Nagyméretű ritkán kitöltött mátrixok számítógépes kezelése a kiegyenlítő számításban

Dr.Völgyesi lajos a BME Általános- és Felsőgeodézia Tanszékének
egyetemi docense

A számítástechnika fejlődésével fokozatosan nőnek az igények az egyre nagyobb méretű lineáris egyenletrendszerek megoldására. A nagyméretű lineáris programozási feladatok megoldhatóságának korlátait a számítógépek operatív (RAM) memória kapacitása szabja meg.

A kiegyenlítési feladatok többségében az alakmátrixok rendszerint nagyon sok zérus elemet tartalmaznak. Ebben az esetben a javítási egyenletek képzésére bemutatunk egy helytakarékos eljárást, amely lehetővé teszi nagy méretű mátrixok kezelését. Célszerű lehetőség ilyenkor a mátrix ortogonalizációs kiegyenlítési eljárás használata, amely az ismeretleneket a közvetítő egyenletekből a normálegyenletek felállítását kikerülve közvetlenül szolgáltatja.

1. Sparse mátrixok

A kiegyenlítő számításban az ismeretlenek számának lineáris növekedésével a normálegyenletek együtthatóinak száma négyzetesen szaporodik. A kiegyenlítési feladatok többségében a nagyméretű mátrixok kezelésére az ad lehetőséget, hogy a mátrixok rendszerint nagyon sok zérus elemet tartalmaznak. Ezeket a sok zérus elemet tartalmazó mátrixokat ritkán kitöltött (sparse) mátrixoknak nevezzük. A kiegyenlítő számításban fellépő sparse mátrixok általában speciális alakú szalagmátrixok (a zérustól különböző elemek a főátlóban és azzal párhuzamos vonalakkal határolt sávban helyezkednek el). Vizsgálatainkban a sparse mátrixok struktúrájában nem feltételezünk semmiféle szabályszerűséget.

Lineáris egyenletrendszerek megoldására és a mátrixok invertálására nagyon sok numerikus módszer ismeretes (*Detrekői*, 1991). Ezek a sparse mátrixok esetére is alkalmazhatók, de nem mindegyikük használata célszerű a sok zérus elem fölösleges tárolása és kezelése miatt. Fontos megjegyeznünk, hogy valamely sparse mátrix inverze általában nem sparse tulajdonságú (*Gergely*, 1974), ezért ha a kiegyenlítési feladatokban a közvetítő egyenletek együttható mátrixa ritkán kitöltött, akkor a normálegyenletek képzése és invertálása helyett célszerűbb a közvetlen megoldás. A közvetlen megoldásra igen jó lehetőség a mátrix ortogonalizációs eljárás (*Charamza*, 1971; *Völgyesi, 1979, 1980*), amely az ismeretleneket a közvetítő egyenletekből a normálegyenletek felállítását kikerülve közvetlenül szolgáltatja. A mátrix ortogonalizációs eljárás alkalmazása esetén az ismeretlenek számának mindössze az szab határt, hogy az operatív memóriába legalább két teljes mátrixoszlopnak kell egyszerre elférni.

2. A javítási egyenletek helytakarékos képzése

Az alábbiakban megadunk egy eljárást, amely alkalmazásával ritkán kitöltött együtthatómátrixok esetén a mátrix ortogonalizációs módszerrel nagy ismeretlen számú kiegyenlítési feladatok oldhatók meg.

Az első lépés a

$$\boldsymbol{v = A\,x - l} \qquad (1)$$

javítási egyenletek ***A*** alakmátrixának és a tisztatagok ***l*** vektorának megfelelő helytakarékos képzése. A helytakarékos tárolás céljából egyetlen ***s*** segédvektorban képezzük és tároljuk mind az ***A*** mátrix, mind az ***l*** vektor elemeit.

Döntő fontosságú, hogy a javítási egyenletek felírásakor az ***A*** mátrix elemeit sorfolytonosan tudjuk meghatározni, ezért első lépésben az ***s*** segédvektor megfelelő részét is csak sorfolytonosan tudjuk feltölteni. Ugyanakkor a mátrix ortogonalizáció során a mátrix oszlopain kell a megfelelő ortogonális transzformációkat végrehajtani, ezért később az ***s*** vektoron belül a sorfolytonos tárolási módról át kell térni oszlopfolytonos tárolási formára.

Amennyiben az (1) javítási egyenletek ***A*** alakmátrixa sok zérus elemet tartalmaz, célszerű csupán a zérustól különböző mátrixelemeket megadni. Ekkor viszont meg kell jelölni a nem zérus elemek mátrixon belüli sor és oszlop koordinátáit is, ami minden nem zérus elem esetén további két adat tárolását igényli.

A kiegyenlítési feladat tényleges megoldása során először alakítsuk ki az ***s*** segédvektor szerkezetét az *1. ábrán* látható módon! Az ***s*** vektor $sor+osz+2*adat+1$ hosszúságú kell legyen (ahol sor = az ***A*** mátrix sorainak - vagyis a felírható egyenletek száma, osz = az ***A*** mátrix oszlopainak - vagyis az ismeretlenek száma, $adat$ = az ***A*** mátrixban lévő nem zérus elemek száma).

Az *1.* lépés az ***s*** vektor létrehozása és az első $osz+1$ elemének feltöltése zérus elemekkel. Az ***s*** vektort a számítógép memóriája által megengedett maximális méretűre kell választani, mivel ekkor még nem tudjuk az $adat$ paraméter értékét.

A *2.* lépés az ***A*** alakmátrix elemeinek képzése és a nem zérus elemek speciális elhelyezése az ***s*** vektorba. Miközben az ***A*** mátrix elemeit a

$$\pm 10000 * oszlopindex + sorindex + 0.1 * mátrixelem \qquad (2)$$

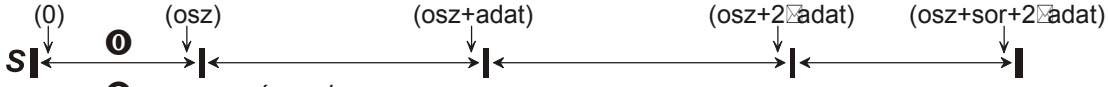
formában sorra elhelyezzük az ***s*** vektor $osz+1$ memóriacímétől, egyúttal minden egyes mátrixelem beírásakor az ***s*** vektor $1 - \ldots\ osz$ megfelelő rekeszében az oszlopok számlálójának értékét megnöveljük 1-gyel. Így a mátrixelemek képzésének befejezése után az $1 - \ldots\ osz$ rekeszekben sorra megkapjuk, hogy az ***A*** mátrix egyes oszlopaiban hány darab nem zérus mátrixelemet találtunk.

(Nagyobb számítási pontosság igénye esetén párosával különválasztva tárolhatjuk az $osz+1$ memóriacímtől a $10000 * oszlopindex + sorindex$ és a hozzá tartozó $\pm mátrixelem$ értékeket, így nem foglalunk el értékes tizedeseket a sor és oszlopindexek számára a mátrix együtthatók rovására.)

Az A mátrix és az I vektor elõállítása az S segédvektorban:

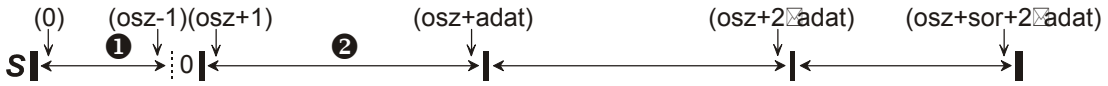
osz = oszlopok száma, sor = sorok száma,
adat = zérustól különbözõ mátrix elemek száma az ***A*** *mátrixban*

1:



❶ = *csupa zérus elem*

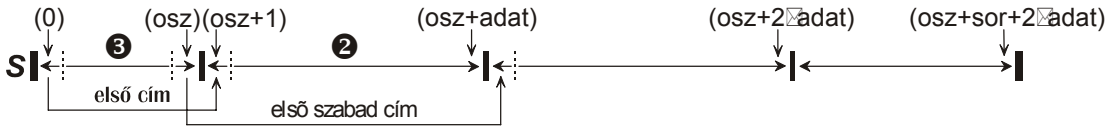
2:



❶ = *sorra megmutatja, hogy az egyes oszlopokban hány db. nem zérus mátrixelem van az* ***A*** *mátrixban*

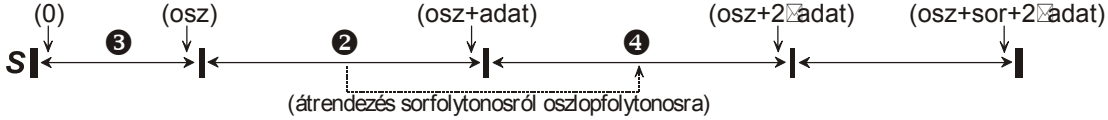
❷ = *sorfolytonosan:* $\pm 1000 * oszlopindex + sorindex + 0.1 * mátrixelem$

3:



❸ = *sorra megmutatja azon rekeszek címét, ahol majd az utolsó fázisban új oszlopindexek kezdődnek a* ❹ *-ben és a* ❻ *-ban*

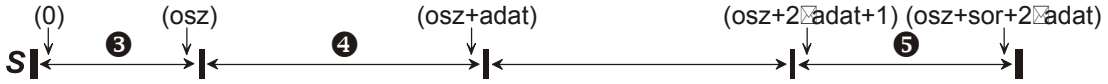
4:



❹ = *oszlopfolytonosan:* $\pm 1000 * oszlopindex + sorindex + 0.1 * mátrixelem$

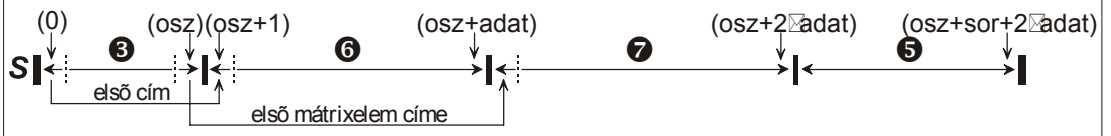
5:

6:



❺ = *jobb oldal (tisztatagok) vektora*

7:



❸ = *sorra megmutatja azon rekeszek címét, ahol új oszlopindexek kezdődnek a* 🕓 *-ban*

❻ = *csak sorindexek oszlopfolytonosan*

❼ = *csak az* ***A*** *mátrix elemei oszlopfolytonosan*

❺ = *a jobb oldal (a tisztatagok) vektora*

1. ábra

A *3.* lépésben a *0 - … osz+1* memóriarekeszek tartalmát átalakítjuk, és az egyes oszlopokban található nem zérus mátrixelemek száma alapján (ezeket sorra összeadva) meghatározzuk, hogy az ***s*** vektor *osz+1 - … osz+adat* területén a végso *7.* fázis után mely memóriacímeken kezdődnek majd az új oszlopindexekhez tartozó sorindexek és mátrixelemek.

A *4.* lépésben az ***s*** vektor *osz+1 - … osz+adat* területéről a sorfolytonosan tárolt (2) típusú adatokat oszlopfolytonosra átrendezve áthelyezzük az ***s*** vektor *osz+adat+1 - … osz+2∗adat* memória területére.

Az *5.* lépésben az ***s*** vektor *osz+adat+1 - … osz+2∗adat* területéről változtatás nélkül visszahelyezzük az adatokat az eredeti *osz+1 - … osz+adat* memória területre.

A *6.* lépésben képezzük a tisztatagok ***l*** vektorát és az értékeket sorfolytonosan betöltjük az ***s*** vektor *osz+2∗adat+1 - … osz+sor+2∗adat* területére.

Végül a 7. lépésben az ***s*** vektor *osz+1 - … osz+adat* területén tárolt adatokból szétválasztjuk a sorindexeket és a mátrix elemeket. A sorindexeket oszlopfolytonosan az ***s*** vektor *osz+1 - … osz+adat* területén, a mátrix elemeket pedig az *osz+adat+1 - … osz+2∗adat* területen képezzük.

Ezzel kialakítottuk az ***s*** vektor *1.* ábrán vázolt végleges szerkezetét:

- a *0 - … osz* területen sorra megtaláljuk hol kezdodnek az új oszlopindexek az ***s*** vektoron belül (az *osz+1* megadja az elso mátrix elem címét),
- az *osz+1 - … osz+adat* területen sorra megtaláljuk a mátrix elemekhez tartozó sorindexeket oszlopfolytonosan,
- az *osz+adat+1 - … osz+2∗adat* területen tároljuk oszlopfolytonosan a mátrix elemeket,
- az *osz+2∗adat+1 - … osz+sor+2∗adat* területen a tisztatagok találhatók.

3. A mátrix ortogonalizációs eljárás alapelve

A következő lépés az adatok előkészítése a mátrix ortogonalizációs kiegyenlítési eljárás számára, - amihez viszont ismernünk kell az eljárás alapelvét (*Charamza*, 1971; *Strang* (1976); *Völgyesi, 1979, 1980*). Ezt a

$$\left[\begin{array}{c|c} \underset{(n,r)}{\mathbf{A}} & \underset{(n,1)}{\mathbf{l}} \\ \hline \underset{(r,r)}{\mathbf{E}} & \underset{(r,1)}{\mathbf{0}} \end{array}\right] \longrightarrow \left[\begin{array}{c|c} \underset{(n,r)}{\mathbf{W}} & \underset{(n,1)}{\mathbf{v}} \\ \hline \underset{(r,r)}{\mathbf{G}^{-1}} & \underset{(r,1)}{\mathbf{x}} \end{array}\right] \qquad (3)$$

hipermátrix transzformáció szemlélteti, ahol ***A*** a javítási egyenletek együtthatómátrixa, ***l*** a tisztatagok vektora, ***E*** egységmátrix, ***0*** zérus vektor; ***W*** egy ortogonális oszlopokkal rendelkező mátrix, $\mathbf{G}^{-1}$ pedig egy felső háromszögmátrix.

A (3) transzformáció algoritmusának szemléltetéséhez vezessük be az alábbi jelöléseket: legyen $\mathbf{a}_j$ az ***A*** mátrix *j* -edik oszlopa, $\mathbf{w}_j$ a ***W*** mátrix *j* -edik oszlopa, $\mathbf{e}_j$ az ***E*** mátrix *j* -edik oszlopa, és végül jelölje $\mathbf{g}_j$ a $\mathbf{G}^{-1}$ mátrix *j* -edik oszlopát! Ezekkel a jelölésekkel a (3) mátrix transzformáció az alábbi lépésekben hajtható végre:

$$\begin{bmatrix} \mathbf{w}_1 \\ \mathbf{g}_1 \end{bmatrix} = \frac{\begin{bmatrix} \mathbf{a}_1 \\ \mathbf{e}_1 \end{bmatrix}}{\|\mathbf{a}_1\|_E}$$

$$\left\{ \begin{array}{l} \begin{bmatrix} \mathbf{a}_j \\ \mathbf{e}_j \end{bmatrix}_{\langle 1 \rangle} = \begin{bmatrix} \mathbf{a}_j \\ \mathbf{e}_j \end{bmatrix} \\ \begin{bmatrix} \mathbf{a}_j \\ \mathbf{e}_j \end{bmatrix}_{\langle k+1 \rangle} = \begin{bmatrix} \mathbf{a}_j \\ \mathbf{e}_j \end{bmatrix}_{\langle k \rangle} - \left((\mathbf{a}_j)_{\langle k \rangle} , \mathbf{w}_k \right) \begin{bmatrix} \mathbf{w}_k \\ \mathbf{g}_k \end{bmatrix} \\ \begin{bmatrix} \mathbf{w}_j^* \\ \mathbf{g}_j^* \end{bmatrix} = \begin{bmatrix} \mathbf{a}_j \\ \mathbf{e}_j \end{bmatrix}_{\langle j \rangle} \end{array} \right\} \qquad (4)$$

$$\begin{bmatrix} \mathbf{w}_j \\ \mathbf{g}_j \end{bmatrix} = \frac{\begin{bmatrix} \mathbf{w}_j^* \\ \mathbf{g}_j^* \end{bmatrix}}{\|\mathbf{w}_j^*\|_E}$$

$j = 2, 3, \ldots, r ; \qquad k = 1, 2, \ldots, j-1$

majd ezt követoen

$$\begin{bmatrix} \mathbf{v} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{l} \\ \mathbf{0} \end{bmatrix} - \sum_{k=1}^{r} \left(\mathbf{l} , \mathbf{w}_k \right) \begin{bmatrix} \mathbf{w}_k \\ \mathbf{g}_k \end{bmatrix} \qquad (5)$$

ahol $\|\mathbf{a}_1\|_E$ és $\|\mathbf{w}_j^*\|_E$ az $\boldsymbol{a_1}$ illetve a $\boldsymbol{w_j^*}$ oszlopvektorok euklideszi normája, az $\left((\mathbf{a}_j)_{\langle k \rangle} , \mathbf{w}_k \right)$ az $(\mathbf{a}_j)_{\langle k \rangle}$ illetve a $\boldsymbol{w}_k$ oszlopvektorok-, az $\left(\mathbf{l}, \mathbf{w}_k \right)$ pedig az $\boldsymbol{l}$ és a $\boldsymbol{w}_k$ vektorok skaláris szorzata.

A (3) mátrix transzformáció a keresett x_i ismeretleneket és a v_i javításokat az $\boldsymbol{x}$ illetve a $\boldsymbol{v}$ vektor helyén közvetlenül szolgáltatja. Az x_i ismeretlenek varianciáját és kovarianciáit a

$$\mathbf{Q}_{(\mathbf{x})} = \mathbf{G}^{-1} (\mathbf{G}^{-1})^*$$

súlykoefficiens mátrix tartalmazza, ahol $(\mathbf{G}^{-1})^*$ a $\mathbf{G}^{-1}$ transzponáltját jelöli.

A mátrix ortogonalizációs kiegyenlítési eljárás további részleteinek tárgyalásával itt nem foglalkozunk, ezek korábbi publikációkban megtalálhatók.

4. Az ortogonalizáció gyakorlati végrehajtása

A fentiek ismeretében kell előállítanunk a

$$\hat{\mathbf{A}} = \left[\begin{array}{c|c} \underset{(n,r)}{\mathbf{A}} & \underset{(n,1)}{\mathbf{l}} \\ \hline \underset{(r,r)}{\mathbf{E}} & \underset{(r,1)}{\mathbf{0}} \end{array}\right]$$

hipermátrix oszlopait az $\boldsymbol{s}$ vektor adatai alapján. Az $\hat{\mathbf{A}}$ oszlopait úgy kell előállítani, hogy most már a zérus elemeket is be kell írni, és sorrendben egyenként külső tárolón elhelyezni. Ha valamennyi oszlopot elhelyeztük a külső tárolón, akkor elkezdhetjük az ortogonalizációt a (4) és az (5) algoritmus szerint: beolvassuk az első új oszlopot, normalizáljuk és visszaírjuk a külső tárolóba; beolvassuk a második új és az első (normalizált) oszlopot, ortogonalizáljuk a másodikat az első oszlopra, normalizáljuk és visszaírjuk a külső tárolóba; beolvassuk az első normalizált és a harmadik új oszlopot, ortogonalizáljuk a harmadikat az első oszlopra, beolvassuk az első oszlop helyére a második (már az első oszlopra ortogonalizált és normalizált) oszlopot, ortogonalizáljuk erre a második oszlopra is a harmadikat, majd normalizálva visszaírjuk a külső tárolóba; … a folyamatot addig folytatjuk, amíg az utolsó oszlopot is ortogonalizáljuk az összes előtte lévő oszlopra, azonban az utolsó oszlopot már nem szabad normalizálni. Ekkor az utolsó oszlop első n db eleme a javításokat, az utolsó r db eleme pedig a keresett ismeretleneket fogja tartalmazni.

Mivel a tényleges ortogonalizáció az operatív memóriában történik, a fenti módszer esetében a megoldható feladat méretének (az ismeretlenek maximális számának) az szab határt, hogy az operatív memóriában legalább két teljes mátrixoszlopnak kell egyszerre elférni. A számítás sebessége fokozható, ha az operatív memóriába egyszerre több oszlop is befér az ortogonalizáció során.

5. Alkalmazások

A fenti módszert sikeresen alkalmaztuk a gravitációs hálózatok kiegyenlítése és az Eötvös-inga mérések alapján végezhető függővonal elhajlás interpoláció megoldására (*Völgyesi*, 1995). Mindkét probléma megoldása során kedvező tapasztalatok adódtak az ismeretlenek nagy száma esetén mind a numerikus stabilitás mind a számítás elvégzéséhez szükséges idő tekintetében.

A szerző ezúton köszöni meg a T-030177 sz. OTKA, valamint az MTA Fizikai Geodézia és Geodinamika Kutatócsoport által a fenti vizsgálatok elvégzéséhez nyújtott támogatást.

IRODALOM

1. *Charamza F.*: Orthogonalization Algorithm … Geofisikálni Sbornik Vol. XIX, (1971)
2. *Detrekői Á.*: Kiegyenlítő Számítások. Tankönyvkiadó, Budapest (1991)
3. *Gergely J.*: Numerikus modellek sparse mátrixokra. *MTA SZTAKI tanulmányok* No.26. (1974)
4. *Strang G.*: Linear Algebra and its Applications. *Academic Press,* New York, … pp. 414 (1976)

5. *Völgyesi L.*: A numerikus modellek választásának néhány kérdése … *Geod. és Kart.* Vol. 31, No.5., pp. 327-334 (1979)
6. *Völgyesi L.*: A mátrix-ortogonalizációs módszer gyakorlati alakalmazása a kiegyenlítő számításban. *Geod. és Kart.* Vol. 32, No. 1. pp. 7-15 (1980)
7. *Völgyesi L.*: Test Interpolation of Deflection of the Vertical … . P*eriodica Polytechnica, Civil Eng.* Vol. 39, No. 1. pp. 37-75 (1995)

Treatment of large sparse matrices by computers in adjustment

L. Völgyesi

Summary

In majority of adjustment problems matrices of observation equations contain a lot of zero elements. It is presented a special method for creation of observation equations which give us a good possibility to treat large sparse matrices in adjustment. Matrix orthogonalization is a suitable adjustment method in this case, which serves the unknowns directly from the observation equations, evading creation of normal equations.

* * *

Völgyesi L (2000): Nagyméretű, ritkán kitöltött mátrixok számítógépes kezelése a kiegyenlítő számításban. Geodézia és Kartográfia, Vol. 52, Nr. 9, pp. 33-36.

Dr. Lajos VÖLGYESI, Department of Geodesy and Surveying, Budapest University of Technology and Economics, H-1521 Budapest, Hungary, Műegyetem rkp. 3.
Web: http://sci.fgt.bme.hu/volgyesi E-mail: volgyesi@eik.bme.hu