tanulmányok

26/1974

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZET

NUMERIKUS MÓDSZEREK SPARSE MÁTRIXOKRA

Irta:

dr. Gergely József

TANULMÁNYOK 26/1974

A kiadásért felelős:
Dr. Arató Mátyás

Technikai szerkesztő
Révész Györgyi

MTA KESZ Sokszorosító. F.v.: Szabó Gyula

TARTALOMJEGYZÉK

1. Sparse mátrixok

1.1 Bevezetés, nagyméretű rendszerek

A számítógépek elterjedése után lehetőség nyílt nagyméretű lineáris rendszerek gépi kezelésére. A lehetőségek birtokában természetesen az igények is megnőttek, illetve a korábban is meglévő igények reális alapot nyertek. De a nagyméretű lineáris programozási feladatok megoldhatóságának, a parciális differenciálegyenletek numerikus gépi megoldhatóságának korlátait napjainkban is a számítástechnikai berendezéseink memóriakapacitása szabja meg. Nagyméretű feladataink megoldásához olyan módszereket kell találnunk, amelyek a korlátozott gépkapacitás mellett is jól használhatók lesznek. Tehát nem feltétlenül új módszert kell keresnünk, hiszen lineáris rendszerek esetén a meglévő módszerek száma úgyis elég nagy, hanem a meglévő (elég gazdag) készletből válasszuk ki az adott feladatra a legmegfelelőbbet. Ha ilyen nincs, akkor természetesen új módszert kell kidolgoznunk.

A számítógépek említett szűk kapacitása operatív memóriájuk korlátozottságában nyilvánul meg. Lineáris rendszerek gépi kezelésénél többnyire téglalap mátrixokkal kell dolgoznunk. Egy $m \cdot k$ méretű mátrix esetén $m \cdot k = 10.000$ nagyságú mátrix egy reális felső korlát, amivel még tudunk könnyen boldogulni, de ha mindkét méret csak a duplájára növekszik, akkor $2m \cdot 2k = 40.000$-es tömböt kapunk, aminek kezelésére mostani gépeink nem igen alkalmasak. Szükség lehet például a módszereink olyan módosítására, amelyek a mátrixnak mindig csak egy részével dolgoznak a gép operatív memóriájában, a többit egy háttér memóriában tárolják.

Szembetűnőbb a méretek növekedése egy háromdimenziós (vagy magasabb dimenziós) probléma megoldásánál. Oldjunk meg egy parciális differenciálegyenlet egy kocka tartományon a véges differenciák módszerével. Ha a rácspontokat úgy vesszük fel, hogy minden él irányában 10 részre bontjuk a kockát, már akkor is 1000 rácspontot kapunk. A megoldandó egyenletrendszer 1000 ismeretlenes, amelynek megoldása az általános egyenletrendszer megoldó programmal lehetetlen.

Feladataink többségében nagyméretű mátrixok gépi kezelésére az ad lehetőséget, hogy a mátrixok nagyon sok 0 mátrixelemet tartalmaznak. A sok 0-t tartalmazó mátrixok a sparse (ritka) mátrixok. Tanulmányunk tárgya olyan módszerek ismertetése, vizsgálata, amelyek sparse mátrixok invertálására, sparse mátrixú lineáris egyenletrendszerek megoldására, sparse mátrixok sajátértékeinek kiszámítására alkalmasak.

A differenciálegyenletek numerikus megoldásánál fellépő sparse mátrixok speciális alakúak, szalagmátrixok (a nem 0 elemek a főátlóban és azzal párhuzamos vonalakkal kijelölt sávban helyezkednek el). Az alkalmazott differencia sémától és a vizsgált tartománytól függően a nem 0 elemek szabályosan helyezkednek el. Sparse mátrixok leggyakrabban a

hálózatszámításban (elektromos és gázhálózatok) lépnek fel.

Tekintsünk egy n csomópontból álló elektromos hálózatot. Legyen a csúcsponti feszültségek illetve áramok vektora $\underline{U}$ illetve $\underline{I}$, ekkor fennáll az

$$Y\underline{U} = \underline{I}$$

egyenletrendszer, ahol Y a csúcsponti admittancia mátrixa. A szokásos mértékben hurkolt hálózatoknál a vezetékek száma körülbelül $3/2 \cdot n$. Ennek megfelelően (minthogy minden vezeték két csomóponthoz kapcsolódik és minden csomópontnak az önmagához viszonyított admittanciája nem 0) az Y mátrix körülbelül $4n$ darab 0-tól különböző elemet tartalmaz a n^2 elemből, a többi 0.

A hálózatszámításban fellépő sparse mátrixok struktúrájában szabályszerűséget általában nem tételezhetünk fel, a nem 0 elemek rendszertelenül helyezkednek el. A sparsitás (ritkaság) mértékét a nem 0 elemek százalékos megadásával szokták jellemezni.

Vizsgálatainkban a sparse mátrix struktúrájában nem tételezünk fel szabályszerűséget. A 4. pontban foglalkozunk szalagstruktúrájú mátrixok esetével.

Nagy rendszerek vizsgálatánál fontos szempontként merül fel a stabilitás kérdése. Egy módszer, ami kis adatrendszer esetén nagyon jó lehet, nagy adathalmazra (nagy mátrixokra) esetleg teljesen használhatatlan. A gépi numerikus problémák kutatásának egyik fő iránya a nagyméretű rendszerek megoldhatósága géppel, ugyanis a kis adatokkal jól működő módszerek nem biztos, hogy alkalmazhatók nagy rendszerek esetén (pl. a módszer nem stabil). Ezért a rendszerek kiválasztásánál a stabilitásra külön gondot kell fordítanunk. Tárgyalt módszereink többnyire stabilok. Az egyes módszereknél a stabilitást külön nem vizsgáljuk meg.

1.2 Módszerek általános jellemzése

Lineáris egyenletrendszerek megoldására és a mátrixok invertálására nagyon sok numerikus módszer ismeretes. (Lásd pl.: 9., 11., 12., 13., 36., 38. és 39.) Az általános módszerek nyilván használhatók sparse esetben is, azonban nem célszerű a használatuk, mert esetleg sok felesleges munkát végzünk azáltal, hogy az együtthatók közt szereplő sok 0-okkal is elvégezzük feleslegesen a kijelölt műveleteket. Másrészt számítógépes számolásnál a gép memóriáját feleslegesen terheljük le, ha a 0-okat is tároljuk. Célszerű kidolgozni speciális módszereket sparse mátrixok esetére.

A sparse mátrixok kezelésére kidolgozott módszerek többsége az általános módszerek valamilyen specialitását használja ki. Az ilyen módszerek arra irányulnak, hogy a mátrixokban fellépő 0 elemeket ne kelljen a számítógépben tárolni és azokkal műveleteket végezni. Természetesen sokszor az alap módszert módosítani is kell, de a módszerek többsége technikai jellegű meggondolásokon alapul.

Egy n ismeretlenes

$$A\underline{x} = \underline{b}$$

lineáris tömörmátrixú egyenletrendszer megoldásához Gauss–eliminációval $\frac{n}{3}(n^2 + 3n - 1)$ nagyságrendű művelet kell, (szorzásokat a és osztásokat számolva) Gauss–Jordan–eliminációval $\frac{n}{2}(n^2 + 4n - 1)$, míg az

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

segítségével $n^3 + n^2$. Sparse esetben a műveleti igény nagyságrendje nyilván a sparsitás %-ától függ. Látható, hogy a mátrixinverzió műveletigényesebb az egyenletrendszer megoldásánál. Akkor célszerű a használata egyenletrendszer megoldásánál, ha az inverzre különben is szükség van, vagy ha ugyanazon baloldalú egyenletet több jobboldallal kell megoldanunk.

Nagyon fontos előrebocsátanunk a következő megjegyzést:
egy sparse mátrix inverze általában nem sparse tulajdonságú. Ezért, ha egy egyenletrendszer megoldásánál ki is tudjuk használni a sparse tulajdonságot, lehet, hogy az inverz alkalmazása esetén már nem. Ebből következik, amint azt látni fogjuk, hogy módszereink az egyenletrendszerek megoldására hatékonyabbak lesznek, mint invertálásra.

A sparse mátrixok számítógép memóriájában való elhelyezése többféle módon történhet:
1) elhelyezzük az egész mátrixot, tehát a 0-okat is. A műveletek végzése előtt vizsgáljuk meg, hogy az illető elem 0-e vagy nem 0, és ettől függően végezzük el a kijelölt műveletet. Ezáltal a számolási időt csökkenthetjük annyival, amennyivel rövidebb időt igényel a 0 vagy nem 0 megnézése, mint a tényleges művelet elvégzése. (Ezt a konkrét számítógép műveleti idő adatai szabják meg, de a vizsgált elem memóriában való megkeresése is időt igényel, ezért ennek a módszernek a hatásfoka nem nagyon jó.)
2) Csak a nem 0 elemeket helyezzük el. Ekkor viszont meg kell jelölni a nem 0 elemek koordinátáit, ami minden nem 0 elemhez annyi segédinformációt jelent, ahány dimenziós tömbről van szó. Ezzel a módszerrel nagyobb sparse mátrixot tudunk tárolni, de a módszerek lesznek komplikáltabbak. A sparsitás %-a dönti el, hogy melyik módot célszerű választani, ha kevesebb a nem 0 elem, nyilván az utóbbi a célszerűbb. A két eset kombinálása és egyéb módszerek is adhatók.

A sparse mátrixú lineáris egyenletrendszer megoldására használatos módszerek három csoportra oszthatók:
1) véges módszerek, amelyek többsége a Gauss–elimináción alapszik, vagy egyéb általános esetre használatos véges módszer speciális alakja;
2) iterációs módszerek, amelyek sparse esetben különösen jók lehetnek;
3) sok módszer gráfelméleti meggondoláson alapszik.
Vannak természetesen más jellegű módszerek, amelyek a vizsgált feladat valamilyen specialitását használják ki.

Részletesen mi a véges és iterációs módszereket elemezzük. A gráfelméleti módszerek lényegét itt csak röviden foglaljuk össze.

A gráfelméleti módszereknél a sparse A mátrixhoz olyan gráfot keresünk, melynek struktúrája megegyezik az A nem 0 elemeinek elhelyezkedési struktúrájával. A gráfon elvégezzük a kívánt egyszerűsítéseket és a kapott megoldást az A matrixra vonatkoztatva, annak célszerű átalakítását érjük el. Ezáltal például az A mátrixot diagonálishoz közeli, nem 0 elemű mátrix-szá transzformáljuk, aminek a további kezelése már egyszerűbb.

Megjegyezzük, hogy az ismertetésre kerülő módszerek mindegyike alkalmazható valós és komplex mátrixelemek esetén is.

1.3 Irodalmi források

A tanulmányhoz bő irodalomjegyzéket mellékeltünk. Alapvetőnek tartjuk az 1 -et és 2-t (melyek körülbelül 35 előadás anyagát tartalmazzák és mintegy 200 irodalmi utalást sorolnak fel).

2. Véges módszerek

2.1 Gauss eliminációs módszer

Az alkalmazott módszerek többsége a Gauss elimináció jól ismert algoritmusát célszerűen alkalmazza sparse esetben. Ezt a 17. cikk alapján ismertetjük.

Induljunk ki az $A\underline{x} = \underline{b}$ egyenletrendszer jobboldalakkal bővített mátrixából

$$\begin{bmatrix} a_{11} & a_{12} \ldots a_{1n} & b_1 \\ a_{21} & a_{22} \ldots a_{2n} & b_2 \\ \ldots\ldots\ldots\ldots & & \\ a_{n1} & a_{n2} \quad a_{nn} & b_n \end{bmatrix} \tag{2.1}$$

Gauss eliminációval a (2.1)-et háromszög mátrix-szá alakítjuk úgy, hogy kiküszöböljük az első oszlop elemeit a_{21}-től kezdve, a második oszlop elemeit a_{32}-től kezdve stb.

A k-adik oszlop eliminálását a következő lépésekben hajtjuk végre:

a) vesszük az $a_{kk}^{(k-1)}$ reciprokát $= d_{kk}$

b) d_{kk}-val megszorozzuk a k-adik sor $a_{kk}^{(k-1)}$-től jobbra lévő elemeit, kapjuk u_{kj}-t, $j = k+1, \ldots, n$, és az utolsó oszlopot is szorozva c_k adódik.

c) $a_{lk}^{(k-1)}$-gyel szorozzuk u_{kj}-t $(l = k+1, \ldots, n,\ j = k+1, \ldots, n)$ és kivonjuk $a_{lj}^{(k-1)}$-ből, kapjuk $a_{lj}^{(k)}$-t.

Az a, b, és c pontokat $k = 1, 2, \ldots, n-1$-re végrehajtva, $l_{ij} = a_{ij}^{(j-1)}$ $(i > j)$ jelölést használva kapjuk a következő elrendezést:

$$\begin{array}{llll} d_{11} & u_{12} & u_{13} \ldots & u_{1n} \quad c_1 \\ l_{21} & d_{22} & u_{23} \ldots & u_{2n} \quad c_2 \\ \ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\ l_{n1} & l_{n2} & l_{n3} \ldots & d_{nn} \quad c_n, \end{array} \tag{2.2}$$

ahol tehát $d_{ii} = \dfrac{1}{a_{ii}^{(i-1)}}$, $u_{ij} = a_{ij}^{(i)}$, $l_{ij} = a_{ij}^{(j-1)}$, $a_{ij}^{(k)}$ jelenti az i-edik sor j-edik elemét a k-adik oszlop eliminálása után. c_i a b_i helyén az elimináció során kialakult szám. (2.2)-ből a megoldás visszahelyettesítésekkel adódik:

$$x_n = c_n, \quad x_i = c_i - \sum_{j=i+1}^{n} u_{ij} x_j, \quad i = n-1, \ldots, 1 \tag{2.3}$$

A Gauss elimináció munkaigényes része a (2.2) táblázat kialakítása. Viszont a (2.2) táblázat segítségével teljesen megfogalmazhatjuk a megoldást, ami tetszőleges más jobboldal mellett is alkalmazható. Több képlet is felírható, pl.:

$$x = A^{-1}\underline{b} = U_1 U_2, \ldots, U_{n-2} U_{n-1} D_n L_{n-1} D_{n-1} L_{n-2}, \ldots, L_2 D_2 L_1 D_1 \underline{b} \tag{2.4}$$

ahol a benne szereplő mátrixok definiciói abban különböznek az n-ed rendű egységmátrixtól, hogy

$$\begin{array}{lll} D_i & i\text{-edik sora} & (0, \ldots, 0, d_{ii}, 0, \ldots, 0) \\ L_i & i\text{-edik oszlopa} & (0, \ldots, 0, 1, -l_{i+1,i}, \ldots, -l_{n,i}) \\ U_i & i\text{-edik sora} & (0, \ldots, 0, 1, -u_{i,i+1}, \ldots, -u_{i,n}) \end{array}$$

(Természetesen a (2.4)-ben kijelölt mátrixszorzások csak formálisak.)

Az A mátrix sparse tulajdonsága jelentkezik a (2.2) táblázatban is, és éppen itt lehet kihasználni a sparsitási sajátosságot és természetesen ennek következtében a (2.4) szorzások végrehajtása közben is.

Tegyük fel, hogy $a_{ij} = 0$ $i = 1, 2, \ldots, k$-ra, ahol $k < j$, akkor $u_{ij} = 0$ is teljesül ugyanazon i-kre, hasonlóan abból, hogy $a_{ij} = 0$ $j = 1, 2, \ldots, k$ $k < j$-re következik, hogy $l_{ij} = 0$ lesz $j = 1, 2, \ldots, k$, $k < i$-re.

A 17. cikk lényege olyan optimális, vagy közel optimális rendezés végrehajtása a rendszer egyenletei közt még az elimináció megkezdése előtt, hogy a (2.2)-ben is minél több 0 legyen. A cikk 3 féle módszert is ajánl, és ezeket nevezi optimális rendezésnek.

Hasonló meggondolásokat tartalmaznak és az optimális elrendezésre adnak eredményeket a 3., 4., 5. és 16. dolgozatok is.

Az elimináció fent leírt sorrendje feltételezi, hogy az A mátrix $n \cdot n$-es tömbje egyidejűleg a számológép memóriájában rendelkezésünkre áll. Nagy n esetén azonban ez nem teljesíthető. Ismeretes az eliminációs lépések sorrendjének olyan megváltoztatása, amelyhez nem szükséges a teljes mátrixot egyidejűleg a számológép belső memóriájában tárolni (lásd pl. 17.).

A Gauss elimináció egyenletrendszerek megoldására a sparse sajátosságot kihasználva nagyon könnyen programozható. Azonban mint már említettük, a sparse mátrix inverze telítetté válhat, ezért a Gauss eliminációnak mátrix invertálásra való használatánál sem nyújt sok előnyt az invertálandó mátrix sparsitása.

Számpéldáinkban elemezzük a Gauss elimináció különböző módozatainak időigényeit egyenletrendszer megoldására és mátrix invertálására.

2.2 Blokkdiagonális mátrix inverze

Legyen az A mátrixunk blokkdiagonális, vagyis

$$A = \begin{bmatrix} B_1 & & & 0 \\ & B_2 & & \\ & & \ddots & \\ 0 & & & B_k \end{bmatrix}$$

alakú, ahol a B_i $k_i \cdot k_i$-s mátrix. Ha az összes B_i-nek létezik inverze, akkor

$$A^{-1} = \begin{bmatrix} B_1^{-1} & & & 0 \\ & B_2^{-1} & & \\ & & \ddots & \\ 0 & & & B_k^{-1} \end{bmatrix} .$$

Ennek segítségével az $n = \sum_i k_i$ rendszámú mátrix invertálása visszavezethető a blokkok invertálására. Célunk a hálózatban olyan jelölés bevezetése, hogy a hálózatra felírt invertálandó mátrix blokkdiagonális, vagy ahhoz minél közelebbi mátrix legyen.

Valamely összefüggő hálózat vágások (diakoptika) segítségével részgráfokra esik szét. Minden részgráf külön mátrixot határoz meg, amelyek az egész gráf mátrixának blokkjai lesznek. Ezáltal a mátrix rendszáma növekszik annyival, ahány vágásra volt szükség és a felvett csúcspontok sorában és oszlopában nem 0 elemek jönnek be.

Az ilyen mátrixok invertálásánál célszerű először a blokkokat invertálni, majd a blokkokon kívüli nem 0 elemek segítségével módosítani az inverzet a később ismertetett módszerek valamelyikével.

A diakoptika módszereinek részletes tárgyalása megtalálható a 21-es könyvben.

2.3 Módosított mátrix inverze

Tegyük fel, hogy ismerjük az $n \cdot n$-es A mátrix inverzét, módosítsuk A-t olyan módon, hogy a módosítás XBY^T alakban legyen felírható, ahol B invertálható, akkor kiszámolható a módosított mátrix inverze a következő képlettel:

$$(A + XBY^T)^{-1} = A^{-1} - A^{-1}X(B^{-1} + Y^TA^{-1}X)^{-1}Y^TA^{-1} \tag{2.5}$$

(2.5)-nek akkor van nagy jelentősége, ha B rendszáma kicsi, ugyanis (2.5) jobboldalán a zárójelben álló mátrix, amit invertálni kell, szintén kis rendszámú.

A (2.5) speciális alakja:

$$(A + \underline{u}\underline{v}^T)^{-1} = A^{-1} - \frac{1}{1 + \underline{v}^T A^{-1} \underline{u}} A^{-1}\underline{u}\underline{v}^T A^{-1} \tag{2.6}$$

(2.6) olyan mátrix inverzére ad képletet, ahol egy már meglévő inverzű mátrixot egy diáddal módosítottunk. Ennek további specialitásait részletezzük:

a) az ismert inverzű mátrix egy sorát, vagy
b) oszlopát módosítjuk, vagy
c) csupán egy elemét.

Az a) esetben u, a b) esetben v egyetlen komponense 1-es lesz, a többi 0, c) esetben u és v is ilyen tulajdonságú lesz, de az egyik 1-es helyén a megváltoztatott számérték áll.

Sparse esetben c) a különösen érdekes eset. Adjunk az A mátrix i-edik sorának j-edik eleméhez b-t. Legyen az $\underline{u}$ vektor i-edik komponense b, a többi 0, a $\underline{v}$ vektor j-edik komponense 1, a többi 0. Jelölje $\underline{u} = b\underline{e}_i$, $\underline{v} = \underline{e}_j$ (e_i és e_j egységvektorok, melyek i-edik illetve j-edik eleme 1, a többi 0.) Ekkor (2.6) szerint

$$(A + b\underline{e}_i \underline{e}_j^T)^{-1} = A^{-1} - \frac{b}{1 + bd_{ji}} \underline{d}_i \underline{d}_j^T \tag{2.7}$$

ahol d_{ji} az A^{-1} j-edik sorának i-edik eleme $\underline{d}_i$ az A^{-1} mátrix i-edik oszlopa $\underline{d}_j^T$ pedig az A^{-1} j-edik sorvektora. Vagyis (2.7) szerint, ha a mátrix egy elemét változtatjuk meg, akkor az inverz egy diáddal módosul.

Sparse esetben (2.7) szukcesszive alkalmazható, ha a mátrix nagyon kevés nem 0 elemet tartalmaz. Ha a sparse mátrixunk fődiagonálisában nem 0 elemek állnak, vehetjük induló

mátrixnak a fődiagonálisból álló mátrixot. Ennek inverzébe a fődiagonális elemeinek reciprokai kerülnek. Vegyünk ezután egy nem 0 fődiagonálison kívüli elemet és alkalmazzuk (2.7)-et. (2.7) nagyon egyszerűen hajtható végre, minthogy A^{-1} diagonálmátrix $d_{ji} = 0$ ($i \neq j$-re) $\underline{d}_i$ és $\underline{d}_j^T$ egyetlen nem 0 elemet tartalmazó vektorok. Ezután vegyünk egy újabb fődiagonálison kívüli nem 0 elemet stb. Az eljárást annyiszor kell ismételni, ahány nem 0 elem van. (2.7) számolása továbbra is nagyon egyszerű lesz, de minden egyes lépés újabb nem 0 elemeket hozhat be a képletbe, az inverz telítetté válhat. A számolás közben szingularitás lép fel, ha (2.7)-ben $1 + bd_{ji} = 0$ lesz. Ekkor viszont módunkban áll más nem 0 elemet választani, esetleg ideiglenesen egy nem 0 elemet felvenni, amit később visszaalakíthatunk 0-sá.

Sparse mátrixoknál az a) eset is érdekes lehet.
Legyen az A i-edik sorában a $j_1, j_2, \ldots, j_k$-adik elemek $b_1, b_2, \ldots, b_k$ nem 0-ok. Ekkor $\underline{u}^T = (\underbrace{00\ldots0}_{i-1}\ \underset{i}{1}\ 0\ldots0)$ alakú, míg v-ben a $j_1, j_2, \ldots, j_k$-adik elem

$b_1, b_2, \ldots, b_k$ a többi 0, azaz $\underline{v}^T = (00\ldots0\ \underset{j_1}{b_1}\ 0\ldots0\ \underset{j_2}{b_2}\ 0\ldots\underset{j_k}{b_k}0\ldots0)$.

A (2.6) jobboldalán lévő tört

$$\frac{1}{1 + \underline{v}^T A^{-1} \underline{u}} = \frac{1}{1 + b_1 d_{j_1 i} + b_2 d_{j_2 i} + \ldots + b_k d_{j_k i}}$$

lesz, ahol d_{ji} jelentése azonos a (2.7)-beli jelentéssel.

A (2.6)-ban szereplő $A^{-1} \underline{u} v^T A^{-1}$ pedig olyan diád lesz, melynek első tényezője $\underline{d}_i$, a második pedig a következő sorvektor

$$\left(\sum_{m=1}^{k} b_m d_{j_m 1}, \quad \sum_{m=1}^{k} b_m d_{j_m 2}, \ldots, \quad \sum_{m=1}^{k} b_m d_{j_m n} \right).$$

Az a) eset számolása segítségével az A mátrix egy–egy sorában álló főátlón kívüli nem 0 elemek bevonhatók a számolásba. Ezáltal a (2.6)-ban szereplő osztást és szingularitás vizsgálatot is kevesebbszer kell elvégezni, mint ha csak egyenkint vonjuk be a nem 0 elemet.

2.4 Invertálás rendszám növeléssel

Sparse esetben nagyon jól használható a 11.-ben ismertetett következő mátrixinvertálási módszer: Legyen ismert a $k \cdot k$-s A_k mátrix inverze A_k^{-1}.
A_k-t egy sor és egy oszlop hozzáadásával $k + 1$-ed rendűre bővítjük a következő módon:

$$A_{k+1} = \begin{bmatrix} A_k & \underline{u}_k \\ \underline{v}_k^T & a_{kk} \end{bmatrix}$$

aminek inverzét

$$A_n^{-1} = \begin{bmatrix} P_{k-1} & \underline{r}_k \\ \underline{q}_k^T & \beta_k \end{bmatrix}$$

alakban keressük. Az $A_{k+1} A_{k+1}^{-1} = E$ egyenlőségből a következő egyenletek adódnak:

$$\begin{aligned} &A_k P_k + \underline{u}_k \underline{q}_k^T = E \\ &\underline{v}_k^T P_k + a_{kk} \underline{q}_k^T = \underline{0} \\ &A_k \underline{r}_k + \beta_k \underline{u}_k = \underline{0} \\ &\underline{v}_k^T \underline{r}_k + \beta_k a_{kk} = 1 \end{aligned}$$

Ezekből pedig rendre nyerhetők a következők:

$$\begin{aligned} \beta_k &= \frac{1}{a_{kk} - \underline{v}_k^T A_k^{-1} \underline{u}_k} \\ \underline{q}_k^T &= -\beta \underline{v}_k^T A_k^{-1} \\ P_k &= A_k^{-1} - A_k^{-1} \underline{u}_k \underline{q}_k^T \\ \underline{r}_k &= -\beta A_k^{-1} \underline{u}_k \end{aligned} \qquad (2.8)$$

Kimutatható, hogy ezzel a módszerrel is n^3 nagyságrendű művelet kell az $n \cdot n$-es inverz számolásához.

A (2.8) képletek számolása közben lehet jól kihasználni a sparse tulajdonságot. Az $A_k^{-1} \underline{u}_k$ és $\underline{v}_k^T A_k^{-1}$ mátrixvektor illetve vektormátrix szorzásnál csak az $\underline{u}_k$ és $\underline{v}_k^T$ nem 0 elemeivel kell szorozni (ezek a kiindulási A_{k+1} mátrixnak $k+1$-edik oszlopvektora illetve sorvektora).

A módszerrel kis rendszámú mátrixból kiindulva (indulhatunk $k = 1$-től is) egyre nagyobb méretű márixot invertálhatunk a sparse tulajdonság jó kihasználása mellett.

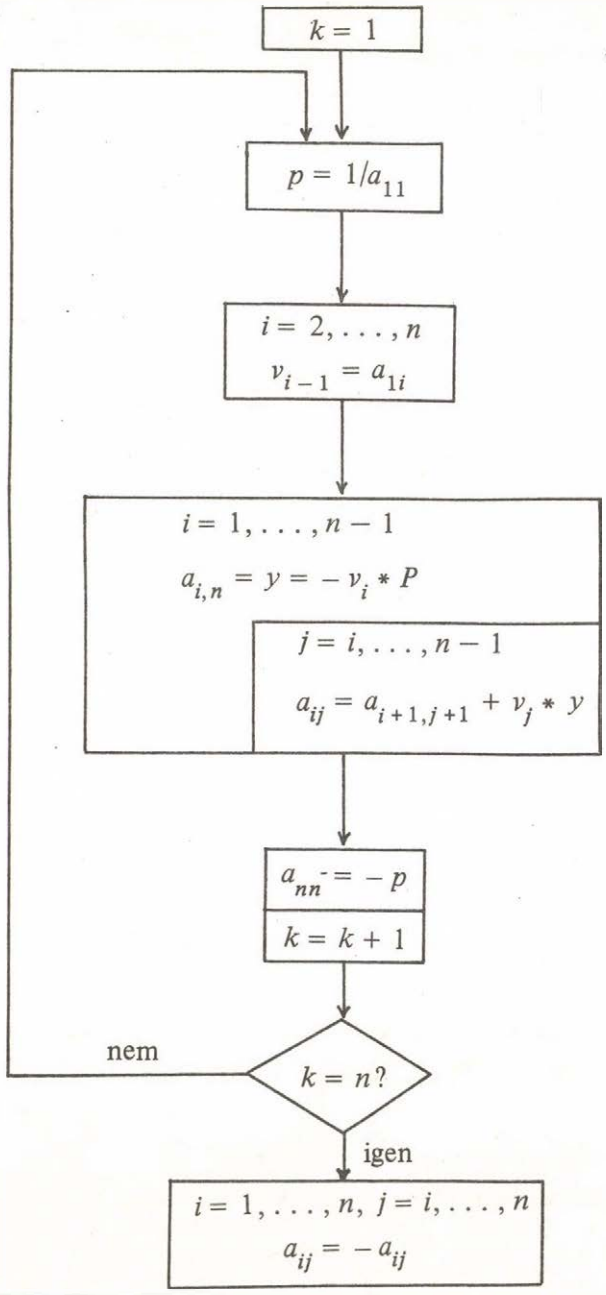
2.5 Módszerek szimmetrikus esetre

A hálózat számolásban előforduló és differenciálegyenletek numerikus megoldása közben előforduló invertálandó mátrixok többsége szimmetrikus. Ezért célszerű külön foglalkozni a szimmetrikus mátrixú egyenletrendszerek megoldási és szimmetrikus mátrixok invertálási módszereivel. Legismertebb a Cholesky módszer, amely egyenletrendszer megoldására és invertálásra egyaránt alkalmazható. A módszer leírása nagyon sok irodalmi forrásban megtalálható (lásd 11., 12., 36., 38.) ezért ezzel itt nem foglalkozunk. Megjegyezzük azonban, hogy a Cholesky módszer számolása közben nagyon jól ki lehet használni azt, hogy a mátrix sparse tulajdonságú. Számpéldánk elemzésénél megmutattuk a Cholesky

módszer művelet és időigényét is más módszerekkel összehasonlítva. Az összehasonlításból látszik, hogy aránylag kevés szorzást és osztást kell végezni, de az n ismeretlenes egyenletrendszer megoldása esetén szükség van n darab négyzetgyökvonásra, ami erősen megterheli a számolási időt.

Kevésbé ismeretes viszont a Gauss–Jordan eliminációs módszer alkalmazása szimmetrikus mátrixok invertálására, ami $n^3/2$ nagyságrendben és csak félmátrix tárolásával invertálja a szimmetrikus mátrixot. Most ismertetjük ennek a blokksémáját (lásd 43.).

Legyen az A mátrixunk szimmetrikus. Helyezzük el csak a fődiagonális és a feletti mátrix elemeket. A blokksémában kijelölt műveleteket végrehajtva az inverz mátrix képződik az eredeti mátrix helyén.



3. Iterációs módszerek

A lineáris egyenletrendszer iterációs módszerrel történő megoldásának alapgondolata a következő: kiindulunk az $\underline{x}$ ismeretlen vektor valamilyen $\underline{x}^{(0)}$ közelítéséből, ami tetszőleges lehet, de előbb célhoz jutunk, ha az a megoldás valamilyen jól becsült értéke. Ezután $k = 1, 2, \ldots,$-re végrehajtunk egy

$$\underline{x}^{(k+1)} = B\underline{x}^{(k)} + \underline{d} \tag{3.1}$$

alakú iterációs műveletsort. A konvergens eljárásoknál az $\underline{x}^0, \underline{x}^{(1)}, \underline{x}^2 \ldots$ vektorsorozat konvergál a megoldáshoz.

A (3.1)-ben szereplő B és $\underline{b}$ megválasztása, a képlet sok lehetséges módosítása, sok különböző eljárást szolgáltat. Az alábbiakban ezekből hármat tárgyalunk. Összehasonlítjuk a véges módszerekkel és rámutatunk használatuk előnyeire, hátrányaira.
(Az iterációs módszerekkel kapcsolatban lásd: 11., 12.,és 40.-et.)

3.1 A módszerek ismertetése

Tekintsük a

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad \text{vagy vektor felírásban az} \quad A\underline{x} = \underline{b} \tag{3.2}$$

lineáris egyenletrendszert. Bontsuk fel az A mátrixot az alábbi módon

$$A = D - E - F,$$

ahol D a főátlóban lévő elemek mátrixa, E az alatta, F a felette lévő háromszögmátrix. Az egyszerű iteráció, vagy Jacobi iteráció ezek segítségével a következő módon írható fel:

$$\underline{x}^{(m+1)} = D^{-1}(E + F)\underline{x}^{(m)} + D^{-1}\underline{b} \tag{3.3}$$

Koordinátás felírásban ez megfelel az

$$x_i^{(m+1)} = -\sum_{\substack{j=1 \\ j \neq 1}}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(m)} + \frac{b_i}{a_{ii}}, \quad 1 \leqslant i \leqslant n, \quad m \geqslant 0$$

iterációnak. A Gauss–Seidel iteráció mátrixos megfogalmazása:

$$\underline{x}^{(m+1)} = (D - E)^{-1}F\underline{x}^{m} + (D - E)^{-1}\underline{b} \tag{3.4}$$

koordinátás alakja pedig:

$$x_i^{(m+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(m+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(m)} + \frac{b_i}{a_{ii}}.$$

A Gauss–Seidel iterációnál gyorsabban konvergál az overrelaxációs módszer, amelynek képletei:

$$\underline{x}^{(m+1)} = (I - \omega L)^{-1}\{(1-\omega)I + \omega U\}\underline{x}^{(m)} + \omega(I - \omega L)^{-1}D^{-1}\underline{b}$$

ahol $L = D^{-1}E$, $U = D^{-1}F$, I pedig az egységmátrix, vagy koordinátásan:

$$x_i^{(m+1)} = x_i^m + \omega\left\{-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{ii}}x_j^{(m+1)} - \sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}x_j^{(m)} + b_i - x_i^{(m)}\right\},$$

ahol ω a reláxiós tényező. A módszer konvergenciagyorsasága ω megválasztásától erősen függhet. ω megválasztása külön vizsgálatot igényel. Az

$$\omega = \frac{2}{1 + \sqrt{1 - \lambda^2}}$$

képletet szokták alkalmazni, ahol λ a $D^{-1}(E + F)$ mátrix legnagyobb abszolút értékű sajátértéke. (Ha ez nem ismeretes, akkor annak valamilyen becslését is használhatjuk.)

A (3.1)-ben szereplő B mátrix és $\underline{d}$ vektor szerepét a Jacobi módszernél $D^{-1}(E + F)$ és $D^{-1}\underline{b}$, Gauss–Seidelnél $(D - E)^{-1}F$ és $(D - E)^{-1}\underline{b}$ míg az ovverelaxációnál $(I - \omega L)^{-1}\{(1-\omega)I + \omega U\}$ és $\omega(I - \omega L)^{-1}D^{-1}\underline{b}$ játsszák. A módszerek konvergenciájának elégséges feltételei, hogy a B-nek megfelelő mátrixok sajátértékei abszolút értékben 1-nél kisebbek legyenek. A konvergenciájuk gyorsaságát is a B spektrál sugara határozza meg.

3.2 A véges és iterációs módszerek összehasonlítása

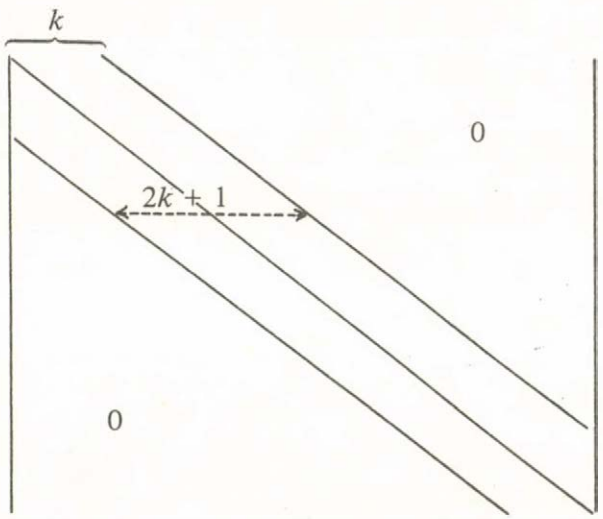
A (3.1) alakú iterációk mátrix–vektor szorzáson alapulnak. Ha a B mátrix sparse tulajdonságú, akkor ez a tény lényegesen lerövidítheti a számolási időt. Másrészt a mátrix elhelyezésénél is lényegesen kihasználhatjuk a sparse tulajdonságot. Ezáltal nagyobb méretű egyenletrendszerek kezelhetők számítógépek segítségével.

Az iterációs módszereknél a B mátrix, amellyel a szorzást végezzük, végig változatlan az iteráció során, így annak tárolásához akár a gép külső memóriája is alkalmas és ennek segítségével nagyobb rendszerek gépi számolására nyilik mód.

Az iterációs eljárást a kívánt pontosság elérése után leállíthatjuk, vagy megszakíthatjuk, majd innen tovább folytathatjuk. A véges módszereknél erre általában nincs lehetőség. Az iterációs módszerek konvergenciája nincs mindig biztosítva, vagy nagyon lassú. A véges módszereknél a pontosság és a számolási idő a számítógép által determinálva van.

4. Szalagmátrixú lineáris egyenletek

A sparse mátrixok speciális alakja a szalagmátrix, amelyben a nem 0 elemek a fődiagonális körül helyezkednek el. Legyenek a fődiagonálistól jobbra és balra, k-nál nagyobb távolságra lévő elemek mind 0-ok, akkor a szalagszélesség $2k + 1$.



A szalagmátrix által megvalósított lineáris transzformációnak speciális jelentés tulajdonítható. Tekintsük az $\underline{y} = A\underline{x}$ mátrix–vektor szorzással felírható lineáris transzformációt. $\underline{y}$ i-edik komponensét úgy kapjuk, hogy az A i-edik sorát komponáljuk az $\underline{x}$ vektorral, de annak csak az $i - k$, $i - k + 1, \ldots, i + k$, indexű komponense különbözhet 0-tól, ami úgy szemléltethető, hogy egyszerre csak ebben a $2k + 1$ dimenziós altérben végezzük a transzformációt. Ez a $2k + 1$ dimenzió persze sorról sorra eggyel eltolódik, úgy, hogy egy újabb dimenzió "irány" jön be a számolásba, közben pedig egy (a legrégibb) elmarad.

$k = 0$ esetben a szalagmátrix a diagonális mátrix lesz. Minél kisebb a k, annál közelebb kerülünk a diagonális mátrixhoz. A diagonalizálás kérdésére az 5. pontban még visszatérünk, itt csak megjegyezzük, hogy a szalagmátrix felfogható úgy is, mint a diagonális mátrixhoz való közeledés egy közbülső állapota. Az 5. pontban láthatjuk, hogy az ott ismertetett módszerek éppen ezt csinálják, ahol a szalagszélesség 3, ($k = 1$).

Szalagmátrixú egyenletrendszer megoldására négy módszert ismertetünk.

4.1 Gauss elimináció szalagmátrix esetén

Szalagmátrix esetén a Gauss elimináció egyszerűsített módon hajtható végre. A főátló alatti elemek eliminálására csak a főátló alatt álló k szám esetén van szükség, hiszen a többi eleve 0. Az eliminálás után kialakult háromszögmátrix a fődiagonális és a fölötte álló k számú, tehát összesen $k+1$ szélességű sávból áll.

Ha a Gauss eliminációt főelem kiválasztással végezzük, akkor szükség lehet sorcserékre, ami viszont a nem 0 elemeknek a szalagsávjából való kitolódásával járhat, mégpedig a sáv felfelé k szélességben kiterjedhet.

4.2 A Gauss elimináció mátrixos alakban

Olyan szalagmátrixot vizsgálunk, mely a következő formában írható fel:

$$A = \begin{bmatrix} B_1 & C_1 & & & \\ A_2 & B_2 & C_2 & & \\ & \cdots & \cdots & \cdots & \\ & & A_{r-1} & B_{r-1} & C_{r-1} \\ & & & A_r & B_r \end{bmatrix}$$

ahol A_i, B_i és C_i $k \cdot k$ méretű mátrixok.

Az $A\underline{x} = \underline{b}$ egyenletrendszer megoldása a következő képletekkel nyerhető ($\underline{b}$-t és x-et felbontjuk r darab k elemű $\underline{v}_i$ és $\underline{u}_i$ vektorra):

$$\begin{aligned}
&\underline{f}_1 = B_1^{-1}\underline{v}_1 \\
&G_1 = -B_1^{-1}C_1 \\
&\underline{f}_j = (A_jG_{j-1} + B_j)^{-1}(\underline{v}_j - A_j\underline{f}_{j-1}) \qquad 2 \leqslant j \leqslant r \\
&G_j = -(A_jG_{j-1} + B_j)^{-1}C_j \\
&\underline{u}_r = \underline{f}_r \\
&\underline{u}_j = \underline{f}_j + G_j\underline{u}_{j+1}, \qquad 1 \leqslant j \leqslant r-1.
\end{aligned}$$

(r darab $\underline{u}_j$ vektor adja az $\underline{x}$ megoldást).

4.3 A progonka módszer

Tekintsük az

$$\begin{aligned}
&a_ix_{i-1} + c_ix_i + b_ix_{i+1} = -d_i \qquad i = 1, 2, \ldots, n-1 \qquad (4.1) \\
&x_0 = h_1x_1 + g_1, \quad x_n = h_2x_{n-1} + g_2
\end{aligned}$$

egyenletrendszert, aminek a mátrixa olyan szalagmátrix, amelyben csak a fődiagonális közvetlen felső és alsó átlós irányú szomszédjában vannak nem 0 elemek.

A megoldás explicite felírható az

$$x_i = \alpha_{i+1} x_{i+1} + \beta_{i+1} \qquad i = n-1, n-2, \ldots, 1$$

alakban, ahol előre kiszámolható

$$\alpha_{i+1} = \frac{b_i}{c_i - \alpha_i a_i}, \quad \beta_{i+1} = \frac{a_i \beta_i + a_i}{c_i - \alpha_i a_i}, \qquad \alpha_1 = h_1, \quad \beta_1 = g_1.$$

Általánosabb speciális tipusú szalagmátrix esetén hasonló explicit képletek vezethetők le. (4.1) alkalmazható akkor is, ha (4.1)-ben szereplő a_i, b_i és c_i helyére négyzetes mátrixokat irunk és ugyanakkor x_i, d_i vektorok lesznek. A progonka módszer ebben az esetben azonos lesz a 4.2 módszerrel.

4.4 Szalagmátrix invertálása rendszámnöveléssel

A 2.4-ben ismertetett rendszámnöveléses módszer nagyon jól alkalmazható szalagmátrix esetén. $2s+1$ szélességű szalagmátrixok esetén a 2.4 képleteiben szereplő $\underline{u}_k$ és $\underline{v}_k^T$ vektoroknak csak az utolsó s eleme különbözhet 0-tól. Ezért a (2.8) képletek számolásában szereplő $A_k^{-1}\,\underline{u}_k$ és $\underline{v}_k^T A_k^{-1}$ mátrix–vektor illetve vektor–mátrix szorzások műveleti igénye k^2 helyett csak $k \cdot s$. Ha az $n \cdot n$-es $2s+1$ szalagszélességű mátrix invertálását 2.4 módszerével $k = 1$-től kezdve iteráltan alkalmazva végezzük el a fenti megjegyzéseink figyelembevételével, akkor a (2.8) számolásához a következő műveleti igény szükséges:

Az $A_k^{-1}\,\underline{u}_k$ és $\underline{v}_k^T A_k^{-1}$ szorzásokhoz külön–külön

$\sum_{k=1}^{n-1} k \cdot s = s\frac{(n-1)\cdot n}{2}$, összesen $s \cdot n \cdot (n-1)$ szorzás kell;

β_k-k számolásához $\sum_{k=1}^{s-1} k + (n-s)s = s(n - s + \frac{s-1}{2})$ szorzás és $n-1$ osztás;

P_k-k számolásához $\sum_{k=1}^{n-1} k^2 = \frac{(n-1)n(2n-1)}{6}$ szorzás;

míg a q_k és r_k^T vektorok számolásához (együtt) $2 \cdot \sum_{k=1}^{n-1} k = n(n-1)$. Összesen tehát

$$s \cdot n(n-1) + s(n - s + \frac{s-1}{2}) + n(n-1) + \frac{(n-1)n(2n-1)}{6} =$$

$$= n(n-1)\left[\frac{2n-1}{6} + s + 1\right] + s(n - s + \frac{s-1}{2})$$ szorzás illetve $n-1$ osztás kell. Ha s kicsi n-hez képest, a műveleti igény körülbelül $n^3/3$ nagyságrendű lesz.

A módszer nagyon jól alkalmazható abban az esetben, amikor szalagmátrixú lineáris egyenletrendszert kell megoldanunk. Ebben az esetben a közbülső inverz mátrixok utolsó s

sora és oszlopa szükséges, ami a memóriában 2 darab $n \cdot s$ nagyságú tömb tárolását igényli. A számolási igény nagyságrendje $n^2 \cdot \frac{s}{2}$ szorzás és $n - 1$ osztás. A módszer jól alkalmazható elliptikus parciális differenciálegyenletek peremértékfeladatainak megoldására.

5. A sparse mátrix sajátértékproblémája

A mátrixszámításban és alkalmazásaiban igen nagy jelentősége van annak a feladatnak, hogy egy mátrixot diagonális vagy diagonálishoz közeli úgynevezett Jordan alakra hozunk olymódon, hogy közben a mátrix sajátértékei és sajátvektorai ne változzanak. A mátrixok diagonális vagy Jordan alakra való ilyen transzformálása ekvivalens a mátrix sajátértékeinek és sajátvektorainak meghatározásával.

Itt nem részletezzük a mátrix sajátértékszámitás problémáit, ami külön hosszadalmas fejtegetést igényel, mégcsak a definiciókat és alapösszefüggéseket sem mondjuk el. Mindez nagyon jól megtalálható a 11., 38. és 39. könyvekben. A 41. cikk egy általa c–g-nek nevezett algoritmust ajánl sparse mátrixokra, amit összehasonlit más algoritmusokkal és kimutatja előnyeit. Itt csupán néhány módszert tárgyalunk, amelyek sparse mátrixok esetén különösen hasznosak lehetnek.

A javasolt módszerek az A $n \cdot n$-es mátrixot kontinuáns mátrix-szá transzformálják, majd a kontinuáns mátrix sajátértékét számolják. Szimmetrikus A esetén a Givens, nemszimmetrikus esetben a Lánczos módszert javasoljuk. Számpéldánkon megmutatjuk ezek időigényességét is.

5.1 A Givens módszer

A szimmetrikus A mátrixot a $B = S^T A S$ ortogonális transzformációval hozzuk kontinuáns alakra, ahol a

$$B_0 = A, \qquad B_k = S_k^T B_{k-1} S_k \tag{5.1}$$

rekurzív transzformáció utolsó $B_k = B_{k_1} = B$ mátrixa lesz kontinuáns. Az összes B_k és így B sajátértékei is (az (5.1) transzformáció miatt) azonosak az A sajátértékeivel. Az előállításból látható, hogy $S = S_1 \; S_2 \; \ldots \; S_{k_1}$. Az (5.1) minden egyes végrehajtása egy $\neq 0$ elemet tüntet el a mátrixból. Igy ezt annyiszor kell végrehajtani, ahány $\neq 0$ elem van a kontinuáns (azaz a főátló és vele párhuzamos szomszédos átló) tartományon kívül. Ez szabja meg a k_1 indexet is.

A transzformáló mátrix felépítése: legyen a B_{k-1} r-edik sorának q-adik eleme $b_{rq}^{(k-1)} \neq 0$, $|r-q| > 1$, $p = r+1$, és legyenek az S mátrix elemei:

$$\left.\begin{aligned} s_{ii} &= \begin{cases} 1, & \text{ha} \quad i \neq p \text{ és } i \neq q \\ a \cdot b_{r,r+1}^{(k-1)} & \text{ha} \quad i = p \text{ és } i = q \end{cases} \\ s_{ij} &= -s_{ji} = \begin{cases} a \cdot b_{rq}^{(k-1)} & \text{ha} \quad i = p \quad j = q \\ 0 & \text{különben,} \end{cases} \end{aligned}\right\} \tag{5.2}$$

ahol $a = 1/\sqrt{(b_{rp}^{(k-1)})^2 + (b_{rq}^{k-1})^2}$.

Az (5.2)-vel meghatározott S egy ortogonális transzformáció mátrixa. A forgatás szöge $\arcsin\{b_{rq}^{(k-1)} / \sqrt{(b_{rp}^{k-1})^2 + (b_{rq}^{(k-1)})^2}\}$

5.2 A Lánczos módszer

A nemszimmetrikus $n \cdot n$-es mátrixot, a $D = S^{-1}AS$ hasonlósági transtformációval hozzuk kontinuáns alakra. S-et egy $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_n$ és egy $\underline{y}_1, \underline{y}_2, \ldots, \underline{y}_n$ vektorsorozat segítségével állítjuk elő.

A $c_0 = 0$, $\underline{x}_0 = \underline{y}_0 = 0$, $\underline{x}_1$ és $\underline{y}_1$ tetszőleges (de $\underline{y}^T\underline{x} \neq 0$) értékekből kiindulva a következő rekurziós képletekből kaphatjuk a vektorsorozatokat:

$$\left.\begin{aligned} b_k &= \underline{y}_k^T A \underline{x}_k / \underline{y}_k^T \underline{x}_k \\ \underline{x}_{k+1} &= A\underline{x}_k - b_k \underline{x}_k - c_{k-1}\underline{x}_{k-1} \\ \underline{y}_{k+1} &= A^T \underline{y}_k - b_k \underline{y}_k - c_{k-1}\underline{y}_{k-1} \\ c_k &= \underline{y}_{k+1}^T \underline{x}_{k+1} / \underline{y}_k^T \underline{x}_k \end{aligned}\right\} \quad k = 1, 2, \ldots, n-1 \tag{5.3}$$

A D mátrixot b_n és c_n-ekből állíthatjuk elő a következőképpen:

$$D = \begin{bmatrix} b_1 & c_1 & & & & & \\ 1 & b_2 & c_2 & & & & \\ & 1 & b_3 & . & 0 & & \\ & & 1 & . & . & & \\ & & & 1 & . & . & \\ & & & & 1 & b_{n-1} & c_{n-1} \\ & & & & & 1 & b_n \end{bmatrix}$$

A mátrix ritka sajátosságát az (5.3) képlet számolása közben használhatjuk ki. Ugyanis a (5.3) képlet számolásának munkaigényes része az $A\underline{x}$ mátrix vektor szorzás, ami ritka A esetén egyszerűsíthető.

5.3 Kontinuáns mátrix sajátértékei

Kontinuáns mátrix sajátértékeinek számolására több, aránylag könnyen kezelhető módszer adható meg. Ilyen mátrixok karakterisztikus polinomjai növekvő fokszámú karakterisztikus polinomokból rekurzív formulával nyerhetők.

Most ismertetjük az LR módszert, ami általános mátrixok esetén is alkalmazható, amennyiben az alábbi felbontás végrehajtható, a végrehajthatóság szükséges és elegendő feltétele, hogy az A mátrix bal felső minorai ne legyenek szinguláris. A módszer kontinuáns mátrixra nagyon jól alkalmazható. Az $A = A_1 = L_1 U_1$ felbontás után legyen $A_2 = U_1 L_1$, majd rendre az $A_k = L_k U_k$ felbontásból $A_{k+1} = U_k L_k$.
Az összes A_k sajátértékei ugyanazok, minthogy

$$A_{k+1} = U_k L_k = L_k^{-1} A_k L_k = U_k A_k U_k^{-1} .$$

ahol L_k alsó, U_k felső háromszög mátrixok.
$k \to \infty$ esetén az $A_k \to A^*$ alulról trianguláris mátrixhoz, amelynek a főátlójában éppen a sajátértékek vannak.

Az LR transzformáció kontinuans mátrixok esetén azért egyszerű, mert az összes A_k, U_k és L_k is kontinuáns lesz, igy azok tárolása kevés helyet igényel és számolásuk is egyszerű.

A szimmetrikus mátrixból Givens módszerrel kapott kontinuáns mátrix is szimmetrikus, míg a Lánczos módszerrel kapott kontinuáns mátrix a fent látott (D) speciális alakú, így az LR transzformáció mindkettőre speciális módon hajtható végre. Ezt néhány numerikus könyv részletezi is (lásd pl. 39.).

6. Példák

1. Példa

A szimmetrikus sparse mátrix sajátértékeinek megkeresésére az 5.1 pontban javasoltuk a Givens, majd ennek folytatásaképpen az 5.3 pontban az LR módszert. Az alábbiakban egy példán szemléltetjük ezen módszerek használatát.
Legyen

$$A = \begin{bmatrix} 1 & 0 & 0{,}1 \cdot j \\ 0 & 2 & 0 \\ 0{,}1 \cdot j & 0 & 3 \end{bmatrix}$$

Az egyetlen főátlón kívüli nem 0 elem $a_{13} = 0{,}1 \cdot j$ (illetve a szimmetrikus megfelelője).
Az 5.1 jelöléseit használva $r = 1,\ q = 3,\ p = 2$

$$a = \frac{1}{\sqrt{a_{12}^2 + a_{13}^2}} = \frac{1}{\sqrt{0{,}01 \cdot j^2}} = -10 \cdot j$$

$$S_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

A mátrix szorzásokat végrehajtva:

$$B_1 = S_1^T A S_1 = \begin{bmatrix} 1 & -0{,}1 \cdot j & 0 \\ -0{,}1 \cdot j & 3 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

ami kontinuáns alakú mátrix.
A B_1 kontinuáns mátrixra alkalmazva az LR sajátérték kereső módszert 18 iterációs lépés után kapjuk a sajátértéket 0.0001 pontossággal, amelyek:

$$\lambda_1 = 2 + \frac{\sqrt{3.96}}{2} = 2.9950$$

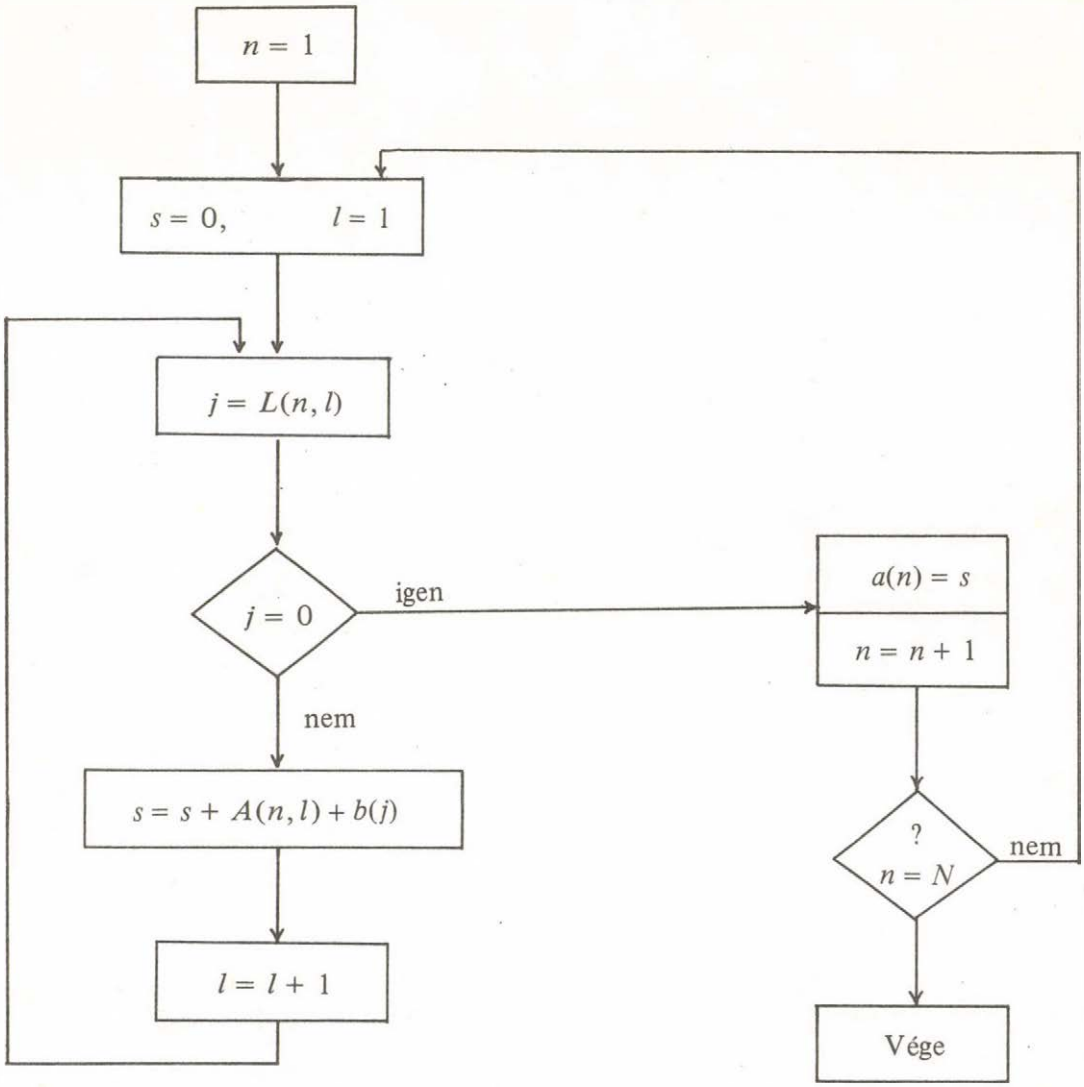
$$\lambda_2 = 2 - \frac{\sqrt{3.96}}{2} = 1.0050$$

$$\lambda_3 = 2.0000$$

2. Példa

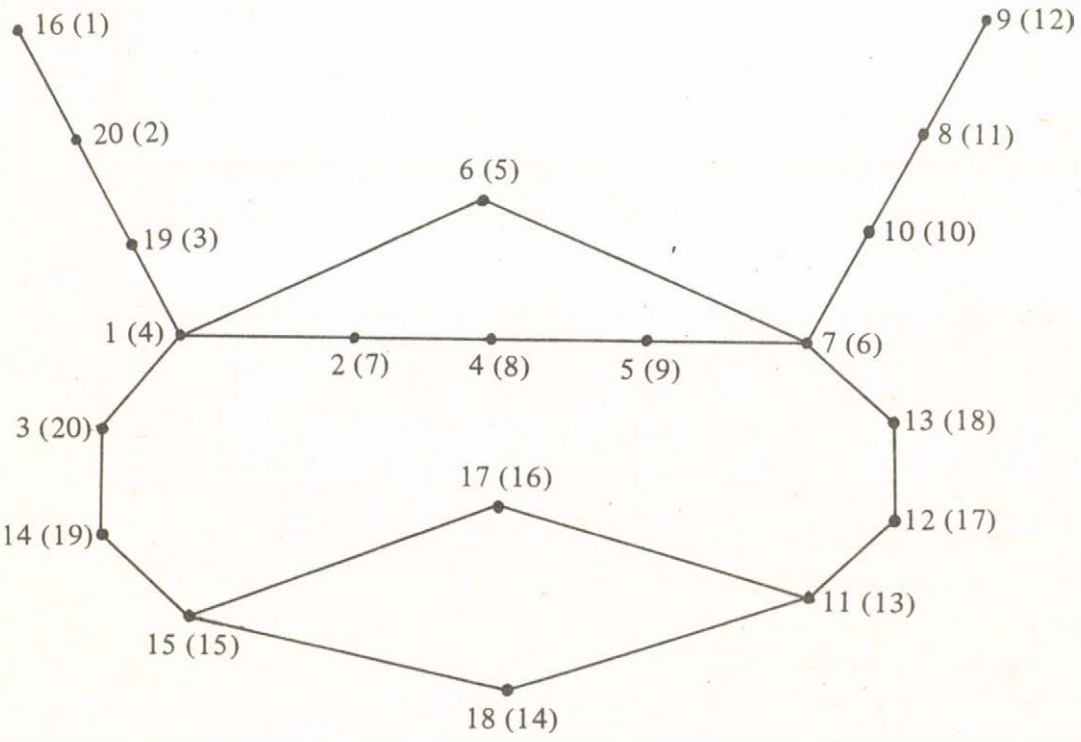
Az egyenletrendszerek iterációs módszerrel történő megoldása lényegében mátrix vektor szorzások sokszor egymás utáni alkalmazásából áll. Egy $N \cdot N$-es mátrixnak egy N elemű vektorral való szorzása N^2 szorzást igényel. Bemutatjuk sparse mátrix esetén a mátrix vektor szorzás számológépen való elvégzésének egy lehetséges változatát.

Legyenek az $N \cdot N$-es mátrix i-edik sora nem 0 elemeinek oszlopindexei $j_1, j_2, \ldots, j_k$ és $\max k = K$. Ekkor az A mátrix elhelyezhető egy $A(N, K)$ kétdimenziós tömbben (N sor, K oszlop). Az $\underline{a}$ és $\underline{b}$ vektorok az egydimenziós N elemű tömbben tárolódnak. Legyen L kétdimenziós tömb a nem 0 elemek sor illetve oszlopindexeinek tömbje, azaz $L(i, j) = l_{ij}$ jelentse az A mátrix i-edik sorában álló j-edik nem 0 elem oszlop indexét. L i-edik sorában annyi index szám szerepel, mint abban a sorban álló nem 0 elemek száma. Az L soraiban ezek után írjunk 0-okat. Legalább 1 db 0 legyen az L minden sorának a végén. Ekkor az $\underline{a} = A\underline{b}$ mátrix–vektor szorzás a következő algoritmussal végezhető el:


n = 1
s = 0,
l = 1
j = L(n, l)
j = 0
igen
a(n) = s
n = n + 1
nem
s = s + A(n, l) + b(j)
?
n = N
nem
l = l + 1
Vége

3. Példa

Most adunk egy példát, amelyen megmutatjuk az ismertetett módszerek alkalmazási lehetőségeit és előnyeit más módszerekkel szemben. A példánkban szereplő hálózatot egy gráf segítségével adjuk meg, amelynek 20 csúcspontja van. A gráf a következő:



Referenciapontnak a 20-as pontot választottuk. A csomoponti admittancia mátrix Y 19.19-es komplex elemekből álló sparse mátrix. Példánk elemzésénél több módszerrel megoldjuk az

$$Y\underline{x} = \underline{b}$$

egyenletrendszert és a megoldásra fordított időket összehasonlítjuk. Foglalkozunk az Y invertálásának és diagonalizálásának kérdéseivel.

Az egyenletrendszer megoldására Gauss elimináció segítségével a sparsitás kihasználása nélkül

$$\frac{n^3 + 3n^2 - n}{3} = \frac{19^3 + 3.19^2 - 19}{3} \sim 2800$$

komplex szorzás és osztás nagyságrendű művelet szükséges. A 2.1-ben ismertetett, sparse mátrixok esetére módosított Gauss elimináció segítségével a művelet igény 405-re csökkenthető.

A feladatban a gráf csúcspontjainak a beszámozása önkényesen történt, és a sparse módszerek szempontjából nem volt a legszerencsésebb. Hajtsuk végre a következő átszámozást (gráfon zárójelben megjelölve)

régi	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	4	7	20	8	9	5	6	11	12	10	13	17	18	19	15	1	16	14	3	2

Az új beszámozáshoz felírható 19.19-es Y admittancia mátrix a sparse módszerek segítségével könnyebben kezelhető. A 2.1 alkalmazásához ebben az esetben mindössze 272 szorzás és osztás kell.

A műveleti igények összehasonlítása jól szemlélteti a sparse módszerek hatékonyságát. Ugyanakkor az is kiderül, hogy alkalmazásuknál nagyon lényeges a gráf csúcsainak beszámozása. Ezzel kapcsolatban érdekes feladatként merül fel az optimális beszámozás megkeresésének feladata, ami matematikailag diszkrét programozási feladatot jelent.

A példát számszerűen is végig számoltuk az MTA CDC 3300-as számítógépen. A programok FORTRAN nyelven készültek. A következő programok futási idejét hasonlítottuk össze a vizsgált $n = 19$ ismeretlenes komplex egyenletrendszerre illetve komplex mátrixinverzióra (az eredeti beszámozást használva).

P1: egyenletrendszer megoldása Gauss eliminációval
(sparsitás kihasználása nélkül)

P2: egyenletrendszer megoldása Gauss eliminációval kihasználva a sparsitás adta lehetőségeket a 2.1 alapján
(a gráf eredeti beszámozását használva)

P3: Cholesky módszer a sparsitás kihasználásával
(ugyanazt a beszámozást használva)

P4: mátrixinverzió Gauss–Jordan módszerrel
(sparsitás kihasználása nélkül)

P5: mátrixinverzió Gauss–Jordan módszerrel, kihasználva a szimmetricitást a 2.5 alapján
(sparsitás nincs kihasználva)

P6: mátrixinverzió a 2.4-ben leírt rendszámnöveléssel a sparsitás és a szimmetricitás felhasználásával.

	P1	P2	P3	P4	P5	P6
szorzások + osztások száma kb.	2800	405	332 + n db gyökvonás	6900	3500	1400
számolási idő	2.9 sec	0.5 sec	1.2 sec	7.4 sec	3.8 sec	2.2 sec

A számpéldán megvizsgáltuk a mátrixinvertálásra alkalmazható sparse sajátosság felhasználása adta lehetőségeket. A mátrixinverzióra közvetlen felhasználva a 2.1 és 2.3-ban leírtakat,

a számolási idő alig rövidült meg. Ezek használata ritkább mátrixok esetében hatásos lehet. A 2.4 módszer használata sparse esetben a táblázatból látható. A következő példában vizsgáljuk a számpéldánk más módszerrel történő invertálását is.

4. Példa

A 3. példában adott Y mátrix invertálása.
Az invertálás előtt a hálózatban egy vágást hajtottunk végre, mégpedig az 1–3 és 7–13 szakaszon. Ezáltal hálózatunk két összefüggő gráfra esett szét, az egyik 8, a másik 12 csúcsponttal. Mindkettőhöz hozzájön még a két vágási csúcspont. Legyen ezek számozása az 1–3 szakaszon 21, a 7–13 szakaszon 22. Vágás után az admittancia mátrixunk a következő struktúrájú lesz:

$$Y_3 = \begin{bmatrix} Y_1 & & \\ & Y_2 & \\ & & \end{bmatrix}$$

ahol Y_1 az 1-től 12-ig, Y_2 13-tól 20-ig terjedő (átszámozott, azaz a zárójelben jelzett számolást használva) csúcspontú hálózat admittancia mátrixa. Y_3 utolsó két sora illetve oszlopa a vágási csúcspontokhoz tartozó nem 0 admittanciákat tartalmazza a 4–21, 20–21, 6–22 és 18–22 szakaszon.

Y_3 invertálását célszerű úgy elvégezni, hogy először Y_1 és Y_2-t külön invertáljuk.

Y_1^{-1} és Y_2^{-1}-et az Y_1 és Y_2 sorrendjében összerakva kapjuk meg az Y_4^{-1}-et. Y_4 abban különbözik Y_3-tól, hogy hiányzik Y_3 utolsó két sora, illetve oszlopa.
Y_4^{-1}-ből Y_3^{-1}-et megkaphatjuk akár a 2.3 pontban, akár a 2.4 pontban leírt módszerek alkalmazásával.

5. Példa

A 3. példában adott Y mátrix sajátértékeit meghatároztuk az 5.-ben adott Givens – LR módszer segítségével. Az Y kontinuáns alakra hozására Givens módszer segítségével a számolási idő 1.6 sec. Ezután az LR módszert alkalmazva 90 iterációs lépésben 4 tizedes pontossággal kaptuk a sajátértékeket. 6 tizedes pontossághoz 113 iterációs lépés kellett. Ennek számolási ideje 15.2 sec.

7. Programok

Lineáris egyenletrendszerek megoldására, mátrixinvertálásra, mátrix sajátértékeinek számolására általában minden számítóközpontban sok számítógépes program áll rendelkezésünkre. Ezért ezek programjaival itt nem foglalkozunk. Itt néhány olyan programnak a használatát ismertetjük, amelyek különösen jók sparse mátrixú lineáris egyenletrendszerek megoldására, ilyen mátrixok invertálására illetve sajátértékeik számolására. Ezek a következők (a 3. példában előfordult elnevezéseket használva)

P2 program: egyenletrendszer megoldása Gauss eliminációval, kihasználva a 2.1 alapján a mátrix sparse tulajdonságát;
P3 program: Cholesky módszer a sparsitás kihasználásával;
P6 program: mátrixinverzió a 2.4-ben leírt rendszámnöveléssel a sparsitás és a szimmetria kihasználásával;
P7 program: azonos a P6-tal de szimmetricitás nincs kihasználva;
P8 program: Givens–LR módszer sparse mátrix sajátértékeinek számolására;

Mindegyik programot úgy készítettük, hogy csak a számolásban használtuk ki a mátrix sparse tulajdonságát, a mátrix tárolásánál nem, mivel, hogy a próbaszámolásoknál a teljes mátrix is könnyen elfér a gép gyorsmemóriájában. A módszerek nagyméretű mátrixra való használatához a programokat módosítani kell.
A P2, P3, P6 és P7 programok egy úgynevezett "karakterisztikus" mátrixot használnak, amelyek más–más módon, de lényegében a mátrix struktúráját írják le. Ennek egy lehetséges módját részletezzük, amit a P2 program használ.
A 2.1 pontban elemeztük, hogy a Gauss elimináció használatának sparse mátrixú egyenletrendszer esetén az az előnye, hogy az elimináció során is megmarad bizonyos sparse tulajdonság és még a számolás megkezdése előtt ki lehet jelölni azokat a helyeket, ahol a nem 0 elemek fellépheznek. Ezt a kijelölést a következőképpen készíthetjük el. Húzzunk vonalat a kiindulási A mátrixunkba oszlopirányba haladva felülről lefelé a főátlóig, az oszlopban fellépő első nem 0 elemtől kezdve, majd soriránýban haladva balról jobbra a főátlóig a sorban fellépő első nem 0 elemtől kezdve. A mátrix behúzott elemeinek indexeit (a nem 0 elemek csak itt léphetnek fel) kétdimenziós integer tömbben (M-ben) adjuk meg.

Legyen $M(i, 1) = s$, ha az i-edik oszlop a főátló alatt $s - 1$ nem 0 elemet (áthúzott helyet) tartalmaz.
$M(i, j + 1) = p$, $1 \leqslant j \leqslant s - 1$ az i-edik oszlop j-edik főátló alatti nem 0 elem (áthúzott hely) sorindexe.
$M(i, s + j) = q$, az i-edik sor főátlótól jobbra eső j-edik nem 0 elemének (áthúzott helyének) oszlopindexe. Legyen M minden sorának a végén legalább egy 0.

A Gauss elimináció során csak az M mátrix-szal kijelölt indexű mátrixelemekkel kell számolnunk. A számolást végző P2 szubrutin használatához el kell készítenünk az M mátrixot a fent

leírt módon.

A szubrutin behívása:

CALL SPG (N, A, B, X, M, L)

ahol A N · N-es mátrix (együtthatók mátrixa) sorfolytonosan, B N elemű jobboldal tömbje, X N elemű megoldás tömbje, M a fentiekben leírt N · L méretű integer tömb. Az A, B és X deklarálható és használható COMPLEX vagy REAL-nak is.

A Cholesky módszer sparse mátrixú lineáris egyenletrendszerek megoldására való használatához is el kell készíteni a karakterisztikus mátrixot, ami a fentihez hasonló módon karakterizálja a mátrix nem 0 elemeinek elhelyezkedését. Hasonlóan járunk el a P6 és P7 programoknál is.

A sparse mátrixok gépi kezelésére alkalmas programokból egy programgyűjteményt állítottunk össze (ami a fent említett és még néhány programot tartalmaz), és a CDC 3300-as felhasználói rendelkezésére bocsáthatjuk. A programgyűjtemény leírása, a programok használata, kezelése, hozzáférhetősége egy későbbi ismertetőben fog megjelenni.

Irodalom

[1] Ralph A. Willaughby: Proceedings of the Symposium on Sparse Matrices and Their Applications, Held at the IBM Watson Research Center September 9–10 1968.

[2] J. K. Reid: Large sparse sets of linear equations Proceedings of the Oxford conference; April 1970. Academic press, London 1971.

[3] R. P. Tewarson: On the Gaussian Elimination Method for Inverting Sparse Matrices, Computing 9, 1–7 (1972).

[4] Y. T. Chen and R. P. Tewarson: On the Optimal Choise of Pivots for the Gaussian Elimination, Computing 9, 245–250 (1972).

[5] Tewarson, R. P., "On the product form of inverse of sparse matrices," SIAM Rev. 8 (1966) 336–42.

[6] Tewarson, R. P., "On the product form of inverses of sparse matrices and graph theory," SIAM Rev. 9 (1967) 91–9.

[7] Householder, A. S., "A survey of some closed methods for inverting matrices," SIAM J. Appl. Math. 5 (1957) 155–69.

[8] Orden, A., "Matrix inversion and related topics by direct methods," Mathematical Methods for Digital Computers (Eds. A. Ralston, and H. S. Wilf) Vol. I, John Wiley and Sons, Inc., New York (1960) 39–49.

[9] Wilkinson, J. H., "Error analysis of direct methods of matrix inversion," J ACM 8 (1961) 281–330.

[10] Dulmage, A. L., and Mendelsohn, N. S., "On the inversion of sparse matrices," Math. Comp. 16 (1962) 494–6.

[11] Faddeev, D. K., and Faddeeva, V. N., "Vücsiszlityelnüe metodü linejnoj algebrü, Leningrád 1963.

[12] Fox, L., Introduction to Linear Algebra, Oxford Univ. Press, New York (1964).

[13] Householder, A. S., The Theory of Matrices in Numerical Analysis, Blaisdell (1964).

[14] Kahan, W., "Numerical linear algebra," Canadian Math. Bull. 9 (1966) 757–801.

[15] Wilkinson, J. H., "The solution of ill–conditioned linear equations," Mathematical Methods for Digital Computers (Eds. A. Ralston and A. S. Wilf), Vol. II. John Wiley and Sons, Inc., New York (1967) 65–93.

[16] Tewarson, R. P., "Solution of a system of simultaneous linear equations with a sparse coefficient matrix by elimination methods," BIT 7 (1967) 226–39.

[17] Tinney, W. F., and Walker, J. W., "Direct solutions of sparse network equations by optimally ordered triangular factorization," Proc. IEEE 55 (1967) 1801–9.

[18] Nathan, A., and Even, R. K., "The inversion of sparse matrices by a strategy derived from their graphs," Comput. J. 10 (1967/68) 190–4.

[19] Westlake, J. R., A Handbook of Numerical Matrix Inversion and Solution of Linear Systems, John Wiley and Sons, Inc., New York (1968).

[20] Tewarson, R. P., "Solution of linear equations with coefficient matrix in band form," BIT 8 (1968) 53–8.

[21] Kron, G., Diakoptics, Macdonald, London (1963).

[22] Spillers, W. R., "Network analogy for linear structures," J. Eng. Mech. Div., Proc. ASCE, Vol. 89, No. EM4 (August 1963) 21–9.

[23] Kuo, F. F., "Network analysis by digital computer," Proc. IEEE 54 (1966) 820–9.

[24] Kuo, F. F., and Kaiser, J. F., Eds., System Analysis by Digital Computer, John Wiley and Sons, Inc., New York (1966).

[25] Branin, F. H., Jr., "Computer methods of network analysis," Proc. IEEE 55 (1967) 1787–1801.

[26] Tinney, W. F., and Hart, C. E., "Power flow solution by Newton's method," IEEE Trans. PAS–86 (1967) 1449–60.

[27] Zienkiewicz, O. C., The Finite Element Method in Structural and Continuum Mechanics, McGraw–Hill, New York (1967)

[28] Przemieniecki, J. S., Theory of Matrix Structural Analysis, McGraw–Hill, New York (1968).

[29] Branin, F. H., Jr., "Kron's method of tearing and its applications," Proc. Second Midwest Symp. on Circuit Theory, Michigan State University, East Lansing, Michigan (December 1956).

[30] Branin, F. H., Jr., "The relation between Kron's method and the classical methods of network analysis," IRE WESCON Convention Record, Part 2 (1959) 1–29.

[31] Swift, G., "A comment on matrix inversion by partition," SIAM Rev. 2 (1960) 132–3.

[32] Dupont, T., "A factorization procedure for the solution of elliptic difference equations," SIAM J. Number. Anal. 5 (1968), to appear.

[33] Babuska, I., "Numerical stability in mathematical analysis," presented at the International Conference on Information processing, Edinburgh, August 1968.

[34] G. Kron, Diakoptics–Piecewise solution of large–scale systems. Electrical Journal, June 1967.

[35] A. Jenninga, A sparse matrix scheme for the computer analysis of structures. Int. J. of Comp. Math. 2 (1968), 1–21.

[36] Gergely J. Lineáris egyenletrendszerek megoldása elektronikus számológépeken. Egyetemi segédkönyv BME 1970.

[37] Gergely J. Szalagmátrixú lineáris egyenletrendszer megoldása. MTA Számítástechnikai Központ, Közlemények 6. 1971.

[38] Berezin–Zsidkov, Methodü vücsiszlenyij I. és II. Moszkva, 1962.

[39] A. Ralston: Bevezetés a numerikus analysisbe Műszaki Kiadó, Budapest, 1969.

[40] Varga, R. S. Matrix Iterative Analysis, Prentice–Hall, Englewood Cliffs, New Jersey (1962).

[41] Axel Ruhe and Torbjörn Wiberg, The method of conjugate gradients used in inverse iteration, BIT 12 (1972) 453–551.

[42] Tewarson, R. P., Sparse matrices, Academic Press, 1973.

[43] Communication of the ACM 1961. 7., 66-os ALGOL program.

A TANULMÁNYOK sorozatban eddig megjelentek:

1/1973 Pásztor Katalin: Módszerek Boole–függvények minimális vagy nem redundáns, $\{\wedge, \vee, \neg\}$ vagy {NOR} vagy {NAND} bázisbeli, zárójeles vagy zárójel nélküli formuláinak előállítására

2/1973 Вашкеви Иштван: Расчленение многосвязных промышленных процессов с помощью вычислительной машины

3/1973 Ádám György: A számítógépipar helyzete 1972 második felében

4/1973 Bányász Csilla: Identification in the Presence of Drift

5/1973* Gyürki J.–Laufer J.–Girnt M.–Somló J.: Optimalizáló adaptív szerszámgépirányítási rendszerek

6/1973 Szelke Erzsébet–Tóth Károly: Felhasználói Kézikönyv (USER MANUAL) a Folytonos Rendszerek Szimulációjára készült ANDISIM programnyelvhez

7/1973 Legendi Tamás: A CHANGE nyelv/multiprocesszor

8/1973 Klafszky Emil: Geometriai programozás és néhány alkalmazása

9/1973 R. Narasimhan: Picture Processing Using Pax

10/1973 Dibuz Ágoston–Gáspár János–Várszegi Sándor: MANU–WRAP hátlaphuzalozó. MSI–TESTER integrált áramköröket mérő, TESTOMAT–C logikai hálózatokat vizsgáló berendezések ismertetése

11/1973 Matolcsi Tamás: Az optimum–számítás egy új módszeréről

12/1973 Makroprocesszorok, programozási nyelvek. Cikkgyűjtemény az NJSzT és SZTAKI közös kiadásában. Szerkesztette: Legendi Tamás

13/1973 Jedlovszky Pál: Uj módszer bonyolult rektifikáló oszlopok vegyészmérnöki számítására

14/1973 Bakó András: MTA Kutatóintézeteinek bérszámfejtése számítógéppel

15/1973 Ádám György: Kelet–nyugati kapcsolatok a számítógépiparban

16/1973 Fidrich Ilona–Uzsoky Miklós: LIDI–72 LIstakezelő rendszer a DIgitális Osztályon, 1972. évi változat

17/1974 Gyürki József: Adaptív termelésprogramozó rendszer (APS) termelő műhelyek irányitására

18/1974 Pikler Gyula: MINI–Számítógépes interaktív alkatrészprogramíró rendszer NC szerszámgépek automatikus programozásához

19/1974 Gertler, J.–Sedlak, J.: Software for process control

20/1974 Vámos, T.–Vassy, Z.: Industrial Pattern Recognition Experiment–A Syntax Aided Approach

21/1974 A KGST I.–15–1.: Diszkrét rendszerek automatikus tervezése c. témában 1973. februárban rendezett szeminárium előadásai

22/1974 Arató, M.–Benczúr, A.–Krámli, A.–Pergel, J.: Stochastic Processes, Part I.

23/1974 Benkó Sándor–Renner Gábor: Erősen telített mágneses körök számítógépes tervezési módszere

24/1974 Kovács György–Franta Lászlóné: Programcsomag elektronikus berendezések hátlaphuzalozásának tervezésére

25/1974 Járdán R. Kálmán: Háromfázisú tirisztoros invertek állandósult tranziens jelenségei és belső impedanciája

A *-gal jelölt kivételével a sorozat kötetei megrendelhetők az Intézet könyvtáránál (Budapest, I. Uri u. 49.).

Jelen tanulmány a 4.9.1. és 4.9.2. "Lineáris és nemlineáris egyenletrendszerek, sajátértékproblémák. Numerikus programkönyvtár kezelése és bővítése" c. intézeti alapkutatási téma keretében készült.

Beérkezett: 1974. május 24.