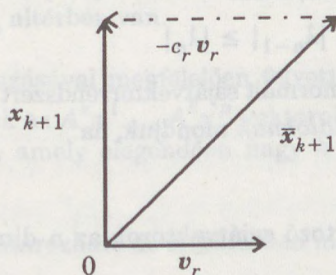
A $P = [v_r, Q]$ mátrix meghatározása nem egyértelmű.

*Vektordefláció*n az iterált vektorok ortogonalizálását értjük a már kiszámított v_r sajátvektorhoz képest.

Tegyük fel, hogy az $\bar{x}_{k+1}$ iterált vektort a (3-1), (3-5), (3-7) eljárások egyikével nyertük. Az

$$\bar{x}_{k+1} = c_1 v_1 + \dots + c_n v_n$$

lineáris kombináció v_r sajátvektor irányú komponense kiküszöbölhető a *Gram-Schmidt ortogonalizálási eljárás* alkalmazásával úgy, ahogy azt a 3.2. ábrán vázoltuk.



3.2. ábra. Vektordefláció

Az

$$x_{k+1} = \bar{x}_{k+1} - c_r v_r$$

összefüggésben a c_r együttható a

$$(v_r, x_{k+1}) = (v_r, \bar{x}_{k+1} - c_r v_r) = 0$$

ortogonalitási feltételből nyerhető, tehát $c_r = (v_r, \bar{x}_{k+1})$, hiszen feltettük, hogy $(v_r, v_r) = 1$.

Tegyük fel, hogy pl. a hatványiteráció alkalmazásával már meghatároztuk a $v_1, \dots, v_p$ sajátvektorokat, és az $\bar{x}_1$ vektort e sajátvektorokra kívánjuk ortogonalizálni. A Gram-Schmidt ortogonalizálásnak az a lényege, hogy a $v_1, \dots, v_p$ sajátvektorokra merőleges x_1 vektort

$$\mathbf{x}_1 = \bar{\mathbf{x}}_1 - \sum_{i=1}^p c_i \mathbf{v}_i \quad (3-12)$$

alakban keressük, ahol a $c_1, \dots, c_p$ együtthatókat a

$$(\mathbf{v}_i, \mathbf{x}_1) = 0, \quad i = 1, \dots, p \quad \text{és} \quad (\mathbf{v}_i, \mathbf{v}_j) = \delta_{ij}$$

(δ_{ij} = Kronecker-delta) feltételekből határozzuk meg. Ha a (3-12) mindkét oldalát balról megszorozzuk a $\mathbf{v}_i^T$ vektorral, akkor

$$c_i = \mathbf{v}_i^T \bar{\mathbf{x}}_1, \quad i = 1, \dots, p.$$

Ha a hatványiterációt $\bar{\mathbf{x}}_1$ helyett az $\mathbf{x}_1$ vektorral kezdjük és feltesszük, hogy $(\mathbf{x}_1, \mathbf{v}_{p+1}) \neq 0$, akkor az iteráció (legalábbis elvileg) a $\mathbf{v}_{p+1}$ sajátvektorhoz konvergál, és a sajátvektor ismeretében a λ_{p+1} sajátérték is (pl. $R(\mathbf{v}_{p+1})$ Rayleigh-hányadosként) meghatározható.

Altér iteráció

A valós szimmetrikus n -ed rendű $\mathbf{A}$ mátrix sajátértékeit a továbbiakban is abszolút értékük csökkenő sorrendjében, tehát

$$|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_{n-1}| \geq |\lambda_n| \quad (3-13)$$

szerint írjuk, a hozzá tartozó ortonormált sajátvektorrendszert pedig jelölje $\mathbf{v}_1, \dots, \mathbf{v}_n$. A $\lambda_1, \dots, \lambda_p$ sajátértékeket *dominálóknak* mondjuk, ha

$$|\lambda_p| > |\lambda_{p+1}|. \quad (3-14)$$

A p domináló sajátértékhez tartozó sajátvektorok az n dimenziós vektortérben kifeszítenek egy

$$S_p = S_p(\mathbf{A}) = \text{span}\{\mathbf{v}_1, \dots, \mathbf{v}_p\} \quad (3-15)$$

alteret, amelyet az $\mathbf{A}$ mátrix p -edik *domináló alterének* nevezünk. Így a $\mathbf{v}_1, \dots, \mathbf{v}_p$ sajátvektorok az S_p egy ortonormált bázisát alkotják és az is nyilvánvaló, hogy S_p dimenziója p . Az $\mathbf{A} \mathbf{v}_j = \lambda_j \mathbf{v}_j$, $j = 1, \dots, p$ összefüggésekből (és abból, hogy

ha $\mathbf{x} \in S_p$, akkor $\mathbf{x} = \sum_{j=1}^p c_j \mathbf{v}_j$) következik, hogy

$$\mathbf{A} S_p = \{\mathbf{y} = \mathbf{A} \mathbf{x} : \mathbf{x} \in S_p\} \subseteq S_p \quad (3-16)$$

vagyis S_p invariáns az $\mathbf{A}$ mátrixszal reprezentált lineáris transzformációval szemben. Az olyan S_p alteret, amelyre a (3-16) teljesül, az $\mathbf{A}$ mátrix *invariáns alterének* nevezzük.

Ha a domináló sajátvektorokat sikerül leválasztani a többitől (hatványiterációval), akkor várható, hogy az S_p jó közelítéséből a $\lambda_1, \mathbf{v}_1, \dots, \lambda_p, \mathbf{v}_p$ sajátpárok jó közelítései is meghatározhatók.

Az S_p altér jó közelítéseinek előállítására képezzük az A mátrix

$$A = V\Lambda V^T = (\lambda_1 v_1 v_1^T + \dots + \lambda_p v_p v_p^T) + (\lambda_{p+1} v_{p+1} v_{p+1}^T + \dots + \lambda_n v_n v_n^T) \quad (3-17)$$

felbontásából következő A^k hatványmátrix

$$A^k = V\Lambda^k V^T = (\lambda_1^k v_1 v_1^T + \dots + \lambda_p^k v_p v_p^T) + (\lambda_{p+1}^k v_{p+1} v_{p+1}^T + \dots + \lambda_n^k v_n v_n^T) \quad (3-18)$$

felbontását. A hatványozás miatt, k növelésével fokozatosan erősödik az első p tag domináló hatása, úgy hogy elegendően nagy k esetén az A^k hatványt gyakorlatilag az első zárójelben levő tagok képviselik.

Ha az A^k operátort tetszőleges n elemű x vektorra alkalmazzuk, akkor

$$A^k x = \sum_{j=1}^n \lambda_j^k v_j (v_j^T x) \approx \lambda_1^k (v_1^T x) v_1 + \dots + \lambda_p^k (v_p^T x) v_p \quad (3-19)$$

adódik. Ha feltesszük, hogy a $v_1^T x, \dots, v_p^T x$ együtthatók egyike sem nulla, akkor az ennek megfelelően felvett x vektor és elegendően nagy k esetén az $A^k x$ vektor már „majdnem” az első p sajátvektor lineáris kombinációja, vagyis az $A^k x$ vektor „majdnem” az S_p altérben van.

Ha a (3-19) alkalmazásával megfelelően felvett, lineárisan független $x^1, \dots, x^p$ vektorokkal képezzük az $A^k x^1, \dots, A^k x^p$ vektorokat, akkor ezek a vektorok kifeszítenek egy alteret, amely elegendően nagy k esetén elég jó közelítése az S_p domináló altérnek.

Az $A^k x$ vektor meghatározható az előbbieken már részletezett

$$\bar{x}_{k+1} = A x_k, \quad x_{k+1} = \bar{x}_{k+1} / \|\bar{x}_{k+1}\|_2, \quad k = 1, 2, \dots \quad (3-20)$$

hatványiterációval.

A hatványiterációval a normált vektoroknak egy $x_k, k = 1, 2, \dots$ sorozatát állítjuk elő úgy, hogy a $\text{span}\{x_k\}, k = 1, 2, \dots$ egydimenziós alterek sorozata approximálja az $S_1 = \text{span}\{v_1\}$ domináló alteret.

Ha a p domináló $\lambda_1, \dots, \lambda_p$ sajátértékekhez tartozó $v_1, \dots, v_p$ sajátvektorokat kívánjuk meghatározni, akkor a $|\lambda_1| > |\lambda_2| > \dots > |\lambda_p| > |\lambda_{p+1}|$ feltétel figyelembevételével az előbbieken már ismertetett *deflációs technikák* alkalmazhatók.

Ennél azonban általában sokkal előnyösebb meghatározni az S_p p -edik domináló altér közvetlen Y_k közelítéseit. E célra az *altériteráció* kínálkozik, amelynek az az alapötlete, hogy kiindulva a valós n dimenziós R^n tér egy p dimenziós Y_1 alteréből, az

$$Y_{k+1} = A Y_k = \{A y : y \in Y_k\}, \quad k = 1, 2, \dots$$

formulával képezzük az Y_k altereknek egy olyan sorozatát, amely bizonyos feltételek mellett tart az S_p altérhez.

Az altériteráció *implementációjához* az Y_k alteret célszerű kifejezni

$$Y_k = \text{span}\{\mathbf{x}_k^1, \dots, \mathbf{x}_k^p\}, \quad \mathbf{X}_k = [\mathbf{x}_k^1, \dots, \mathbf{x}_k^p]$$

alakban, vagyis megadni egy $n \times p$ típusú $\mathbf{X}_k$ mátrixot, amelynek oszlopai kifeszítik az Y_k alteret.

Ekkor az

$$\bar{\mathbf{X}}_{k+1} = [\bar{\mathbf{x}}_{k+1}^1, \dots, \bar{\mathbf{x}}_{k+1}^p] = \mathbf{A} \mathbf{X}_k = [\mathbf{A} \mathbf{x}_k^1, \dots, \mathbf{A} \mathbf{x}_k^p] \quad (3-21)$$

mátrix oszlopai kifeszítik az $Y_{k+1} = \mathbf{A} Y_k$ alteret. A nagyon nagy és a nagyon kicsi mennyiségekkel való számolás elkerülésére alkalmazható az oszlopok

$$\mathbf{X}_{k+1} = \bar{\mathbf{X}}_{k+1} \begin{bmatrix} 1 / \|\bar{\mathbf{x}}_{k+1}^1\| & & \\ & \ddots & \\ & & 1 / \|\bar{\mathbf{x}}_{k+1}^p\| \end{bmatrix} \quad (3-22)$$

szerinti normálása, hiszen $\text{span}\{\bar{\mathbf{x}}_{k+1}^1, \dots, \bar{\mathbf{x}}_{k+1}^p\} = \text{span}\{\mathbf{x}_{k+1}^1, \dots, \mathbf{x}_{k+1}^p\}$.

Ez az eljárás nem más, mint p vektorral *szimultán* végrehajtott *hatványiteráció*. Ezért ha $|\lambda_1| > |\lambda_2|$, akkor az $\mathbf{X}_k$ mátrix oszlopai mind ugyanahhoz a $\mathbf{v}_1$ sajátvektorhoz konvergálnak. Tehát k növelésével az $\mathbf{x}_k^1, \dots, \mathbf{x}_k^p$ vektorok fokozatosan válnak párhuzamossá, és így mindkevésbé lesznek alkalmasak az Y_k altér kifeszítésére.

Ez a numerikus instabilitás kiküszöbölhető, ha pl. a Gram-Schmidt ortonormalizálási eljárással az Y_{k+1} alterekben ortonormált bázisokat képezünk. Ekkor az altériteráció kifejezhető

$$\bar{\mathbf{X}}_{k+1} = \mathbf{A} \mathbf{X}_k \quad (3-23)$$

$$\bar{\mathbf{X}}_{k+1} = \mathbf{X}_{k+1} \mathbf{R}_{k+1}, \quad k = 1, 2, \dots \quad (3-24)$$

alakban, ahol $\mathbf{R}_{k+1}$ olyan p -ed rendű nem szinguláris felső háromszögmátrix, amelyre teljesül, hogy $\mathbf{X}_{k+1}^T \mathbf{X}_{k+1} = \mathbf{I}_p$.

Ha az $\mathbf{X}_1$ mátrix oszlopai, mint kezdővektorok, komponensként tartalmazzák a $\mathbf{v}_1, \dots, \mathbf{v}_p$ sajátvektorokat, akkor $k \rightarrow \infty$ esetén

$$\mathbf{X}_k \rightarrow \mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_p] \quad \text{és} \quad \mathbf{R}_k \rightarrow \mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_p \end{bmatrix}.$$

A Rayleigh–Ritz-módszer

Megmutatjuk, hogy az X_k mátrixok által reprezentált Y_k altérközelítések ismeretében hogyan határozhatók meg az A mátrix keresett λ_j , v_j , $j = 1, \dots, p$ sajátpárja-inak meglehetősen jó μ_j , u_j , $j = 1, \dots, p$ közelítései. Ha a közelítés pontosságát azzal „mérjük”, hogy az $A u_j - \mu_j u_j$ hibavektorok normáinak négyzetösszege mennyire tér el a zérustól, akkor a következő eredményt kapjuk. (Emlékeztetünk arra, hogy ha A egy $n \times p$ típusú valós mátrix, akkor annak euklideszi normája kifejezhető

$$||A||_E = \left(\sum_{i=1}^n \sum_{j=1}^p a_{ij}^2 \right)^{1/2} = \left(\sum_{j=1}^p ||a_j||^2 \right)^{1/2}$$

alakban, ahol $||a_j|| = \left(a_j^T a_j \right)^{1/2}$ jelöli az A mátrix j -edik oszlopának euklideszi vektornormáját.)

3.1. tétel. Tegyük fel, hogy az $n \times p$ típusú X mátrix ortonormált oszlopai az n dimenziós térben egy $Y = \text{span}\{x_1, \dots, x_p\}$ p dimenziós alteret feszítenek ki.

Az X mátrixszal és az n -ed rendű szimmetrikus A mátrixszal képezzük a p -ed rendű

$$R = (r_{ij}) = X^T A X \quad (3-25)$$

mátrixot, majd az R mátrix

$$R = Q M Q^T \quad (3-26)$$

sajátérték-feladatának megoldásával képezzük az $M = \langle \mu_1, \dots, \mu_p \rangle$ diagonálmátrixot és a p -ed rendű ortogonális Q mátrixot. Ekkor az

$$U = [u_1, \dots, u_p] = X Q \quad (3-27)$$

mátrixszal képezett M , U pár megoldása a

$$||A U - U M||_E^2 = \sum_{j=1}^p ||A u_j - \mu_j u_j||^2 = \min ! \quad (3-28)$$

feladatnak az $M = \langle \mu_1, \dots, \mu_p \rangle$ alakú diagonálmátrixok halmazán és az olyan $n \times p$ típusú U mátrixok halmazán, amelyeknek oszlopai ortonormáltak és kifeszítik az Y alteret, vagyis amelyekre érvényes, hogy $\text{span}\{u_1, \dots, u_p\} = \text{span}\{x_1, \dots, x_p\}$.

Bizonyítás. Mivel $\text{span}\{u_1, \dots, u_p\} = \text{span}\{x_1, \dots, x_p\}$ az $U = XQ$ szorzatban a p -ed rendű Q mátrix nem szinguláris és az oszlopokra vonatkozó $I_p = U^T U = Q^T X^T X Q = Q^T Q$ ortonormáltsági követelmény miatt, Q ortogonális is. Az ilyen U mátrixszal képezett $H = A U - U M$ hibamátrixra

$$||H||_E = ||H Q^T||_E = ||A X - X(Q M Q^T)||_E$$

adódik (mivel $H^T = [h_1, \dots, h_p]$ jelöléssel azt kapjuk, hogy $||H Q^T||_E^2 = ||(H Q^T)^T||_E^2 = ||Q [h_1, \dots, h_p]||_E^2 = \sum_{j=1}^p ||Q h_j||^2 = \sum_{j=1}^p h_j^T Q^T Q h_j = ||H||_E^2$).

Ezért feladatunk egyenértékű a

$$||H||_E^2 = ||A X - X R||_E^2 = \sum_{j=1}^p ||A x_j - \sum_{i=1}^p r_{ij} x_i||^2 \quad (3-29)$$

norma (R szerinti) minimalizálásával, ahol $R = (r_{ij}) = Q M Q^T$, M diagonális, Q pedig ortogonális mátrix.

Ha egyelőre nem vesszük figyelembe, hogy az R mátrix szimmetrikus, akkor a minimalizálandó összegben a tagok függetlenek egymástól, hiszen a j -edik tag az R mátrixnak kizárólag a j -edik oszlopától függ. Ezért a (3-29) összeg minimalizálása ekvivalens a tagok egyenkénti minimalizálásával.

Tehát minden egyes $j = 1, \dots, p$ indexre meg kell oldani a

$$||A x_j - \sum_{i=1}^p r_{ij} x_i|| = \min!$$

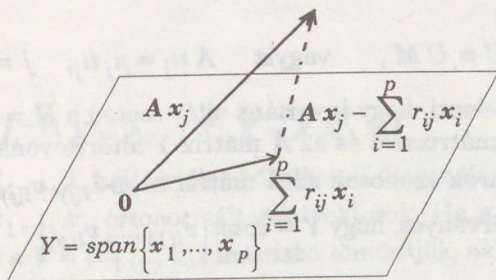
feladatot.

Rögzített j index esetén, az $A x_j$ vektor távolsága az X oszlopai által kifeszített Y altér vektoraitól akkor és csak akkor minimális, ha az $A x_j - \sum_{i=1}^p r_{ij} x_i$ vektor ortogonális az Y altérre, vagyis ha

$$\left(A x_j - \sum_{i=1}^p r_{ij} x_i, x_i \right) = 0, \quad i = 1, \dots, p \quad (3-30)$$

érvényes, amely nem más, mint az ún. projekció elv alkalmazása (lásd a 3.3. ábrát). Az X mátrix oszlopai ortonormáltak, ezért (3-30)-ból

$$r_{ij} = x_i^T A x_j, \quad i = 1, \dots, p$$



3.3. ábra. A projekció elv szemléltetése

következik. Mivel a fentiek minden $j = 1, \dots, p$ indexre is érvényesek, ez azt jelenti, hogy az

$$R = X^T A X$$

mátrix minimalizálja a (3-29) normát. Mivel az eredményként kapott R mátrix szimmetrikus, sajátérték-feladatának megoldása után előállítható (3-26) alakban és ezzel a tétel bizonyítását befejeztük.

A 3.1. tételben megfogalmazott eljárást, amellyel az A, X mátrixpárhoz hozzárendeljük az M, U mátrixpárt, Rayleigh–Ritz-algoritmusnak nevezzük. A $\mu_j, u_j, j = 1, \dots, p$ mennyiségeket az $Y = \text{span}\{x_1, \dots, x_p\}$ altérre vonatkozó Ritz-féle sajátértékeknek, ill. sajátvektoroknak nevezzük. Az $r_{ij} = x_i^T A x_j$ elemű R mátrix a Rayleigh-hányados egy általánosítása, amelyet az A mátrix $Y = \text{span}\{x_1, \dots, x_p\}$ altérre vonatkozó vetületének nevezzük.

Megjegyzés. Emlékeztetünk a Rayleigh-hányados, azaz a (3-6) alatti formula értelmezését követő geometriai szemléltetésre. A Rayleigh-hányados általánosításaként kapott $R = X^T A X$ mátrix is hasonló okoskodással megkapható a projekció elv következő alkalmazásával: Az $\|AX - XR\|_E$ távolság pontosan akkor minimális, ha az $AX - XR$ mátrix ortogonális az X mátrixra, azaz ha $(AX - XR, X) = 0$, vagyis ha $R = X^T A X$, hiszen $X^T X = I_p$.

Ha Y az A mátrix invariáns altere, azaz ha $\text{span}\{Ax_1, \dots, Ax_p\} \subseteq \text{span}\{x_1, \dots, x_p\}$, akkor teljesül, hogy $AX = [Ax_1, \dots, Ax_p] = [c_{11}x_1 + \dots + c_{p1}x_p, \dots, c_{1p}x_1 + \dots + c_{pp}x_p] = XC$, ahol C egy p -ed rendű mátrix. Az $AX = XC$ összefüggésből $X^T A X = C$ és $R = C$ választással $AX = XR$ következik. (Ha Y nem invariáns altere A -nak, akkor nincs olyan $R = C$ mátrix, amelyre $AX = XC$.) Ha az

$AX = XR = XQMQ^T$ egyenlőséget jobbról megszorozzuk a Q mátrixszal, akkor

$$AU = UM, \quad \text{vagyis} \quad Au_j = \mu_j u_j, \quad j = 1, \dots, p$$

adódik. Ez azt jelenti, hogy invariáns altér esetén a $H = AU - UM$ hibamátrix egyenlő a zérusmátrixszal és az A mátrix Y altérre vonatkozó $\mu_j, u_j, j = 1, \dots, p$ Ritz-féle saját párok azonosak az A mátrix azon $\lambda_{l(j)}, v_{l(j)}, j = 1, \dots, p$ saját páraival, amelyekre érvényes, hogy $Y = \text{span}\{v_{l(1)}, \dots, v_{l(p)}\}$.

Ha tehát az altér iterációval generált X_k mátrixok oszlopai már csak a keresett sajátvektorok lineáris kombinációi, akkor a Rayleigh–Ritz-algoritmus egyetlen lépésének alkalmazása a keresett saját párokat eredményezi.

Megjegyezzük, hogy a p -ed rendű R mátrix sajátérték-feladatának megoldására a továbbiakban ismertetendő Jacobi-eljárást célszerű alkalmazni.

Az altér iteráció és a Rayleigh–Ritz-módszer a legegyszerűbben a következőképpen kombinálható. Miután a (3–23), (3–24) képletekkel kiszámítottuk az ortogonális oszlopokkal rendelkező X_{k+1} mátrixot, azt a Rayleigh–Ritz-módszernek megfelelően módosítjuk. Vagyis az $X := X_{k+1}$ mátrixszal képezzük az $R = X^T AX$ mátrixot és elvégezzük annak $R = QMQ^T$ spektrálfelbontását, majd a (3–23) iterációt az $X_{k+1} := U = XQ$ mátrixszal folytatjuk.

Általánosított sajátérték-feladat

Az $Av = \lambda v$, ún. speciális sajátérték-feladat mellett az alkalmazásokban gyakori az

$$Av = \lambda Bv, \quad \text{vagyis} \quad (A - \lambda B)v = 0 \quad (3-31)$$

ún. lineáris általánosított sajátérték-feladat is, ahol A és B valós szimmetrikus n -ed rendű mátrix és a két mátrix közül legalább az egyik, pl. a B mátrix pozitív definit.

Ezért elvégezhető a $B = LL^T$ Cholesky-felbontás, ahol L valós nem szinguláris alsó háromszögmátrix. Következésképpen

$$Av = \lambda LL^T v,$$

és bevezetve az

$$L^T v = x \quad (3-32)$$

transzformációt a (3–31) általánosított feladat visszavezethető az

$$L^{-1}AL^{-T}x = \lambda x \quad (3-33)$$

speciális sajátérték-feladatra, ahol $L^{-1}AL^{-T}$ valós szimmetrikus n -ed rendű mátrix.

Ezért érvényesek a

$$X^T(L^{-1}AL^{-T})X = \Lambda, \quad X^TX = XX^T = I \quad (3-34)$$

összefüggések, ahol Λ a $\lambda_1, \dots, \lambda_n$ sajátértékekből alkotott diagonálmátrix, X oszlopai pedig a hozzá tartozó $x_1, \dots, x_n$ ortonormált sajátvektorok. Ha a (3-31) egyenlet nemtriviális megoldásait a $V = [v_1, \dots, v_n]$ mátrixba tömörítjük, akkor (3-32) miatt azt kapjuk, hogy

$$X = [x_1, \dots, x_n] = L^T V.$$

Ennek behelyettesítésével a (3-34) összefüggésbe

$$V^T A V = \Lambda \quad \text{és} \quad V^T B V = I \quad (3-35)$$

adódik. Tehát beláttuk, hogy ha az A, B mátrixpár szimmetrikus és B pozitív definit, akkor a (3-31) általánosított sajátérték-feladatnak létezik n valós $\lambda_1, \dots, \lambda_n$ sajátértéke és n B -ortonormált¹ hozzá tartozó $v_1, \dots, v_n$ sajátvektora.

Az alkalmazásokban gyakran előforduló eset, amikor keressük a (3-31) lineáris sajátérték-feladat p legkisebb abszolút értékű, vagyis a

$$0 < |\lambda_1| \leq |\lambda_2| \leq \dots \leq |\lambda_p| < |\lambda_{p+1}| \leq \dots \leq |\lambda_n|$$

egyenlőtlenségeket kielégítő sajátértékeket és a hozzá tartozó B -ortonormált $v_1, \dots, v_p$ sajátvektorokat.

Az $A v_j = \lambda_j B v_j$, $j = 1, \dots, p$ egyenletek az egyetlen

$$A V = B V \Lambda \quad (3-36)$$

egyenletbe tömöríthetők, ahol $\Lambda = \langle \lambda_1, \dots, \lambda_p \rangle$ diagonálmátrix, $V = [v_1, \dots, v_p]$ és $V^T B V = I$.

Nem szinguláris A mátrix esetén a megoldásra alkalmazható az

$$A \bar{X}_{k+1} = B X_k, \quad k = 1, 2, \dots \quad (3-37)$$

szimultán inverziteráció p vektorral. A p vektor (oszlopvektorok) által alkotott X_k mátrix

$$X_k^T B X_k = I_p \quad (3-38)$$

B -ortonormált.

¹ A V mátrixot B -ortonormáltnak nevezzük, ha a (3-35) második összefüggése teljesül.

Miután a (3-37) lineáris algebrai egyenletrendszer megoldásával meghatároztuk az $\bar{X}_{k+1}$ mátrixot, ezt kell B -ortonormálni. E célra előnyösen alkalmazható a *Rayleigh-Ritz-algoritmus* a következőképpen: Kiszámítjuk az

$$A_{k+1} = \bar{X}_{k+1}^T A \bar{X}_{k+1}, \quad B_{k+1} = \bar{X}_{k+1}^T B \bar{X}_{k+1} \quad (3-39)$$

mátrixokat és megoldjuk az

$$A_{k+1} Q_{k+1} = B_{k+1} Q_{k+1} \Lambda_{k+1} \quad (3-40)$$

p -ed rendű mátrixpár sajátérték-feladatát, ahol a Q_{k+1} sajátvektorokból álló mátrixot

$$Q_{k+1}^T B_{k+1} Q_{k+1} = I_p \quad (3-41)$$

szerint B_{k+1} -ortonormáljuk. Ekkor az

$$X_{k+1} = \bar{X}_{k+1} Q_{k+1} \quad (3-42)$$

mátrix oszlopai is B -ortonormáltak. Valóban, (3-41) miatt

$$I_p = Q_{k+1}^T B_{k+1} Q_{k+1} = Q_{k+1}^T (\bar{X}_{k+1}^T B \bar{X}_{k+1}) Q_{k+1} = X_{k+1}^T B X_{k+1}$$

és ezzel az állítást bebizonyítottuk.

Ha az X_1 kezdőmátrix oszlopai a $v_1, \dots, v_p$ sajátvektorok egyikére sem ortogonálisak, akkor $k \rightarrow \infty$ esetén

$$\Lambda_k \rightarrow \Lambda \quad \text{és} \quad X_k \rightarrow V.$$

E módszer képezi az elvi alapját a következő, ún. *Alteriteracio* eljárásnak, amely alkalmas a fentiekben részletezett mátrixpár részleges sajátérték-feladatának megoldására akkor is, ha a mátrixok rendszáma, ill. szalagszélessége nagy.

A (3-40) alatti p -ed rendű mátrixpár sajátérték-feladatának megoldására a továbbiakban ismertetett általánosított Jacobi-féle diagonalizálást alkalmaztuk.


```

Procedure Alteriteracio(      A: tvNa;
                             B: tvNb;
                             iatlo: tInl;
                             var V: tvNxq;
                             var lambda: tvq;
                             n, na, nb, p, q: integer;
                             var maxite: integer;
                             var h: tvq;
                             eps: real);

```

{ Alkalmas az $A v = \lambda B v$ általánosított sajátérték-feladat p legkisebb sajátértékének és hozzá tartozó sajátvektorának meghatározására, ahol A és B pozitív definit, valós szimmetrikus n -ed rendű mátrix. Elemeiket na és nb elemű A és B vektorokban tároljuk. A tárolása a körvonaltechnikának megfelelő (részletesebben 1. az LDLTfelbont eljárást), Az $n+1$ elemű iatlo vektor A megfelelő indexeit tartalmazza. B vagy azonos körvonalú A -val ($na=nb$ esetén), vagy diagonálmátrix ($nb=n$). A megoldás során p helyett több, q ($q \geq p$) vektorral végezzük az iterációt. A lambda kimenő vektor az eredményként nyert p legkisebb sajátértéket tartalmazza. A V tömb bemenetkor az altér kezdő közelítését, kimenetkor a p számú B -ortonormált sajátvektort tartalmazza. A kezdővektorok jó megadása nagyon fontos, ezért előállításukat célszerű a későbbiekben ismertetendő KezdoAlter eljárással végezni. Az iterációt befejezzük, ha az iterációk száma nagyobb a maxite értéknél, vagy az egymás után kiszámított legkisebb p sajátértékek relatív hibája kisebb mint eps, ahol az eps bemenő paraméter jelzi a hibakorlátot. A maxite változó kimenetkor a végrehajtott iterációk számát mutatja. A q elemű kimenő h vektor elemei a megfelelő sajátértékek relatív hibái ($||A v - \lambda B v|| / ||A v||$).

```

procedure Cserel(var a,b: real);

```

```

var s: real;
begin s:=a; a:=b; b:=s; end;

```

```

procedure Szoroz(var v: tvN;      {eredményvektor}
                 var A ;          {bal oldali tag (körvonal)}
                 r: tvNxq;        {jobb oldali tag}
                 iatlo: tInl;      {A főátlójának indexei}
                 n, na, j: integer);

```

{ $v = A * r[j]$ kiszámítása, ahol v és $r[j]$ n elemű vektor, az n -ed rendű A mátrix a körvonaltechnikának megfelelően na elemű vektor, iatlo a főátlóelemek indexeit tartalmazó vektor (1. LDLTfelbont eljárás).}

```

type

```

```

tv = array [1..10000] of real;

```

```

var
  i, k, ik: integer;      {indexek}
  s: double;

```

```

begin

```

```

  if na<=n then

```

```

    for i:=1 to n do      {A diagonális}

```



```

v[i]:=tv(A)[i]*r[i,j]
else
  for i:=1 to n do
    begin k:=i; s:=tv(A)[iatlo[i]]*r[i,j];
      for ik:=iatlo[i]+1 to iatlo[i+1]-1 do
        begin dec(k); s:=s+tv(A)[ik]*r[k,j];
          v[k]:=v[k]+tv(A)[ik]*r[i,j];
        end;
      v[i]:=s;
    end;
  end;
end;

var
  LD: tvna;
  x, y: tvn;
  Qk1, Ak1, Bk1: tvqxq;
  s, vnorm, wnrm: real;
  nite, i, j, k, ii: integer;
begin
  for i:=1 to na do LD[i]:=A[i]; {A átmásolása LD-be}
  LDLTfelbont(LD, iatlo, n, i); {A felbontása LD-ben}
  if i<=0 then Hiba('A nem pozitív definit.');
```

{iteráció előkészítése}

```

  nite:=0;
  repeat inc(nite); {—— ITERÁCIÓ KEZDETE ——}
    for j:=1 to q do {B Xk számítása}
      begin Szoroz(x, B, V, iatlo, n, nb, j);
        for i:=1 to n do V[i,j]:=x[i];
      end;
    for j:=1 to q do {A vetületének számítása}
      begin for k:=1 to n do x[k]:=V[k,j];
        LDLTmegold(LD, iatlo, n, x);
        for i:=j to q do
          begin s:=0.0;
            for k:=1 to n do s:=s+V[k,i]*x[k];
            Ak1[i,j]:=s; Ak1[j,i]:=s;
          end;
        for k:=1 to n do V[k,j]:=x[k];
      end;
    for j:=1 to q do {B vetületének számítása}
      begin Szoroz(x, B, V, iatlo, n, nb, j);
        for i:=j to q do
          begin s:=0.0;
            for k:=1 to n do s:=s+V[k,i]*x[k];
            Bk1[i,j]:=s; Bk1[j,i]:=s;
          end;
        end;
    {A qxq-s általánosított sajátérték-feladat megoldása}
    AltJacobi(Ak1, Bk1, q, h, Qk1, 15);
    ii:=q;
    repeat k:=ii-1; ii:=0; {rendezés növekvő sorrendben}
      for i:=1 to k do
        if h[i+1]<h[i] then
          begin ii:=i;
            Cserel(h[i], h[i+1]);
          end;
      for j:=1 to q do
```



```

        cserel(Qk1[j,i],Qk1[j,i+1]);
    end;
until ii=0;
for i:=1 to n do          {A következő altér kiszámítása}
begin
    for j:=1 to q do
        begin s:=0.0;
            for k:=1 to q do s:=s+V[i,k]*Qk1[k,j];
                x[j]:=s;
            end;
        for j:=1 to q do V[i,j]:=x[j];
        end;
    ii:=1; s:=0;          {konvergencia ellenőrzése}
    if nite>1 then
        while (ii<=q) and (s<eps) do
            begin s:=abs((h[ii]-lambda[ii])/h[ii]); inc(ii); end;
            for i:=1 to q do lambda[i]:=h[i];          {lambdaák mentése}
        until (ii>p) or (nite>maxite);          {—— ITERÁCIÓ VÉGE ——}
    if ii>p then
        begin maxite:=nite;
            for i:=1 to p do          {hiba számítása}
                begin Szoroz(x,A,V,iatlo,n,na,i);
                    vnorm:=0.0;
                    for j:=1 to n do vnorm:=vnorm+sqr(x[j]);
                        Szoroz(y,B,V,iatlo,n,nb,i);
                            wnorm:=0.0;
                            for j:=1 to n do
                                wnorm:=wnorm+sqr(x[j]-lambda[i]*y[j]);
                            h[i]:=sqr(wnorm)/sqr(vnorm);
                        end;
                    end
                end;
            end;
end;
end;

```

Annak ellenőrzésére, hogy a (3-37) – (3-42) altér iterációval valóban a kívánt p legkisebb abszolút értékű sajátértéket határoztuk meg, előnyösen felhasználható a következő tulajdonság (lásd Bathe–Wilson [2]):

Az $A - \lambda B$ karakterisztikus mátrix $\tilde{L} D \tilde{L}^T$ alakú felbontásában a D diagonál-mátrix főátlójában a negatív elemek száma megadja a $p_n(\lambda) = \det(A - \lambda B)$ karakterisztikus polinomjában a λ -nál kisebb sajátértékek számát.

Ez a tulajdonság annak a következménye, hogy az $A - \lambda B$ mátrix bal felső sarokminorai Sturm-sorozatot alkotnak (lásd pl. Parlett [9]).


```

procedure AlterEll(      A: tVna;
                        B: tVnb;
                        iatlo: tInl;
                        lambda: tVq;
                        n, na, nb, p, q: integer;
                        h: tVq;
                        eps: real;
                        var nell: integer);
{ Alkalmas az Alteriteracio eljárás eredményeként nyert
lambda sajátértékek ellenőrzésére. A bemenő paraméterek
ismertetése az Alteriteracio eljárásnál található. Az
eljárás eredménye a nell paraméter, amely két szám
különbsége. Az egyik az eljárás során meghatározott mu
eltolásnál kisebb sajátértékek száma, a másik az
eltolással nyert A - mu B mátrix L D LT felbontásában
a D negatív főátlóelemeinek száma. A sajátértékek
akkor fogadhatók el, ha nell értéke 0.}
const
  epsh = 0.01;          {azonos csoportba tartozás kritériuma}
var
  As: tVna;              {segédvektor A-mu*B tárolásához}
  i, j,                  {ciklusváltozók}
  nm,                    {mu-nél kisebb sajátértékek száma}
  nc,                    {csoportok száma}
  ns: integer;           {a megfelelő sajátértékek száma}
  mu: real;              {az eltolás}
  s,                      {segédvektor}
  csh: tVq;              {csoportok felső határai}
  ncs: tIq;              {sajátértékek száma a csoportokban}
begin ns:=0;             {a sajátértékek számának meghatározása}
  for i:=1 to q do
    if h[i]<eps then
      begin inc(ns); ncs[i]:=1; end;
  if ns=0 then
    Hiba('Nincs a hibahatáron belül sajátérték.');
```

for i:=2 to q do
 s[i]:=(lambda[i]-lambda[i-1])/(lambda[i]+lambda[i-1]);
 i:=1; nc:=0; {a csoportok száma, határai, elemeinek száma}
 repeat inc(nc); inc(i); {felső határok keresése}
 while (i<=ns) and (s[i]<epsh) do
 begin inc(ncs[nc]); inc(i); end;
 csh[nc]:=lambda[i-1]*(1+epsh);
 until i>ns;
 while (ns<q) and (s[ns]<epsh) and (h[ns]<=eps) do
 begin csh[nc]:=lambda[ns-1]*(1+epsh);
 inc(ns); inc(ncs[nc]);
 end;
 for j:=nc downto 2 do {eltolás keresése}
 for i:=1 to j-1 do
 ncs[j]:=ncs[j]+ncs[i];
 i:=1; while (i<=nc) and (ncs[i]<ns) do inc(i);
 mu:=csh[i]; nm:=ncs[i]; {mu-nél lévő sajátértékek száma}
 for i:=1 to na do As[i]:=A[i]; {A eltolása mu-vel As-ben}
 if nb<=n then
 for i:=1 to n do


```

begin j:=iatlo[i];
  As[j]:=As[j]-mu*B[i];
end
else
  for i:=1 to na do
    As[i]:=As[i]-mu*B[i];
  LDLTfelbont(As,iatlo,n,nell);           {A-mu*B felbontása}
  nell:=nmu+nell;                         {a különbség számítása}
end;

```

Az altér iterációhoz szükség van a (3-37) egyenletben szereplő X_1 kezdő altér megadására. A jó megválasztás nagyon fontos, pl. ha a kezdővektorok a domináló altérben fekszenek, akkor az iteráció egyetlen lépésben a kívánt sajátvektorokat eredményezi. Egy, a gyakorlatban jól használható eljárást Bathe-Wilson [2] könyve alapján ismertetünk.

```

Procedure KezdoAlter(      A: tvNa;
                          B: tvNb;
                          iatlo: tInl;
                          var X1: tvNxq;
                          n,na,nb,q: integer);
{ Ez az eljárás az altér iterációhoz szükséges
  kezdővektorokat az X1 mátrixban állítja elő, amely az
  Alteriteracio eljárás v paramétereként felhasználható. A
  bemenő paraméterek és a típusok jelentését l. az
  Alteriteracio-nál. }
var
  i,j,ii,nd,l: integer;
  s: real;                      {segédváltozó}
  w: tvN;                      {segédvektor}
begin l:=0;                    {B pozitív elemei, ha B diagonális}
  for i:=1 to n do
    begin
      if nb<=n then              {B diagonális?}
        begin ii:=i; if B[i]>0.0 then inc(l); end
      else
        ii:=iatlo[i];
        X1[i,1]:=B[ii]; w[i]:=B[ii]/A[iatlo[i]];
        for j:=2 to q do X1[i,j]:=0.0;
      end;
      if (nb<=n) and (q>1) then
        Hiba('q nagyobb mint B rangja (B diagonális)');
      nd:=n div q; l:=n;
      for j:=2 to q do
        begin s:=0.0; l:=l-nd;
          for i:=1 to l do {w[i] legnagyobb elemének keresése}
            if w[i]>=s then
              begin s:=w[i]; ii:=i; end;
          for i:=l+1 to n do
            if w[i]>s then
              begin s:=w[i]; ii:=i; end;
          w[ii]:=0.0; X1[ii,j]:=1.0;
        end;
      end;
    end;
end;

```


Az ismertetett vektoriterációs technikák csak akkor gazdaságosak, ha viszonylag kevés sajátérték és sajátvektor meghatározására van szükség. Ha azonban a sajátértékeket már meghatároztuk valamely más módszerrel, pl. a háromatlós mátrixra alkalmazott Givens-féle felezési eljárással, akkor a hozzá tartozó sajátvektorok meghatározására az inverziteráció látszik a legalkalmasabb eljárásnak. (Ez azért van így, mert ha μ spektráleltolásként pl. a λ_q sajátérték közelítését választjuk, akkor

$$|\lambda_q - \mu| = \min_{i=1}^n |\lambda_i - \mu| \ll \min_{i \neq q} |\lambda_i - \mu|,$$

és így a konvergenciatényező elegendően kicsiny ahhoz, hogy a v_q sajátvektort már kevés iterációs lépés után a kívánt pontossággal megkapjuk.)

4. Transzformációs módszerek

Az (1–3) alatti $V^T A V = \Lambda$ egyenlet olyan transzformációk (mátrixok) megkeresését sugallja, amelyeket egymás után alkalmazva az A mátrixra, azt diagonál-mátrixra redukálják. A megfelelő iteráció csak

$$A_{k+1} = W^{-1} A_k W$$

alakú lehet, mert csak a *hasonlósági transzformáció* biztosítja, hogy az A_{k+1} mátrixnak ugyanazok a sajátértékei mint A_k -nak (lásd az 1.2. tételt). Különösen előnyös, ha *ortogonális* hasonlósági transzformációkat alkalmazunk, mert ekkor $W^{-1} = W^T$ miatt a számítások nemcsak egyszerűbbé válnak, hanem a *stabilitásuk* is biztosítva van.

Jacobi-diagonalizálás

A Jacobi-iteráció egy egyszerű és nagyon megbízható transzformációs módszer, amely az összes sajátértéket és az ortonormált sajátvektorrendszert egyszerre szolgáltatja. Ha a főátlón kívüli elemek abszolút értéke kicsi, akkor hatékony is.

Az iteráció kifejezhető az

$$A_{k+1} = P_k^T A_k P_k, \quad k = 1, 2, \dots, \quad A_1 = A \quad (4-1)$$

képlettel, ahol $k \rightarrow \infty$ esetén $A_k \rightarrow \Lambda$ és

$$P_k = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & \dots & \cos \varphi & \dots & \sin \varphi & \dots & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & \dots & -\sin \varphi & \dots & \cos \varphi & \dots & \\ & & & & & & 1 & \dots & \\ & & & & & & & \ddots & & \\ & & & & & & & & & 1 \end{bmatrix} \begin{matrix} \text{p-edik} \\ \text{oszlop} \end{matrix} \quad \begin{matrix} \text{q-adik} \\ \text{oszlop} \end{matrix} \quad \begin{matrix} \text{p-edik sor} \\ \text{q-adik sor} \end{matrix} \quad (4-2)$$

alakú ún. *Jacobi-forgatómátrixok*, amelyeknek csak négy eleme különbözik az egység mátrixtól.

A $\mathbf{P}_k^T \mathbf{A}_k$ szorzat $\mathbf{A}_k$ -nak csak a p -edik és q -edik sorát változtatja meg, míg $\mathbf{A}_k \mathbf{P}_k$ csak a p -edik és q -edik oszlopokat módosítja. Tehát a (4-1) transzformáció az (rögzített k esetén) $\mathbf{A}_k$ mátrixnak csak a következő sémán feltüntetett elemeit változtatja meg.

$$A_{k+1} = \begin{bmatrix} & & a_{1p}^{k+1} & & a_{1q}^{k+1} & & \\ & & \vdots & & \vdots & & \\ & & & & & & \\ a_{p1}^{k+1} & \dots & a_{pp}^{k+1} & \dots & a_{pq}^{k+1} & \dots & a_{pn}^{k+1} \\ & & \vdots & & \vdots & & \\ a_{q1}^{k+1} & \dots & a_{qp}^{k+1} & \dots & a_{qq}^{k+1} & \dots & a_{qn}^{k+1} \\ & & \vdots & & \vdots & & \\ & & & & & & \\ & & a_{np}^{k+1} & & a_{nq}^{k+1} & & \end{bmatrix} \quad (4-3)$$

Ha (rögzített k esetén) elvégezzük a (4-1) alatti szorzásokat és figyelembe vesszük, hogy A_k szimmetrikus, akkor a következő képleteket nyerjük:

$$a_{ip}^{k+1} = a_{ip}^k \cos \varphi - a_{iq}^k \sin \varphi$$

$$a_{iq}^{k+1} = a_{iq}^k \cos \varphi + a_{ip}^k \sin \varphi \quad i \neq p, i \neq q \quad (4-4)$$

$$a_{pp}^{k+1} = a_{pp}^k \cos^2 \varphi + a_{qq}^k \sin^2 \varphi - 2 a_{pq}^k \sin \varphi \cos \varphi \quad (4-5)$$

$$a_{qq}^{k+1} = a_{pp}^k \sin^2 \varphi + a_{qq}^k \cos^2 \varphi + 2 a_{pq}^k \sin \varphi \cos \varphi \quad (4-6)$$

$$a_{pq}^{k+1} = a_{pq}^k (\cos^2 \varphi - \sin^2 \varphi) + (a_{pp}^k - a_{qq}^k) \sin \varphi \cos \varphi \quad (4-7)$$

A φ szöget abból a követelményből határozzuk meg, hogy a transzformáció nullává tegye az A_k mátrix $a_{pq}^k = a_{qp}^k$ indexű elemét. Ekkor a (4-7) egyenletből $a_{pq}^{k+1} = 0$ miatt a φ szögre

$$\theta \equiv \cotg 2\varphi = \frac{\cos^2 \varphi - \sin^2 \varphi}{2 \sin \varphi \cos \varphi} = \frac{a_{qq}^k - a_{pp}^k}{2 a_{pq}^k} \quad (4-8)$$

adódik. A θ definíciója átírható

$$tg^2 \varphi + 2 \theta tg \varphi - 1 = 0 \quad (4-9)$$

alakra. Ennek az egyenletnek a kisebbik gyöke a $|\varphi| \leq \frac{\pi}{4}$ szöggel való forgatásnak felel meg; ez a választás a számítási pontosság szempontjából kedvezőbb.

A (4-9) egyenlet megoldása kifejezhető

$$tg \varphi = -\theta \pm (\theta^2 + 1)^{1/2} = \frac{1}{\theta \pm (\theta^2 + 1)^{1/2}}$$

tört alakban és így a kisebbik gyökre azt kapjuk, hogy

$$tg \varphi = \frac{\text{sign}(\theta)}{|\theta| + (\theta^2 + 1)^{1/2}}. \quad (4-10)$$

Ha a θ annyira nagy, hogy θ^2 a számítógép túlsordulását okozná, akkor a $tg \varphi = 1 / (2 \theta)$ értékkel folytatjuk a számítást.

A $tg \varphi$ ismeretében (a φ szög kiszámítása nélkül) közvetlenül meghatározhatók a

$$\cos \varphi = \frac{1}{(tg^2 \varphi + 1)^{1/2}} \quad (4-11)$$

és a

$$\sin \varphi = tg \varphi \cos \varphi \quad (4-12)$$

függvényértékek.

Hogy a kerekítési hibák hatását a minimumra csökkentjük, a (4-4)–(4-7) egyenleteket célszerű átírni más alakra. A (4-7) egyenletet az

$$a_{pq}^{k+1} = 0 \quad (4-13)$$

egyenlettel helyettesítjük. A többi egyenlet átírásánál azt az elvet követjük, hogy az új mennyiséget a régi mennyiség és egy kis korrekció összegeként fejezzük ki. Így pl. ha a (4-7) és a (4-13) egyenlethől kifejezzük az a_{qq}^k együtthatót, akkor azt kiküszöbölhetjük a (4-5) egyenlethől és így az

$$a_{pp}^{k+1} = a_{pp}^k - a_{pq}^k \operatorname{tg} \varphi \quad (4-14)$$

egyenletet nyerjük. Hasonlóképpen kapjuk, hogy

$$a_{qq}^{k+1} = a_{qq}^k + a_{pq}^k \operatorname{tg} \varphi, \quad (4-15)$$

$$a_{ip}^{k+1} = a_{ip}^k - (a_{iq}^k + \tau a_{ip}^k) \sin \varphi, \quad (4-16)$$

$$a_{iq}^{k+1} = a_{iq}^k + (a_{ip}^k - \tau a_{iq}^k) \sin \varphi, \quad (4-17)$$

ahol $\tau = (\operatorname{tg} \varphi / 2)$ mennyiséget a

$$\tau \equiv \frac{\sin \varphi}{1 + \cos \varphi} \quad (4-18)$$

összefüggéssel értelmezzük.

A Jacobi-diagonalizálás konvergenciája jól követhető az

$$S_k = \sum_{i \neq j} \left(a_{ij}^k \right)^2 \quad (4-19)$$

főátlón kívüli elemek négyzetösszegének alakulásával. A (4-4)–(4-7) összefüggésekből következik, hogy

$$S_{k+1} = S_k - 2 \left(a_{pq}^k \right)^2. \quad (4-20)$$

(Minthogy az $A_{k+1} = P_k^T A_k P_k$ transzformáció ortogonális, $||A_{k+1}||_E^2 = ||A_k||_E^2 = \sum_{i,j=1}^n \left(a_{ij}^k \right)^2$ és így a főátlóelemek négyzetösszege a $2 \left(a_{pq}^k \right)^2$ értékkel nő.) Tehát nem-negatív számok S_k , $k = 1, 2, \dots$ sorozata, mely monoton csökkenő, ezért konvergens. Mivel az a_{pq}^k nullázandó elemként bármely, főátlón kívüli elem választható, a sorozat zérushoz tartónak tehető.

Mivel $k \rightarrow \infty$ esetén $A_k \rightarrow \Lambda$, ahol Λ az A sajátértékeiből alkotott diagonálmátrix (hiszen hasonlósági transzformációkat alkalmaztunk), a

$$\Lambda = V^T A V, \quad V = P_1 P_2 P_3 \dots \quad (4-21)$$

összefüggésekből V ortogonalitása miatt $AV = V\Lambda$ következik, vagyis a sajátvektorok a V mátrix oszlopai. Így a sajátvektorok a

$$V_{k+1} = V_k P_k, \quad k = 1, 2, \dots, \quad V_1 = I \quad (4-22)$$

összefüggéssel állíthatók elő, vagyis részletesebben

$$v_{ij}^{k+1} = v_{ij}^k \quad (i \neq p, i \neq q),$$

$$v_{ip}^{k+1} = v_{ip}^k \cos \varphi - v_{iq}^k \sin \varphi,$$

$$v_{iq}^{k+1} = v_{ip}^k \sin \varphi + v_{iq}^k \cos \varphi.$$

Hogy a kerekítési hibákat a minimumra csökkentsük, ezeket az összefüggéseket a τ mennyiség bevezetésével ugyanúgy átírhatjuk, mint ahogy a (4-16) és a (4-17) összefüggéseket nyertük.

Végül stratégiát kell adni az elemek nullázásának sorrendjére vonatkozóan. Jacobi az 1846-ból származó algoritmusában minden egyes lépésben kikereste a legnagyobb abszolút értékű, főátlón kívüli elemet és azt nullázta. Kézi számolásnál ez egy nagyon is ésszerű stratégia, számítógép alkalmazása esetén azonban hátrányos, mivel minden egyes Jacobi-rotációt a keresés maga n helyett egy n^2 rendű eljárásra változtatja.

Céljainknak jobban megfelel az ún. *ciklikus Jacobi-módszer*, amelynél az elemek nullázása egy előre megadott sorrend szerint megy végbe. Ilyen egyszerű sorrend pl. a sorfolytonos haladás, amikor az indexeket a következőképpen változtatjuk: (1, 2), (1, 3), ..., (1, n) majd (2, 3), (2, 4) és így tovább. Egyszeres sajátértékeket feltételezve belátható, hogy mindkét stratégia általában másodfokú konvergenciát biztosít. Egy ilyen ciklust, amely $n(n-1)/2$ Jacobi-rotációból áll, *turnusnak* nevezzük.

A következő program az első három turnusban a megfelelő rotációt csak akkor hajtja végre, ha

$$|a_{pq}^k| > \frac{1}{5n^2} \sum_{i < j} |a_{ij}^k|. \quad (4-23)$$

Négy turnus után a program az a_{pq}^k elemet akkor tekinti zérusnak, ha $|a_{pq}^k|$ sokkal kisebb, mint az $|a_{pp}^k|$ és az $|a_{qq}^k|$ értékek valamelyike, és ekkor kihagyja a megfelelő rotációt. Az erre vonatkozó kritérium $|a_{pq}^k| < 10^{-(D+2)} |a_{pp}^k|$, ill. $|a_{pq}^k| < 10^{-(D+2)} |a_{qq}^k|$, ahol D a számítógép értékes decimális jegyeinek a száma. (Ezek a programfinomítások először a Numerical Recipes-ban szereplő Jacobi-eljárásban szerepelnek.)


```

procedure Jacobi(      var A: tVnxn;
                      n: integer;
                      var d: tVn;
                      var V: tVnxn;
                      var nrot: integer);

```

{ Az eljárás a valós szimmetrikus n -ed rendű A mátrix összes sajátértékének és sajátvektorának meghatározására szolgál. Az A mátrixot a $tVnxn$ típusú kétindexes A tömbben tároljuk. Az A tömb főátló feletti elemeit az eljárás módosítja. A tVn típusú d vektor az A mátrix sajátértékeit tartalmazza. Az 1-re normált sajátvektorokat az A -val azonos módon a V tömbben tároljuk. $Nrot$ a szükséges Jacobi-rotációk számát jelzi.

A Jacobi eljárást használó programban definiálni kell a következő globális típusokat

```

tVn      =array [1..nmax] of real;
tVnxn    =array [1..nmax,1..nmax] of real;

```

ahol $nmax$ az n legnagyobb lehetséges értéke.

Ha az adott Pascal implementáció nem rendelkezik a `double` típussal, akkor definiálni kell azt a `double=real;` utasítással. }

```

var
  i, j, iq, ip: integer;
  tresh, theta, tau, t, s, h, g, c: real;
  sm: double;
  b, z: tVn;

begin
  for i:=1 to n do                {egységmátrix előállítása v-ben}
    begin v[i,i]:=1;
      for j:=1 to i-1 do
        begin v[i,j]:=0; v[j,i]:=0; end;
      end;
    for i:=1 to n do
      begin b[i]:=a[i,i];
        d[i]:=b[i];                {b és d A diagonáleinelei}
        z[i]:=0;                  {z-ben az a[p,q]*tau(fi) lesz l. (4-14)}
      end;
    nrot:=0;
    for i:=1 to 50 do
      begin sm:=0;
        for ip:=1 to n-1 do
          for iq:=ip+1 to n do
            sm:=sm+abs(a[ip,iq]);
          if sm=0 then
            exit;                  {normál kilépés}
          if i<4 then tresh:=0.2*sm/sqr(n)
          else tresh:=0;
          for ip:=1 to n-1 do
            for iq:=ip+1 to n do
              begin g:=100*abs(a[ip,iq]);
                if (i>4) and (abs(d[ip]+g)=abs(d[iq]))

```



```

        and (abs(d[iq]+g)=abs(d[iq])) then
a[ip,iq]:=0
else if abs(a[ip,iq])>tresh then
begin h:=d[iq]-d[ip];
  if (abs(h)+g)=abs(h) then
    t:=a[ip,iq]/h
  else
    begin theta:=0.5*h/a[ip,iq];
      t:=1/(abs(theta)+sqrt(1+sqr(theta)));
      if theta<0 then
        t:=- t;
    end;
    c:=1/sqrt(1+sqr(t)); s:=t*c;
    tau:=s/(1+c); h:=t*a[ip,iq];
    z[ip]:=z[ip]-h; z[iq]:=z[iq]+h;
    d[ip]:=d[ip]-h; d[iq]:=d[iq]+h;
    a[ip,iq]:=0;
    for j:=1 to ip-1 do
      begin g:=a[j,ip]; h:=a[j,iq];
        a[j,ip]:=g-s*(h+g*tau);
        a[j,iq]:=h+s*(g-h*tau);
      end;
    for j:=ip+1 to iq-1 do
      begin g:=a[ip,j]; h:=a[j,iq];
        a[ip,j]:=g-s*(h+g*tau);
        a[j,iq]:=h+s*(g-h*tau);
      end;
    for j:=iq+1 to n do
      begin g:=a[ip,j]; h:=a[iq,j];
        a[ip,j]:=g-s*(h+g*tau);
        a[iq,j]:=h+s*(g-h*tau);
      end;
    for j:=1 to n do
      begin g:=v[j,ip]; h:=v[j,iq];
        v[j,ip]:=g-s*(h+g*tau);
        v[j,iq]:=h+s*(g-h*tau);
      end;
    inc(nrot);
  end;
end;
for ip:=1 to n do
  begin b[ip]:=b[ip]+z[ip];
    d[ip]:=b[ip]; z[ip]:=0;
  end;
end;
Hiba('Az iterációk száma meghaladta az 50-et!');
end;

```


A Jacobi-eljárás a sajátértékeket nem nagyság szerinti sorrendben szolgáltatja. A következő rutin alkalmas a Jacobi-eljárás vagy e fejezet további eljárásai által kiszámított sajátértékek sorba rendezésére nagyságuk csökkenő sorrendjében.

```
procedure Rendez( var lambda: tvN;
                  var V: tvNxn;
                  n: integer);
```

```
{ Tegyük fel, hogy a Jacobi, vagy a továbbiakban szereplő
ImplicitQL eljárás eredményeként nyert lambda vektor a
sajátértékeket, a V mátrix oszlopai pedig a megfelelő
sajátvektorokat tartalmazza. Ez az eljárás a sajátértékeket
csökkenő sorrendbe állítja és ennek megfelelően átrendezi a
V oszlopait. A típusdefiníciókat l. pl. a Jacobi
eljárásban.}
```

```
procedure Cserel(var a,b: real);
var s: real;
begin s:=a; a:=b; b:=s; end;
```

```
var
  i, j, cserek, meddig: integer;

begin cserek:=n; {rendezés buborék módszerrel}
  repeat meddig:=cserek-1; cserek:=0;
    for i:=1 to meddig do
      if lambda[i+1]>lambda[i] then
        begin cserek:=i; {cserek elvégzése}
          cserel(lambda[i],lambda[i+1]);
          for j:=1 to n do
            cserel(V[j,i],V[j,i+1]);
          end;
        until cserek=0; {vége, ha nem volt csere}
    end;
```

Általánosított Jacobi-módszer. Ha a Jacobi-módszert a (3-31) alatti $A v = \lambda B v$ általánosított sajátérték-feladatra kívánjuk alkalmazni, akkor elsősorban az az eljárás kínálkozik, mely szerint az általánosított feladatot visszavezetjük a (3-33) speciális sajátérték-feladatra, és a Jacobi-módszert az $L^{-1} A L^{-T}$ mátrixra alkalmazzuk.

A Jacobi-módszer azonban speciális sajátérték-feladatra való visszavezetés nélkül is közvetlenül alkalmazható az A, B mátrixok egyidejű diagonalizálására. Ez esetben a (4-2) alatti P_k mátrixot felváltja a

$$P_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & \ddots \\ \dots & & & \gamma & \dots & 1 & \dots \\ & & & & & & \ddots \\ & & & & & & & 1 \end{bmatrix} \begin{matrix} \text{p-edik} \\ \text{oszlop} \end{matrix} \quad \begin{matrix} \text{q-adik} \\ \text{oszlop} \end{matrix}$$

p-edik sor

q-adik sor

mátrix, ahol az α és γ állandókat abból a követelményből határozzuk meg, hogy a P_k transzformáció egyidejűleg tegye nullává az $A_{k+1} = P_k^T A_k P_k$, $B_{k+1} = P_k^T B_k P_k$ mátrixok p, q indexű elemeit. A részletekre vonatkozóan lásd pl. Bathe-Wilson [2] könyvét.

```
procedure AltJacobi(var A,B: tvqxq;
                    n: integer;
                    var d: tvq;
                    var V: tvqxq;
                    maxiter: integer);
```

{ Ez az eljárás alkalmas az $A v = \lambda B v$ általánosított sajátérték-feladat összes sajátértékének és sajátvektorának meghatározására, ahol A és B n -ed rendű valós szimmetrikus pozitív definit mátrixok. Az A és B mátrixot a tvqxq globális típusú qmax * qmax elemű kétindexes A és B tömbben tároljuk. Az A és B tömb főátló feletti elemei az eljárás során módosulnak. A tvq típusú qmax elemű d tömb első q eleme a sajátértékeket tartalmazza. A B -ortonormált sajátvektorokat tartalmazó V mátrixot, mely az A -val teljesen azonos módon van tárolva, a V tömb tartalmazza. A maxiter bemenő paraméter az iterációk maximális számát mutatja.

Az AltJacobi eljárást használó programban definiálni kell a következő globális típusokat

```
tvq      = array [1..qmax] of real;
tvqxq    = array [1..qmax,1..qmax] of real;
```

ahol qmax az n legnagyobb lehetséges értéke.

Ha az adott Pascal implementáció nem rendelkezik a double típussal, akkor definiálni kell azt a

```
double = real;
utasítással. }
```



```

procedure forgat(var a: tvqxq;
  i1,j1,i2,j2: integer;
  alfa,gamma: real);
var s1,s2: real;
begin s1:=a[i1,j1]; s2:=a[i2,j2];
  a[i1,j1]:=s1+gamma*s2; a[i2,j2]:=alfa*s1+s2;
end;

var
  iter,                                {iteráció száma}
  i, p, q: integer;                   {ciklusváltozók}
  alfa, gamma,                         {a forgatómátrix elemei}
  s, ss, s1, s2: real;                {segédváltozók}
  vege: boolean; {az iterációk végét jelző logikai változó}
begin iter:=0; vege:=false;
  for p:=1 to n do                    {egységmátrix előállítása v-ben}
    begin v[p,p] := 1;
      for q:=1 to p-1 do
        begin v[p,q]:=0; v[q,p]:=0; end;
      end;
    repeat s1:=0; s2:=0; inc(iter);
      for q:=2 to n do
        for p:=1 to q-1 do
          begin s1:=s1+abs(a[q,p]);
            s2:=s2+abs(b[q,p]);
          end;
        if (s1<1.0e-7) and (s2<1.0e-7) or (iter>maxiter) then
          vege:=true                    {Kilépés}
        else
          for q:=2 to n do
            for p:=1 to q-1 do
              begin
                s1:=a[p,p]*b[q,p]-a[q,p]*b[p,p];
                s2:=a[q,q]*b[q,p]-a[q,p]*b[q,q];
                s:=a[q,q]*b[p,p]-a[p,p]*b[q,q];
                ss:=0.25*(sqr(s)+4*s1*s2);
                if ss<0 then
                  Hiba('A, ill. B nem pozitív definit.');
```



```

end;
s1:=a[p,p]; s2:=a[q,q];
a[p,p]:=s1+2*gamma*a[q,p]+sqr(gamma)*s2;
a[q,q]:=sqr(alfa)*s1+2*alfa*a[q,p]+s2;
s1:=b[p,p]; s2:=b[q,q];
b[p,p]:=s1+2*gamma*b[q,p]+sqr(gamma)*s2;
b[q,q]:=sqr(alfa)*s1+2*alfa*b[q,p]+s2;
a[q,p]:=0; b[q,p]:=0;
for i:=1 to n do
    forgat(v,i,p,i,q,alfa,gamma);
end;
until vege;
if iter>maxiter then
    Hiba('Az iterációk száma meghaladta az 50-et!')
else
    for p:=1 to n do
        begin d[p]:=a[p,p]/b[p,p]; s:=1/sqrt(b[p,p]);
        for q:=1 to n do
            v[q,p]:=v[q,p]*s;
        end;
    end;
end;

```

Ha valós szimmetrikus mátrix összes sajátértékét és sajátvektorát kívánjuk meghatározni, akkor a következő két módszerből kombinált eljárás látszik e célra a legelőnyösebbnek.

1. A *Householder-transzformációk* alkalmazásával az A mátrixot szimmetrikus háromatlós mátrixra redukáljuk.
2. *QR-iterációval* kiszámítjuk a háromatlós mátrix sajátértékeit, ha szükséges akkor a sajátvektorait is, majd azokat az A mátrix sajátvektoraira transzformáljuk.

Householder-módszer

A háromatlós mátrixra való redukció $n-2$ ortogonális hasonlósági transzformáció alkalmazásával megy végbe, ahol n az A rendszáma. Az eljárás kifejezhető

$$A_{k+1} = P_k^T A_k P_k, \quad k = 1, \dots, n-2, \quad A_1 = A \quad (4-24)$$

alakban, ahol

$$P_k = I - 2 w_k w_k^T, \quad w_k^T w_k = 1. \quad (4-25)$$

Tekintsük a $k = 1$ esetet, amely az egész eljárásra jellemző. Ha az A_1 mátrixot

$$A_1 = \left[\begin{array}{c|c} a_{11} & a_1^T \\ \hline a_1 & A_{11} \end{array} \right]$$

szerint particionáljuk, ahol A_{11} egy $(n-1)$ -ed rendű mátrix, és ha w_1 egy n elemű

$$\mathbf{w}_1 = \begin{bmatrix} 0 \\ \bar{\mathbf{w}}_1 \end{bmatrix}$$

alakú vektor, amelynek zérus az első eleme, akkor a $\mathbf{P}_1$ ortogonális mátrix felírható

$$\mathbf{P}_1 = \begin{bmatrix} 1 & | & \mathbf{0}^T \\ \hline \mathbf{0} & | & \bar{\mathbf{P}}_1 \end{bmatrix}, \quad \bar{\mathbf{P}}_1 = \mathbf{I}_{n-1} - 2 \bar{\mathbf{w}}_1 \bar{\mathbf{w}}_1^T$$

particionált alakban és

$$\mathbf{P}_1^T \mathbf{A}_1 \mathbf{P}_1 = \begin{bmatrix} a_{11} & | & \mathbf{a}_1^T \bar{\mathbf{P}}_1 \\ \hline \bar{\mathbf{P}}_1^T \mathbf{a}_1 & | & \bar{\mathbf{P}}_1^T \mathbf{A}_{11} \bar{\mathbf{P}}_1 \end{bmatrix} \quad (4-26)$$

következik.

Mivel a (4-26) mátrix első oszlopáról feltételezzük, hogy egy háromatlós mátrix első oszlopa, a $\bar{\mathbf{P}}_1^T \mathbf{a}_1$ vektortól megköveteljük az $s \mathbf{e}_1$ alakot, ahol $\mathbf{e}_1$ az $(n-1)$ -ed rendű egységmátrix első oszlopa és így az s szám a

$$||\bar{\mathbf{P}}_1^T \mathbf{a}_1|| = ||s \mathbf{e}_1||$$

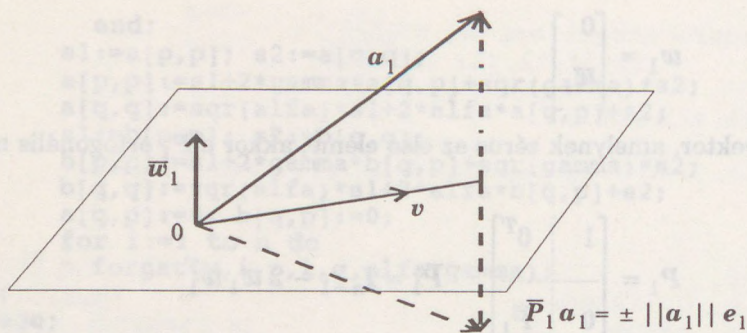
feltételből határozható meg. Minthogy a $\bar{\mathbf{P}}_1^T$ ortogonális transzformáció nem változtatja meg az $\mathbf{a}_1$ vektor euklideszi normáját, $||\mathbf{a}_1|| = |s| ||\mathbf{e}_1||$, vagyis $s = \pm ||\mathbf{a}_1||$ következik.

A $\bar{\mathbf{w}}_1$, $||\bar{\mathbf{w}}_1|| = 1$ vektor szintén a $\bar{\mathbf{P}}_1^T \mathbf{a}_1 = s \mathbf{e}_1$, azaz a

$$(\mathbf{I} - 2 \bar{\mathbf{w}}_1 \bar{\mathbf{w}}_1^T) \mathbf{a}_1 = \pm ||\mathbf{a}_1|| \mathbf{e}_1 \quad (4-27)$$

egyenletből határozható meg, ahol a + és a - előjelek közül majd azt választjuk, amely kisebb kerekítési hibát okoz.

Geometriai szemlélettel, az $\mathbf{a}_1$ vektort tükrözzük a $\bar{\mathbf{w}}_1$ normálisú hipersíkra, ahol a $\bar{\mathbf{w}}_1$ vektort úgy határozzuk meg, hogy a $\bar{\mathbf{P}}_1^T \mathbf{a}_1$ képvektornak csak az első eleme különbözik a nullától. (Valóban, mint minden vektor, így $\mathbf{a}_1$ is felírható $\mathbf{a}_1 = c_1 \mathbf{v} + c_2 \bar{\mathbf{w}}_1$ alakban, ahol $\mathbf{v}$ a hipersík egy eleme. A $\bar{\mathbf{w}}_1^T \mathbf{v} = 0$ ortogonalitási feltétel miatt azt kapjuk, hogy $\bar{\mathbf{P}}_1 \mathbf{a}_1 = (\mathbf{I} - 2 \bar{\mathbf{w}}_1 \bar{\mathbf{w}}_1^T) (c_1 \mathbf{v} + c_2 \bar{\mathbf{w}}_1) = c_1 \mathbf{v} + c_2 \bar{\mathbf{w}}_1 - 2 c_2 \bar{\mathbf{w}}_1 \bar{\mathbf{w}}_1^T \bar{\mathbf{w}}_1 = c_1 \mathbf{v} - c_2 \bar{\mathbf{w}}_1$ (lásd a 4.1. ábrát).)



4.1. ábra. Householder tükrözés

Csak a $\bar{w}_1$ vektort kell még meghatározni a (4-27) egyenletből. Ez az egyenlet átírható

$$(2 \bar{w}_1^T a_1) \bar{w}_1 = a_1 \mp ||a_1|| e_1$$

alakra, amely azt fejezi ki, hogy a $\bar{w}_1$ vektor számmal való szorzata egyenlő az $a_1 \pm ||a_1|| e_1$ vektorral. Ezért és a $||\bar{w}_1|| = 1$ feltétel miatt

$$\bar{w}_1 = \frac{a_1 \pm ||a_1|| e_1}{||a_1 \pm ||a_1|| e_1||}$$

írható. Az

$$a_1 \pm ||a_1|| e_1 = \begin{bmatrix} a_{21} \pm ||a_1|| \\ a_{31} \\ \vdots \\ a_{n1} \end{bmatrix}$$

vektor részletes felírásából a nevezőre

$$||a_1 \pm ||a_1|| e_1||^2 = 2 ||a_1||^2 \pm 2 a_{21} ||a_1||$$

adódik, és az osztás kisebb hibát okoz, ha az előjelet a

$$\bar{w}_1 = \frac{a_1 + \text{sign}(a_{21}) ||a_1|| e_1}{\left(2 ||a_1||^2 + 2 |a_{21}| ||a_1||\right)^{1/2}} \quad (4-28)$$

összefüggésnek megfelelően választjuk meg.

A (4-26) transzformáció végrehajtásához nem kell ténylegesen elvégezni a $P_1^T A P_1$ mátrixszorzásokat. A

$$p_1 \equiv 2 A_1 w_1 \quad (4-29)$$

jelöléssel

$$A_1 P_1 = A_1 (I - 2 w_1 w_1^T) = A_1 - p_1 w_1^T$$

$$A_2 = P_1 A P_1 = A_1 - p_1 w_1^T - w_1 p_1^T + 2 K_1 w_1 w_1^T$$

adódik, ahol

$$K_1 = w_1^T p_1.$$

Alkalmazva a

$$q_1 \equiv p_1 - K_1 w_1 \quad (4-30)$$

jelölést, a gyakorlati számítások elvégzésére alkalmas

$$A_2 = A_1 - q_1 w_1^T - w_1 q_1^T \quad (4-31)$$

formulát nyerjük.

A redukció k -adik lépése, $k = 2, \dots, n-2$, az eddigiek ismétlése egy n elemű

$$w_k = \begin{bmatrix} 0 \\ \bar{w}_k \end{bmatrix} \quad (4-32)$$

vektorral, ahol 0 a k elemű zérusvektort jelöli.

Összhangban a „Numerical Recipes” gyűjteményben szereplő Householder-redukcióval, a következő eljárás is az A mátrix n -edik oszlopával kezdi a redukciót, tehát nem az első oszloppal, mint ez a fenti ismertetésben szerepel. Ennek megfelelően a redukció k -adik lépésében a (4-32) alatti vektor szerepét egy olyan n elemű vektor tölti be, amelynek nem az első k eleme, hanem az utolsó k eleme egyenlő zérussal.

Ha a háromatlós mátrix sajátvektorait már meghatároztuk, pl. a következő szakaszban szereplő eljárás alkalmazásával, és azokat megszorozzuk a

$$Q = P_1 P_2 \dots P_{n-2} \quad (4-33)$$

mátrixszal, akkor ($Q^T A Q x = \lambda x$, vagyis $A Q x = \lambda Q x$ miatt) megkapjuk az A sajátvektorait. Ezért miután már meghatároztuk az összes P_k mátrixot, a

$$Q_{n-2} = P_{n-2}$$

$$Q_j = P_j Q_{j+1}, \quad j = n-3, \dots, 1, \quad (4-34)$$

$$Q = Q_1$$

rekurzióval képezzük a Q mátrixot.

A következő eljárás bemenő paramétere az n -ed rendű valós szimmetrikus A mátrix, amelyet egy np -ed rendű tömbben tárolunk. Kimenetnél A az ortogonális Q mátrix elemeit tartalmazza. A háromátlós mátrix főátlóelemeit a d vektor tartalmazza, a főátlóval párhuzamos szomszédos elemeket pedig az e vektor $e(2), \dots, e(n)$ indexű elemei tartalmazzák és $e(1) = 0$.

Amennyiben a redukció k -adik lépésében, $k = 1, 2, \dots, n-2$ az

$$||a_i||^2 = a_{i1}^2 + a_{i2}^2 + \dots + a_{i,i-1}^2, \quad i = n-k+1$$

mennyiség zérus vagy „kicsi”, akkor a megfelelő transzformáció elhagyható. Ezt a mennyiséget az eljárás akkor tekinti kicsinek, ha

$$\varepsilon = |a_{i1}| + |a_{i2}| + \dots + |a_{i,i-1}| = 0$$

a gépi pontosság vonatkozásában. Egyébként az a_{ij} elemek helyett az a_{ij}/ε skálázott elemekkel folytatjuk a számítást. (A Householder-transzformációkat, azaz a w vektorokat a skálázás nem változtatja meg.) Ezt a Numerical Recipes-ban szereplő kritériumot a következő Red3Atlos eljárásban mi is alkalmaztuk.

Megjegyezzük, hogy ha a mátrix elemei több nagyságrendben különböznek egymástól, akkor amennyiben ez lehetséges, a mátrixot célszerű úgy permutálni, hogy a kisebb elemek a bal felső sarokba kerüljenek. Erre azért van szükség, mert a redukciót a mátrix jobb alsó sarkával kezdjük, és így elkerülhető a jelentős kerekítési hibákat okozható kis és nagy elemek együttes szerepeltetése.

A Red3Atlos eljárást a következőkben ismertetendő ImplicitQL eljárással való együttes alkalmazásra szánták. Az ImplicitQL eljárás egy háromátlós valós szimmetrikus mátrix valamennyi sajátértékének és sajátvektorának meghatározására szolgál. Ha a Red3Atlos eljárást az ImplicitQL eljárással kombináljuk, akkor úgy tűnik, hogy ez a leghatékonyabb technika egy szimmetrikus valós mátrix összes sajátértékének és sajátvektorának (esetleg csak a sajátértékeinek) meghatározására.

```
procedure Red3Atlos(      var A: tvn $\times$ n;
                          n: integer;
                          var d,e: tvn);
```

{ Az eljárás alkalmas a valós szimmetrikus, n -ed rendű A mátrix háromátlós alakra redukálására. Az A mátrixot a $tvn \times n$ globális típusú $nmax \times nmax$ elemű kétindexes A tömbben tároljuk. Kimenetkor az A tömb a (4-33)-mal definiált Q tömböt tartalmazza.

Eredményként az n elemű d vektor a háromátlós mátrix főátlóelemeit, az e vektor pedig a főátlóval párhuzamos szomszédos elemeket tartalmazza a másodiktól az n -edik elemig. d és e tvn típusú vektor.

A Red3Atlos eljárást használó programban definiálni kell a következő globális típusokat

```
tvn      = array [1..nmax] of real;
tvn $\times$ n    = array [1..nmax,1..nmax] of real;
```


ahol $n_{\max}$ az n legnagyobb lehetséges értéke.

Ha az adott Pascal implementáció nem rendelkezik a `double` típussal, akkor definiálni kell azt a `double = real;` utasítással. }

```
function sign(a,b: real): real;
begin
  if b>=0 then sign:=abs(a) else sign:=-abs(a);
end;

var
  l, k, j, i: integer;
  hh, h, g, f: real;
  scale: double;

begin
  for i:=n downto 2 do
    begin l:=i-1; h:=0.0; scale:=0.0;
      if l>1 then
        begin
          for k:=1 to l do
            scale:=scale+abs(a[i,k]);
          if scale=0.0 then
            e[i]:=a[i,l]
          else
            begin
              for k:=1 to l do
                begin
                  a[i,k]:=a[i,k]/scale;
                  h:=h+sqr(a[i,k]);
                end;
              f:=a[i,l]; g:=-sign(sqrt(h),f);
              e[i]:=scale*g;
              h:=h-f*g; a[i,l]:=f-g; f:=0;
              for j:=1 to l do
                begin a[j,i]:=a[i,j]/h; g:=0;
                  for k:=1 to j do
                    g:=g+a[j,k]*a[i,k];
                  if l>j then
                    for k:=j+1 to l do
                      g:=g+a[k,j]*a[i,k];
                    e[j]:=g/h; f:=f+e[j]*a[i,j];
                  end;
                  hh:=f/(h+h);
                  for j:=1 to l do
                    begin f:=a[i,j]; g:=e[j]-hh*f;
                      e[j]:=g;
                      for k:=1 to j do
                        a[j,k]:=a[j,k]-f*e[k]-g*a[i,k];
                      end;
                    end;
                  end
                else
                  e[i]:=a[i,l];
```



```

d[i]:=h;
end;
d[1]:=0; e[1]:=0;
for i:=1 to n do
begin l:=i-1;
  if d[i]<>0 then
    for j:=1 to l do
      begin g:=0;
        for k:=1 to l do
          g:=g+a[i,k]*a[k,j];
        for k:=1 to l do
          a[k,j]:=a[k,j]-g*a[k,i];
        end;
      d[i]:=a[i,i]; a[i,i]:=1;
      if l>=1 then
        for j:=1 to l do
          begin a[i,j]:=0; a[j,i]:=0; end;
        end;
      end;
end;
end;

```

Ha a szimmetrikus valós mátrixot már redukáltuk háromatlós mátrixra, akkor a sajátértékek pl. a 2. szakaszban ismertetett *Givens-féle felezési eljárás*s, a sajátvektorok pedig inverziterációval határozhatók meg. Ez az eljárás viszont csak akkor hatékony, ha viszonylag kevés sajátértéket és sajátvektort kívánunk meghatározni. Minden más esetben sokkal hatékonyabb a következőekben részletezett felbontási módszer.

A QR és a QL algoritmus

A QR algoritmusnak az az alapötlete, hogy bármely *valós mátrix* felbontható

$$A = Q R \quad (4-35)$$

szorzatra, ahol Q ortogonális, R pedig felső háromszögmátrix. Általános valós mátrix esetében ez a felbontás pl. Householder-transzformációk alkalmazásával úgy valósítható meg, hogy egymás után nullává tesszük A oszlopainak főátló alatti elemeit.

Tekintsük a

$$B = R Q \quad (4-36)$$

mátrixot, vagyis a (4-35)-ben szereplő tényezők fordított sorrendű szorzatát. Mivel Q ortogonális mátrix, a (4-35) felbontásból $R = Q^T A$ következik, és így a (4-36) egyenlet átírható

$$B = Q^T A Q \quad (4-37)$$

alakra. Tehát az A és a B mátrixok hasonlóak.

A (4-37)-ből közvetlenül következik, hogy ha $A^T = A$, akkor $B^T = B$ is érvényes, vagyis a QR transzformáció megőrzi a szimmetriát, és azt sem sokkal nehezebb belátni, hogy a háromatlós alakot is.

Nem lényeges kikötés, hogy az A egyik tényezője felső háromszögmátrix legyen. Lehet alsó háromszögmátrix is. Ekkor a felbontás

$$A = QL \quad (4-38)$$

alakja miatt QL algoritmusról beszélünk, ahol L alsó háromszögmátrix.

Emlékeztetünk arra, hogy a szimmetrikus mátrixnak háromátlós mátrixra való Householder-redukcióját az eredeti mátrix n -edik, azaz utolsó oszlopával kezdtük meg. Hogy a redukció kerekítési hibáit minimálisra csökkentsük, a legnagyobb elemeket igyekeztünk áthelyezni a jobb alsó sarokba. Ha az ilyen módon nyert háromátlós mátrixot diagonalizálni kívánjuk, akkor a QR és a QL algoritmusok közül előnyösebb a QL algoritmust választani, mert a kerekítési hibák kisebbek lesznek.

A QL algoritmust az

$$\begin{aligned} A_k &= Q_k L_k \\ A_{k+1} &= L_k Q_k (= Q_k^T A_k Q_k) \end{aligned} \quad (4-39)$$

ortogonális transzformációk sorozata alkotja.

Az algoritmus a következő, általános A mátrix esetére megfogalmazott tételen alapszik, amelynek bizonyítása hosszadalmas és pl. Stoer és Bulirsch [13] könyvében található meg.

4.1. tétel.

(1) Ha az A mátrix sajátértékeinek abszolút értéke mind különböző, akkor az A_k , $k = 1, 2, \dots$ mátrixsorozat egy olyan alsó háromszögmátrixhoz tart, amelynek főátlóelemei az A sajátértékei abszolút értékük szerinti növekedő sorrendben.

(2) Ha az A mátrixnak van p számú azonos abszolút értékű sajátértéke, akkor az A_k , $k = 1, 2, \dots$ mátrixsorozat tart egy olyan alsó háromszög alakú hiper-mátrixhoz, amelynek van olyan p -ed rendű diagonálblokkja, hogy a blokk sajátértékei megegyeznek az A mátrix megfelelő p számú sajátértékével. (A konvergencia a blokk sajátértékeire és nem az elemekre vonatkozik.)

A QL algoritmus iterációs lépésenkénti műveletigénye általános mátrix esetén n^3 nagyságrendű, háromátlós mátrix esetén ez a nagyságrend csak n .

Az alábbiakban kizárólag arra az esetre szorítkozunk, amikor A valós, szimmetrikus, háromátlós mátrix. Ekkor a sajátértékek valósak, és a problémát tovább egyszerűsíti a következő, Wilkinsontól származó állítás.

4.2. tétel. Ha a

$$\begin{bmatrix} d_1 & e_2 & & & \\ e_2 & d_2 & e_3 & & \\ & e_3 & d_3 & & \\ & & & \ddots & \\ & & & & e_n \\ & & & & e_n & d_n \end{bmatrix} \quad (4-40)$$

háromátlós szimmetrikus mátrixnak van p -szeres sajátértéke, akkor az $e_2, e_3, \dots, e_n$ elemek közül legalább $p-1$ elem zérussal egyenlő.

Ezért többszörös sajátértékek előfordulása esetén a háromátlós mátrix kisebb rendszámú háromátlós mátrixokra esik szét, amelyeknek csak egyszeres sajátértékeik vannak. Ezekre a mátrixokra külön-külön alkalmazható a QL algoritmus, és ekkor a 4.1. tételben szereplő határértékmátrix főátlójában legfeljebb másodrendű blokkok lehetnek és ezért nem okoznak nehézségeket.

A 4.1. tétel bizonyításából következik, hogy általános, valós mátrix főátló feletti elemei az

$$a_{ij}^{(k)} \sim \left(\frac{\lambda_i}{\lambda_j} \right)^k \quad (4-41)$$

összefüggés szerint tartanak a zérushoz. Ha $|\lambda_i|$ csak kevéssel kisebb mint $|\lambda_j|$, akkor a konvergencia lassú. A konvergencia meggyorsítható *spektráleltolások* alkalmazásával. Emlékeztetünk arra, hogy ha az A mátrix sajátértékei rendre $\lambda_1, \dots, \lambda_n$, akkor az $A - \mu I$ mátrixnak $\lambda_1 - \mu, \dots, \lambda_n - \mu$ a sajátértékei. Ha elvégezzük az $A_k - \mu_k I$ mátrix

$$A_k - \mu_k I = Q_k L_k \quad (4-42)$$

felbontását és képezzük az

$$A_{k+1} = L_k Q_k + \mu_k I = Q_k^T (A_k - \mu_k I) Q_k + \mu_k I = Q_k^T A_k Q_k \quad (4-43)$$

mátrixot, akkor a konvergencia a

$$\frac{\lambda_i - \mu_k}{\lambda_j - \mu_k} \quad (4-44)$$

hányadossal jellemezhető.

Az iteráció minden lépésében a μ_k eltolást olyannak választjuk, hogy biztosítsa a lehető leggyorsabb konvergenciát. Ha kezdő eltolásként a λ_1 legkisebb abszolút értékű sajátérték jó közelítését választanánk, akkor ez biztosítaná, hogy az első

sor főátlón kívüli elemei gyorsan tartsanak a zérushoz. A λ_1 sajátértéket azonban általában nem ismerjük előre. A gyakorlatban ezért a következő nagyon hatékonynak mutakozó stratégiát szokás alkalmazni: Kiszámítjuk a háromátlós A mátrix másodrendű bal felső sarokblokkjának sajátértékeit, és μ_k spektráleltolásként azt a sajátértéket választjuk, amely közelebb van a d_1 elemhez.

Általánosabban, tegyük fel, hogy már meghatároztuk a háromátlós A mátrix $r-1$ sajátértékét. Akkor az első $r-1$ sorának és oszlopának nullázásával A deflátható

$$\begin{bmatrix} 0 & & & & & 0 \\ & \ddots & & & & \\ & & 0 & & & \\ & & & d_r & e_{r+1} & \\ & & & e_{r+1} & d_{r+1} & \\ & & & & & \ddots \\ & & & & & & 0 \\ & & & & & & d_{n-1} & e_n \\ 0 & & & & & & 0 & e_n & d_n \end{bmatrix} \quad (4-45)$$

mátrixra. Spektráleltolásként a (4-45)-ben szereplő

$$\begin{bmatrix} d_r & e_{r+1} \\ e_{r+1} & d_{r+1} \end{bmatrix}$$

diagonál blokk azon sajátértékét választjuk, amely közelebb van a d_r elemhez. Bebizonyítható, hogy ezzel a stratégiával az algoritmus konvergenciája harmadrendű, kivéve a többszörös sajátértékek előfordulását, amikor a konvergencia legrosszabb esetben is másodrendű. Ez a gyors konvergencia teszi annyira vonzóvá az algoritmust.

Megjegyezzük, hogy a spektráleltolások alkalmazása elronthatja a sajátértékek abszolút értékük szerinti növekedő sorrendjét a határértékmátrix főátlójában. Ekkor a sajátértékek sorba rendezésére a Jacobi-iteráció után szereplő Rendez eljárás alkalmazható.

Amint már említettük, egy általános valós mátrix QL felbontása Householder-transzformációk egy sorozatával valósítható meg. Háromátlós mátrix esetében azonban előnyösebb, ha az $A = QL$, azaz a $Q^T A = L$ felbontásra (Jacobi) forgatómátrixokat alkalmazunk. A $P_1, P_2, \dots, P_{n-1}$ sorozattal rendre nullázzuk a főátló feletti $e_2, e_3, \dots, e_n$ elemeket. Az $LQ = Q^T A Q$ szimmetria miatt majd a főátló alatti $e_2, \dots, e_n$ elemek is nullává válnak. Ezért minden egyes Q_k mátrix felírható

$$Q_k^T = P_1^{(k)} P_2^{(k)} \dots P_{n-1}^{(k)} \quad (4-46)$$

alakban, ahol a $P_i^{(k)}$ forgatómátrix nullázza az $e_{i+1}^{(k)}$ elemet.

QL algoritmus implicit eltolásokkal

A spektráleltolásokkal kiegészített QL algoritmus sok esetben nagyon hatékony lehet. Ha azonban A elemei nagyságrendben nagyon különbözőek, akkor nagy μ_k érték kivonása az A mátrix főátlóelemeiből nagyon ronthatja a kis sajátértékek pontosságát. Ez a nehézség kiküszöbölhető, ha a QL algoritmust *implicit eltolásokkal* alkalmazzuk. Az implicit eltolásokkal alkalmazott QL algoritmus matematikailag egyenértékű az eredeti QL algoritmussal, ugyanakkor nem igényli, hogy az A mátrixból ténylegesen kivonjuk a $\mu_k I$ mátrixot.

Az algoritmus elvi alapját a következő állítás képezi, amelynek bizonyítása a Numerical Recipes-ban szereplő bizonyítás részletesebb változata.

4.3. tétel. Tegyük fel, hogy

$$B = Q^T A Q,$$

ahol A tetszőleges valós, szimmetrikus mátrix, Q ortogonális és B olyan háromátlós mátrix, amelynek főátlón kívüli elemei pozitívak. Akkor a Q^T mátrix utolsó sora és az A mátrix egyértelműen meghatározza a Q mátrix többi oszlopát és a (szimmetrikus) B mátrixot.

Bizonyítás. Jelölje q_i^T a Q^T mátrix i -edik sorát, vagyis q_i a Q mátrix i -edik oszlopa. A $B Q^T = Q^T A$ egyenlet részletesebben felírható

$$\begin{bmatrix} \beta_1 & \alpha_2 & & & \\ \alpha_2 & \beta_2 & \alpha_3 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \alpha_n \\ & & & & \alpha_{n-1} & \beta_{n-1} & \alpha_n \\ & & & & & \alpha_n & \beta_n \end{bmatrix} \begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ \vdots \\ q_{n-1}^T \\ q_n^T \end{bmatrix} = \begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ \vdots \\ q_{n-1}^T \\ q_n^T \end{bmatrix} A \quad (4-47)$$

alakban. A (4-47) utolsó egyenlete

$$\alpha_n q_{n-1}^T + \beta_n q_n^T = q_n^T A. \quad (4-48)$$

Mivel a Q mátrix ortogonális,

$$q_i^T q_j = \begin{cases} 1, & \text{ha } i = j \\ 0, & \text{ha } i \neq j \end{cases} \quad (4-49)$$

érvényes. Ha a (4-48) egyenletet megszorozzuk a q_n vektorral, akkor

$$\beta_n = \mathbf{q}_n^T \mathbf{A} \mathbf{q}_n \quad (4-50)$$

adódik, és mivel a $\mathbf{q}_n$ vektor ismert, a β_n szám is az. Ezután a (4-48) egyenlet kifejezhető

$$\alpha_n \mathbf{q}_{n-1}^T = \mathbf{z}_{n-1}^T \quad (4-51)$$

alakban, ahol

$$\mathbf{z}_{n-1}^T \equiv \mathbf{q}_n^T \mathbf{A} - \beta_n \mathbf{q}_n^T \quad (4-52)$$

ismert vektor. Ezért (4-51)-ből

$$\alpha_n^2 = \mathbf{z}_{n-1}^T \mathbf{z}_{n-1} \quad (4-53)$$

vagyis (pozitív előjelet választva)

$$\alpha_n = ||\mathbf{z}_{n-1}|| \quad (4-54)$$

és

$$\mathbf{q}_{n-1}^T = \frac{\mathbf{z}_{n-1}^T}{\alpha_n} \quad (4-55)$$

következik, hiszen az α_n elemről feltettük, hogy pozitív. Hasonlóképpen, indukcióval belátható, hogy ha ismertek a $\mathbf{q}_n, \mathbf{q}_{n-1}, \dots, \mathbf{q}_{n-j}$ vektorok és az $\alpha_n, \alpha_{n-1}, \dots, \alpha_{n-(j-1)}, \beta_n, \beta_{n-1}, \dots, \beta_{n-(j-1)}$ együtthatók, akkor egyértelműen meghatározhatók az $\alpha_{n-j}, \beta_{n-j}$ együtthatók és a $\mathbf{q}_{n-(j+1)}$ vektor.

E tétel gyakorlati alkalmazása céljából tegyük fel, hogy valamilyen módon előálítható egy olyan $\bar{\mathbf{A}}_{k+1}$ háromatlós mátrix, amelyre teljesül az

$$\bar{\mathbf{A}}_{k+1} = \bar{\mathbf{Q}}_k^T \mathbf{A}_k \bar{\mathbf{Q}}_k \quad (4-56)$$

összefüggés, ahol $\bar{\mathbf{Q}}_k^T$ olyan ortogonális mátrix, amelynek utolsó sora azonos az eredeti QL algoritmusban szereplő $\mathbf{Q}_k^T$ mátrix utolsó sorával. Akkor teljesül, hogy $\bar{\mathbf{Q}}_k = \mathbf{Q}_k$ és $\bar{\mathbf{A}}_{k+1} = \mathbf{A}_{k+1}$.

A (4-46) összefüggésből (a $\mathbf{P}_i^{(k)}$ forgatómátrixok speciális alakja miatt) kitűnik, hogy az eredeti QL algoritmusban szereplő $\mathbf{Q}_k^T$ mátrix utolsó sora azonos a $\mathbf{P}_{n-1}^{(k)}$ mátrix utolsó sorával. Ugyanakkor emlékeztetünk arra, hogy $\mathbf{P}_{n-1}^{(k)}$ olyan síkbeli forgatómátrix, amellyel nullázható az $\mathbf{A}_k - \mu_k \mathbf{I}$ mátrix $(n-1, n)$ indexű eleme. Figyelembe véve a síkbeli forgatómátrixok (4-2) alakját, azt kapjuk, hogy a $\mathbf{P}_{n-1}^{(k)}(\mathbf{A}_k - \mu_k \mathbf{I})$ mátrix $(n-1, n)$ indexű eleme kifejezhető

$$(d_n^{(k)} - \mu_k) \sin \varphi_k + e_n^{(k)} \cos \varphi_k$$

alakban, és ez egyenlő zérussal, ha

$$\cos \varphi_k = \frac{d_n^{(k)} - \mu_k}{\left(\left(e_n^{(k)} \right)^2 + \left(d_n^{(k)} - \mu_k \right)^2 \right)^{1/2}},$$

$$\sin \varphi_k = \frac{-e_n^{(k)}}{\left(\left(e_n^{(k)} \right)^2 + \left(d_n^{(k)} - \mu_k \right)^2 \right)^{1/2}}.$$
(4-57)

A $P_{n-1}^{(k)} A_k P_{n-1}^{(k)T}$ mátrix azonban

$$\begin{bmatrix} x & x & & & \\ x & x & x & & \\ . & . & . & & \\ & x & x & x & \\ & & x & x & x & \underline{x} \\ & & & x & x & x \\ & & & \underline{x} & x & x \end{bmatrix}$$
(4-58)

alakú, vagyis 2 (aláhúzott) elem kivételével háromátlós mátrix. Ezt a mátrixot háromátlós mátrixra kell redukálni, méghozzá olyan ortogonális mátrix alkalmazásával, amelynek utolsó sora a $[0, 0, \dots, 0, 1]$ vektor, mert így a $\bar{Q}_k^T$ utolsó sora azonos marad a $P_{n-1}^{(k)}$ utolsó sorával. A redukció Householder-transzformációk vagy síkbeli forgatómátrixok egy sorozatával hajtható végre. A (4-58) mátrix speciális alakja miatt előnyösebb a forgatómátrixok alkalmazása. Az $(n-2)$ -edik és az $(n-1)$ -edik koordinát tengely által meghatározott $(n-2, n-1)$ síkbeli forgatással az $(n-2, n)$ indexű elemet tesszük zérussá. (A szimmetria miatt az $(n, n-2)$ indexű elem is nullává válik.) Ezzel háromátlós alak keletkezik, azonban ismét csak két, az $(n-3, n-1)$ és az $(n-1, n-3)$ indexű elemek kivételével. Ezek az elemek zérussá tehetők az $(n-3, n-2)$ síkbeli forgatással és így tovább. Összesen $n-2$ ilyen ún. *Givens-forgatás* szükséges ahhoz, hogy a (4-58) által vázolt mátrixot háromátlós mátrixra redukáljuk. Eredményként

$$Q_k^T = \bar{Q}_k^T = \bar{P}_1^{(k)} \bar{P}_2^{(k)} \dots \bar{P}_{n-2}^{(k)} P_{n-1}^{(k)} \quad (4-59)$$

adódik, ahol a $\bar{P}_i^{(k)}$ mátrixok (Givens) forgatások, a $P_{n-1}^{(k)}$ mátrix pedig azonos az eredeti algoritmusban alkalmazott síkbeli forgatómátrixszal. A következő A_{k+1} iteráltat a (4-56) összefüggésből nyerjük. Megjegyezzük, hogy a μ_k eltolás a (4-57) paraméterek által implicit módon kerül be a számításba.

A következő, ImplicitQL eljárás a Numerical Recipes-ban és a Wilkinson-Reinsch kézikönyvbeli eljárás változata. A gyakorlatban rendkívül jól működik és egy sajátérték meghatározásához átlagosan 1.3–1.6 iteráció végrehajtására van szükség. Az összes sajátérték meghatározása kb. $30n^2$ műveletet igényel. Ha a sajátvektorokra is szükség van, akkor a megjegyzések által jelzett programrészeket is figyelembe véve, a szükséges műveletek száma $3n^3$ érték lesz, tehát nagyon megnő.

```

Procedure ImplicitQL( var d,e: tvn;
                     n: integer;
                     var z: tvn $\times$ n);
{ Az implicit eltolásokkal kiegészített QL algoritmus
  alkalmas háromátlós valós szimmetrikus, vagy a Red3Atlos
  eljárással erre az alakra redukált valós szimmetrikus
  mátrix összes sajátértékének és sajátvektorának
  meghatározására. Az n elemű d vektor bemenetkor a
  háromátlós mátrix főátlóelemeit, kimenetkor a
  sajátértékeket tartalmazza. A bemenő e vektor a
  háromátlós mátrix főátlóval párhuzamos szomszédos elemeit
  tartalmazza. Az e[1] elem tetszőleges. Az e vektor
  elemeit az eljárás megváltoztatja. Ha csak a sajátértékeket
  kívánjuk meghatározni, akkor az eljárás megjegyzésekkel
  jelölt sorai elhagyhatók. Ha a háromátlós mátrix
  sajátvektorait kívánjuk meghatározni, akkor a bemenő z
  egységmátrix. Ha Red3Atlos eljárással redukált mátrix
  sajátvektorait kívánjuk meghatározni, akkor a bemenő z
  azonos a Red3Atlos kimenő A mátrixával. Ekkor a z k-adik
  oszlopa tartalmazza a d[k] sajátértékhez tartozó normált
  sajátvektort.

```

Az ImplicitQL eljárást használó programban definiálni kell a következő globális típusokat

```

tvn      = array [1..nmax] of real;
tvn $\times$ n   = array [1..nmax,1..nmax] of real;

```

ahol nmax az n legnagyobb lehetséges értéke. }

```

function sign(a,b: real): real;
begin
  if b>=0 then sign:=abs(a) else sign:=-abs(a);
end;

```

```

var
  m, l, iter, i, k: integer;
  s, r, p, g, f, dd, c, b: real;

```

```

begin
  for i:=2 to n do
    e[i-1]:=e[i];
  e[n]:=0;
  for l:=1 to n do
    begin iter:=0; m:=0;
      while (iter<=30) and (m<>1) do
        begin m:=l-1;

```



```

repeat inc(m);
  if m<n then dd:=abs(d[m])+abs(d[m+1]);
  until (m=n) or (abs(e[m])+dd=dd);
  if m<>1 then
    begin
      if iter = 30 then
        Hiba('Az ImplicitQL nem konvergens.');
```

inc(iter); g:=(d[l+1]-d[l])/(2*e[l]);
r:=sqrt(sqr(g)+1);
g:=d[m]-d[l]+e[l]/(g+sign(r,g));
s:=1; c:=1; p:=0;
for i:=m-1 downto 1 do
 begin f:=s*e[i]; b:=c*e[i];
 if abs(f)>=abs(g) then
 begin c:=g/f; r:=sqrt(sqr(c)+1);
 e[i+1]:=f*r; s:=1/r; c:=c*s;
 end
 else
 begin s:=f/g; r:=sqrt(sqr(s)+1);
 e[i+1]:=g*r; c:=1/r; s:=s*c;
 end;
 g:=d[i+1]-p; r:=(d[i]-g)*s+2*c*b;
 p:=s*r; d[i+1]:=g+p; g:=c*r-b;
 for k:=1 to n do {elhagyható a ciklus}
 begin f:=z[k,i+1]; {<-}
 z[k,i+1]:=s*z[k,i]+c*f; {<-}
 z[k,i]:=c*z[k,i]-s*f; {<-}
 end; {ha nem kellene a sajátvektorok}
 end;
 d[l]:=d[l]-p; e[l]:=g; e[m]:=0;
 end;
 end;
end;
end;

5. Szimmetrikus mátrix sajátérték-feladatára vonatkozó hibaanalízis

Egy számítási eljárás sikeres végrehajtása nemcsak magán az eljáráson múlik, hanem függ a probléma stabilitásától, vagyis a megoldás érzékenységétől az adatok kis változtatásaira. A lineáris egyenletrendszerek megoldásával kapcsolatosan végzett hibaanalízisnél láttuk, hogy az $A + \delta A$ mátrix inverze akkor is nagyon eltérhet az A mátrix inverzétől, ha a δA perturbáló mátrix viszonylag kicsi. Az ilyen mátrixokat az invertálás szempontjából gyengén meghatározott mátrixoknak nevezzük. Hasonlóképpen léteznek olyan mátrixok, amelyek a *sajátérték-feladat* megoldásával kapcsolatosan *gyengén meghatározottak*. Tekintsük pl. az

$$A(\varepsilon) = \begin{bmatrix} 20 & 20 & & & & \\ & 19 & 20 & & & \\ & & 18 & 20 & & \\ & & & \ddots & & \\ & & & & 2 & 20 \\ \varepsilon & & & & & 1 \end{bmatrix}$$

mátrixot, amelynek

$$p_A(\lambda) = \det(A - \lambda I) = (20 - \lambda)(19 - \lambda) \dots (2 - \lambda)(1 - \lambda) - 20^{19}\varepsilon$$

a karakterisztikus polinomja. Ha $\varepsilon = 0$, akkor $\lambda_{20} = 1$ a legkisebb sajátérték. Ha viszont $\varepsilon = 20! \cdot 20^{-19} \approx 5 \cdot 10^{-7}$, akkor $\lambda_{20} = 0$.

Ebből a példából is kitűnik, hogy egy nem szimmetrikus mátrix sajátértékei érzékenyek a mátrix elemeinek kis megváltoztatására.

A továbbiakban bebizonyítjuk, hogy szimmetrikus mátrixok esetén ez nincs így, vagyis a szimmetrikus mátrixok sajátértékei „jól meghatározottak”.

Bizonyítsuk be először a következő állítást:

5.1. tétel. *Courant–Fischer min–max reprezentáció*
Legyen A n -ed rendű szimmetrikus mátrix, amelynek

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n \quad (5-1)$$

a sajátértékei. Akkor

$$\lambda_k = \min_{R^k} \max \{x^T A x : x \in R^k, x^T x = 1\} \quad (5-2)$$

ahol R^k az R^n , n -dimenziós tér egy tetszőleges k -dimenziós altere.

Bizonyítás. Mivel az A mátrix szimmetrikus, a $v_1, \dots, v_n$ sajátvektorairól feltehetjük, hogy azok ortonormáltak:

$$v_i^T v_j = \begin{cases} 1, & \text{ha } i = j \\ 0, & \text{ha } i \neq j \end{cases}$$

A $v_k, \dots, v_n$ sajátvektorok által kifeszített alteret jelölje S^k . Akkor S^k dimenziója $(n-k+1)$ -gyel egyenlő, úgyhogy ha R^k egy tetszőleges k -dimenziós altér, akkor fennáll, hogy $R^k \cap S^k \neq \{0\}$. Ezért létezik olyan x vektor, hogy $x \in R^k \cap S^k$, $x^T x = 1$. Így az

$$x = c_k v_k + \dots + c_n v_n$$

vektorral az (5-1) egyenlőtlenségek miatt

$$\begin{aligned} \mathbf{x}^T \mathbf{A} \mathbf{x} &= \left(c_k \mathbf{v}_k^T + \dots + c_n \mathbf{v}_n^T \right) \left(\lambda_k c_k \mathbf{v}_k + \dots + \lambda_n c_n \mathbf{v}_n \right) = \\ &= \lambda_k c_k^2 + \dots + \lambda_n c_n^2 \geq \lambda_k \left(c_k^2 + \dots + c_n^2 \right) = \lambda_k \end{aligned}$$

adódik (hiszen $c_k^2 + \dots + c_n^2 = \mathbf{x}^T \mathbf{x} = 1$).

Mivel $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq \lambda_k$, szintén $\max \mathbf{x}^T \mathbf{A} \mathbf{x} \geq \lambda_k$ és mivel az R^k altérrel feltételeztük, hogy tetszőleges, írhatjuk, hogy

$$\min \max \mathbf{x}^T \mathbf{A} \mathbf{x} \geq \lambda_k \quad (5-3)$$

Hogy bebizonyítsuk az egyenlőtlenség fordítottját, tételezzük fel, hogy R_0^k a $\mathbf{v}_1, \dots, \mathbf{v}_k$ sajátvektorok által kifeszített altér és legyen $\mathbf{x} \in R_0^k$, $\mathbf{x}^T \mathbf{x} = 1$. Így az

$$\mathbf{x} = d_1 \mathbf{v}_1 + \dots + d_k \mathbf{v}_k$$

vektorral és figyelembe véve az (5-1) összefüggést, kapjuk:

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = \lambda_1 d_1^2 + \dots + \lambda_k d_k^2 \leq \lambda_k \left(d_1^2 + \dots + d_k^2 \right) = \lambda_k,$$

úgyhogy

$$\max \{ \mathbf{x}^T \mathbf{A} \mathbf{x} : \mathbf{x} \in R_0^k, \mathbf{x}^T \mathbf{x} = 1 \} \leq \lambda_k.$$

Az utóbbi és az (5-3) egyenlőtlenségből közvetlenül következik az (5-2) összefüggés, amivel a tételt bebizonyítottuk.

Az 5.1. tétel felhasználásával már egyszerűen belátható, hogy egy *valós szimmetrikus mátrix sajátértékei jól meghatározottak*.

5.2. tétel. Ha $\lambda_1 \leq \dots \leq \lambda_n$ a szimmetrikus $\mathbf{A}$ mátrix sajátértékei és $\mu_1 \leq \dots \leq \mu_n$ a perturbált szimmetrikus $\mathbf{A} + \delta \mathbf{A}$ mátrix sajátértékei, akkor érvényes az

$$|\lambda_i - \mu_i| \leq \|\delta \mathbf{A}\|_S, \quad i = 1, \dots, n$$

összefüggés.

Bizonyítás. A spektrálnorma definíciója szerint: $\|\delta \mathbf{A}\|_S = \sigma_1$, ahol σ_1 a $\delta \mathbf{A}$ mátrix legnagyobb szinguláris értéke. Vagyis

$$\|\delta \mathbf{A}\|_S = [\text{a } (\delta \mathbf{A})^T \delta \mathbf{A} \text{ maximális sajátértéke}]^{1/2}.$$

Mivel $\delta \mathbf{A}$ szimmetrikus, a $\|\delta \mathbf{A}\|_S$ norma egyenlő a $\delta \mathbf{A}$ mátrix legnagyobb abszolút értékű sajátértékének abszolút értékével. Ennek alapján a $\delta \mathbf{A} + \|\delta \mathbf{A}\|_S \mathbf{I}$

mátrix sajátértékei nemnegatívak, tehát ez a mátrix nemnegatív definit és az 5.1. tétel alapján

$$\begin{aligned}\lambda_i &= \min \max \mathbf{x}^T \mathbf{A} \mathbf{x} \leq \min \max \left[\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{x}^T (\delta \mathbf{A} + ||\delta \mathbf{A}||_S \mathbf{I}) \mathbf{x} \right] = \\ &= \min \max \mathbf{x}^T \left[\mathbf{A} + \delta \mathbf{A} + ||\delta \mathbf{A}||_S \mathbf{I} \right] \mathbf{x} = \mu_i + ||\delta \mathbf{A}||_S.\end{aligned}$$

Ebből viszont

$$\lambda_i - \mu_i \leq ||\delta \mathbf{A}||_S, \quad i = 1, \dots, n$$

következik.

Hasonlóképpen a $\delta \mathbf{A} - ||\delta \mathbf{A}||_S \mathbf{I}$ mátrix nempozitív definit és ezért

$$\begin{aligned}\lambda_i &= \min \max \mathbf{x}^T \mathbf{A} \mathbf{x} \geq \\ &\geq \min \max \left[\mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{x}^T (\delta \mathbf{A} - ||\delta \mathbf{A}||_S \mathbf{I}) \mathbf{x} \right] = \mu_i - ||\delta \mathbf{A}||_S,\end{aligned}$$

amiből

$$\lambda_i - \mu_i \geq -||\delta \mathbf{A}||_S, \quad i = 1, \dots, n$$

következik. Ezzel a tételt bebizonyítottuk.

Bár egy szimmetrikus mátrix sajátértékei „érzékenynek” a mátrix elemeinek kis változtatásaira, a sajátvektorokra ez már nem vonatkozik. Mint a következő példa mutatja, a sajátvektorok nem okvetlenül folytonos függvényei a mátrix elemeinek. Legyen

$$\mathbf{A} = \begin{bmatrix} 1 + \varepsilon \cos \frac{2}{\varepsilon} & -\varepsilon \sin \frac{2}{\varepsilon} \\ -\varepsilon \sin \frac{2}{\varepsilon} & 1 - \varepsilon \cos \frac{2}{\varepsilon} \end{bmatrix}, \quad \varepsilon \neq 0.$$

Mint hogy az $\mathbf{A}$ sajátértékei $1 \pm \varepsilon$ és

$$\left[\cos \frac{1}{\varepsilon}, \quad -\sin \frac{1}{\varepsilon} \right]^T, \quad \left[\sin \frac{1}{\varepsilon}, \quad \cos \frac{1}{\varepsilon} \right]^T$$

a megfelelő sajátvektorok, ha $\varepsilon \rightarrow 0$, a sajátvektorok nem tartanak határértékhez. Ennek következtében az $\mathbf{A}$ mátrix együtthatóinak kis megváltoztatása a sajátvektorok nagy megváltozását okozhatja.

A sajátvektor „érzékenysége” azzal kapcsolatos, hogy a hozzá tartozó sajátérték mennyire különbözik a vele szomszédos sajátértékektől. A következő tételben bebizonyítjuk, hogy ha egy sajátérték a többi sajátértéktől jól elkülönített, akkor az ehhez tartozó sajátvektor is jól meghatározott.

5.3. tétel. Legyenek az A , ill. az $A + \delta A$ szimmetrikus valós mátrixok sajátértékei

$$\lambda_1 \leq \dots \leq \lambda_n, \quad \text{ill.} \quad \mu_1 \leq \dots \leq \mu_n$$

és a megfelelő – egyre normált – sajátvektorai

$$u_1, \dots, u_n \quad \text{ill.} \quad v_1, \dots, v_n.$$

Ha teljesül az

$$|\lambda_k - \lambda_i| \geq \alpha > \|\delta A\|_S, \quad i \neq k$$

feltétel, akkor fennáll a

$$\|u_k - v_k\| \leq \gamma (1 + \gamma^2)^{1/2}, \quad \gamma = \frac{\|\delta A\|_S}{\alpha - \|\delta A\|_S}$$

egyenlőtlenség.

Bizonyítás. Tegyük fel, hogy

$$v_k = \beta_1 u_1 + \dots + \beta_n u_n,$$

ahol v_k előjelét úgy választjuk meg, hogy $\beta_k \geq 0$ legyen.

A sajátérték-feladat

$$(A + \delta A - \mu_k I) v_k = 0$$

alapegyenletéből

$$(A - \mu_k I) v_k = -\delta A v_k$$

következik. Tekintettel a $\|\delta A v_k\| \leq \|\delta A\| \|\|v_k\|$ Schwarz-egyenlőtlenségre és mivel a tétel feltételezése miatt $\|v_k\| = 1$, írhatjuk, hogy

$$\begin{aligned} \|\delta A\|^2 &\geq \|\delta A v_k\|^2 = \|(A - \mu_k I) v_k\|^2 = \\ &= \|A \sum \beta_i u_i - \mu_k \sum \beta_i u_i\|^2 = \|\sum \beta_i \lambda_i u_i - \mu_k \sum \beta_i u_i\|^2 = \\ &= \|\sum \beta_i (\lambda_i - \mu_k) u_i\|^2 = \left[\sum \beta_i (\lambda_i - \mu_k) u_i \right]^T \left[\sum \beta_i (\lambda_i - \mu_k) u_i \right] = \\ &= \sum_{i=1}^n \beta_i^2 (\lambda_i - \mu_k)^2 \geq \sum_{i \neq k} \beta_i^2 (\lambda_i - \mu_k)^2 \end{aligned}$$

(hiszen az $u_1, \dots, u_n$ sajátvektorrendszer ortonormált). Tekintettel az $|a - b| \leq |a| + |b|$ egyenlőtlenségre, azt írhatjuk, hogy

$$|\lambda_k - \lambda_i| \equiv |\lambda_k - \mu_k - (\lambda_i - \mu_k)| \leq |\lambda_k - \mu_k| + |\lambda_i - \mu_k|$$

amiből

$$|\lambda_i - \mu_k| \geq |\lambda_k - \lambda_i| - |\lambda_k - \mu_k|$$

következik. Mivel feltételeztük, hogy $|\lambda_k - \lambda_i| \geq \alpha$, $i \neq k$ és mivel az 5.2. tétel szerint $|\lambda_k - \mu_k| \leq \|\delta A\|_S$, azt kapjuk, hogy

$$|\lambda_i - \mu_k| \geq \alpha - \|\delta A\|_S, \quad i \neq k.$$

Ennek következtében

$$\|\delta A\|_S^2 \geq (\alpha - \|\delta A\|_S)^2 \sum_{i \neq k} \beta_i^2.$$

A $\|v_k\|^2 = v_k^T v_k = \sum_{i=1}^n \beta_i^2 = 1$ feltétel miatt

$$\beta_k^2 = 1 - \sum_{i \neq k} \beta_i^2 \geq 1 - \frac{\|\delta A\|_S^2}{(\alpha - \|\delta A\|_S)^2} = 1 - \gamma^2,$$

vagyis

$$1 - \beta_k^2 \leq \gamma^2,$$

úgyhogy

$$(1 - \beta_k)^2 = \frac{(1 - \beta_k^2)^2}{(1 + \beta_k)^2} \leq (1 - \beta_k^2)^2 \leq \gamma^4.$$

Felhasználva az utóbbi egyenlőtlenséget, írhatjuk, hogy

$$\begin{aligned} \|u_k - v_k\|^2 &= \|u_k - \sum_{i=1}^n \beta_i u_i\|^2 = \|(1 - \beta_k) u_k - \sum_{i \neq k} \beta_i u_i\|^2 = \\ &= (1 - \beta_k)^2 + \sum_{i \neq k} \beta_i^2 \leq \gamma^4 + \gamma^2 = \gamma^2(1 + \gamma^2), \end{aligned}$$

amiből már

$$\|u_k - v_k\| \leq \gamma(1 + \gamma^2)^{1/2}$$

következik és ezáltal a tételt bebizonyítottuk.

Ha a λ_k sajátérték többszörös, akkor az 5.3. tétel olyan értelemben szigorítható, hogy ha λ_k a vele szomszédos sajátértékektől jól elkülönített, akkor a λ_k -hoz tartozó sajátaltér jól meghatározott.

6. Lineáris algebrai egyenletrendszerek általánosított megoldása

A korszerű numerikus analízisben fontos szerepe van a mátrix szinguláris értékek szerinti felbontásának, vagy rövidebben az SVD-nek (Singular Value Decomposition). Ezen olyan nagyon hatékony technikákat értünk, amelyekkel jól elemezhetők és bizonyos (általánosított) értelemben meg is oldhatók olyan lineáris algebrai egyenletrendszerek, amelyeknek az együtthatómátrixa szinguláris vagy numerikusan majdnem szinguláris.

Mátrix szinguláris értékek szerinti felbontása

Az SVD módszerek elvi alapját a következő állítás képezi.

6.1. tétel. Bármely $m \times n$ típusú, $m \geq n$, valós A mátrix felbontható

$$A = U \Sigma V^T = \begin{array}{c} \begin{array}{|c|} \hline n \\ \hline \end{array} \\ \begin{array}{|c|} \hline m \\ \hline \end{array} \end{array} \begin{array}{c} n \\ \diagdown \\ n \end{array} \begin{array}{c} \begin{array}{|c|} \hline n \\ \hline \end{array} \\ \begin{array}{|c|} \hline n \\ \hline \end{array} \end{array} \quad (6-1)$$

alakra, ahol

$$U^T U = V^T V = V V^T = I_n$$

és $\Sigma = \langle \sigma_1, \dots, \sigma_n \rangle$ diagonálmátrix. Az U mátrix n oszlopát az $A A^T$ mátrix n legnagyobb sajátértékéhez tartozó ortonormált sajátvektorok alkotják. A V mátrix oszlopai az $A^T A$ mátrix ortonormált sajátvektorai. A Σ főátlóelemei az $A^T A$ mátrix sajátértékeinek nemnegatív négyzetgyökei. Ezek az A mátrix szinguláris értékei, amelyeket $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0$ sorrendben jelölünk.

Ha az A mátrix rangja r , akkor $\sigma_r > 0$ és vagy $r = n$, vagy $\sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_n = 0$ (vagyis a pozitív szinguláris értékek száma egyenlő az A rangjával).

Bizonyítás. Az $A^T A$ szorzat valós, szimmetrikus n -ed rendű mátrix, amely nemnegatív definit is, hiszen tetszőleges n elemű $x \neq 0$ vektorra $x^T A^T A x = (A x)^T (A x) = \|A x\|^2 \geq 0$. Ezért sajátértékei nemnegatív valós számok, így $\sigma_1^2, \dots, \sigma_n^2$ -tel jelölhetők úgy, hogy $\sigma_1 \geq \dots \geq \sigma_n \geq 0$. Jelölje r azt az indexet, amelyre $\sigma_r > 0$ és vagy $r = n$, vagy $\sigma_{r+1} = 0$.

A valós, szimmetrikus mátrix mindig kifejezhető

$$V^T A^T A V = \Sigma^2 \quad (6-2)$$

alakban, ahol $\Sigma^2 = \langle \sigma_1^2, \dots, \sigma_n^2 \rangle$ és $V = [v_1, \dots, v_n]$ oszlopai az $A^T A$ ortonormált sajátvektorai. A (6-2) egyenlet részletesebb felírással

$$\begin{bmatrix} (A v_1)^T \\ \vdots \\ (A v_n)^T \end{bmatrix} \begin{bmatrix} A v_1, \dots, A v_n \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_n^2 \end{bmatrix}. \quad (6-3)$$

A főátlóelemek egyenlőségéből

$$||A v_i|| = \sigma_i, \quad i = 1, \dots, n \quad (6-4)$$

következik, a többi elem egyenlőségéből pedig az, hogy az $A v_1, \dots, A v_n$ vektorok páronként ortogonálisak. A $\sigma_1 \geq \dots \geq \sigma_r > 0, \sigma_{r+1} = \dots = \sigma_n = 0$ fenti feltételezésünk miatt (6-4)-ből következik, hogy

$$A v_i \neq 0, \quad \text{ha} \quad i = 1, \dots, r \quad (6-5)$$

és

$$A v_i = 0, \quad \text{ha} \quad i = r+1, \dots, n.$$

Ha az

$$A^T A v_i = \sigma_i^2 v_i, \quad i = 1, \dots, n \quad (6-6)$$

sajátérték-feladat definíciós egyenletét balról szorozzuk A -val, akkor

$$A A^T (A v_i) = \sigma_i^2 (A v_i), \quad i = 1, \dots, n. \quad (6-7)$$

Ebből (6-5) miatt következik, hogy az $A^T A$ mátrix $\sigma_1^2, \dots, \sigma_r^2$ sajátértékei egyben az $A A^T$ mátrix sajátértékei és a páronként ortogonális $A v_i, i = 1, \dots, r$ vektorok az $A A^T$ mátrix sajátvektorai. Ezért az

$$u_i = \frac{A v_i}{\sigma_i}, \quad i = 1, \dots, r \quad (6-8)$$

vektorok az $A A^T$ ortonormált sajátvektorai, hiszen (6-4) miatt $||u_i|| = 1, i = 1, \dots, r$ érvényes.

A továbbiakban felhasználjuk a mátrixszámítás következő eredményeit (lásd pl. Kielbasinski-Schwetlick [5]):

$$r = \text{rang}(A) = \text{rang}(\Sigma),$$

$$\text{rang}(A) = \text{rang}(A^T A) = \text{rang}(A A^T).$$

Ebből következik, hogy a zérus az n -ed rendű $A^T A$ mátrixnak $(n-r)$ -szeres sajátértéke, az m -ed rendű $A A^T$ mátrixnak pedig pontosan $(m-r)$ -szeres sajátértéke. Következésképpen $\sigma_1^2, \dots, \sigma_r^2$ mindkét $-A^T A$ és $A A^T$ - mátrixnak a zérustól különböző sajátértékei.

Az $A A^T$ szintén valós szimmetrikus mátrix. Ezért létezik az $u_1, \dots, u_m$ ortonormált sajátvektorrendszere. A pozitív sajátértékekhez tartozó u_i , $i = 1, \dots, r$ sajátvektorokra a (6-8) összefüggés érvényes. Jelölje U azt az $m \times n$ típusú mátrixot, amelynek oszlopai az $A A^T$ mátrix $\sigma_1^2, \dots, \sigma_n^2$ sajátértékeihez tartozó $u_1, \dots, u_n$ ortonormált sajátvektorok. Ezért $U^T U = I_n$. A (6-8) összefüggésből és a fentiekből következik, hogy

$$A v_i = \sigma_i u_i, \quad i = 1, \dots, n,$$

vagyis tömörebben

$$A V = U \Sigma.$$

Végül, ha ezt az egyenletet jobbról szorozzuk az ortogonális V mátrix transzponáltjával, akkor megkapjuk a (6-1) egyenletet, és ezzel a bizonyítást befejeztük.

Eljárás az SVD numerikus megoldására

A (6-1) alatti $A = U \Sigma V^T$ szinguláris értékek szerinti felbontás elvégzésére a legalkalmasabbnak látszik az a numerikusan nagyon stabilis eljárás, amelynél az A mátrixot először Householder-transzformációk alkalmazásával kétátlós alakra redukálják, majd a kétátlós mátrix szinguláris értékeit a QR algoritmussal egyenértékű eljárással határozzák meg.

Redukció kétátlós mátrixra. A Householder-féle ortogonális transzformációk két

$$P_k = I - 2 p_k p_k^T, \quad p_k^T p_k = 1, \quad k = 1, \dots, n$$

és

$$Q_k = I - 2 q_k q_k^T, \quad q_k^T q_k = 1, \quad k = 1, \dots, n-2$$

sorozatának alkalmazásával A redukálható

$$P_n \dots P_1 A Q_1 \dots Q_{n-2} = \left[\begin{array}{cccc} q_1 e_2 & & & \\ & q_2 e_3 & & \\ & & \ddots & \\ & & & e_n \\ & & & & q_n \\ \hline & & & & 0 & \end{array} \right] \equiv B_0 \quad \left. \vphantom{\begin{array}{c} P_n \dots P_1 A Q_1 \dots Q_{n-2} \\ B_0 \end{array}} \right\} (m-n) \times n$$

felső kétátlós mátrixra. A P_k mátrixot úgy határozzák meg, hogy nullává tegye az aktuális mátrix k -edik oszlopának utolsó $m-k$ elemét. A Q_k mátrix pedig nullázza az aktuális mátrix k -edik sorának utolsó $n-(k+1)$ elemét.

Mivel $B_0 = P^T A Q$, ahol $P^T \equiv P_n \dots P_1$ és $Q \equiv Q_1 \dots Q_{n-2}$ ortogonális mátrixok, azt kapjuk, hogy

$$B_0^T B_0 = Q^T A^T P P^T A Q = Q^T A^T A Q,$$

ami azt is jelenti, hogy a B_0 mátrixnak ugyanazok a szinguláris értékei, mint A -nak. Az $A = P B_0 Q^T$ összefüggésből az is következik, hogy ha ismerjük a B_0 mátrix $B_0 = \bar{U} \Sigma \bar{V}^T$ szinguláris értékek szerinti felbontását, akkor az $A = P \bar{U} \Sigma \bar{V}^T Q^T = U \Sigma V^T$ mátrix szinguláris értékek szerinti felbontása már egyszerűen képezhető: $U = P \bar{U}$, $V = Q \bar{V}$.

A kétátlós mátrix szinguláris értékek szerinti felbontása. A B_0 mátrix diagonalizálható a QR algoritmus egy változatával úgy, hogy a

$$B_{i+1} = S_i^T B_i T_i$$

összefüggéssel képezünk egy kétátlós mátrixokból álló $B_1, B_2, \dots$ sorozatot, amely a Σ diagonálmátrixhoz konvergál. Az S_i és T_i ortogonális mátrixokat úgy választjuk meg, hogy a T_i forgatómátrixok lehetővé tegyék a

$$M_{i+1} \equiv B_{i+1}^T B_{i+1} = T_i^T B_i^T B_i T_i \equiv T_i^T M_i T_i \quad (6-9)$$

szimmetrikus háromátlós mátrixok konvergenciáját a diagonálmátrixhoz, az S_i forgatómátrixok pedig biztosítsák a B_i mátrixok kétátlós alakját. A $B_i \rightarrow B_{i+1}$ transzformációt Givens-forgatásokkal végezzük úgy, hogy azokat váltakozva, jobbról és balról alkalmazzuk a B_i mátrixra.

Így írható, hogy

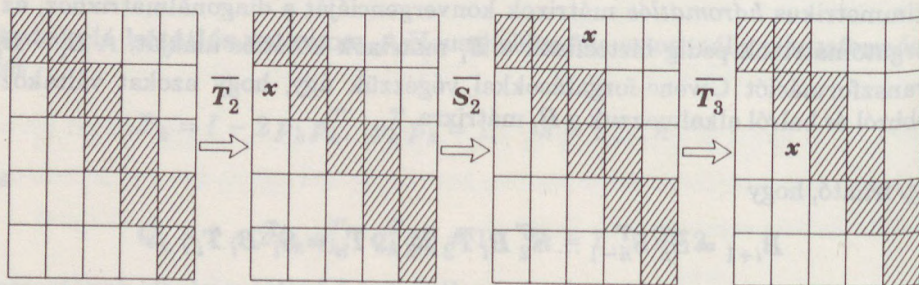
$$B_{i+1} = \bar{S}_n^T \bar{S}_{n-1}^T \dots \bar{S}_2^T B_i \bar{T}_2 \bar{T}_3 \dots \bar{T}_n = S_i^T B_i T_i,$$

ahol

$$\bar{S}_k = \begin{array}{c} \begin{array}{cc} k-1\text{-edik} & k\text{-edik} \\ \text{oszlop} & \text{oszlop} \end{array} \\ \left[\begin{array}{cccccc} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & \cos \psi_k & -\sin \psi_k & \\ \cdots & & & \sin \psi_k & \cos \psi_k & \\ & & & & & 1 & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{array} \right] \end{array} \begin{array}{l} \\ \\ \\ k-1\text{-edik sor} \\ k\text{-adik sor} \\ \\ \\ \end{array}$$

és a $\bar{T}_k$ mátrix csak annyiban tér el $\bar{S}_k$ -től, hogy ψ_k helyén φ_k szerepel.

Egyelőre legyen a φ_2 szög tetszőleges. Az összes többi szöveget úgy választjuk meg, hogy a B_{i+1} mátrix alakja pontosan megegyezzen B_i alakjával. Részletesebben: $\bar{T}_2$ a B_i mátrix egyetlen elemét sem teszi nullává, azonban megváltoztatja a $(2, 1)$ indexű elem helyén álló zérust; $\bar{S}_2^T$ újra nullázza ezt az elemet és módosítja az aktuális mátrix $(1, 3)$ indexű elem helyén álló zérust, $\bar{T}_3$ ezt az elemet nullázza és megváltoztatja az aktuális mátrix $(3, 2)$ indexű elemét és így tovább. Végül $\bar{S}_n^T$ nullázza a $(n, n-1)$ indexű elemet és nem változtat meg egyetlen zérus elemet sem (lásd a 6.1. ábrát).



6.1 ábra. A $B_i \rightarrow B_{i+1}$ transzformáció kétátlós alakjának megőrzése

Ha végül a $\bar{T}_2$ mátrixot (a még határozatlan φ_2 szöget) úgy választjuk meg, hogy annak első oszlopa legyen egyenlő az

$$M_i - sI$$

mátrix első oszlopával (ahol s pl. a $M_i = B_i^T B_i$ másodrendű jobb alsó sarokblokkjának sajátértéke), akkor bebizonyítható, hogy a (6–9) alatti $M_i \rightarrow M_{i+1}$ transzformáció egyenértékű azzal, mintha *implicit eltolással* alkalmaztuk volna a QR algoritmus

$$M_i - sI = T_s R_s$$

$$M_{i+1} = R_s T_s + sI$$

lépését, ahol $T_s^T T_s = I$ és R_s felső háromszögmátrix.

Valóban, a $\bar{T}_2$ mátrixot úgy választottuk meg, hogy első oszlopa egyenlő a T_s ortogonális mátrix első oszlopával. A forgatómátrixok alakjából következik, hogy a $\bar{T}_2$ első oszlopa azonos a $T_i = \bar{T}_2 \dots \bar{T}_n$ ortogonális mátrix első oszlopával. Így a T_i és a T_s ortogonális mátrixok első oszlopai azonosak. A (6–9) összefüggésből és a 4.3. tételből következik, hogy $T_s = T_i$.

```

Procedure SzEfelbont(   var A: tVmxn;
                        m,n: integer;
                        var w: tVn;
                        var V: tVnxn);

```

{ Ez az eljárás alkalmas az mxn típusú A mátrix $A = U W V^T$ alakú szinguláris értékek szerinti felbontására. Kimenetkor az A tömb az U mátrixot tartalmazza. A W diagonálmátrix szinguláris értékeit az n elemű w vektor tartalmazza. A V mátrixot a V tömb tartalmazza. Ha nem teljesül az $m \geq n$ követelmény, akkor az A mátrixot zérus sorokkal kell négyzetesre kiegészíteni.

A SzEfelbont, ill. SzEmegold eljárást használó programban definiálni kell a következő globális típusokat

```

tvn      = array [1..nmax] of real;
tvm      = array [1..mmax] of real;
tvnxn    = array [1..nmax,1..nmax] of real;
tvmxn    = array [1..mmax,1..nmax] of real;

```

ahol az nmax konstans az n, mmax pedig az m legnagyobb lehetséges értéke.

Ha az adott Pascal implementáció nem rendelkezik a double típussal, akkor definiálni kell azt a

```

double = real;

```

utasítással. }


```

function sign(a,b: real): real;
begin
  if b>=0 then sign:=abs(a) else sign:=-abs(a);
end;

function max(a,b: real): real;
begin
  if a>b then max:=a else max:=b;
end;

const
  maxite = 30;                                {iteráció maximális száma}
var
  nm,l,k,j,jj,nite,i: integer;
  z,y,x,h,g,f,c,anorm: real;
  scale,s: double;
  rv1: tvn;

begin g:=0.0; scale:=0.0; anorm:=0.0;
  for i:=1 to n do                             {redukció kétátlós alakra}
    begin l:=i+1; rv1[i]:=scale*g; g:=0.0; s:=0.0;
      scale:=0.0;
      if i<=m then
        begin
          for k:=i to m do
            scale:=scale+abs(a[k,i]);
          if scale<>0.0 then
            begin
              for k:=i to m do
                begin a[k,i]:=a[k,i]/scale;
                  s:=s+a[k,i]*a[k,i];
                end;
              f:=a[i,i]; g:=-sign(sqrt(s),f);
              h:=f*g-s; a[i,i]:=f-g;
              if i<>n then
                for j:=1 to n do
                  begin s:=0.0;
                    for k:=i to m do
                      s:=s+a[k,i]*a[k,j];
                    f:=s/h;
                    for k:=i to m do
                      a[k,j]:=a[k,j]+f*a[k,i];
                    end;
                  for k:=i to m do
                    a[k,i]:=scale*a[k,i];
                  end;
                end;
              end;
            w[i]:=scale*g; g:=0.0; s:=0.0; scale:=0.0;
            if (i<=m) and (i<>n) then
              begin
                for k:=1 to n do
                  scale:=scale + abs(a[i,k]);
                if scale<>0.0 then
                  begin
                    for k:=1 to n do

```



```

begin a[i,k]:=a[i,k]/scale;
s:=s+a[i,k]*a[i,k];
end;
f:=a[i,l]; g:=-sign(sqrt(s),f);
h:=f*g-s; a[i,l]:=f-g;
for k:=1 to n do
  rv1[k]:=a[i,k]/h;
if i<>m then
  for j:=1 to m do
    begin s:=0.0;
      for k:=1 to n do
        s:=s+a[j,k]*a[i,k];
      for k:=1 to n do
        a[j,k]:=a[j,k]+s*rv1[k];
      end;
    for k:=1 to n do
      a[i,k]:=scale*a[i,k];
    end;
  end;
anorm:=max(anorm,(abs(w[i])+abs(rv1[i]))));
end;
v[n,n]:=1.0; g:=rv1[n]; l:=n;
for i:=n-1 downto 1 do
  begin
    if g<>0.0 then
      begin
        for j:=1 to n do
          v[j,i]:=(a[i,j]/a[i,l])/g;
        for j:=1 to n do
          begin s:=0.0;
            for k:=1 to n do
              s:=s+a[i,k]*v[k,j];
            for k:=1 to n do
              v[k,j]:=v[k,j]+s*v[k,i];
            end;
          end;
        for j:=1 to n do
          begin v[i,j]:=0.0; v[j,i]:=0.0; end;
          v[i,i]:=1.0; g:=rv1[i]; l:=i;
        end;
        l:=n+1; g:=w[n]; if g<>0 then g:=1.0/g;
        for j:=n to m do a[j,n]:=a[j,n]*g;
        a[n,n]:=a[n,n]+1.0;
        for i:=n-1 downto 1 do
          begin l:=i+1; g:=w[i];
            for j:=1 to n do a[i,j]:=0.0;
            if g<>0.0 then
              begin g:=1.0/g;
                for j:=1 to n do
                  begin s:=0.0;
                    for k:=1 to m do
                      s:=s+a[k,i]*a[k,j];
                    f:=(s/a[i,i])*g;
                    for k:=i to m do
                      a[k,j]:=a[k,j]+f*a[k,i];

```



```

        end;
    end;
    for j:=i to m do a[j,i]:=a[j,i]*g;
    a[i,i]:=a[i,i]+1.0
end;

for k:=n downto 1 do {ciklus a szinguláris értékekre}
    begin nite:=0;
    repeat inc(nite); l:=k; {Iteráció szing. értékenként}
    while (l>1) and ((abs(rv1[l])+anorm)<>anorm) and
    ((abs(w[l-1])+anorm)<>anorm) do dec(l);
    nm:=l-1;
    if (abs(rv1[l])+anorm)<>anorm then
        begin c:=0.0; s:=1.0;
        for i:=1 to k do
            begin f:=s*rv1[i];
            if (abs(f)+anorm)<>anorm then
                begin g:=w[i]; h:=sqrt(f*f+g*g); w[i]:=h;
                h:=1.0/h; c:=g*h; s:=-f*h;
                for j:=1 to m do
                    begin y:=a[j,nm]; z:=a[j,i];
                    a[j,nm]:=y*c+z*s;
                    a[j,i]:=-y*s+z*c;
                end;
            end;
        end;
        end;
    end;
    z:=w[k];
    if (nite<maxite) and (l<>k) then
        begin x:=w[l]; {eltolásszám az alsó 2x2 minorból}
        nm:=k-1; y:=w[nm]; g:=rv1[nm]; h:=rv1[k];
        f:=((y-z)*(y+z)+(g-h)*(g+h))/(2.0*h*y);
        g:=sqrt(f*f+1.0);
        f:=((x-z)*(x+z)+h*((y/(f+sign(g,f)))-h))/x;
        c:=1.0; s:=1.0; {a következő QR transzformáció}
        for j:=1 to nm do
            begin i:=j+1; g:=rv1[i]; y:=w[i]; h:=s*g;
            g:=c*g; z:=sqrt(f*f+h*h); rv1[j]:=z;
            c:=f/z; s:=h/z; f:=x*c+g*s;
            g:=-x*s+g*c; h:=y*s; y:=y*c;
            for jj:=1 to n do
                begin x:=v[jj,j]; z:=v[jj,i];
                v[jj,j]:=x*c+z*s;
                v[jj,i]:=-x*s+z*c;
            end;
            z:=sqrt(f*f+h*h); w[j]:=z;
            if z<>0.0 then
                begin z:=1.0/z; c:=f*z; s:=h*z; end;
            f:=c*g+s*y; x:=-s*g+c*y;
            for jj:=1 to m do
                begin y:=a[jj,j]; z:=a[jj,i];
                a[jj,j]:=y*c+z*s;
                a[jj,i]:=-y*s+z*c;
            end;
        end; {j}
    end;
end;

```



```

    rv1[l]:=0.0; rv1[k]:=f; w[k]:=x;
    end; {if}
until (nite=maxite) or (l=k);
if nite=maxite then
    Hiba('A SzEfelbont eljárás nem konvergens.')
else if z<0.0 then
    begin w[k]:=-z;
        for j:=1 to n do
            v[j,k]:=-v[j,k];
        end;
    end; {k}
end;
end;

```

Pszéudoinverz mátrix

Az $m \times n$ típusú valós A mátrix *Penrose–Moore pszéudoinverzének* nevezzük az $n \times m$ típusú X mátrixot, ha kielégíti a következő négy feltételt:

$$AXA = A, \quad XAX = X, \quad AX \text{ és } XA$$

szimmetrikus mátrixok.

Bebizonyítható, hogy mindig létezik egyetlen ilyen X mátrix. Ha A nem szinguláris (tehát négyzetes) mátrix, akkor $X = A^{-1}$. Ha A egy $m \times n$ típusú $r(A) = n$ rangú mátrix, akkor $X = (A^T A)^{-1} A^T$. Ha $r(A) < n$, akkor a pszéudoinverz mátrixot, amelyet a továbbiakban A^+ jelöli, a szinguláris értékek szerinti felbontás alkalmazásával állítjuk elő. Nem nehéz ellenőrizni, hogy ha $A = U \Sigma V^T$, akkor $A^+ = V \Sigma^+ U^T$, ahol $\Sigma^+ = \langle \sigma_1^+, \dots, \sigma_n^+ \rangle$ diagonálmátrix

$$\sigma_i^+ = \begin{cases} 1/\sigma_i, & \text{ha } \sigma_i > 0 \\ 0, & \text{ha } \sigma_i = 0 \end{cases}$$

elemekkel.

Így tehát az A^+ pszéudoinverz mátrix egyszerűen kiszámítható miután elvégeztük az A mátrix SVD felbontását, azaz előállítottuk az U , Σ és V mátrixokat.

Lineáris egyenletrendszerek legkisebb négyzetes megoldása

Tekintsük az

$$A x = b \quad (6-10)$$

valós elemű lineáris algebrai egyenletrendszert, ahol A $m \times n$ típusú mátrix, $m \geq n$, b pedig m elemű vektor. Ha $m > n$, akkor a (6-10) egyenletrendszer „túlhatározott” és általános esetben nincs megoldása. Ilyen esetben célszerű bevezetni az *általánosított megoldás*, illetve a *legkisebb négyzetes megoldás* fogalmát. Ezen olyan n elemű x vektor meghatározását értjük, amely minimummá teszi a $b - Ax$ hibavektor euklideszi normáját. Megmutatjuk, hogy ha az A mátrix $r(A) \equiv r$ rangja kisebb, mint n , akkor a

$$||b - Ax|| \equiv [(b - Ax)^T (b - Ax)]^{1/2} = \min ! \quad (6-11)$$

feladatnak a megoldása nem egyértelmű. A megoldás egyértelművé tehető, ha a (6-11) feladat x megoldásai közül a legkisebb euklideszi normájút keressük, vagyis azt az x^* vektort, amelyre

$$||x^*|| = \min !$$

Bebizonyítjuk, hogy

$$x^* = A^+ b = V \Sigma^+ U^T b. \quad (6-12)$$

Először is megmutatjuk, hogy a (6-11) feladat egyenértékű az

$$A^T A x = A^T b, \quad (6-13)$$

ún. *normálegyenletek* megoldásával. A projekció elv alapján ugyanis a $||b - Ax||$ norma, azaz a $b \in R^m$ vektor „távolsága” az $Ax \equiv x_1 a_1 + \dots + x_n a_n$ alakú vektortól, azaz az A mátrix oszlopai által kifeszített R^r altértől akkor és csak akkor minimális, ha a $b - Ax$ vektor ortogonális az R^r altérre, tehát az alteret kifeszítő $a_1, \dots, a_n$ vektorokra. Tehát ha

$$(b - Ax, a_i) \equiv a_i^T (b - Ax) = 0, \quad i = 1, \dots, n,$$

ami nem más, mint a (6-13) normálegyenletek.

Megjegyezzük, hogy a (6-13) normálegyenletek klasszikus példája a gyengén meghatározott egyenletrendszereknek.

Ha az $A = U \Sigma V^T$ összefüggést behelyettesítjük (6-13)-ba, akkor

$$V \Sigma U^T U \Sigma V^T x = V \Sigma U^T b$$

adódik, és az $U^T U = V^T V = V V^T = I_n$ ortogonalitási feltétel miatt

$$\Sigma^2 V^T x = \Sigma U^T b \quad (6-14)$$

következik.

Ha az A rangja $r = n$, akkor Σ nem szinguláris és így

$$\mathbf{x} = \mathbf{V} \Sigma^{-1} \mathbf{U}^T \mathbf{b} = \mathbf{V} \Sigma^+ \mathbf{U}^T \mathbf{b} = \mathbf{A}^+ \mathbf{b}$$

a feladat egyetlen megoldása.

Ha A rangja $r < n$, akkor a (6–14) egyenlet

$$\begin{bmatrix} \Sigma_r^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_r \\ \mathbf{y}_s \end{bmatrix} = \begin{bmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_r \\ \mathbf{c}_s \end{bmatrix}$$

szerint particionálható, ahol

$$\mathbf{V}^T \mathbf{x} = \mathbf{y} = \begin{bmatrix} \mathbf{y}_r \\ \mathbf{y}_s \end{bmatrix} \quad \text{és} \quad \mathbf{U}^T \mathbf{b} = \mathbf{c} = \begin{bmatrix} \mathbf{c}_r \\ \mathbf{c}_s \end{bmatrix}$$

A (6–14) egyenlet particionált alakjából következik, hogy $\mathbf{y}_r$ a $\Sigma_r^2 \mathbf{y}_r = \Sigma_r \mathbf{c}_r$ egyenlet egyetlen megoldása és $\mathbf{y}_s$ tetszőleges vektor. Ha $\mathbf{y}_s = \mathbf{0}$, akkor nyilvánvaló, hogy

$\|\mathbf{y}\| = \min$. Ezzel az $\mathbf{y}^*$ vektorral képezett $\mathbf{x}^* = \mathbf{V} \mathbf{y}^*$ vektorra is $\|\mathbf{x}^*\| = \min$ érvényes, hiszen $\|\mathbf{x}^*\| = \|\mathbf{V} \mathbf{y}^*\| = (\mathbf{y}^{*T} \mathbf{V}^T \mathbf{V} \mathbf{y}^*)^{1/2} = \|\mathbf{y}^*\|$.

Ha az

$$\mathbf{y}_r = \Sigma_r^{-1} \mathbf{c}_r, \quad \mathbf{y}_s = \mathbf{0}$$

egyenleteket kifejezzük

$$\begin{bmatrix} \mathbf{y}_r \\ \mathbf{y}_s \end{bmatrix} = \begin{bmatrix} \Sigma_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{c}_r \\ \mathbf{c}_s \end{bmatrix},$$

vagyis

$$\mathbf{V}^T \mathbf{x}^* = \Sigma^+ \mathbf{U}^T \mathbf{b}$$

alakban, akkor ebből már következik, hogy

$$\mathbf{x}^* = \mathbf{V} \Sigma^+ \mathbf{U}^T \mathbf{b} = \mathbf{A}^+ \mathbf{b},$$

és ezzel az állítást igazoltuk.

6.1. példa. Lineáris regresszió. Tekintsük az

$$ax + b \equiv \begin{bmatrix} x & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = y \quad (6-15)$$

egyenes egyenletét. Ha az (x_i, y_i) , $i = 1, \dots, m$, $m > 2$ pontokhoz egyenest kívánunk illeszteni, akkor e pontok koordinátáinak a (6-15) egyenletbe való behelyettesítésével a

$$\begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix}$$

túlhatározott lineáris algebrai egyenletrendszert nyerjük. A legkisebb négyzetes megoldására a (6-12) összefüggés alkalmazható, miután elvégeztük az együtthatómátrix SVD felbontását.

A (6-12) implementációja a következő Pascal eljárás:

```

procedure SzEmegold(      U: tVmxn;
                          W: tVn;
                          V: tVnxn;
                          m,n: integer;
                          b: tVm;
                          var x: tVn);
{ Ez az eljárás alkalmas az A x = b egyenletrendszer
legkisebb normájú x legkisebb négyzetes megoldásának
meghatározására. Az A együtthatómátrixot a SzEfelbont
eljárással már felbontottuk és az eredményt az U, W és
V tömbökben tároljuk. A típusdefiníciók és paraméterek
leírását l. a SzEfelbont eljárásban.}
var
  i,j: integer;           {ciklus-, ill. indexváltozók}
  s: double;             {összegzés duplapontos változóban}
  xs: tVn;                {segédvektor a szorzások elvégzéséhez}
begin
  for i:=1 to n do                {xs = W+ UT b kiszámítása}
    begin s:=0.0;
      if w[i]<>0.0 then
        begin
          for j:=1 to m do
            s:=s+u[j,i]*b[j];
            s:=s/w[i];
          end;
          xs[i]:=s;
        end;
  for i:=1 to n do                {x = V xs kiszámítása}
    begin s:=0.0;
      for j:=1 to n do
        s:=s+v[i,j]*xs[j];
        x[i]:=s;
      end;
  end;

```


A SzEfelbont és az SzEmegold eljárás ugyanúgy használható, mint az LUFelbont és LUMegold eljárások: Mindkét esetben elvégezzük az A mátrix megfelelő felbontását, azután a jobb oldalak számától függően hívjuk a megfelelő egyenletrendszer megoldó eljárást. Különbség abban van, hogy az SzEmegold eljárás hívása előtt a következő programrészlettel előbb módosítani kell a szinguláris értékeket:

```

Procedure SzEnullaz(      var w: tvn;
                        n: integer);
{A SzEfelbont eljárással nyert, a szinguláris értékeket
 tartalmazó n elemű w tömb módosítása. Ez a kicsi
 elemek nullázását jelenti. A SzEnullaz eljárást a SzEmegold
 eljárás hívása előtt kell hívni.}
var
  i: integer;
  wmax: real;
begin wmax:=0;
  for i:=1 to n do {legnagyobb szinguláris érték keresése}
    if w[i]>wmax then wmax:=w[i];
  for i:=1 to n do {wmax-hoz képest kicsi elemek nullázása}
    if (wmax+w[i])=wmax then w[i]:=0;
end;
```


NEMLINEÁRIS EGYENLETRENDSZEREK

1. Alapfogalmak

Tekintsük az

$$f_i(x_1, \dots, x_n) = 0, \quad i = 1, \dots, m, \quad m \geq n$$

nemlineáris egyenletrendszert, amely tömörebben is kifejezhető

$$f(x) = 0, \quad f: R^n \rightarrow R^m, \quad m \geq n \quad (1-1)$$

alakban. Tehát az egyenlet bal oldalát alkotó $f(x) = [f_1(x), \dots, f_m(x)]^T$ vektor-vektor függvény egy leképezés az n -dimenziós valós térből az m -dimenziós valós térbe.

Az (1-1) egyenlet megoldása minden olyan $x \in R_n$ vektor, amelyre $f(x) = 0$.

Ha $m > n$, akkor az (1-1) egyenlet „túlhatározott” és általános esetben nincs megoldása.

Az $\|f(x)\|$ normát minimalizáló vektorokat az (1-1) egyenlet *általánosított megoldásának* nevezzük (az alkalmazott normára vonatkoztatva).

Speciálisan az

$$F(x) = f^T(x) f(x) \quad (1-2)$$

euklideszi normát (négyzetét) minimalizáló vektorokat az (1-1) egyenlet *legkisebb négyzetes megoldásának* nevezzük.

Mivel $F(x)$ az $f(x)$ függvény normája (négyzete), ezért $F(x) \geq 0$ és $F(x) = 0$ pontosan akkor, ha $f(x) = 0$. Tehát $F(x)$ abszolút minimumát, a zérust az (1-1) egyenlet megoldásain veszi fel, ezért az (1-2) funkcionál minimalizálására kidolgozott eljárások egyben a nemlineáris egyenletrendszerek megoldására is alkalmasak.

A továbbiakban szükségünk lesz a következő két *szimbólumra*:

(1) Az

$$f(h) = O(\|h\|^p)$$

szimbólum, ahol p nemnegatív egész szám, azt fejezi ki, hogy az $f(h) / \|h\|^p$ függvény korlátos, ha $h \rightarrow 0$. Vagyis létezik olyan $0 < c < \infty$ állandó, amelyre

$$\|f(h)\| \leq c \|h\|^p, \quad \text{ha } h \rightarrow 0.$$

(2) Az

$$f(h) = o(||h||^p)$$

szimbólum pedig azt jelenti, hogy

$$\lim_{h \rightarrow 0} \frac{||f(h)||}{||h||^p} = 0.$$

Az $O(\cdot)$ szimbólumon lényegében azt értjük, hogy „ $(\cdot)$ rendű”, $o(\cdot)$ pedig azt jelenti, hogy „elhagyható a $(\cdot)$ -hoz viszonyítva”.

Fréchet-derivált. Az $f: R^n \rightarrow R^m$ függvényt *Fréchet-deriválhatónak* nevezzük az x pontban, ha létezik x -ben a

$$\frac{df}{dx}(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x), \dots, \frac{\partial f_1}{\partial x_n}(x) \\ \vdots \\ \frac{\partial f_m}{\partial x_1}(x), \dots, \frac{\partial f_m}{\partial x_n}(x) \end{bmatrix} \quad (1-3)$$

Jacobi-mátrix és ha

$$f(x+h) = f(x) + \frac{df}{dx}(x)h + o(||h||). \quad (1-4)$$

Az $o(||h||)$ szimbólum értelmezésének megfelelően (1-4) azt jelenti, hogy

$$\lim_{h \rightarrow 0} \frac{||f(x+h) - f(x) - \frac{df}{dx}(x)h||}{||h||} = 0.$$

Tehát a határértékben a számláló gyorsabban tart a nullához, mint a nevező.

Megjegyezzük, hogy ha $f(x)$ az x pontban Fréchet-deriválható, akkor *folytonos* is x -ben (vagyis $||f(x+h) - f(x)|| \rightarrow 0$, ha $h \rightarrow 0$). Ez a $||f(x+h) - f(x)|| = ||f(x+h) - f(x) - \frac{df}{dx}(x)h + \frac{df}{dx}(x)h|| \leq ||f(x+h) - f(x) - \frac{df}{dx}(x)h|| + ||\frac{df}{dx}(x)h||$ egyenlőtlenségből közvetlenül következik, ha $h \rightarrow 0$.

1.1. tétel. Ha a Jacobi-mátrix folytonos az x pontban, vagy ami ugyanaz, ha

a $\frac{\partial f_i}{\partial x_j}$, $i = 1, \dots, m$, $j = 1, \dots, n$ parciális deriváltak folytonosak x -ben,

akkor az $f: R^n \rightarrow R^m$ függvény Fréchet-deriválható az x pontban.

A bizonyítás megtalálható pl. Ortega–Rheinboldt [8] könyvében.

1.1. *példa.* Ha az $F: R^n \rightarrow R$ függvény Fréchet-deriválható az x pontban, akkor (1-3)-ból $m = 1$ esetre következik, hogy a derivált mátrixreprezentációja a

$$\frac{dF}{dx}(x) = \left[\frac{\partial F}{\partial x_1}(x), \dots, \frac{\partial F}{\partial x_n}(x) \right] \quad (1-5)$$

sorvektor (a gradiensvektor transzponáltja).

Ha az $F: R^n \rightarrow R$ függvény Fréchet-deriválható az R^n tér valamely D résztartományának minden pontjában, akkor a $dF(x)/dx$ egy leképezés, amely minden $x \in D$ vektorhoz hozzárendeli az R^n tér egy vektorát. Ha a

$$\frac{dF}{dx}: D \subset R^n \rightarrow R^n$$

függvény Fréchet-deriválható az x pontban, akkor a

$$\frac{d}{dx} \left(\frac{dF}{dx}(x) \right)^T$$

kifejezés, amelyet $d^2F(x)/dx^2$ szimbólummal jelölünk, az $F: R^n \rightarrow R$ függvény másodrendű Fréchet-deriváltjának mátrixreprezentációja az x pontban.

1.2. *példa.* E definíciónak megfelelően képezzük az $F: R^n \rightarrow R$ függvény másodrendű Fréchet-deriváltjának a mátrixreprezentációját, vagyis alkalmazzuk az (1-3) alatti Jacobi-mátrixot az (1-5) alatti $dF(x)/dx$ függvényre (vektor transzponáltjára). Így a

$$\begin{aligned} \frac{d^2F}{dx^2}(x) &= \frac{d}{dx} \left(\frac{dF}{dx}(x) \right)^T = \frac{d}{dx} \begin{bmatrix} \frac{\partial F}{\partial x_1}(x) \\ \vdots \\ \frac{\partial F}{\partial x_n}(x) \end{bmatrix} = \\ &= \begin{bmatrix} \frac{\partial}{\partial x_1} \left(\frac{\partial F}{\partial x_1}(x) \right) & \dots & \frac{\partial}{\partial x_n} \left(\frac{\partial F}{\partial x_1}(x) \right) \\ \vdots & & \vdots \\ \frac{\partial}{\partial x_1} \left(\frac{\partial F}{\partial x_n}(x) \right) & \dots & \frac{\partial}{\partial x_n} \left(\frac{\partial F}{\partial x_n}(x) \right) \end{bmatrix} = H_F(x) \end{aligned} \quad (1-6)$$

szimmetrikus n -ed rendű mátrixot nyerjük, amelyet az F függvény Hesse-mátrixának nevezünk. Vagyis, az $F: R^n \rightarrow R$ függvény másodrendű Fréchet-deriváltjának mátrixreprezentációja az (1-6) alatti Hesse-mátrix.

1.3. példa.

- a) Ha az $\mathbf{a} = [a_1, \dots, a_n]^T$ vektor elemei állandók és $\mathbf{x} = [x_1, \dots, x_n]^T$ elemei változók, akkor

$$\frac{d}{dx} (\mathbf{a}^T \mathbf{x}) = \left[\frac{\partial}{\partial x_1} (\mathbf{a}^T \mathbf{x}), \dots, \frac{\partial}{\partial x_n} (\mathbf{a}^T \mathbf{x}) \right] = [a_1, \dots, a_n] = \mathbf{a}^T,$$

$$\frac{d^2}{dx^2} (\mathbf{a}^T \mathbf{x}) = \frac{d}{dx} \left(\frac{d}{dx} (\mathbf{a}^T \mathbf{x}) \right)^T = \frac{d}{dx} \mathbf{a} = \begin{bmatrix} 0, \dots, 0 \\ \vdots \\ 0, \dots, 0 \end{bmatrix} = \mathbf{0}.$$

- b) Ha az n -ed rendű $\mathbf{A}$ mátrix elemei állandók és $\mathbf{x} = [x_1, \dots, x_n]^T$ a változók vektora, akkor

$$\frac{d}{dx} (\mathbf{x}^T \mathbf{A} \mathbf{x}) = \left[\frac{\partial}{\partial x_1} (\mathbf{x}^T \mathbf{A} \mathbf{x}), \dots, \frac{\partial}{\partial x_n} (\mathbf{x}^T \mathbf{A} \mathbf{x}) \right] = \mathbf{x}^T (\mathbf{A}^T + \mathbf{A})$$

és

$$\frac{d^2}{dx^2} (\mathbf{x}^T \mathbf{A} \mathbf{x}) = \frac{d}{dx} (\mathbf{x}^T (\mathbf{A}^T + \mathbf{A}))^T = \mathbf{A}^T + \mathbf{A}.$$

Ezért szimmetrikus $\mathbf{A}$ mátrix esetén

$$\frac{d}{dx} (\mathbf{x}^T \mathbf{A} \mathbf{x}) = 2 \mathbf{x}^T \mathbf{A}$$

és

$$\frac{d^2}{dx^2} (\mathbf{x}^T \mathbf{A} \mathbf{x}) = 2 \mathbf{A}.$$

A konvergencia gyorsasága. Az $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k, \dots$ vektorsorozatról akkor mondjuk, hogy p -ed rendű gyorsasággal tart az $\mathbf{x}^*$ vektorhoz, ha $k \rightarrow \infty$ esetén fennáll az

$$||\mathbf{x}_{k+1} - \mathbf{x}^*|| = c ||\mathbf{x}_k - \mathbf{x}^*||^p + o(||\mathbf{x}_k - \mathbf{x}^*||^p) \quad (1-7)$$

összefüggés, ahol $0 \leq c < \infty$. Tehát ha

$$\lim_{k \rightarrow \infty} \frac{||\mathbf{x}_{k+1} - \mathbf{x}^*||}{||\mathbf{x}_k - \mathbf{x}^*||^p} = c. \quad (1-8)$$

Ha $p = 1$, azaz lineáris konvergencia esetén feltételezzük, hogy $c < 1$, és $c = 0$ esetben a konvergenciát szuperlineárisnak mondjuk.

2. Newton-típusú módszerek

Newton-módszer

Tegyük fel, hogy az $F : D \subset R^n \rightarrow R$ függvény az $x_k \in D$ pontban kétszer Fréchet-deriválható. Akkor minden $x = x_k + h \in D$ pontban érvényes az

$$F(x) = F(x_k) + \frac{dF}{dx}(x_k)(x - x_k) + \frac{1}{2}(x - x_k)^T \frac{d^2F}{dx^2}(x_k)(x - x_k) + o(||x - x_k||^2) \quad (2-1)$$

összefüggés, ahol dF/dx az (1-5) alatti gradiensvektor (transzponáltja), d^2F/dx^2 pedig az F függvény (1-6) alatti Hesse-mátrixa (lásd pl. Ortega-Rheinboldt [8], 8.1 pontját). Ezért az x_k pont elegendően kis környezetében az F függvény közelíthető az

$$F(x_k + h) = F(x_k) + \frac{dF}{dx}(x_k)h + \frac{1}{2}h^T H_F(x_k)h + \dots \quad (2-2)$$

Taylor-sorának első három tagjával.

Ha x_k az $F(x)$ függvény $x^* = x_k + h^k$ minimalizáló vektorának közelítése, akkor a minimum (stacionaritási pont) szükséges feltétele miatt, a (2-1) összefüggés x vektor szerinti deriválásával

$$\frac{dF}{dx}(x^*) \equiv \frac{dF}{dx}(x_k) + (x^* - x_k)^T \frac{d^2F}{dx^2}(x_k) + \dots = 0^T,$$

vagyis

$$\left(\frac{dF}{dx}(x_k) \right)^T + H_F(x_k)h^k + O(||h^k||^2) = 0 \quad (2-3)$$

adódik.

Ha csak a lineáris tagokra szorítkozunk, akkor ez az egyenlet a

$$H_F(x_k)h = - \left(\frac{dF}{dx}(x_k) \right)^T \quad (2-4)$$

lineáris algebrai egyenletrendszerre egyszerűsödik. A csonkítás miatt azonban megoldásával a pontos h^k értéknek csak egy h_k közelítését nyerjük és így

$$x_{k+1} = x_k + h_k \quad (2-5)$$

az $x^* = x_k + h^k$ minimalizáló vektornak csak egy új közelítése.

A (2-4), (2-5) összefüggésekkel értelmezett iterációt, amely egy $F: R^n \rightarrow R$ függvény lokális minimumpontjának meghatározására szolgál, *Newton-módszernek* nevezzük.

A (2-3) egyenletből kitűnik, hogy a Newton-módszer *konvergenciagyorsasága másodrendű*. Valóban, feltéve, hogy a $H_F(x_k)$ mátrix nem szinguláris, szorozzuk meg a (2-3) egyenletet balról a $H_F^{-1}(x_k)$ inverz mátrixszal. Így a

$$H_F^{-1}(x_k) \left(\frac{dF}{dx}(x_k) \right)^T + h^k + O(||h^k||^2) = 0$$

egyenletet nyerjük, amelyben az első tag a (2-4) egyenlet megoldásaként kapott $-h^k = x_k - x_{k+1}$ vektor, s mivel $h^k = x^* - x_k$ ez az egyenlet átírható az

$$x^* - x_{k+1} + O(||h^k||^2) = x^* - x_{k+1} + O(||x^* - x_k||^2) = 0$$

alakra. Az $O(\cdot)$ definíciója alapján létezik egy olyan $0 < c < \infty$ állandó, hogy

$$||x_{k+1} - x^*|| \leq c ||x_k - x^*||^2, \quad \text{ha} \quad x_k \rightarrow x^*,$$

vagyis

$$\lim_{k \rightarrow \infty} \frac{||x_{k+1} - x^*||}{||x_k - x^*||^2} = c,$$

ami nem más mint az (1-8) formula $p = 2$ esetén.

Gauss-Newton-módszer

Emlékeztetünk arra, hogy $m > n$ esetén az

$$f(x) = 0, \quad f: R^n \rightarrow R^m \quad (2-6)$$

„túlhatározott” egyenletrendszernek általános esetben nincs megoldása, és az

$$F(x) = f^T(x) f(x), \quad f: R^n \rightarrow R^m, \quad m \geq n \quad (2-7)$$

funkcionál minimalizáló vektora a (2-6) egyenlet *legkisebb négyzetes megoldása*.

A (2-7) alatti $F(x)$ függvény x vektor szerinti deriválásával

$$\frac{dF}{dx} = \frac{dF}{df} \frac{df}{dx} = 2 f^T(x) J(x) \quad (2-8)$$

adódik, ahol $J(x)$ az (1-3) alatti $m \times n$ típusú Jacobi-mátrix. Az F függvény Hesse-mátrixára pedig

$$\begin{aligned}
 H_F &= \frac{d^2 F}{dx^2} = \frac{d}{dx} \left(\frac{dF}{dx} \right)^T = \frac{d}{dx} (2 J^T(x) f(x)) = \\
 &= 2 \frac{dJ^T}{dx}(x) f(x) + 2 J^T(x) \frac{df}{dx}(x) = 2 \sum_{i=1}^m f_i H_{f_i} + 2 J^T(x) J(x) \quad (2-9)
 \end{aligned}$$

adódik, ahol H_{f_i} az $f(x)$ vektor-vektor függvény i -edik elemének Hesse-mátrixa.

Ugyanis:

$$\begin{aligned}
 \frac{dJ^T}{dx}(x) &= \frac{d}{dx} \left[\left(\frac{df_1}{dx} \right)^T, \dots, \left(\frac{df_m}{dx} \right)^T \right] = \left[\frac{d}{dx} \left(\frac{df_1}{dx} \right)^T, \dots, \frac{d}{dx} \left(\frac{df_m}{dx} \right)^T \right] = \\
 &= \left[\frac{d^2 f_1}{dx^2}, \dots, \frac{d^2 f_m}{dx^2} \right] = [H_{f_1}, \dots, H_{f_m}].
 \end{aligned}$$

Ha a (2-9)-ben szereplő $2 \sum_{i=1}^m f_i H_{f_i}$ tagot elhagyjuk és az ezáltal csonkított H_F mátrixot és a (2-8) alatti dF/dx vektort behelyettesítjük a (2-4) alatti Newton-formulába, akkor a

$$J^T(x_k) J(x_k) h = -J^T(x_k) f(x_k) \quad (2-10)$$

formulát nyerjük, amely a (2-5) alatti

$$x_{k+1} = x_k + h_k$$

összefüggéssel együtt képezi a Gauss-Newton-módszert, amely alkalmas a (2-7) funkcionál minimalizálására, vagyis a (2-6) túlhatározott nemlineáris egyenletrendszer legkisebb négyzetes megoldására.

A Gauss-Newton-módszer így nem egyenértékű a (2-4), (2-5) alatti Newton-iteráció alkalmazásával az $F(x) = f^T(x) f(x)$ speciális alakú funkcionál minimalizálására, mivel annál – a $2 \sum_{i=1}^m f_i H_{f_i}$ tag elhagyása miatt – pontatlanabb. A (2-8) alatti dF/dx vektorral és a (2-9) alatti H_F mátrixszal a (2-3) egyenlet a

$$\begin{aligned}
 2 J^T(x_k) f(x_k) + 2 \sum_{i=1}^m f_i H_{f_i} h^k + 2 J^T(x_k) J(x_k) h^k + \\
 + O(\|h^k\|^2) = 0 \quad (2-11)
 \end{aligned}$$

alakot ölti. Mivel a (2-10) alatti Gauss-Newton-formula alkalmazása a h^k -t lineárisan tartalmazó $2 \sum_{i=1}^m f_i H_{f_i} h^k$ tagot általában nem tűnteti el, így a Gauss-Newton-módszer konvergenciagyorsasága csak elsőrendű. (Részletesebben, a (2-10) for-

mula alapján a (2-11)-ben szereplő első tag kifejezhető $-2\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)\mathbf{h}_k$ alakban és így (2-11) átírható a

$$\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)(-\mathbf{h}_k + \mathbf{h}^k) + \sum_{i=1}^m f_i \mathbf{H}_{f_i} \mathbf{h}^k + O(\|\mathbf{h}^k\|^2) = 0$$

alakra, amelyből $\mathbf{h}^k - \mathbf{h}_k = \mathbf{x}^* - \mathbf{x}_k - (\mathbf{x}_{k+1} - \mathbf{x}_k) = \mathbf{x}^* - \mathbf{x}_{k+1}$ miatt már $\mathbf{x}^* - \mathbf{x}_{k+1} + \left(\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)\right)^{-1} \sum_{i=1}^m f_i \mathbf{H}_{f_i} \mathbf{h}^k + O(\|\mathbf{h}^k\|^2) = \mathbf{x}^* - \mathbf{x}_{k+1} + O(\|\mathbf{h}^k\|) = 0$ következik.)

Newton-Raphson-módszer

Ha $m = n$, azaz ha a (2-6) egyenletrendszer nem túlhatározott, akkor várható, hogy az

$$\mathbf{f}(\mathbf{x}) = 0, \quad \mathbf{f}: \mathbb{R}^n \rightarrow \mathbb{R}^n \quad (2-12)$$

nemlineáris egyenletrendszernek van megoldása. Ekkor a (2-10) formulában szereplő Jacobi-mátrix általában nem szinguláris és így a Gauss-Newton-módszer a (2-12) nemlineáris egyenletrendszer megoldására alkalmas

$$\mathbf{J}(\mathbf{x}_k)\mathbf{h} = -\mathbf{f}(\mathbf{x}_k) \quad (2-13)$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k$$

Newton-Raphson-módszerrel ekvivalens. Ennek az ekvivalenciának ellenére a Newton-Raphson-módszer konvergenciája másodrendű. Ez azért van így, mert $m = n$ esetben, amikor létezik az $\mathbf{f}(\mathbf{x}^*) = 0$ megoldás, akkor $k \rightarrow \infty$ esetén $f_i(\mathbf{x}_k) \rightarrow 0$ minden i indexre és így (2-11)-ben a $2 \sum_{i=1}^m f_i \mathbf{H}_{f_i} \mathbf{h}^k$ lineáris tag is eltűnik.

A következő Newton-Raphson eljárás, kiindulva egy n elemű $\mathbf{x}$ vektorból mint a (2-12) egyenlet megoldásának kezdő közelítéséből, iter számú Newton-Raphson-iterációs lépést hajt végre. Az iterációt befejezi, ha a $\sum_{i=1}^n |f_i(\mathbf{x}_k)| < \text{tol}f$,

$\sum_{j=1}^n |h_j| < \text{tol}x$ feltételek egyike teljesül, ahol $h_1, \dots, h_n$ a $\mathbf{h}$ vektor elemei és $\text{tol}f$, $\text{tol}x$ két adott korlátszám. A felhasználónak még az fgvnx1 rutinként meg kell adni a (2-13) egyenlet baloldalán szereplő $\mathbf{J}(\mathbf{x}_k)$ Jacobi-mátrix elemeit és a jobb oldalon szereplő $-\mathbf{f}(\mathbf{x}_k)$ vektor elemeit (tehát negatív előjellel). Az iter korlátszámot nem célszerű nagynak megadni. Inkább néhány lépés végrehajtása után (mielőtt az iterációt folytatnánk) nézzük meg a részeredményeket.


```

procedure NewtonRaphson(var xk: tvN;
                        n: integer;
                        tolX, tolf: real;
                        var iter: integer);
{ Alkalmas n egyenletből álló n ismeretlenes nemlineáris
  egyenletrendszer megoldására tolX és tolf pontossággal. Az
  eljárás legfeljebb iter számú Newton-Raphson-iterációs
  lépést hajt végre. Az xk vektor bemenetkor a kezdővektort,
  kimenetkor a megoldást tartalmazza. Az iter paraméter
  kimenetkor a végrehajtott iterációk számát mutatja.
  A NewtonRaphson eljárást használó programban definiálni
  kell a következő globális típusokat
  tvNxn = array [1..nmax, 1..nmax] of real;
  tIn = array [1..nmax] of integer;
  tvN = array [1..nmax] of real;
  ahol az nmax konstans az n legnagyobb lehetséges értéke. A
  minimalizálandó függvényt a
  procedure fgvnxn( x: tvN;
                  n: integer;
                  var J: tvNxn;
                  var fneg: tvN);
  eljárásként kell megadni, ahol x az aktuális pontot
  tartalmazó n elemű vektor, n az egyenletek (és
  ismeretlenek) száma, J a parciális deriváltak értékeit
  tartalmazó n-ed rendű Jacobi-mátrix ( $J[i,j] = df_i/dx_j(x)$ ),
  fneg a függvényértékeket tartalmazza negatív előjellel. }
var
  i, k: integer;           {ciklusváltozó és iterációs számláló}
  s: real;                 {segédváltozó a normák számításához}
  hk: tvN;                 {segédvektor}
  J: tvNxn;                {a Jacobi-mátrixot tartalmazó mátrix}
  icsere: tIn;              {az LU felbontáshoz}
  vege: boolean;           {az iteráció sikeres befejezése}
begin k:=0;
  repeat inc(k); fgvnxn(xk,n,J,hk);           {hk-ban -f lesz}
    s:=0; for i:=1 to n do s:=s+abs(hk[i]); { |f| számítása}
    vege:=s<=tolf;
    if not vege then
      begin
        LUFelbont(J,n,icsere,i);           {(2-13) első egyen-}
        LUMegold(J,n,icsere,hk);           { letének megoldása}
        s:=0;
        for i:=1 to n do
          begin s:=s+abs(hk[i]);           { |hk| számítása}
            xk[i]:=xk[i]+hk[i];           {(2-13) második egyenlete}
          end;
        vege:=s<=tolX;
      end;
  until vege or (k>iter);           {iteráció vége}
  if not vege then
    Hiba('Az iterációk száma nagyobb a megengedettnél.')
  else iter:=k;
end;

```


Gauss–Newton-iteráció egydimenziós kereséssel

Ha a (2–10) alatti egyenlet jobb oldalát átírjuk a (2–8) összefüggés felhasználásával, akkor a Gauss–Newton-egyenletrendszer a

$$\mathbf{J}^T(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k) \mathbf{h} = -\frac{1}{2} \left(\frac{dF}{dx}(\mathbf{x}_k) \right)^T \quad (2-14)$$

alakot ölti. Az együtthatómátrix nyilvánvalóan nemnegatív definit (hiszen minden $\mathbf{h} \neq \mathbf{0}$ vektorra $\mathbf{h}^T \mathbf{J}^T \mathbf{J} \mathbf{h} = \|\mathbf{J} \mathbf{h}\|^2 \geq 0$). Ha feltesszük, hogy pozitív definit és $\frac{dF}{dx}(\mathbf{x}_k) \neq 0^T$, azaz $\mathbf{x}_k$ nem stacionaritási pont, akkor (2–14)-ből

$$\mathbf{h}_k^T \mathbf{J}^T(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k) \mathbf{h}_k = -\frac{1}{2} \mathbf{h}_k^T \left(\frac{dF}{dx}(\mathbf{x}_k) \right)^T > 0$$

következik. Tehát ha $\mathbf{x}_k$ nem stacionaritási pont, akkor teljesül a „csökkenési tulajdonságnak” nevezett

$$\frac{dF}{dx}(\mathbf{x}_k) \mathbf{h}_k < 0 \quad (2-15)$$

egyenlőtlenség.

Keressük most azt a λ_k számot, amelyre $F(\mathbf{x}_k + \lambda \mathbf{h}_k)$ felveszi a minimumát! Vagyis, az $\mathbf{x}_k$ ponton áthaladó, $\mathbf{h}_k$ vektor irányú $x(\lambda) = \mathbf{x}_k + \lambda \mathbf{h}_k$ egyenes mentén keressük az $F(x(\lambda))$ függvényt minimalizáló λ_k számot.

Ha a (2–2) Taylor-sorban $\mathbf{h}$ -t $\lambda \mathbf{h}_k$ -val helyettesítünk, akkor

$$F(\mathbf{x}_k + \lambda \mathbf{h}_k) = F(\mathbf{x}_k) + \frac{dF}{dx}(\mathbf{x}_k) \lambda \mathbf{h}_k + \frac{1}{2} (\lambda \mathbf{h}_k)^T \frac{d^2F}{dx^2}(\mathbf{x}_k) (\lambda \mathbf{h}_k) + \dots$$

adódik és a minimum létezésének szükséges feltétele:

$$\frac{\partial F}{\partial \lambda}(\mathbf{x}_k + \lambda \mathbf{h}_k) \equiv \frac{dF}{dx}(\mathbf{x}_k) \mathbf{h}_k + \lambda \mathbf{h}_k^T \frac{d^2F}{dx^2}(\mathbf{x}_k) \mathbf{h}_k + \dots = 0 \quad (2-16)$$

Ha a Gauss–Newton-módszernek megfelelően $\frac{d^2F}{dx^2}(\mathbf{x}_k) \approx 2 \mathbf{J}^T(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k)$ és a

fenti feltételezés miatt ez a mátrix pozitív definit, akkor a (2–16) egyenlet csonkított alakjából – a (2–15) egyenlőtlenség miatt – következik, hogy $\lambda > 0$. Tehát ha $\mathbf{x}_k$ nem minimalizáló vektor, akkor létezik olyan λ_k pozitív szám, amely minimalizálja az $F(\mathbf{x}_k + \lambda \mathbf{h}_k)$ függvényt.

Ha tehát a Gauss–Newton-módszert kombináljuk ezzel a λ -szerinti egyváltozós minimalizálással, akkor ez elvileg biztosítja a Gauss–Newton-módszer konvergenciáját.

A megfelelő számítási algoritmus ekkor egy iteráció, amelynek k -adik lépése a következő három részfeladatból tevődik össze:

(1) A $\mathbf{h}_k$ irány meghatározása az (2–10) alatti lineáris algebrai egyenletrendszer megoldásával.

(2) Annak a $\lambda = \lambda_k$ számnak a kiszámítása, amely minimalizálja az $F(\mathbf{x})$ függvényt az $\mathbf{x}(\lambda) = \mathbf{x}_k + \lambda \mathbf{h}_k$ egyenes mentén.

(3) Az

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{h}_k$$

további közelítés kiszámítása.

3. Egydimenziós minimalizálás

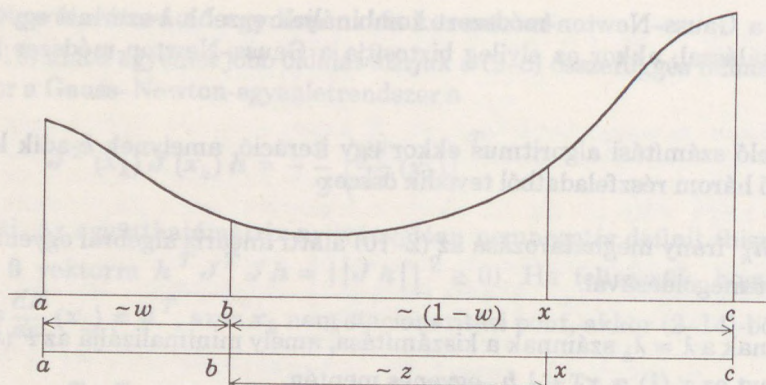
Az egydimenziós függvény minimalizálására a szakirodalomban több módszer kínálkozik. Ezek közül két, a gyakorlatban jól bevált eljárás ismertetésére szorítkozunk. Az egyik a mindig konvergáló, de meglehetősen lassú *arany metszés szerinti keresés*, a másik a *Brent-módszer*. Ez utóbbi kombinálja az arany metszés szerinti keresést a gyorsabb, de nem mindig konvergens parabolikus interpoláció módszerével. A Brent-módszernek azt a változatát ismertetjük, amelynél a függvény deriváltjára is szükség van.

Arany metszés szerinti keresés

Az $f: R \rightarrow R$ függvény minimumát akkor tekintjük *közrefogottnak*, ha ismerünk egy olyan $a < b < c$ *ponthármas*t, amelyre $f(b) < f(a)$ és $f(b) < f(c)$ érvényes. Ekkor ugyanis a (nem szinguláris) függvénynek az (a, c) intervallumon biztosan van (lokális) minimuma.

Célunk a közrefogás finomítása, azaz az (a, c) intervallum hosszának fokozatos csökkentése. E célból az (a, b) , (b, c) intervallumok egyikében felveszünk egy új x pontot. Legyen ez pl. a (b, c) intervallumban. Kiszámítjuk az $f(x)$ értéket. Ha $f(b) < f(x)$, akkor $a < b < x$ az új közrefogási ponthármas; ha $f(b) > f(x)$, akkor $b < x < c$ az új ponthármas. Az esetek mindegyikében az a szabály, hogy az új ponthármas középpontja az az abszcissza, amelynek ordinátája kisebb a két szélső pont ordinátájánál (lásd a 3.1. ábrát). Ezt az eljárást addig folytatjuk, ameddig a ponthármas külső pontjainak távolsága még nagyobb, mint a megadott tolerancia.

Felvetődik a kérdés, hogy ez a tolerancia mennyire kicsinek adható meg. Ha ϵ a számítógép lebegőpontos pontossága (azaz, ha ϵ az a legkisebb ábrázolható szám,



3.1. ábra. A közrefogás finomítása

amelyre $1 + \varepsilon > 1$ és $1 - \varepsilon < 1$ és ha b a minimum helye, akkor azt hihetnénk, hogy a minimum közrefogható az $(1 - \varepsilon)b < b < (1 + \varepsilon)b$ intervallumban. Ez azonban nincs így! A Taylor-tétel alkalmazásával az $f(x)$ függvény a b pont közelében kifejezhető

$$f(x) \approx f(b) + \frac{1}{2} f''(b) (x - b)^2 \quad (3-1)$$

alakban. A második tag elhagyható az elsőhöz viszonyítva, ha

$$\frac{\frac{1}{2} f''(b) (x - b)^2}{|f(b)|} < \varepsilon,$$

azaz ha

$$|x - b| < \sqrt{\varepsilon} |b| \left(\frac{2 |f(b)|}{b^2 f''(b)} \right)^{1/2}. \quad (3-2)$$

A jobb oldalt azért fejeztük ki ilyen alakban, mert a függvények többségére az utolsó négyzetgyök nagyságrendje 1-gyel egyenlő. Ezért a közrefogási intervallum hosszát nem érdemes kisebbnek megadni, mint amilyen középértékének $\sqrt{\varepsilon}$ -szorosa. Ennek tudatában sok felesleges számítást takaríthatunk meg. Az ebben a szakaszban ismertetett eljárásokhoz a ± 1 paramétert célszerű valamivel kisebbnek megadni, mint amilyen a gépi lebegőpontos pontosságának négyzetgyöke. Ekkor a minimalizáló abszcisszákat ± 1 pontossággal nyerjük.

Még meg kell mutatni, hogy adott a, b, c pontok ismeretében milyen stratégiával határozzuk meg az új x pontot. Tegyük fel, hogy a b pont helyzetét az a, c pontokhoz viszonyítva a következő w tört fejezi ki

$$\frac{b - a}{c - a} = w, \text{ következésképpen } \frac{c - b}{c - a} = 1 - w. \quad (3-3)$$

Tegyük fel továbbá, hogy az x pont helyzetét a b ponttól jobbra az

$$\frac{x-b}{c-a} = z \quad (3-4)$$

tört határozza meg. Ekkor a következő közrefogási szakasz hossza az aktuális szakaszhoz viszonyítva vagy $w + z$, vagy $1 - w$. Annak érdekében, hogy a legkedvezőtlenebb eset előfordulását a minimumra csökkentsük, a z értéket olyannak választjuk, hogy $w + z = 1 - w$, vagyis

$$z = 1 - 2w \quad (3-5)$$

legyen. Ekkor az x pont az (a, c) intervallumon a b ponttal szimmetrikusan helyezkedik el, vagyis $|b - a| = |x - c|$ (lásd a 3.1. ábrát). Ebből az is következik, hogy az x pont a két szakasz közül a nagyobbikban van (z csak akkor pozitív, ha $w < 1/2$). A nagyobbikban van, azonban pontosan hol? Ha feltesszük, hogy a w értéket az előbbi lépésben ugyanúgy optimálisnak vettük fel, mint ebben a lépésben a z értéket, akkor ez azt is jelenti, hogy az x pontnak a (b, c) intervallumon ugyanolyan arányokkal kell elhelyezkednie, mint a b pontnak az (a, c) intervallumon. Tehát teljesülnie kell az

$$\frac{x-b}{c-b} = \frac{b-a}{c-a}$$

összefüggésnek, amely átírható

$$\frac{z}{1-w} = w \quad (3-6)$$

alakra. A (3-5) és a (3-6) egyenletből

$$w^2 - 3w + 1 = 0$$

következik és ennek az egyenletnek

$$w = \frac{3 - \sqrt{5}}{2} \approx 0.38197 \quad (3-7)$$

az egyik megoldása. (A másik megoldás nagyobb 1-nél, ezért nem használható.)

Másképpen, az $a < b < c$ optimális közrefogási intervallum belső pontja $0.38197 * (c - a)$ távolságra van az egyik, pl. az a végponttól és $0.61803 * (c - a)$ a távolsága a másik, pl. a c végponttól. Tehát a $b - a$ és a $c - b$ szakaszok hosszának aránya azonos az ókori görögök által is ismert, ún. *aranymetszéssel* vagy *aranyközéppel*. A függvény minimalizálásnak ezt az optimális módját felhasználó és a (zérushelyek finomítására szolgáló) felezési eljárásra emlékeztető módszert ezért *aranymetszés szerinti keresésnek* nevezzük. A módszer a következőképpen foglalható össze:

Minden lépésben adva van egy $a < b < c$ közrefogási ponthármas. Jelölje l az (a, b) és a (b, c) intervallumok nagyobbikának hosszát. Az új x pont a hosszabb intervallumba kerül, $0.38197 * l$ távolságra a b ponttól. A kezdő közrefogási

ponthármas szakaszainak hossza általában nincs arányban, az eljárás folyamán azonban ez a kívánt arány gyorsan beáll.

Az aranymetszés szerinti keresés minden egyes lépése közrefogja a minimalizáló pontot egy olyan intervallummal, amelynek hossza az előző intervallum hosszának a 0.61803-szorosa. (Ez az arány csak kevéssel rosszabb, mint a (zérushelyek közrefogására szolgáló) felezési eljárás által biztosított 0.50000 tényező.) Megjegyezzük, hogy az aranymetszés szerinti keresés konvergenciája *lineáris*, vagyis érvényes az (1-8) formula $p = 1$ kitevővel. Az aranymetszés szerinti keresés implementációját a következő Pascal eljárás tartalmazza:

```
function Aranymetszes( a,b,c,
                      tol: real;
                      var xmin: real): real;
{ A függvényeljárás alkalmas a következőekben részletezett
  fgv(x) eljárással értelmezett függvény xmin minimalizáló
  pontjának meghatározására. A minimalizáló pontot az a<b<c
  ponthármas fogja közre. A keresés addig tart, amíg a
  közrezárt intervallum  $|b-a|/0.5/(|a|+|b|)$  relatív hossza
  nagyobb, mint a tol korlátszám. A függvényeljárás értéke az
  xmin minimalizáló pontbeli függvényérték.
  Az Aranymetszes függvényeljárást hívó programnak
  tartalmaznia kell a
  function fgv(x: real): real;
  deklarációjú függvényeljárást, mely a minimalizálandó
  függvény értékét határozza meg az x pontban.}
const
  r=0.61803399;
var
  f0,f1,f2,f3,s: real;
  x0,x1,x2,x3: real;
begin s:=1.0-r; x0:=a; x3:=c; {kezdőpontok meghatározása}
  if abs(c-b)>abs(b-a) then {A nagyobb részt osztjuk!}
    begin x1:=b; x2:=b+s*(c-b); end
  else
    begin x2:=b; x1:=b-s*(b-a); end;
  f1:=fgv(x1); f2:=fgv(x2);
  while abs(x3-x0)>tol*(abs(x1)+abs(x2)) do
    if f2<f1 then
      begin x0:=x1; x1:=x2; x2:=r*x1+s*x3; {szűkítés:}
        f0:=f1; f1:=f2; f2:=fgv(x2); {- balról}
      end
    else
      begin x3:=x2; x2:=x1; x1:=r*x2+s*x0; {-jobbról}
        f3:=f2; f2:=f1; f1:=fgv(x1);
      end;
    if f1<f2 then {eredmény a kimenő változóba}
      begin Aranymetszes:=f1; xmin:=x1; end
    else
      begin Aranymetszes:=f2; xmin:=x2; end;
  end;
```


A minimum kezdő közrefogása

Az előbbi eljárás indításához szükség van egy kezdő közrefogási ponthármas ismeretére. Meghatározására pl. a következő egyszerű eljárás alkalmazható, amelynek az a lényege, hogy kiindulva két adott pontból, az azok által meghatározott egyenes mentén lépegetünk „lefelé”. A lépésközt lépésenként növeljük, vagy egy állandó szorzó tényezővel, vagy azt az előző pontokon átfektetett parabola fordulópontjáiig nyújtjuk. Az sem baj, ha a lépések túl nagyra sikerülnek. A lefelé haladás miatt a közrefogási ponthármas egyik szélső és belső pontját már meghatároztuk, és így már csak elegendően hosszú lépésre van szükség ahhoz, hogy befejezzük a lefelé haladást és megkapjuk az elég „magas” harmadik pontot.

A fentieket a következő eljárás valósítja meg:

```

procedure Kozrefog(var a,b,c,
                   fa,fb,fc: real);
{ Alkalmas az fgv függvényeljárással értelmezett függvény
minimumpontjának kezdő közrefogására. Az eljárás az a és b
pontokkal indul és addig tart, ameddig a minimumot közre nem
zárta. A közrezáró ponthármaszt az a, b, c változókbán
kapjuk meg. Fa, fb és fc a megfelelő függvényértékeket
tartalmazza.
  A Kozrefog eljárást felhasználó programnak tartalmaznia
kell a
  function fgv(x: real): real;
deklarációjú függvényeljárást, mely a minimalizálandó
függvény értékét határozza meg az x pontban.}
const
  gold    = 1.618034;
  glimit  = 100.0;
  tiny    = 1.0e-20;
var
  ulim,u,r,q,fu,dum: real;
  n:boolean;
begin fa:=fgv(a); fb:=fgv(b);
  if fb>fa then {keresési irány megfordítása}
    begin dum:=a; a:=b; b:=dum;
      dum:=fb; fb:=fa; fa:=dum;
    end;
  c:=b+gold*(b-a); fc:=fgv(c); {első próbálkozás c-re}
  while fb>fc do
    begin r:=(b-a)*(fb-fc); q:=(b-c)*(fb-fa); n:=true;
      u:=abs(q-r); if tiny>u then u:=tiny;
      if r>q then u:=-u;
      u:=b-((b-c)*q-(b-a)*r)/(2*u); {a parabola minimuma}
      ulim:=b+glimit*(c-b); {ulim-nél nem mehet messzebb}
      if (b-u)*(u-c)>0.0 then {u b és c között}
        begin fu:=fgv(u);
          if fu<fc then {a minimum b és c között}
            begin a:=b; fa:=fb; b:=u; fb:=fu; n:=false; end
          else if fu>fb then {a minimum a és u között}
            begin c:=u; fc:=fu; n:=false end
          else {a parabolaefektetés nem sikerült}

```



```

begin u:=c+gold*(c-b); fu:=fgv(u); end;
end
else if (c-u)*(u-ulim)>0.0 then {u c és ulim között}
begin fu:=fgv(u);
if fu<fc then
begin b:=c; c:=u; u:=c+gold*(c-b);
fb:=fc; fc:=fu; fu:=fgv(u);
end;
end
else if (u-ulim)*(ulim-c)>=0.0 then
begin u:=ulim; fu:=fgv(u); end
else {parabolafektetés helyett aranymetszés}
begin u:=c+gold*(c-b); fu:=fgv(u); end;
if n then
begin a:=b; b:=c; c:=u;
fa:=fb; fb:=fc; fc:=fu;
end;
end;
end;
end;

```

Parabolikus interpoláció és a Brent-módszer

Az aranymetszés szerinti keresés az egydimenziós függvényminimalizálás legkedvezőtlenebb eseteinek megoldására is alkalmas. Az eljárás „megbízhatóságának” lassú konvergencia az ára. Ha a minimum közelében a függvény elegendően sima (legalább kétszer folytonosan deriválható), akkor a minimalizálás meggyorsítható a *másodfokú inverz interpoláció* alkalmazásával. Ennek az a lényege, hogy az $[a, f(a)]$, $[b, f(b)]$ és $[c, f(c)]$ pontokon másodfokú parabolát fektetünk át, és meghatározzuk annak minimalizáló pontját, azaz az

$$x = b + \frac{1}{2} \frac{(b-a)^2 [f(b) - f(c)] - (b-c)^2 [f(b) - f(a)]}{(b-a)[f(b) - f(c)] - (b-c)[f(b) - f(a)]} \quad (3-8)$$

abszcisszát. Ez a formula csak akkor nem használható, ha mind a három illesztési pont egy egyenesen van, ekkor ugyanis a nevezőben zérus áll. A (3-8) formula azonban a parabola stacionaritási pontját eredményezi, így x nemcsak minimalizáló, hanem maximalizáló pont is lehet. Ezért nem várható, hogy csupán a (3-8) formula alapján tervezett, „gyorsabb” függvényminimalizálás kielégítően működjön a gyakorlatban.

A cél egy olyan technika kidolgozása, amely megőrzi az aranymetszés szerinti keresés megbízhatóságát, ugyanakkor azonnal átvált a (3-8) képlet szerinti minimalizálásra, amint ezt a függvény alakja megengedi. Az ilyen technika megvalósítása több ok miatt sem egyszerű. Bonyolult pl. a két módszer közötti váltás gazdaságos, azaz olyan megszervezése, amelynél nem számolunk feleslegesen függvényértékeket. A „végjáték” megtervezése sem okoz kisebb gondot, hiszen ekkor már a (3-2) egyenlet kerekítési korlátjához közeli függvényértékekkel számolunk. Nagyon nehéz annak az eldöntése is, hogy a függvény alakja mikor indokolja a két módszer közötti váltást.

A felsorolt nehézségek figyelembevételével készült az 1973-ban közölt *Brent-módszer*. A módszer minden lépésében a következő hat (nem szükségképpen

különböző) függvénypont nyilvántartása alapján alakítja taktikáját. Az a, b pontok közrefogják a minimumot, x az eddigi számítások során kapott legkisebb függvényértékű pont (ha több ilyen pont van, akkor x az utolsóként kiszámított pont abszcisszája); w a második legkisebb függvényértékű pont (abszcisszája) és v az előző w értéke; végül u az a pont, amelyben utoljára számítottunk függvényértéket. Az algoritmus nyilvántartja még az $x_m = (a + b) / 2$ pontot is, azonban az x_m -beli függvényérték kiszámítására nincs szükség. További megjegyzéseink a következőkben adott eljárás könnyebb megértését szolgálják. Az x, v és w pontokon parabolát kívánunk átfektetni. Ez csak akkor előnyös, ha a parabola minimalizáló pontja nem vezet ki az (a, b) intervallumból, és ha a kurrens legjobb x értékből kisebb elmozdulást idéz elő, mint amilyen az előző lépésbeli elmozdulás fele. Ez utóbbi feltétel biztosítja a parabolikus lépések valamely értékhez való konvergenciáját, megakadályozva ezzel pl. a végtelen ciklus létrejöttét. A legkedvezőtlenebb esetben, amikor a parabolikus lépések előnyösnek kínálóznak, azonban hasznavehetetlenek, a két eljárás nagyjából felváltva működik, és a konvergenciát ekkor az aranymetszés szerinti keresés biztosítja.

A (3-2) egyenlettel kapcsolatos, kerekítésre vonatkozó megjegyzéseink miatt nem nyerünk újabb információt, ha olyan pontban számítunk ki függvényértéket, amely a tol távolságnál közelebb van egy olyan ponthoz, amelyben a függvényértéket már ismerjük. A következő eljárásban az ilyen felesleges függvényértékek számítására nem kerül sor.

Az AranyMetszes eljáráshoz hasonlóan a BrentDeriv Pascal függvényeljárás is a függvényminimum kezdő $a < b < c$ közrefogásának finomítására alkalmas. A BrentDeriv a Brent-módszer azon változatának implementációja, amelynél a függvényértékeken kívül az első deriváltak kiszámítása sem okoz gondot. (A deriváltak kiszámítása gondot okozhat pl. akkor, ha a minimalizálandó függvény értékeit csak diszkrét pontokban ismerjük, képletét azonban nem.)

Elsőnek az az ötlet kínálkozik, hogy a problémát az első derivált zéruspontjainak meghatározására egyszerűsítsük, anélkül hogy figyelembe vennénk a függvényértékekre vonatkozó információkat. Ekkor válogathatnánk az egyismeretlenes egyenletek gyökeinek finomítására kidolgozott számos eljárás között. Ez az egyszerűsítés azonban nem alkalmazható, s ennek lényegében két oka van: hogyan különböztethető meg a minimum a maximumtól, és merre lépjen az eljárás, ha a külső közrefogási pontok legalább egyikében a derivált a „csökkenés” irányát a közrefogási intervallumon kívülre jelzi.

Ezért a BrentDeriv eljárás a deriváltakat csak kiegészítő információként, a közrefogási intervallumon belüli, új próbapont meghatározásánál használja. Részletesebben, a derivált előjele az $a < b < c$ közrefogási ponthármas belső pontjában egyértelműen jelzi, hogy a következő próbapont az (a, b) , (b, c) intervallumok közül melyikbe kerüljön. Az e pontbeli derivált és az eddigiek során nyert második legkisebb derivált értékű ponton egyenest fektetünk át és meghatározzuk annak zérus pontját (lineáris inverz interpoláció). Ez nem más, mint a derivált függvény zéruspontjának meghatározására alkalmazott szelómódszer első lépése. Ha a szelómódszerrel nyert, új próbapont nem felel meg a Brent-módszer által támasztott követelményeknek, akkor a szelómódszert felváltja a felezési eljárás.


```

function BrentDeriv(  a,b,c,
                      tol: real;
                      var xmin: real): real;
{ A függvényeljárás alkalmas az alábbiakban részletezett
fgv(x) eljárással értelmezett függvény xmin minimalizáló
pontjának meghatározására a dfgv(x) deriváltfüggvény
felhasználásával. A minimalizáló pontot az  $a < b < c$ 
ponthármas fogja közre. A keresés addig tart, amíg a
közrezárt intervallum  $|b-a|/0.5/(|a|+|b|)$  relatív hossza
nagyobb, mint tol. A függvényeljárás értéke az xmin pontbeli
függvényérték.
A BrentDeriv függvényt használó programnak tartalmaznia
kell a
    function fgv(x: real): real;
    function dfgv(x: real): real;
deklarációjú függvényeljárásokat, melyek a minimalizálandó
függvény értékét és deriváltjának értékét határozzák meg az
x pontban.}
const
    iter = 100;
    zeps  = 1.0e-10;
var
    sa,sb,d,d1,d2: real;
    du,dv,dw,dx: real;
    e,fu,fv,fw,fx: real;
    k: integer;
    olde,toll,tol2: real;
    u,u1,u2,v,w,x,xm: real;
    ok1,ok2,vege: boolean;

    function sign(a,b: real): real;
    begin if b>=0 then sign:=abs(a) else sign:=-abs(a); end;

begin
    if a<c then sa:=a else sa:=c;
    if a>c then sb:=a else sb:=c;
    v:=b; w:=v; x:=v; e:=0.0;
    fx:=fgv(x); fv:=fx; fw:=fx;
    dx:=dfgv(x); dv:=dx; dw:=dx;
    k:=0;
    repeat inc(k);
        xm:=0.5*(sa+sb); toll:=tol*abs(x)+zeps;
        tol2:=2.0*toll;
        vege:=abs(x-xm)<=(tol2-0.5*(sb-sa));
        if not vege then
            begin ok1:=false;
                if abs(e)>toll then
                    begin d1:=2.0*(sb-sa); d2:=d1;
                        if dw<>dx then d1:=(w-x)*dx/(dx-dw);
                        if dv<>dx then d2:=(v-x)*dx/(dx-dv);
                        u1:=x+d1; u2:=x+d2;
                        ok1:=((sa-u1)*(u1-sb)>0.0) and (dx*d1<=0.0);
                        ok2:=((sa-u2)*(u2-sb)>0.0) and (dx*d2<=0.0);
                        olde:=e; e:=d;
                        if ok1 and ok2 then

```



```

begin
    if abs(d1)<abs(d2) then d:=d1
    else d:=d2;
end
else if ok1 then d:=d1
else if ok2 then begin d:=d2; ok1:=true; end;
if abs(d)>abs(0.5*olde) then
    ok1:=false;
if ok1 then
    begin u:=x+d;
        if ((u-sa)<tol2) or ((sb-u)<tol2) then
            d:=sign(toll,xm-x);
        end;
    end;
if not ok1 then
    begin
        if dx>=0.0 then e:=sa-x else e:=sb-x;
        d:=0.5*e;
    end;
if abs(d)>=toll1 then
    begin u:=x+d; fu:=fgv(u); end
else
    begin u:=x+sign(toll1,d); fu:=fgv(u);
        vege:=fu>fx;
    end;
if not vege then
    begin
        du:=dfgv(u);
        if fu<=fx then
            begin
                if u>=x then sa:=x else sb:=x;
                v:=w; fv:=fw; dv:=dw;
                w:=x; fw:=fx; dw:=dx;
                x:=u; fx:=fu; dx:=du;
            end
        else
            begin
                if u<x then sa:=u else sb:=u;
                if (fu<=fw) or (w=x) then
                    begin v:=w; fv:=fw; dv:=dw;
                        w:=u; fw:=fu; dw:=du;
                    end
                else if (fu<fv) or (v=x) or (v=w) then
                    begin v:=u; fv:=fu; dv:=du; end;
            end;
        end;
    end;
until vege or (k>iter);
if not vege then
    Hiba('Az iterációk száma nagyobb a megengedettnél.');
```

{iteráció vége}

```

    xmin:=x; BrentDeriv:=fx;
end;
```


4. Gradiensmódszer

Az $F: R^n \rightarrow R$ függvény minimalizálására

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{h}_k, \quad k = 1, 2, \dots \quad (4-1)$$

alakban iterációs eljárások fogalmazhatók meg, ahol a λ_k számot *egydimenziós minimalizálás* alkalmazásával, azaz úgy határozzuk meg, hogy a λ_k szám λ szerint minimalizálja az $F(\mathbf{x}_k + \lambda \mathbf{h}_k)$ egyváltozós függvényt.

Ha az egydimenziós minimumkeresést az F függvény negatív gradiensének, azaz a legmeredekebb csökkenésének irányában végezzük, vagyis ha

$$\mathbf{h}_k = - \left(\frac{dF}{d\mathbf{x}}(\mathbf{x}_k) \right)^T, \quad (4-2)$$

akkor a (4-1) iterációt *gradiensmódszernek* nevezzük.

4.1. példa. Ha

$$F(\mathbf{x}) = \mathbf{f}^T(\mathbf{x}) \mathbf{f}(\mathbf{x}), \quad \mathbf{f}: R^n \rightarrow R^m,$$

akkor (4-2)-ből

$$\begin{aligned} \mathbf{h}_k &= -\text{grad } F(\mathbf{x}_k) = - \left(\frac{dF}{d\mathbf{x}}(\mathbf{x}_k) \right)^T = - \left(\frac{dF}{d\mathbf{f}} \frac{d\mathbf{f}}{d\mathbf{x}} \right)^T = \\ &= - \left(2 \mathbf{f}^T(\mathbf{x}_k) \mathbf{J}(\mathbf{x}_k) \right)^T = -2 \mathbf{J}^T(\mathbf{x}_k) \mathbf{f}(\mathbf{x}_k) \end{aligned} \quad (4-3)$$

következik.

A (4-1) iterációban szereplő λ_k szám meghatározható például úgy is, hogy az F függvényt az

$$F(\mathbf{x}_k + \lambda \mathbf{h}_k) \approx F(\mathbf{x}_k) + \frac{dF}{d\mathbf{x}}(\mathbf{x}_k) \lambda \mathbf{h}_k + \frac{1}{2} \lambda \mathbf{h}_k^T \frac{d^2 F}{d\mathbf{x}^2}(\mathbf{x}_k) \lambda \mathbf{h}_k$$

Taylor-sorának első három tagjával közelítjük, és megkeressük a λ változós parabola minimalizáló pontját (ha ilyen pont létezik). A

$$\frac{\partial F}{\partial \lambda}(\mathbf{x}_k + \lambda \mathbf{h}_k) \approx \frac{dF}{d\mathbf{x}}(\mathbf{x}_k) \mathbf{h}_k + \lambda \mathbf{h}_k^T \frac{d^2 F}{d\mathbf{x}^2}(\mathbf{x}_k) \mathbf{h}_k = 0$$

stacionaritási feltételből

$$\lambda_k = - \frac{\frac{dF}{d\mathbf{x}}(\mathbf{x}_k) \mathbf{h}_k}{\mathbf{h}_k^T \frac{d^2 F}{d\mathbf{x}^2}(\mathbf{x}_k) \mathbf{h}_k} \quad (4-4)$$

adódik.

4.2. példa. Ha $F(x) = f^T(x) f(x)$, $f: R^n \rightarrow R^m$, akkor

$$\begin{aligned} \frac{d^2 F}{dx^2} &= \frac{d}{dx} \left(\frac{dF}{dx} \right)^T = \frac{d}{dx} (2 J^T(x) f(x)) = \\ &= 2 \frac{dJ^T}{dx} f + 2 J^T \frac{df}{dx} \approx 2 J^T(x) J(x), \end{aligned}$$

ahol a Gauss–Newton-módszerhez hasonlóan most is elhagytuk a $\frac{dJ^T}{dx} f$ tagot.

Ennek és a (4-3) összefüggésnek a felhasználásával (4-4) átírható

$$\begin{aligned} \lambda_k &= - \frac{2 f^T(x_k) J(x_k) \left(-2 J^T(x_k) f(x_k) \right)}{\left(-2 J^T(x_k) f(x_k) \right)^T \left(2 J^T(x_k) J(x_k) \right) \left(-2 J^T(x_k) f(x_k) \right)} = \\ &= \frac{1}{2} \frac{\|J^T(x_k) f(x_k)\|_2^2}{\|J(x_k) J^T(x_k) f(x_k)\|_2^2} \end{aligned} \quad (4-5)$$

alakra. A (4-5) és a (4-3) összefüggések (4-1)-be való behelyettesítésével az

$$x_{k+1} = x_k - \frac{\|J^T(x_k) f(x_k)\|_2^2}{\|J(x_k) J^T(x_k) f(x_k)\|_2^2} J^T(x_k) f(x_k), \quad k = 1, 2, \dots$$

iterációs formulát nyerjük.



4.1. ábra. Gradiensmódszer hosszúkás, keskeny „völgy” esetén

A gradiensmódszer nem tekinthető valami nagyon hatékony eljárásnak, mivel egy hosszúkás, keskeny völgy aljának eléréséhez akkor is sok rövid lépésre van szükség, ha a völgy alakja pontosan másodfokú függvény (lásd a 4.1. ábrát). Két dimenziós esetben pl. azt várnánk, hogy az első lépéssel a völgy alját érjük el, a másodikkal pedig pontosan a hosszú tengely mentén fogunk haladni. Vegyük azonban észre, hogy bármely egyenes menti minimalizálás esetén a minimumponti, új gradiensvektor merőleges az előbbi haladás irányára. Ez nyilvánvaló az

$$F(x) = F(x_k + \lambda h_k) = \min! \text{ és a}$$

$$\frac{\partial F}{\partial \lambda} = \frac{dF}{dx}(x_k + \lambda h_k) \Big|_{\lambda = \lambda_k} \frac{dx(\lambda)}{d\lambda} \equiv (\text{grad } F(x_{k+1}))^T h_k = 0 \quad (4-6)$$

összefüggésekből.

A gradiensmódszernek ez a tulajdonsága, hogy minden egyes lépését az előző lépés irányára merőleges irányba kényszeríti, általában csak nagyon lassú konvergenciát tesz lehetővé (lásd a 4.1. ábrát).

Így aztán nem is az annyira fontos, hogy az új gradiens irányában folytassuk a minimalizálást, inkább az, hogy olyan irányokat állítsunk elő, amelyeken haladva nem rontunk az előző lépésekben már elért függvény csökkenésén. Az ilyen irányok meghatározása képezi a *konjugált gradiensmódszerek* alap gondolatát.

5. Konjugált gradiensmódszerek

Konjugált irányok. A zérustól különböző $h_1, \dots, h_n$ vektorokat konjugáltaknak nevezzük a valós *szimmetrikus* n -ed rendű H mátrixhoz viszonyítva, ha teljesül a

$$h_i^T H h_j = 0, \quad i \neq j \quad (5-1)$$

összefüggés.

5.1. lemma. Ha a $h_1, \dots, h_n$ vektorok konjugáltak a *pozitív definit* szimmetrikus H mátrixhoz viszonyítva, akkor *lineárisan függetlenek*.

Bizonyítás. Ha a

$$c_1 h_1 + \dots + c_n h_n = 0$$

egyenlőséget balról szorozzuk $h_i^T H$ -val, akkor az (5-1) és a $h_i^T H h_i > 0, h_i \neq 0$ összefüggések miatt $c_i = 0$ következik. Mivel ez minden i -re igaz, ezzel az állítást beláttuk.

Az előbbi pontban ismertetett gradiensmódszer olyan függvényminimalizálási eljárás, amelynél egydimenziós minimumkeresést alkalmazunk egymás után, mindig a legmeredekebb csökkenés irányában.

A *konjugált irányok módszere* is olyan függvényminimalizálási eljárás, amelynek alapját az egydimenziós minimumkeresés egymás utáni alkalmazása képezi. A minimumkeresés irányait maga az eljárás úgy állítja elő, hogy ha a módszert az

$$F(x) = a + b^T x + \frac{1}{2} x^T H x \quad (5-2)$$

n változós kvadratikus függvény minimalizálására alkalmazzuk, ahol a , b , H állandók és H pozitív definit szimmetrikus mátrix, akkor ezek az irányok konjugáltak a H Hesse-mátrixhoz viszonyítva.

5.1. tétel. A konjugált irányok módszere legfeljebb n egydimenziós pontos minimumkereséssel éri el az (5-2) alatti n változós kvadratikus függvény minimumát.

Bizonyítás. Az (5-2) függvény x vektor szerinti deriváltja

$$\frac{dF}{dx}(x) = b^T + x^T H. \quad (5-3)$$

Az

$$x_{k+1} = x_k + \lambda_k h_k, \quad k = 1, 2, \dots$$

alakú iterációs formulából

$$\begin{aligned} x_2 &= x_1 + \lambda_1 h_1 \\ x_3 &= x_2 + \lambda_2 h_2 = x_1 + \lambda_1 h_1 + \lambda_2 h_2 \\ &\vdots \\ x_{j+1} &= x_1 + \lambda_1 h_1 + \lambda_2 h_2 + \dots + \lambda_j h_j \\ &\vdots \\ x_{i+1} &= \left(x_1 + \lambda_1 h_1 + \dots + \lambda_j h_j \right) + \lambda_{j+1} h_{j+1} + \dots + \lambda_i h_i, \end{aligned}$$

vagyis

$$x_{i+1} = x_{j+1} + \sum_{k=j+1}^i \lambda_k h_k \quad (5-4)$$

következik. Ha az (5-4) összefüggést behelyettesítjük (5-3)-ba, akkor

$$\begin{aligned} \frac{dF}{dx}(x_{i+1}) &= b^T + x_{i+1}^T H = b^T + x_{j+1}^T H + \sum_{k=j+1}^i \lambda_k h_k^T H = \\ &= \frac{dF}{dx}(x_{j+1}) + \sum_{k=j+1}^i \lambda_k h_k^T H \end{aligned}$$

adódik, és ha ezt az összefüggést jobbról szorozzuk a h_j vektorral, akkor

$$\frac{dF}{dx}(x_{i+1}) h_j = \frac{dF}{dx}(x_{j+1}) h_j + \sum_{k=j+1}^i \lambda_k h_k^T H h_j \quad (5-5)$$

következik.

Itt emlékeztetünk arra, hogy az iteráció j -edik lépésében alkalmazott egydimenziós pontos minimumkeresés egy olyan λ_j szám meghatározását jelenti, amely λ szerint minimalizálja az $F(x_j + \lambda h_j)$ függvényt. A stacionaritási pontra vonatkozó (4-6) alatti, azaz a

$$\frac{\partial F}{\partial \lambda}(\mathbf{x}_j + \lambda \mathbf{h}_j) = \frac{dF}{d\lambda}(\mathbf{x}_j + \lambda \mathbf{h}_j) \Big|_{\lambda=\lambda_j} \frac{d\lambda}{d\lambda} = \frac{dF}{d\lambda}(\mathbf{x}_{j+1}) \mathbf{h}_j = 0$$

feltétel miatt az (5-5) jobb oldalán az első tag zérus, a második tag pedig azért zérus mert a $\mathbf{h}_1, \dots, \mathbf{h}_n$ vektorok konjugáltak a $\mathbf{H}$ mátrixhoz viszonyítva. Következésképpen

$$\frac{dF}{d\mathbf{x}}(\mathbf{x}_{i+1}) \mathbf{h}_j = 0, \quad j = 1, 2, \dots, i$$

és így $i = n$ speciális esetben is

$$\frac{dF}{d\mathbf{x}}(\mathbf{x}_{n+1}) \mathbf{h}_j = 0, \quad j = 1, \dots, n \quad (5-6)$$

Feltételezésünk értelmében $\mathbf{H}$ pozitív definit mátrix, ezért az 5.1. lemma állítása miatt a $\mathbf{h}_1, \dots, \mathbf{h}_n$ vektorok bázist alkotnak az R^n térben. Ezért az (5-6) összefüggés akkor és csak akkor teljesül, ha

$$\frac{dF}{d\mathbf{x}}(\mathbf{x}_{n+1}) \equiv \mathbf{b}^T + \mathbf{x}_{n+1}^T \mathbf{H} = \mathbf{0}^T,$$

vagyis ha $\mathbf{x}_{n+1}$ az (5-2) kvadratikus függvény *stacionaritási* pontja. Mivel $\mathbf{H}$ -ról feltettük, hogy pozitív definit mátrix, az $\mathbf{x}_{n+1}$ vektor egyben minimalizáló vektor. (Ez közvetlenül következik az (5-2) függvény $\mathbf{x}_{n+1}$ pont körüli

$$\begin{aligned} F(\mathbf{x}_{n+1} + \mathbf{y}) &= F(\mathbf{x}_{n+1}) + \frac{dF}{d\mathbf{x}}(\mathbf{x}_{n+1}) \mathbf{y} + \frac{1}{2} \mathbf{y}^T \frac{d^2 F}{d\mathbf{x}^2}(\mathbf{x}_{n+1}) \mathbf{y} = \\ &= F(\mathbf{x}_{n+1}) + \frac{1}{2} \mathbf{y}^T \mathbf{H} \mathbf{y} \end{aligned}$$

Taylor-sorának felírásából, ahonnan az $F(\mathbf{x}_{n+1} + \mathbf{y}) > F(\mathbf{x}_{n+1})$ tetszőleges $\mathbf{y} \neq \mathbf{0}$ vektorra az állítás már nyilvánvaló.) Ezzel beláttuk, hogy a minimumot legfeljebb n iterációs lépéssel nyerjük; ha valamely lépésben $\lambda_k = 0$, akkor korábban.

Az 5.1. tételben megfogalmazott tulajdonság miatt előnyös a gradiensmódszert kombinálni a konjugált irányok módszerének különböző változataival. Így keletkeznek az ún. *konjugált gradiensmódszerek*, amelyek közül a *Fletcher-Reeves-módszer* és a *Polak-Ribiere-változat* látszik a legelőnyösebbnek.

Mindkét módszer alapját a következő tétel képezi:

5.2. tétel. Tegyük fel, hogy $\mathbf{H}$ pozitív definit szimmetrikus valós n -ed rendű mátrix, $\mathbf{g}_0$ tetszőleges n elemű vektor és $\mathbf{h}_0 = \mathbf{g}_0$. A

$$\mathbf{g}_{k+1} = \mathbf{g}_k - \lambda_k \mathbf{H} \mathbf{h}_k, \quad \mathbf{h}_{k+1} = \mathbf{g}_{k+1} + \gamma_k \mathbf{h}_k, \quad k = 0, 1, 2, \dots \quad (5-7)$$

összefüggésekkel értelmezzünk két vektorsorozatot, ahol a λ_k, γ_k számokat úgy határozzuk meg, hogy teljesüljön a $\mathbf{g}_{k+1}^T \mathbf{g}_k = 0$ ortogonalitási és a $\mathbf{h}_{k+1}^T \mathbf{H} \mathbf{h}_k = 0$ konjugáltsági feltétel, vagyis

$$\lambda_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{H} \mathbf{h}_k}, \quad \gamma_k = -\frac{\mathbf{g}_{k+1}^T \mathbf{H} \mathbf{h}_k}{\mathbf{h}_k^T \mathbf{H} \mathbf{h}_k} \quad (5-8)$$

legyen (a sorozatok képzését addig folytatjuk, amíg a nevezők nem egyenlők nullával, ezután felvesszük a $\lambda_k = 0, \gamma_k = 0$ értékeket). Ekkor minden $k \neq l$ indexre érvényesek a

$$\mathbf{g}_k^T \mathbf{g}_l = 0, \quad \mathbf{h}_k^T \mathbf{H} \mathbf{h}_l = 0 \quad (5-9)$$

összefüggések (és van olyan $m \leq n - 1$ nemnegatív egész szám, hogy $\mathbf{g}_k = \mathbf{h}_k = 0$, ha $k > m$).

Tehát ha az (5-7), (5-8) képletekkel, mint egyféle Gram-Schmidt biortogonalizálási eljárással, két vektorsorozatot úgy állítunk elő, hogy minden egyes $\mathbf{g}_k$ vektor ortogonális az azt közvetlenül megelőző vektorra és minden egyes $\mathbf{h}_k$ vektor konjugált az azt közvetlenül megelőző vektorra, akkor a $\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \dots$ sorozatot páronként ortogonális, a $\mathbf{h}_0, \mathbf{h}_1, \mathbf{h}_2, \dots$ sorozatot pedig páronként konjugált vektorok alkotják (és a véges dimenziós tér miatt nyilvánvalóan van olyan $m \geq 0$, hogy minden $k > m$ indexre $\mathbf{g}_k = \mathbf{h}_k = 0$).

A tétel bebizonyítható matematikai indukcióval és a bizonyítás megtalálható pl. Polak [10] könyvének 2.3. pontjában. E bizonyításnak részeredménye a

$$\mathbf{g}_k^T \mathbf{h}_l = 0, \quad k \neq l \quad (5-10)$$

összefüggés. Felhasználásával már egyszerűen belátható a következő állítás:

5.1. következmény. Az (5-8) összefüggésekkel egyenértékűek a

$$\gamma_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k} = \frac{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k} \quad (5-11)$$

$$\lambda_k = \frac{\mathbf{g}_k^T \mathbf{h}_k}{\mathbf{h}_k^T \mathbf{H} \mathbf{h}_k} \quad (5-12)$$

összefüggések.

(Valóban, az (5-7), (5-9) és (5-10) képletek felhasználásával az (5-8) formulák átírhatók a következőképpen:

$$\begin{aligned}\lambda_k &= \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{H} \mathbf{h}_k} = \frac{\mathbf{g}_k^T (\mathbf{h}_k - \gamma_{k-1} \mathbf{h}_{k-1})}{(\mathbf{h}_k - \gamma_{k-1} \mathbf{h}_{k-1})^T \mathbf{H} \mathbf{h}_k} = \frac{\mathbf{g}_k^T \mathbf{h}_k}{\mathbf{h}_k^T \mathbf{H} \mathbf{h}_k}, \\ \gamma_k &= - \frac{\mathbf{g}_{k+1}^T \mathbf{H} \mathbf{h}_k}{\mathbf{h}_k^T \mathbf{H} \mathbf{h}_k} = - \frac{\mathbf{g}_{k+1}^T \left(-\frac{1}{\lambda_k} (\mathbf{g}_{k+1} - \mathbf{g}_k) \right)}{\mathbf{h}_k^T \left(-\frac{1}{\lambda_k} (\mathbf{g}_{k+1} - \mathbf{g}_k) \right)} = \\ &= - \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{h}_k^T (\mathbf{g}_{k+1} - \mathbf{g}_k)} = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{h}_k^T \mathbf{g}_k} = \\ &= \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{(\mathbf{g}_k + \gamma_{k-1} \mathbf{h}_{k-1})^T \mathbf{g}_k} = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k} = \\ &= \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1} - \mathbf{g}_k^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k} = \frac{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}.\end{aligned}$$

E technika máris alkalmazható az

$$F(\mathbf{x}) \approx a + \mathbf{b}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x}, \quad (5-13)$$

vagyis az (5-2) kvadratikus függvénnel approximált $F: \mathbb{R}^n \rightarrow \mathbb{R}$ függvény minimalizálására, ha feltesszük, hogy ismerjük a $\mathbf{H}$ Hesse-mátrixot. Ekkor ugyanis az (5-7) konstrukcióval egymás után előállíthatók azok a $\mathbf{h}_k$ konjugált irányok, amelyek mentén végrehajtható az egydimenziós minimumkeresés. Az 5.1. tétel állítása alapján legfeljebb n ilyen lépés végrehajtása után megkapjuk a kvadratikus függvény minimalizáló vektorát.

A H mátrixot azonban nem ismerjük. Ez mégsem okoz gondot a következő tételben megfogalmazott ötlet miatt.

5.3. tétel. Jelölje g_k és h_k ugyanazoknak a vektorsorozatoknak az elemeit mint az eddigiekben. Tegyük fel, hogy valamely x_k pontban $g_k = -\left(\frac{dF}{dx}(x_k)\right)^T$ adódott, ahol F egy (5-13) alakú kvadratikusan függvény. Kiindulva az x_k pontból és h_k irány mentén haladva, keressük meg azt az x_{k+1} pontot, amelyben $F(x)$ felveszi lokális minimumát. A $g_{k+1} = -\left(\frac{dF}{dx}(x_{k+1})\right)^T$ összefüggéssel határozzuk meg a g_{k+1} vektort. Akkor ez a g_{k+1} vektor nem más, mint az a vektor, amelyet az (5-7) összefüggés eredményezne. (Tehát a g_{k+1} vektort a H mátrix ismerete nélkül állítottuk elő.)

Bizonyítás. Az (5-13) összefüggésből

$$g_k = -\left(\frac{dF}{dx}(x_k)\right)^T = -\left(b^T + x_k^T H\right)^T = -H x_k - b$$

következik és így

$$g_{k+1} = -\left(\frac{dF}{dx}(x_{k+1})\right)^T = -H x_{k+1} - b,$$

vagyis

$$g_{k+1} = -H(x_k + \lambda_k h_k) - b = g_k - \lambda_k H h_k \quad (5-14)$$

ahol a λ_k lépéshosszt az

$$F(x_{k+1}) = F(x_k + \lambda_k h_k) = \min_{\lambda} F(x_k + \lambda h_k)$$

összefüggésnek megfelelően határoztuk meg. Ebben az egyenes menti minimálzó pontban azonban

$$\frac{\partial F}{\partial \lambda}(x_k + \lambda h_k) = \frac{dF}{dx}(x_k + \lambda h_k) \Big|_{\lambda = \lambda_k} \frac{dx(\lambda)}{d\lambda} = 0$$

vagyis

$$g_{k+1}^T h_k = 0$$

érvényes. Ez a feltétel az (5-14) felhasználásával átírható

$$(g_k - \lambda_k H h_k)^T h_k = 0$$

alakra, amely pontosan az (5-12) kifejezést eredményezi. Ezzel a λ_k értékkel viszont az (5-14) egyenlőség megegyezik az (5-7) összefüggéssel, s ezt kellett belátni.

E tétel az alapja az olyan algoritmusnak, amely nem igényli a H Hesse-mátrix ismeretét, így tárolását sem. A $\mathbf{h}_k$ irányok sorozatát úgy állítjuk elő, hogy azokhoz csupán egyenes menti minimalizálásra, gradiensvektor-képzésre és a $\mathbf{g}$ vektorok közül az utolsó tárolására van szükség.

Az előbbieket összefoglalva, a *konjugált gradiensmódszer* egy iteráció, amelynek k -edik lépése a következő három részfeladat megoldásából tevődik össze:

(1) Az $F(\mathbf{x}_k + \lambda \mathbf{h}_k)$ függvény λ szerinti minimalizálásával a λ_k szám meghatározása, majd az $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda_k \mathbf{h}_k$ vektor előállítása.

(2) A

$$\mathbf{g}_{k+1} = - \left(\frac{dF}{d\mathbf{x}}(\mathbf{x}_{k+1}) \right)^T$$

negatív gradiensvektor kiszámítása az $\mathbf{x}_{k+1}$ pontban.

(3) A

$$\mathbf{h}_{k+1} = \mathbf{g}_{k+1} + \gamma_k \mathbf{h}_k$$

irány meghatározása, ahol γ_k (5-11) alapján a

$$\gamma_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}, \quad \gamma_k = \frac{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}$$

képletek egyikéből számítható ki.

Az iteráció kezdővektorai:

$$\mathbf{x}_0, \quad \mathbf{g}_0 = - \left(\frac{dF}{d\mathbf{x}}(\mathbf{x}_0) \right)^T, \quad \mathbf{h}_0 = \mathbf{g}_0$$

Attól függően, hogy a γ_k kiszámítására az (5-11) alatti első vagy második képletet alkalmazzuk, a konjugált gradiensmódszer *Fletcher-Reeves*-, illetve *Polak-Ribiere-változatáról* beszélünk. Kvadratikusan konjugált függvény esetén a két képlet egyenlő egymással, hiszen $\mathbf{g}_k^T \mathbf{g}_{k+1} = 0$. Az alkalmazásokban azonban a minimalizálandó függvény nem egyezik meg pontosan a kvadratikusan konjugált függvénnyel. Miután meghatároztuk a kvadratikusan konjugált alak minimumát, általában az eljárást folytatni kell; további iterációra van szükség. E további iterációra való áttérés valamivel kedvezőbb, ha a Polak-Ribiere-formulát részesítjük előnyben a γ_k kiszámításánál. A szóban forgó áttérést a $\mathbf{h}$ vektor megváltoztatásával valósítjuk meg úgy, hogy $\mathbf{h}$ a lokális gradiensvektor irányában lefelé mutasson. Ez pedig nem más, mint a konjugált gradienseljárás újraindítása.

A következő KonjGradiens eljárás a Polak–Ribiere-változat implementációja, azonban egyetlen sorának megváltoztatásával megkaphatjuk a Fletcher–Reeves-változatot. A minimalizálandó függvényt az $fgvnx1(x)$ függvényeljárással kell megadni, ahol x egy n elemű tömb és a függvény x pontbeli g gradiensenek meghatározására a $dfgvnx1(x, g)$ eljárást is meg kell adni.

```

procedure KonjGradiens(var xk: tvN;
                        n: integer;
                        tolf: real;
                        var iter: integer;
                        var fmin: real);
{ Az eljárás alkalmas az fgvnx1 függvényeljárással
értelmezett n változós függvény minimalizálására. Az
eljárás felhasználja a dfgvnx1 eljárással megvalósított, a
gradiensvektor meghatározására szolgáló eljárást is. Az n
elemű xk pontból indul az iteráció, és ugyanebben a
változóban lesz a megoldás is. Az eljárás pontosságát a
tolf paraméterrel szabályozhatjuk. Az fmin kimenő paraméter
a megoldásként nyert xk pont függvényértéke. Az iter
paraméter az iterációk számát mutatja.
A KonjGradiens eljárást használó főprogramban meg kell
adni a
function fgvnx1(x: tvN): real;
procedure dfgvnx1(x: tvN; var g: tvN);
eljárásokat, melyek a függvényértéket adják meg az x
pontban (fgvnx1), ill. az x pontbeli g gradiensvektort
szolgáltatják (dfgvnx1). Definiálni kell még
tvN = array [1..nmax] of real;
típust is.}
const
    eps = 1.0e-10;
var
    i, k: integer;           {ciklusváltozó és az iteráció száma}
    gammak, f: real;         {segédváltozók}
    s1, s2: double;          {segédváltozó az összegzésekhez}
    gk, hk, gkl: tvN;        {segédvektorok}
    vege: boolean;           {az iteráció sikeres végét jelzi}
begin k:=0; f:=fgvnx1(xk); dfgvnx1(xk, gk);
    for i:=1 to n do          {g0, h0 számítása}
        begin gk[i]:=-gk[i]; hk[i]:=gk[i]; gkl[i]:=hk[i] end;
    repeat inc(k);             {iteráció kezdete}
        LinMin(xk, gkl, n, fmin); {gkl hk helyett van!}
        vege:=(2*abs(fmin-f))<=(tolf*(abs(fmin)+abs(f)+eps));
        if not vege then
            begin f:=fgvnx1(xk); dfgvnx1(xk, gkl); {gkl negatív!}
                s2:=0.0; s1:=0.0;
                for i:=1 to n do
                    begin s2:=s2+sqr(gk[i]);
{→          s1:=s1+sqr(gkl[i]);          <- Fletcher-Reeves}
                    s1:=s1+(gkl[i]+gk[i])*gkl[i]; {Polak-Ribiere}
                    end;
                if s2=0.0 then vege:=true
                else
                    begin gammak:=s1/s2;

```



```

    for i:=1 to n do
        begin gk[i]:=-gkl[i]; {gk helyes előjellel!}
              hk[i]:=gk[i]+gammak*hk[i];
              gkl[i]:=hk[i];
        end;
    end;
end;
until vege or (k>iter);
if not vege then
    Hiba('Az iterációk száma nagyobb a megengedettnél!')
else iter:=k;
end;

```

Az $F(x)$ függvény λ szerinti $x(\lambda) = x_k + \lambda h_k$ egyenes menti minimalizálásra a KonjGradiens eljárás hívja a LinMin eljárást. Ha a már ismertetett BrentDeriv eljárást kívánjuk alkalmazni, akkor az $f(x)$ és $f'(x)$ egyváltozós függvényeket $f(\lambda) = F(x_k + \lambda h_k)$ és $f'(\lambda) = \frac{dF}{d\lambda} = \frac{dF}{dx} \frac{dx}{d\lambda} = \frac{dF}{dx} (x_k + \lambda h_k) h_k$ alapján kell előállítani.

```

procedure LinMin(   var xk,hk: tvN;
                   n: integer;
                   var fmin: real);
{ Az eljárás alkalmas a KonjGradiens és DFPMIn eljárások
  által igényelt xk+lambd h k egyenes menti minimalizálásra.
  A minimalizálást a BrentDeriv eljárással megvalósított
  Brent-módszerrel végzi. Xk bemenetkor a kezdő pontot,
  kimenetkor a minimalizáló pontot tartalmazza, hk bemenetkor
  a minimalizálás irányát, kimenetkor a minimumhoz mutató
  irányvektort tartalmazza. Az fmin kimenő paraméter az
  egyenes menti minimum függvényértéke.
  A LinMin eljárást használó programban definiálni kell a
  tvN = array [1..nmax] of real;
  típust. A fenti típuson kívül szükség van a
  glN: integer;
  glxk, glhk: tvN;
  változók definiálására is. Az eljárás hívja a Kozrefog és
  BrentDeriv eljárásokat. A BrentDeriv eljáráshoz szükséges
  fgV(x) és dfgv(x) függvényeljárásokat a következők szerint
  kell deklarálni:

```

```

function fgV(lambda: real): real;
var i:integer; x:tvN;
begin
    for i:=1 to glN do
        x[i]:=glxk[i]+lambda*glhk[i];
    fgV:=fgvnx1(x);
end;

```

```

function dfgv(lambda: real): real;
var i:integer; x,df:tvN; s: double;
begin
    for i:=1 to glN do
        x[i]:=glxk[i]+lambda*glhk[i];

```



```

    dfgvnx1(x,df); s:=0;
    for i:=1 to gln do
        s:=s+df[i]*glhk[i];
    dfgv:=s;
end;
}
const
    tol = 1.0e-4;           {tolerancia a BrentDeriv eljáráshoz}
var
    j: integer;              {ciklus változó}
    lambdak,                 {minimum helye az egyenes mentén}
    x,fx,fb,fa,b,a: real;    {egydimenziós kereséshez}
begin gln:=n;               {adatok átrakása a globális változókbá}
    for j:=1 to n do
        begin glxk[j]:=xk[j]; glhk[j]:=hk[j]; end;
    a:=0.0; x:=1.0; b:=2.0;   {kezdő ponthármas megadása}
    Kozrefog(a,x,b,fa,fx,fb); {egyenes menti minimalizálás}
    fmin:=BrentDeriv(a,x,b,tol,lambdak);
    for j:=1 to n do         {kimenő paraméterek számítása}
        begin hk[j]:=lambdak*glhk[j]; xk[j]:=xk[j]+hk[j]; end;
    end;

```

6. Kvázi-Newton-módszerek

A néha *változó metrikájú módszereknek* (variable metric methods) is nevezett *kvázi-Newton-módszerek* lényegében ugyanazt a célt szolgálják mint a konjugált gradiensmódszerek: összegyűjtik az egymást követő, egyenes menti minimalizálás során szerzett információkat úgy, hogy n változós kvadratikus függvényt feltételezve, n egydimenziós minimalizálás végrehajtása a pontos minimumot eredményezi. E tulajdonság általánosabb sima függvények minimalizálásánál is előnyösnek mutatkozik.

A konjugált gradiensmódszerekhez hasonlóan, a kvázi-Newton-módszereknél is szükség van arra, hogy bármely pontban képesek legyünk kiszámítani a függvény gradiensét, vagyis az első parciális deriváltakat. A kvázi-Newton-módszerek azonban az információk gyűjtése és tárolása vonatkozásában különböznek a konjugált gradiensmódszerektől. Míg a konjugált gradiensmódszereknél a közbeni eredmények n elemű vektorban tárolhatók, ahol n a független változók száma, a kvázi-Newton-módszereknél ez a tárolás $n \times n$ -es mátrixot igényel.

A kvázi-Newton-módszerek közül a leginkább a *Davidon-Fletcher-Powell*, tömörebben a *DFP-algoritmust* implementálják. Ehhez az eljáráshoz szorosan kapcsolódik a *Broyden-Fletcher-Goldfarb-Shanno*, vagyis a *BFGS-algoritmus*, amely a DFP-eljárástól csupán a kerekítési hibákra és a konvergenciatoleranciákra vonatkozó részletekben különbözik. A gyakorlati tapasztalatok a BFGS-algoritmus-sal kedvezőbbek.

Az előbbiekhöz hasonlóan, most is tegyük fel, hogy a minimalizálandó $F(x)$ függvény lokálisan approximálható az

$$F(x) \approx a + b^T x + \frac{1}{2} x^T H x, \quad (6-1)$$

kvadratikus függvénnyel, ahol H pozitív definit szimmetrikus mátrix. A (6-1)-ben szereplő b vektor és H mátrix elemeinek numerikus értékeit azonban nem ismerjük. Az ezekre vonatkozó információk függvényérték-számítások és egyenes menti minimalizálások segítségével gyűjthetők össze.

A kvázi-Newton-módszereknek az az alapötlete, hogy iterációval állítsuk elő a H Hesse-mátrix inverzének jó közelítéseit, vagyis képezzük az A_k mátrixok olyan sorozatát, amelyekre

$$\lim_{k \rightarrow \infty} A_k = H^{-1} \quad (6-2)$$

érvényes. Az csak előny, ha a határértéket (∞ helyett) már n iterációs lépés végrehajtásával érjük el.

Ha a (6-2) határérték előállítására sikerrel járt, akkor ez az A határérték a következőképpen használható fel:

A (6-1) függvény stacionaritási pontjára vonatkozó

$$\frac{dF}{dx}(x) \approx b^T + x^T H = 0^T \quad (6-3)$$

feltételből következik, hogy az x^* minimalizáló vektor kielégíti a

$$H x^* = -b \quad (6-4)$$

egyenletet. Az aktuális x_k pontban (6-3)-ból közelítően

$$H x_k = \left(\frac{dF}{dx}(x_k) \right)^T - b \quad (6-5)$$

következik. Ha a (6-4) egyenletből kivonjuk a (6-5) egyenletet és a különbséget balról szorozzuk a H^{-1} inverz mátrixszal, akkor

$$x^* - x_k = H^{-1} \left(-\frac{dF}{dx}(x_k) \right)^T \quad (6-6)$$

adódik.

A (6-6) bal oldala annak a *véges lépésnek* a kifejezője, amelyet hozzá kell adni az x_k közelítéshez, hogy megkapjuk a pontos minimumot; a jobb oldal pedig ismertnek tekinthető, miután előállítottuk a (6-2) határérték egy jó $A \approx H^{-1}$ közelítését.

Nem kívánjuk precízen levezetni, hogy a DFP-algoritmus az A_k mátrixból hogyan képezi az A_{k+1} mátrixot. Erre vonatkozóan lásd pl. Polak [10] vagy Fletcher [3] könyvét. Inkább az eljárás heurisztikus motivációjának a bemutatására szorítkozunk.

Ha a (6-6) egyenletből kivonjuk az x_{k+1} vektorral feltüntetett (6-6) egyenletet, akkor az

$$x_{k+1} - x_k = H^{-1} \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right)^T \quad (6-7)$$

összefüggést nyerjük. Az x_k -ből az x_{k+1} vektort képező lépésben ésszerűnek kínálkozik az a követelmény, hogy a (6-7) egyenlet a pontos H^{-1} helyett annak az A_{k+1} közelítésével teljesüljön. Vagyis az új A_{k+1} közelítés elégítse ki az

$$x_{k+1} - x_k = A_{k+1} \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right)^T \quad (6-8)$$

egyenletet. A módosító formulát $A_{k+1} = A_k + \text{korrekció}$ alakban célszerű feltételezni.

Mi legyen azoknak az „elemeknek” a köre, amelyek alkalmasak a korrekció tag megszerkesztésére? Mindenekelőtt az $x_{k+1} - x_k$ és a $\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k)$ vektorokat és természetesen az A_k mátrixot kell figyelembe venni. Ezekből az elemekből mátrix ésszerűen nem nagyon sokféleképpen alkotható meg, különösen akkor nem, ha teljesülni kell a (6-8) összefüggésnek!

Az egyik ilyen lehetőség az

$$A_{k+1} = A_k + \frac{(x_{k+1} - x_k)(x_{k+1} - x_k)^T}{\left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right) (x_{k+1} - x_k)} - \frac{A_k \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right)^T \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right) A_k}{\left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right) A_k \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right)^T} \quad (6-9)$$

alakú DFP-módosító formula alkalmazásában, egy másik pedig a BFGS-módosító formula alkalmazásában rejlik.

A BFGS-módosító formula pontosan megegyezik az előbbi formulával, azonban tartalmaz még egy, éspedig a

$$\dots + \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right) A_k \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right)^T u u^T \quad (6-10)$$

kiegészítő tagot, ahol u az

$$u \equiv \frac{x_{k+1} - x_k}{\left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right) (x_{k+1} - x_k)} - \frac{A_k \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right)^T}{\left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right) A_k \left(\frac{dF}{dx}(x_{k+1}) - \frac{dF}{dx}(x_k) \right)^T} \quad (6-11)$$

összefüggéssel értelmezett vektor.

Bebizonyítható, hogy mind a (6-9) formula önmagában, mind annak a (6-10) taggal való kiegészített változata n lépésben tart a H^{-1} mátrixhoz, feltéve, hogy az iterációt (6-1) alakú kvadratikus F függvényre alkalmaztuk.

Az iteráció k -adik lépése a következő három részfeladat megoldásából tevődik össze:

(1) A

$$h_k = -A_k \frac{dF}{dx}(x_k)$$

irány kiszámítása.

(2) Az x_k ponton keresztülhaladó, h_k irányú egyenes menti $F(x(\lambda))$ függvény λ szerinti minimalizálásával a λ_k szám meghatározása, majd az

$$x_{k+1} = x_k + \lambda_k h_k$$

vektor meghatározása.

(3) Az

$$A_k \rightarrow A_{k+1}$$

mátrixmódosítás végrehajtása.

Az iteráció kezdő adatai:

x_0 , és $A_0 =$ tetszőleges pozitív definit szimmetrikus, pl. az egységmátrix.

A következő DFPMIn eljárás a Davidon–Fletcher–Powell függvényminimalizálás Broyden–Fletcher–Goldfarb–Shanno változatának implementációja. A minimalizálandó függvényt az $fgvnx1(x)$ függvényeljárással kell megadni, ahol x egy n elemű tömb és a függvény x pontbeli g gradienseinek meghatározására a $dfgvnx1(x, g)$ eljárást is meg kell adni.

Az egyenes menti minimalizálások végrehajtására az eljárás hívja a LinMin eljárást.

```
procedure DFPMIn(          var xk: tVn;
                           n: integer;
                           tolf: real;
                           var iter: integer;
                           var fmin: real);
{ Az eljárás alkalmas az fgvnx1 függvényeljárással
értelmezett n változós függvény minimalizálására. Az
eljárás felhasználja a dfgvnx1 eljárással megvalósított, a
gradiensvektor meghatározására szolgáló eljárást is. Az n
elemű xk pontból indul az iteráció, és ugyanebben a
változóban lesz a megoldás is. Az eljárás pontosságát a
tolf paraméterrel szabályozhatjuk. Az fmin kimenő paraméter
a megoldásként nyert xk pont függvényértéke. Az iter
paraméter az iterációk számát mutatja.
```

A DFPMIn eljárást használó főprogramban meg kell adni a

```
function fgvnx1(x: tVn): real;
procedure dfgvnx1(x: tVn; var g: tVn);
eljárásokat, melyek a függvényértéket adják meg az x
pontban (fgvnx1), ill. az x pontbeli g gradiensvektort
szolgáltatják (dfgvnx1). Definálni kell még
```

```
tVn = array [1..nmax] of real;
tVnxn = array [1..nmax, 1..nmax] of real;
típusokat is.)
```

```
const
```

```
eps = 1.0e-10;
```

```
var
```

```
j, i, k: integer;          {ciklus változók és az iteráció száma}
f: real;                   {az F(xk) függvényérték}
s, sl, s2: double;         {segédváltozók az összegzésekhez}
hk, g, u, Adg: tVn;        {segédvektorok}
Ak: tVnxn;                 {a Hesse-mátrix inverzének közelítéséhez}
vege: boolean;             {az iteráció sikeres végét jelzi}
```

```
begin k:=0; f:=fgvnx1(xk); dfgvnx1(xk, g);
```

```
for i:=1 to n do
```

```
begin
```

```
{A0 egységmátrix}
```

```
for j:=1 to n do
```

```
Ak[i, j]:=0.0;
```

```
Ak[i, i]:=1.0; hk[i]:=-g[i];
```

```
{h0 = -dF/dx}
```

```
end;
```

```
repeat inc(k);
```

```
{iteráció kezdete}
```

```
LinMin(xk, hk, n, fmin);
```

```
{(2) xk+1 meghatározása}
```

```
vege:=(2*abs(fmin-f))<=(tolf*(abs(fmin)+abs(f)+eps));
```

```
f:=fmin;
```

```
{F(xk+1) lesz a következő F(xk)}
```

```
for i:=1 to n do
```

```
u[i]:=g[i];
```

```
{dF/dx (xk)}
```

```
fmin:=fgvnx1(xk); dfgvnx1(xk, g);
```



```

for i:=1 to n do                                     {(3) Ak+1 számítása}
  u[i]:=g[i]-u[i];                                   {dF/dx (xk+1) - dF/dx (xk)}
for i:=1 to n do
  begin s:=0.0;
    for j:=1 to n do
      s:=s+Ak[i,j]*u[j];
    Adg[i]:=s;                                         {Ak (dF/dx (xk+1) - dF/dx (xk))T}
  end;
sl:=0.0; s:=0.0;
for i:=1 to n do                                     {(6-9) nevezőinek számítása}
  begin sl:=sl+u[i]*hk[i];
    s:=s+u[i]*Adg[i];
  end;
if sl<>0 then sl:=1.0/sl;
if s<>0 then s2:=1.0/s else s2:=0;
for i:=1 to n do
  u[i]:=sl*hk[i]-s2*Adg[i];                           {1. (6-11) képlet}
for i:=1 to n do
  for j:=1 to n do                                   {1. (6-9) képlet}
    Ak[i,j]:=Ak[i,j]+sl*hk[i]*hk[j]-
      s2*Adg[i]*Adg[j]+s*u[i]*u[j];
  for i:=1 to n do                                   {(1) hk számítása}
    begin hk[i]:=0.0;
      for j:=1 to n do
        hk[i]:=hk[i]-Ak[i,j]*g[j];
      end;
until vege or (k>iter);
if not vege then                                     {iteráció vége}
  Hiba('Az iterációk száma nagyobb a megengedettnél!')
else iter:=k;
end;

```


7. Gauss-szelő módszer

Tekintsük az

$$f(x) = 0, \quad f: R^n \rightarrow R^m, \quad m \geq n \quad (7-1)$$

nemlineáris egyenletrendszer *legkisebb négyzetes megoldását*, vagyis az

$$F(x) = ||f(x)||^2 = f^T(x) f(x) \quad (7-2)$$

n változós függvény minimalizálásának a feladatát.

Az alkalmazásokban az $x \rightarrow f(x)$ függvényt általában egy meglehetősen gépidő-igényes szubrutin valósítja meg, és gyakran az f függvény deriváltjai sem állnak rendelkezésre, esetleg csak nagyon nehézkesen számíthatók ki. Ezért a (7-2) funkcionál minimalizálására csak ún. *deriváltmentes* módszerek alkalmazhatók.

Néhány deriváltmentes módszer összehasonlításából a Gauss-szelő módszer látszik a leggazdaságosabbnak, mivel a minimalizáló vektor azonos pontosságú közelítéséhez általában egy nagyságrenddel kevesebb függvényértékszámítást igényel, mint az összehasonlításban szereplő többi deriváltmentes módszer.

Jelölje $x_1, \dots, x_{n+1}$ a (7-2) alatti $F(x)$ függvény x^* minimalizáló vektorának ismert közelítéseit. A három $r_k, k = 1, 2, 3$ helyvektorú ponton keresztül kétdimenziós sík fektethető át, amelynek

$$r = r_1 + (r_2 - r_1) q_1 + (r_3 - r_1) q_2$$

az egyenlete, ehhez hasonlóan az $n+1$ számú

$$r_k = \begin{bmatrix} x_k \\ f(x_k) \end{bmatrix}, \quad k = 1, \dots, n+1, \quad x_k \in R^n, \quad f(x_k) \in R^m$$

ponthoz olyan n dimenziós sík illeszthető, amelynek

$$\begin{bmatrix} x \\ y(x) \end{bmatrix} = \begin{bmatrix} x_1 \\ f(x_1) \end{bmatrix} + \begin{bmatrix} x_2 - x_1 \\ f(x_2) - f(x_1) \end{bmatrix} q_1 + \dots + \begin{bmatrix} x_{n+1} - x_1 \\ f(x_{n+1}) - f(x_1) \end{bmatrix} q_n \quad (7-3)$$

az egyenlete.

Ezzel az egyenlettel $y: R^n \rightarrow R^m$ affin interpolációs függvényt értelmeztünk, amely az $f(x)$ nemlineáris függvény közelítése. Ezért a (7-2) alatti $F(x)$ nemlineáris függvény is helyettesíthető a

$$||y(x)||^2 = y^T(x) y(x)$$

n változós kvadratikus függvénnyel, amelynek minimalizáló vektora a (7-2) alatti $F(x)$ függvényt minimalizáló x^* vektor következő, x_{n+2} -edik közelítésének tekinthető.

A (7-3) második egyenlete kifejezhető

$$y = b - A q, \quad b = f(x_1), \quad q = [q_1, \dots, q_n]^T,$$

$$\mathbf{A}_{m \times n} = [\mathbf{f}(x_1) - \mathbf{f}(x_2), \dots, \mathbf{f}(x_1) - \mathbf{f}(x_{n+1})] \quad (7-4)$$

alakban, és így a minimalizálandó kvadratikus függvény a

$$\|y(x(q))\|^2 = (\mathbf{b} - \mathbf{A}q)^T (\mathbf{b} - \mathbf{A}q) \quad (7-5)$$

alakot ölti. A keresett következő közelítést, az

$$x_{n+2} = (1 - q_1 - \dots - q_n)x_1 + q_1x_2 + \dots + q_nx_{n+1}$$

vektort a (7-3) első egyenletéből nyerjük.

Ezzel be is fejeztük az iteráció első lépését, amely a szelőmódszer lényegét képezi.

Egy további közelítés kiszámításához ismét $n+1$ közelítő vektor ismeretére van szükség. A rendelkezésre álló $n+2$ közelítő vektor közül egyet el kell hagyni. Célszerűnek látszik azt a vektort elhagyni, amelyhez a legnagyobb (7-2) függvényérték tartozik, esetleg azt, amelyet a legrégebben számítottunk ki.

A Gauss-szelőmódszer (mint iteráció) minden egyes lépésében q vektor szerint minimalizálni kell a (7-5) másodfokú függvényt, vagyis meg kell határozni az $\mathbf{A}q = \mathbf{b}$ túlhatározott lineáris algebrai egyenletrendszer legkisebb négyzetes megoldását. A második fejezet 6. pontjában megmutattuk, hogy ha az $m \times n$ típusú ($m \geq n$) $\mathbf{A}$ mátrix r rangja kisebb, mint n , akkor a q legkisebb négyzetes megoldás nem egyértelmű, de egyértelművé tehető, ha a q megoldások közül a legkisebb euklideszi normájú q^* vektort választjuk. Beláttuk azt is, hogy ha elvégezzük az $\mathbf{A}$ mátrix $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ alakú szinguláris értékek szerinti felbontását, akkor

$$q^* = \mathbf{V} \Sigma^+ \mathbf{U}^T \mathbf{b}, \quad (7-6)$$

ahol Σ^+ n -ed rendű diagonálmátrix (főátlóelemeit a zérustól különböző szinguláris értékek reciprokai és nullák alkotják).

A Gauss-szelőmódszer alapvető tulajdonsága, hogy a (7-4) alatti $\mathbf{A}$ mátrix tart a nullamátrixhoz. A (7-2) függvényt minimalizáló x^* vektor közelében fokozatosan romlik az $\mathbf{A}$ mátrix oszlopainak lineáris függetlensége, ami miatt $\mathbf{A}$ szingulárisává válik. Ez mégsem akadályozza meg a Gauss-szelőmódszer működését, ha a (7-5) alatti kvadratikus függvény minimalizálására, vagyis az

$$\mathbf{A}q = \mathbf{b}, \quad \mathbf{A} : R^n \rightarrow R^m, \quad m \geq n$$

lineáris algebrai egyenletrendszer legkisebb négyzetes megoldására az $\mathbf{A}$ mátrix szinguláris értékek szerinti felbontásán alapuló (7-6) összefüggést alkalmazzuk.

Ugyanis ekkor az r rangú és (a spektrál normára vonatkozó) σ_1 / σ_n kondíciószámú $\mathbf{A}$ mátrixot σ_1 / σ_r kondíciószámúra „defláljuk”, ahol σ_1 , σ_n és σ_r rendre az $\mathbf{A}$ legnagyobb, legkisebb és legkisebb pozitív szinguláris értéke. Ez az eljárás addig működik, amíg az $\mathbf{A}$ mátrix zérusmátrix nem lesz, de ekkor már megkaptuk a megoldást.


```

procedure GaussSzelo(      n,m:integer;
                           xx: tVn1xn;
                           c: real;
                           fpx: tVm;
                           var fnpx: real;
                           var iter: integer);

```

{ A GaussSzelo eljárás alkalmas a (7-1) alakú nemlineáris egyenletrendszer legkisebb négyzetes megoldására, vagyis a (7-2) funkcionált minimalizáló x^* vektor meghatározására.

A bemenő paraméterek: n az x vektor elemeinek száma; m az $f(x)$ vektor elemeinek száma; xx egy $n+1$ elemű vektor, melynek elemei n elemű vektorok ($n+1 \times n$ elemű mátrix), az $n+1$ vektor az iteráció indításához szükséges vektorokat tartalmazza. Egyetlen $xx[1]$ kezdővektor megadása is elegendő ($c > 0$), ekkor a GaussSzelo eljárás a többi n vektort az

$xx[j] = xx[1] + c * e[j]$, $j = 2, n+1$
összefüggéssel generálja, ahol $e[j]$ jelöli az egységmátrix j -edik oszlopát. Iter az iterációs lépések számának felső korlátja, ha a számítás befejezésére vonatkozó feltétel iter számú lépésen belül nem teljesül, akkor ez kifrödik és a vezérlést a hívó program veszi át.

A kimenő paraméterek: Az $xx[1]$ vektor a nemlineáris egyenletrendszer legkisebb négyzetes megoldását (az x^* vektort); az fpx vektor az $f(x^*)$ vektort; az $fnpx$ változó az $f(x^*)$ vektor euklideszi normánegyzetét; iter az iterációk számát (ha $iter=0$, akkor hiba volt) tartalmazza.

A GaussSzelo eljárást használó programban definiálni kell a következő globális típusokat

```

tVn      = array [1..nmax] of real;
tVn1     = array [1..nmax+1] of real;
tVm      = array [1..mmax] of real;
tVn1xn   = array [1..nmax+1] of tVn;
tVn1xm   = array [1..nmax+1] of tVm;

```

ahol az $nmax$ és $mmax$ konstansok az n és m legnagyobb lehetséges értékei. A minimalizálandó függvényt a

```

procedure fgvnxm( n,m: integer;
                  x: tVn;
                  var f: tVm);

```

eljárásként kell megadni, ahol x az aktuális pontot tartalmazó n elemű vektor, m az egyenletek, n az ismeretlenek száma, f a függvényértékeket tartalmazó m elemű vektor. }

```

var
  fx: tVn1xm;           {az n+1 pont függvényértékei}
  fnx: tVn1;            {az n+1 pont függvéynormája}
  A: tVmxn;             {1. (7-4)}
  V: tVnxn;             {szinguláris értékek szerinti felbontáshoz}
  b: tVm;               {1. (7-4)}
  q,D: tVn;            {(7-4) ill. szing. ért. felbontáshoz}
  i,j,k,                {ciklus változók és az iteráció száma}
  imax,imin: integer;   {legnagyobb és legkisebb normájú pont}
  sl,s2,s: real;        {segédváltozók}

```



```

begin
  if c<>0.0 then {kezdőpontok kiszámolása}
    for i:=1 to n do
      begin
        for j:=1 to n do
          xx[i+1,j]:=xx[1,j];
          xx[i+1,i]:=xx[i+1,i]+c;
        end;
      for i:=1 to n+1 do {függvényértékek, normák számítása}
        begin fgvnxm(n,m,xx[i],fx[i]);
          s:=0; for j:=1 to m do s:=s+sqr(fx[i,j]); nfx[i]:=s;
        end;
      k:=0; s2:=1; s1:=1;
      while (k<iter) and (s2>1.0e-9*s1) do {iteráció}
        begin inc(k);
          for j:=1 to m do b[j]:=-fx[1,j];
          for i:=1 to n do
            for j:=1 to m do A[j,i]:=fx[i+1,j]-fx[1,j];
            imax:=1; imin:=1;
            for i:=2 to n+1 do
              begin {min. és max. függvényérték keresése}
                if fnx[i]>fnx[imax] then imax:=i
                else if fnx[i]<fnx[imin] then imin:=i;
              end;
            SzEfelbont(A,m,n,D,V); {A felbontása}
            SzEnullaz(d,n);
            SzEmegold(A,D,V,m,n,b,q); {A q = b megoldása}
            s:=1.0; for i:=1 to n do s:=s-q[i];
            for i:=1 to n do
              begin s2:=s*xx[1,i];
                for j:=1 to n do
                  s2:=s2+xx[j+1,i]*q[j];
                  xx[imax,i]:=s2; {a legnagyobb normájú pont helyén}
                end;
              fgvnxm(n,m,xx[imax],fx[imax]);
              s:=0; for j:=1 to m do s:=s+sqr(fx[imax,j]);
              fnx[imax]:=s;
              if fnx[imax]<fnx[imin] then imin:=imax;
              s1:=0.0; s2:=0.0; {konvergenciafeltétel ellenőrzése}
              for i:=1 to n+1 do s1:=s1+fnx[i]; s1:=s1/(n+1);
              for i:=1 to n+1 do s2:=s2+sqr(fnx[i]-s1);
              s2:=sqrt(s2/(n+1));
              s1:=abs(fnx[imin]); if s1<1.0e-5 then s1:=1.0e-5;
            end;
          if k>=iter then {iteráció vége}
            Hiba('Az iteráció sorszáma nagyobb a megengedettnél.')
          else
            begin iter:=k; {Kimenő paraméterek beállítása}
              if imin<>1 then
                for i:=1 to n do
                  xx[1,i]:=xx[imin,i];
                for i:=1 to m do fpx[i]:=fx[imin,i];
                fnpx:=fnx[imin+1];
              end;
            end;
          end;
end;

```


IRODALOMJEGYZÉK

- [1] *Bahvalov, N. Sz.*: A gépi matematika numerikus módszerei. Budapest, Műszaki Könyvkiadó, 1977.
- [2] *Bathe, K. J. – Wilson, E.*: Numerical Methods in Finite Element Analysis. Englewood Cliffs, N. J., Prentice-Hall, 1976.
- [3] *Fletcher, R.*: Practical Methods of Optimization (Vol. 1, Unconstrained Optimization). Chichester, John Wiley and Sons, 1972.
- [4] *Forsythe, G. E. – Moler, C. B.*: Lineáris algebrai problémák megoldása számítógéppel. Budapest, Műszaki Könyvkiadó, 1976.
- [5] *Kielbasinski, A. – Schwetlick, H.*: Numerische lineare Algebra. Berlin, VEB Deutscher Verlag der Wissenschaften, 1988.
- [6] *Míka, S.*: MVŠT. Numerické metody algebry. Praha, SNTL, 1985.
- [7] *Ortega, J. M.*: Numerical Analysis. New York, Academic Press, 1972.
- [8] *Ortega, J. M. – Rheinboldt, W.*: Iterative Solution of Nonlinear Equations in Several Variables. New York, Academic Press, 1970.
- [9] *Parlett, B. N.*: The Symmetric Eigenvalue Problem. Englewood Cliffs, N. J., Prentice-Hall, 1980.
- [10] *Polak, E.*: Computational Methods in Optimization. New York, Academic Press, 1971.
- [11] *Press, W. H. – Flannery B. P. – Teukolsky S. A. – Vetterling W. T.*: Numerical Recipes. New York, Cambridge University Press, 1988.
- [12] *Ralston, A.*: Bevezetés a numerikus analízisbe. Budapest, Műszaki Könyvkiadó, 1969.
- [13] *Stoer, J. – Bulirsch, R.*: Introduction to Numerical Analysis. New York, Springer-Verlag, 1983.
- [14] *Wilkinson, J. H.*: The Algebraic Eigenvalue Problem. Oxford, Clarendon Press, 1969.
- [15] *Wilkinson, J. H. – Reinsch, C.*: Linear Algebra (Handbook for Automatic Computation). New York, Springer-Verlag, 1971.

ELJÁRÁSOK JEGYZÉKE

LINEÁRIS ALGEBRAI EGYENLETRENDSZEREK

LUfelbont	LU felbontás	15
LUmegold	Lineáris egyenletrendszer megoldása LU felbontással	17
	Inverz mátrix kiszámítása	18
	Determináns kiszámítása	19
	Determináns pontosabb kiszámítása	19
LUjavit	A megoldás iterációs javítása LU felbontás esetén	27
LLTfelbont	Pozitív definit, szimmetrikus mátrix Cholesky-felbontása	32
LLTmegold	Lineáris egyenletrendszer megoldása Cholesky-felbontással	33
LLTjavit	A megoldás iterációs javítása Cholesky-felbontás esetén	33
LDLTfelbont	Körvonal technikával tárolt pozitív definit szalagmátrix LDL ^T felbontása	37
LDLTmegold	Lineáris egyenletrendszer megoldása LDL ^T felbontás esetén	38

VALÓS SZIMMETRIKUS MÁTRIX SAJÁTÉRTÉKEI ÉS SAJÁTVEKTORAI

Felez	Givens-féle felezési eljárás	58
Alteriteracio	Altériteráció	77
AlterEll	Az altériterációval nyert értékek ellenőrzése	80
KezdoAlter	Az iterációhoz szükséges kezdővektorok meghatározása	81
Jacobi	Jacobi-diagonalizálás	87
Rendez	Sajátpárok rendezése	89
AltJacobi	Altalánosított sajátérték-feladatra alkalmazott Jacobi-diagonalizálás	90
Red3Atlos	Redukció háromátlós alakra	96
ImplicitQL	Implicit QL algoritmus	105
SzEfelbont	Szinguláris értékek szerinti felbontás	117
SzEmegold	Lineáris egyenletrendszer megoldása szinguláris érték szerinti felbontással	124
SzEnullaz	A kis szinguláris értékek nullázása	125

NEMLINEÁRIS EGYENLETRENDSZEREK

NewtonRaphson	Newton–Raphson-módszer	135
Arany Metsz es	Arany metszés szerinti keresés	140
Kozrefog	A minimum kezdő közrefogása	141
BrentDeriv	Brent-módszer a deriváltak felhasználásával	144
KonjGradiens	Konjugált gradiensmódszer	155
LinMin	Egyenes menti minimalizálás	156
DFPMin	Kvázi-Newton-módszer	161
GaussSzelo	Gauss-szelőmódszer	165

TÁRGYMUTATÓ

A

affin interpolációs függvény 163

algoritmus

BFGS-~ 157

DFP-~ 157

QR és QL ~ 98-106

altér

domináló ~ 68

invariáns ~ 68

altériteráció 68-82

aranymetszés szerinti keresés 137-140

Á

általánosított

~ Jacobi-módszer 89

~ megoldás 122, 127

~ sajátértékfeladat 74-76

B

B-ortonormált 75

Brent-módszer 142-145

Broyden-Fletcher-Goldfarb-Shanno
(BFGS-algoritmus) 157

C

Cholesky-felbontás 28

Ciklikus Jacobi-módszer 86

Courant-Fischer min-max
reprezentáció 107

Cramer-szabály 9

Cs

csúsztatott háromszögtechnika 39

D

Davidon-Fletcher-Powell
(DFP-algoritmus) 157

defláció 66, 164

deriváltmentes 163

determináns 18, 19

diagonalizálás

Jacobi-~ 82

direkt módszerek 10

domináló

~ altér 68

~ sajátérték 60, 68

~ sajátvektor 60

E

egydimenziós minimalizálás 137-145

egyenletrendszer 9-10

gyengén meghatározott ~ 20

homogén lineáris ~ 9, 51

túlhatározott ~ 122, 127

előjelváltások száma 56

F

felbontás

Cholesky-~ 28

pozitív definit szalagmátrixé 34

trianguláris ~ 13

felülrelaxálás 47

Fletcher-Reeves-formula 154

forgatómátrix 83

főelem 12

~ kiválasztás 12

részleges ~ 12

teljes ~ 12

főminor 11

Fréchet-derivált 128

frontális technika 39

G

Gauss-elimináció 11

Gauss-Newton-módszer 132-134

Gauss-Newton-iteráció egydimenziós
kereséssel 136-137

Gauss-Seidel-eljárás 45

Gauss-szelőmódszer 163-166

Givens-féle felezési eljárás 54

gradiensmódszer 146

gradiensvektor 129, 131

Gram-Schmidt

~ ortogonalizálás 67

~ biortogonalizálás 151

Gy

gyengén meghatározott 20, 106

H

hasznosítási transzformáció 53, 82
 ortogonális ~ 82
hatványiteráció 60
háromszögfelbontás 10
Hesse-mátrix 129, 131
hibaanalízis 20, 106
hibavektor 21
homogén lineáris egyenletrendszer 9
Householder
 ~-transzformációk 92
 ~-tükrözés 93

I

instabilis 20
interpoláció 163
 parabolikus ~ 142
invariáns altér 68
inverz interpoláció
 másodfokú ~ 142
invertálás 10, 18
inverziteráció 62
 szimultán ~ 75
iteráció befejezésére vonatkozó feltétel 43
iterációk
 ~ javítás 27
 ~ módszer
 ~ konvergenciája 40-42
 stacionárius ~ 40

J

Jacobi
 általánosított ~-módszer 89
 ~-diagonalizálás 82-92
 ~ forgatómátrix 83
 ~-iteráció 44
 ~-mátrix 128

K

karakterisztikus polinom 54
kerekítési hibák 12
 halmozódó ~ 20
kondíciós szám 25, 164
 spektrál ~ 25
konjugált gradiensmódszer
 Fletcher-Reeves-változat 154
 Polak-Ribiere-változat 154
konjugált gradiensmódszerek 148-157
konjugált irányok módszerek 148-150

konvergencia 40-42

 ~ gyorsasága 44, 130
 lineáris 130, 133, 140
 másodrendű 132, 134
 szuperlineáris 130
 ~ gyorsítása Rayleigh-hányadossal 64
 ~ tényező 61-63

konvizstensen rendezett mátrix 48
korlátos pontosságú aritmetika 9, 12
körvonal (sky-line) technika 36
kvázi-Newton-módszerek 157-162

L

lineáris algebrai egyenletrendszer
 ~ megoldásának létezése és egyértelműsége 9
 ~ általánosított megoldása 112-125
lineáris egyenletrendszerek legkisebb négyzetes megoldása 122
lineáris konvergencia 130
 ~ tényező 60
LU felbontási tétel 11

M

másodfokú inverz interpoláció 142
mátrix
 forgató ~ 83
 ~ háromszögfelbontása 10
 ~ invertálása 10, 18
 ~ LU felbontása 10
 ~ rangja 9
 ~ szinguláris értékek szerinti felbontása 112

mátrixdefláció 66

minimum

 ~ kezdő közrefogása 141
 ~ közrefogása 137-140

módszer

 Householder-~ 92-98

 Jacobi-~ 82-89

 általánosított Jacobi-~ 89-92

multiplicitás

 sajátértéké 51

N

- nemlineáris egyenletrendszer
 - ~ általánosított megoldása 127
 - ~ legkisebb négyzetes megoldása 127, 163
- nem szinguláris 10, 20, 23, 25, 29, 40
- Newton-Raphson-módszer 134-135
- norma 22
 - euklideszi vagy Frobenius-féle ~ 22
 - végtelen vagy maximum ~ 22
 - spektrál ~ 22
- normálegyenletek 122

O

- optimális relaxációtényező 47

P

- permutációs mátrix 13
- perturbáció 23, 24
- Polak-Ribiere-formula 154
- pozitív definit mátrix Cholesky-felbontása 28
- pozitív definit szalagmátrix 34-39
- projekció elv 72
- pszeudoinverz mátrix 121

Q

- QL algoritmus implicit eltolásokkal 102-106
- QR-iteráció 92
- QR és QL algoritmus 98-106

R

- rang 9, 164
- Rayleigh-hányados 64
 - ~ iteráció 65
- Rayleigh-Ritz-módszer 71-74
- Rayleigh-Ritz-algoritmus 76
- redukció
 - ~ kétátlós mátrixra 114
 - ~ háromátlós mátrixra 92
- relaxáció 47
- Ritz-féle sajátértékek és sajátvektorok 73
- rosszul kondicionált 20

S

- sajátaltér 51
- Schwarz-egyenlőtlenség 22
- skaláris szorzatok kettős pontosságú számítása 15
- skalázás 15
- sky-line (körvonal) technika 36
- SOR módszer 46

- sorcserék 12
- sorok kiegyensúlyozása 14, 15
- spektrál
 - ~ norma 22
 - ~ kondíciós szám 25
- spektráletolások 63
- stacionaritási
 - ~ pont 131
 - ~ feltétel 146
- Sturm-sorozat 55

Sz

- szelőmódszer 163
- szimultán
 - ~ hatványiteráció 70
 - ~ inverziteráció 75
- szimmetrikus pozitív definit mátrix 28-39, 46
- szimmetrikus pozitív definit szalagmátrix 34-39
 - ~ tárolása 35-39
- szinguláris értékek 22, 112, 164
 - ~ szerinti felbontás 112
- szukcesszív felülrelaxálás módszere 46
- szuperlineáris 130

T

- tényező 60
- turnus 86
- túlhatározott egyenletrendszer 122, 127
- transzformáció
 - hasonlósági ~ 53, 82
 - ortogonális ~ 82

U

- utóiteráció 26

V

- változó metrikájú módszerek 157-162
- vektordefláció 67
- visszahelyettesítés 11

A TYPOT_EX KIADÓ AJÁNLATA

Róka Sándor

1000 feladat az elemi matematika köréből

Az elemi matematika — definíció szerint — nem lép túl a középiskolai anyagon. Ez az 1000 feladatos feladatgyűjtemény érdekes, sőt időnként meglepő feladatokat tartalmaz. A megoldások tömörek és frappánsak. Jó néhány fejezete gazdag tárházát nyújtja olyan feladatoknak, amelyeket más példatárak mellőznek, pl. Függvényegyenletek, Geometriai konstrukciók stb., de a klasszikus témakörökből is tartalmaz friss feladatokat.

Az olvasó eldöntheti, hogy a nehézségi fok szerint skálázott feladatok közül melyekkel birkózik meg.

Ára: 210,- Ft

Stoyan Gisbert – Takó Galina

Numerikus módszerek I.

Ez a tankönyv elvezeti az olvasót az alapoktól a posztgraduális szintig a numerikus matematika terén. A legfontosabb numerikus módszereket tárgyalja a következő területeken: lineáris egyenletrendszerek, legkisebb négyzetek, sajátérték-feladatok, interpoláció, approximáció, közelítő integrálás, nemlineáris egyenletek és egyenletrendszerek.

Az elméleten kívül a numerikus feladatok tipikus alkalmazásairól is ad információt, valamint számítási tapasztalatokat és pszeudokódos algoritmusokat is közöl.

A kiadványnak fontos részét képezik a floppy lemezen mellékelt programok, amelyek MS-DOS alatt futnak, menüvezéreltek és színes ablaktechnikát használnak.

Ára: kb. 450 Ft

Kőnig Dénes

Mathematikai mulatságok

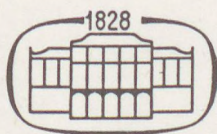
A magyar matematikai iskola század eleji vonulatának egyik kiemelkedő alakja volt Kőnig Dénes. Könyve idegen a hagyományos feladatgyűjteményektől, az iskolás szemlélettől, annak minden merevségétől. Azóta többségünkben klasszikussá vált feladatokról ír, sallangmentes megfogalmazásban. Szakmai színvonalán túl egy további nagy érdeme is van a műnek: Kőnig stílusa kicsit archaikus, de csodaszép. A mindenütt burjánzó sematikus és igénytelen beszéd és írás korában különös érték.

Ára (minisztériumi támogatással): 69,- Ft

TYPOT_EX Kft. Elektronikus Kiadó

1015 Budapest, Batthyány u. 14.

Tel.: 202 13 65 Fax: 115 42 12



AZ AKADÉMIA KIADÓ AJÁNLATA

Matematikai zseblexikon

Szerkesztette: Oláh Judit

A matematika jól körülhatárolt fogalmai mind műszaki, mind gazdasági gondolkodásunk háttéréül szolgálnak. Jogos tehát az igény a gyakorlati nézőpontú rendszerezésre. A Matematikai Zseblexikonban mintegy 1000 fogalom rövid, lényegretörő ismertetését adja az olvasó kezébe az Akadémia Kiadó és a TYPOTEX Kft.

Ára: 415,- Ft

Baranyi Károly

A fizikai gondolkodás iskolája

1. kötet: Mechanika

2. kötet: Termodinamika; elektrodinamika és optika; atomfizika

3. kötet: Feladatmegoldások

A felvételi vizsgára készülő diákoknak, középiskolásoknak, tanároknak, tanárjelölteknek és az érdeklődő szülőknek készült ez a nagyszabású mű. A fizika elméletét 300, részletesen tárgyalt problémán keresztül mutatja be, és ezekhez a kérdéskörökhöz 800 feladatot tartalmazó példatár tartozik a feladatok részletes megoldásával.

A három kötet ára: 1850,- Ft

Ian Stewart

A matematika problémái

Mi a matematika? Mivel foglalkoznak ma a matematikusok? Mi az, ami igazán fontos? Mit mond minderről — és egyéb hasonló kérdésekről — egy matematikus? Hogyan próbálja mindezt elmagyarázni és beavatni a kívülállókat? Erről szól a könyv, Ian Stewart személyes élménybeszámolója a matematikáról.

Ára: 500,- Ft

A kiadványok megvásárolhatók:

STÚDIUM Akadémia Könyvesbolt

1052 Budapest, V. Váci u. 22.

MAGISZTER Akadémia Könyvesbolt

1052 Budapest, V. Városház u. 1.

FAMULUS Akadémia Könyvesbolt

1137 Budapest, XIII. Újpesti rkp. 5.

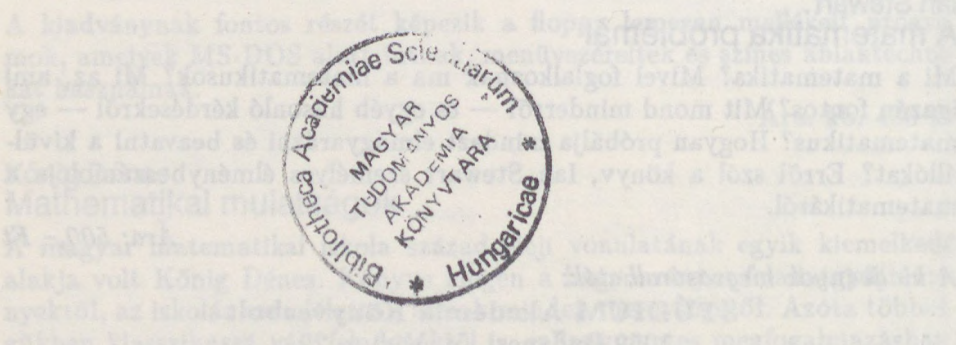
BÁLINT SÁNDOR Akadémia Könyvesbolt

6720 Szeged, Aradi vértanúk tere 8.

Megrendelhetők:

Akadémia Kiadó Kereskedelmi Osztálya

1117 Budapest, XI., Prielle Kornélia u. 19-35.



Kiadta az Akadémiai Kiadó és a TYPOT_EX Kft. Elektronikus Kiadó.

A kiadásért felelős:

Zöld Ferenc igazgató és Votisky Zsuzsa ügyvezető igazgató.

A nyomás és kötés az Akadémiai Nyomdában készült.

Felelős vezető: Zöld Ferenc. Felelős szerkesztő: Gyömöre Mihályné.

A fedéltervet készítette: Fodor Attila. Műszaki szerkesztő: Csizmás Ferenc.

Megjelent: 15,7 (A/5) fv terjedelemben.

Akadémiai Kiadó ISBN 963 05 6454 8

TYPOT_EX Kft. ISBN 963 7546 23 5

Ez a könyv a numerikus módszereket, számítástechnikát tanuló mérnökhallgatók számára tankönyv, a tervező és kutató mérnököknek „receptkönyvként” is használható programgyűjtemény.

A szerkezettervezésben alkalmazott finitizálási eljárások – pl. a végeelem-módszer – nagyméretű egyenletrendszereket eredményeznek. Ezek általában speciális tulajdonságú lineáris vagy nemlineáris egyenletrendszerek, attól függően, hogy az eredeti feladatban a fizikai és geometriai nemlinearitások hatását figyelembe vették-e. A stabilitási feladatok és a dinamikai hatások figyelembevétele algebrai sajátérték-feladatok megoldására vezet.

Ebből adódóan a könyv a következő fejezeteket tartalmazza:

- Lineáris algebrai egyenletrendszerek megoldása
- Valós szimmetrikus mátrix sajátértékeinek és sajátvektorainak meghatározása
- Nemlineáris egyenletrendszerek megoldása

A könyvben a gyakorlatban jól használható számítógépes eljárások megjegyzésekkel ellátott Pascal forrásnyelvű listái is megtalálhatók. A listák mágneses lemezen 100 Ft-os áron a TypoT_EX Kft. boltjában (1015 Budapest, Batthány u. 14.) megvásárolhatók, ill. megrendelhetők.

Reméljük, könyvünket mérnökök és mérnökhallgatók, továbbá operációkutatással foglalkozó közgazdászok és programozó matematikusok is haszonnal forgatják majd.

AKKADÉMIAI KIADÓ · TYPOTEX

Popper–Csizmás

NUMERIKUS MÓDSZEREK

