ALGORITMI ZA REINVERZIJO BAZNE MATRIKE V LINEARNEM PROGRAMU

Janez Barle, Janez Grad
Univerza Edvarda Kardelja v Ljubljani,
Ekonomska fakulteta Borisa Kidriča, Ljubljana

UDK: 519.852.11

V prispevku opišemo nekatere algoritme za invertiranje matrik, ki so uporabni pri reševanju linearnih programov. Njihova značilnost je specifičen način predstavitve inverzne matrike in razdelitev postopka v dve fazi. Poseben poudarek dajemo problemom, ki se nanašajo na računalniško realizacijo algoritmov.

BASIS MATRIX REINVERSION ALGORITHMS IN LINEAR PROGRAMMING. In this paper we describe some of the algorithms for matrix reinversion which can be applied in the procedures for solving the linear programming problem. They use a specific presentation of the inverse matrix and two-phase computation process. Special attention is paid to those problems associated with the computer processing of these algorithms.

UVOD

Reševanje sistema linearnih enačb in s tem povezan problem invertiranja matrike spadata med klasične probleme numerične linearne algebre. Običajne metode za reševanje linearnih enačb uporabljajo Gaussov eliminacijski postopek, pri katerem se zaradi zagotavljanja numerične stabilnosti izvaja t.i. pivotiranje (delno ali popolno). Pri delu na programski opremi za linearno programiranje smo prišli do spoznanja, da omenjene metode niso vedno učinkovite. Zaradi lastnosti matrik, ki se pojavljajo pri praktičnih problemih linearnega programiranja, je treba za njihovo invertiranje uporabiti posebne postopke. Uporaba teh postopkov lahko precej poveča hitrost in natančnost pri reševanju linearnih programov velikih dimenzij. Zato je podprogram za invertiranje matrike ena od najbolj pomembnih sestavin komercialnih programskih paketov za linearno programiranje. V nadaljevanju podajamo opis nekaterih najbolj znanih algoritmov za invertiranje matrike pri linearnem programiranju. Opisujemo tudi nekatere izkušnje, ki smo jih pridobili pri programiranju in testiranju teh algoritmov.

Problem linearnega programiranja lahko v matrični obliki opišemo takole:

$$\text{minimiziraj } c^T x$$

$$\text{pri pogojih}$$

$$Ax = b \text{ in } x \geq 0$$

kjer je A realna matrika dimenzije m*n, b, c in x pa realni vektorji ustreznih dimenzij. Optimalna rešitev linearnega programa je tak vektor x, ki reši gornji problem.

Upoštevati moramo, da pri večini praktično pomembnih problemov linearnega programiranja velja:

1) Matrika A je zelo velikih dimenzij (tudi do nekaj tisoč vrstic in stolpcev)

2) Matrika A je razpršena. To pomeni, da je delež njenih neničelnih elementov zelo majhen (pogosto tudi manj od 1%). Zaradi tega se v praksi za reševanje linearnih programov skoraj vedno uporablja t.i. revidirana metoda simpleksov, ki jo je možno prilagoditi delu z matrikami velikih dimenzij. To je iterativna metoda, ki je podrobno opisana v literaturi (npr. [1]). Omenimo le, da se na vsakem koraku te metode rešuje zaporedje sistemov linearnih enačb

$$\pi B = c_B,\ Bv = u \text{ in } Bx_B = b$$

kjer je B nesingularna kvadratna podmatrika matrike A (imenujemo jo bazna matrika), c_B in x_B sta ustrezna podvektorja vektorjev c in b, u pa je stolpec matrike A, ki je določen z rešitvijo prvega sistema enačb. Reševanje gornjih sistemov enačb olajšuje dejstvo, da se bazni matriki pri dveh zaporednih korakih algoritma razlikujeta le v enem stolpcu.

PREDSTAVITEV INVERZNE BAZNE MATRIKE

Revidirana metoda simpleksov ima več inačic, ki izhajajo iz različnih načinov predstavitve matrike B (ali B^{-1}). Predstavitev matrike B^{-1} v eksplicitni obliki lahko uporabimo le pri problemih majhnih dimenzij. V obstoječih programskih paketih je najbolj običajna predstavitev inverzne bazne matrike v obliki produkta t.i. elementarnih matrik:

$$B^{-1} = E_k E_{k-1} \ldots E_1$$

Elementarne matrike E_i (i = 1, ... k) se razlikujejo od enotne matrike le v enem stolpcu, tako da zadostuje shraniti neničelne elemente tega stolpca in podatke o tem, kje se le-ti nahajajo. Ena od prednosti produktne predstavitve inverzne bazne matrike je tudi enostaven način njenega ažuriranja. Po vsakem koraku revidirane metode simpleksov se v produkt dodaja nova elementarna matrika, stare elementarne

matrike pa ostanejo nespremenjene. Tako predstavitev matrike imenujemo tudi ETA-datoteka, tisti stolpec, v katerem se elementarna matrika razlikuje od enotne pa imenujemo ETA-vektor.

Znano je, da pri reševanju linearnih enačb po Gaussovem eliminacijskem postopku računanje inverzne matrike ni potrebno. Boljša pot za reševanje sistema enačb (Bx = b) je izvršiti razcep permutirane matrike na spodnjo in zgornjo trikotno matriko (PB = LU) in potem reševati dva trikotna sistema enačb (Ly = b in Ux = y). Ta ugotovitev v določeni meri velja tudi za bazno matriko linearnega programa. Za njo lahko uporabimo razcep B = LU, kjer vsakega od obeh faktorjev predstavimo z zaporedjem ustreznih trikotnih elementarnih matrik:

$L = L_1 L_2 \ldots L_k$ in $U = U_k \ldots U_1$

Tako predstavitev imenujemo eliminacijska oblika inverzne matrike. V zapisu smo zaradi poenostavitve izpustili permutacijske matrike, ki so podane z zaporedjem, v katerem obravnavamo vrstice in stolpce matrike B. Izkaže se, da v splošnem uporaba eliminacijske oblike matrike omogoča ekonomičnejšo izrabo pomnilnika in časovno hitrejši izračun rešitve linearnega programa v primerjavi s postopkom, ki uporablja produktno obliko inverzne matrike. Vseeno so trikotni razcep začeli vgrajevati v komercialne programske pakete šele potem, ko so bili razviti učinkoviti postopki za ažuriranje trikotnih faktorjev (npr. metoda Forresta in Tomlina, glej |1|). Ugotavljamo pa, da uporaba trikotnega razcepa zahteva precej bolj zapletene algoritme in podatkovne strukture kot postopek, ki uporablja produktno obliko inverzne matrike.

REINVERZIJA BAZNE MATRIKE

Ker se med izvajanjem simpleksnega algoritma število elementarnih matrik v predstavitvi B^{-1} ali v trikotnih faktorjih L in U povečuje, se izkaže za nujno občasno izračunati novo predstavitev B^{-1} ali novi trikotni razcep matrike B. Ta postopek, imenujemo ga reinverzija bazne matrike, sprošča pomnilnik in omogoča hitrejše in natančnejše izvajanje algoritma na preostalih korakih revidirane metode simpleksov. Naziv reinverzija je primeren tudi za postopek določanja novih elementarnih matrik trikotnega razcepa matrike, ker je ta naloga v tesni povezavi z invertiranjem matrike. Tako lahko iz znane eliminacijske oblike matrike takoj dobimo produktno obliko inverzne matrike:

$$B^{-1} = U^{-1}L^{-1} = U_1^{-1} \ldots U_k^{-1} L_k^{-1} \ldots L_1^{-1}$$

Za produktno in eliminacijsko obliko matrike uporabljamo tudi skupen naziv ETA-faktorizacija matrike. Omenimo še, da je inverzum elementarne matrike zopet elementarna matrika. Vzemimo, da se prvotna elementarna matrika razlikuje od enotne matrike v stolpcu, ki ga označimo z y. Pri inverzumu se ta stolpec spremeni po naslednjih pravilih: diagonalni element y_k postane $1/y_k$, vsi ostali elementi y_i pa se spremenijo v $-y_i/y_k$.

Pri reinverziji izhajamo iz prvotne matrike B, ki je praviloma razpršena. Lahko se zgodi, da ima matrika B^{-1} mnogo več neničelnih elementov kot B. To velja tudi za skupno število neničelnih elementov v ETA-faktorizaciji inverzne matrike. Ta pojav, imenujemo ga napolnitev, je posebno izrazit takrat, ko pri izvajanju Gaussovega eliminacijskega postopka uporabimo delno ali popolno pivotiranje. To sta postopka, pri katerih se teži k izbiri ključnega elementa (pivota) s čim večjo absolutno vrednostjo ("pivoting for size"). Zato je pri reinverziji bazne matrike treba uporabiti take kriterije za izbor pivotov, ki vodijo k ETA-faktorizaciji s čim manjšim številom neničelnih elementov ("pivoting for sparsity"). Poskusi optimalne rešitve tega problema, ki je znan tudi pod imenom "problem minimalne napolnitve", so pripeljali do ugotovitve, da je ta problem NP-poln. To pomeni, da je malo upanja, da bo za njegovo reševanje kdaj oblikovan ekonomičen algoritem (glej |10|). Zato so bili razviti številni algoritmi, osnovani na različnih heurističnih pravilih za izbor pivota, ki so bolj ali manj uspešno reševali zastavljeno nalogo. Pomemben dogodek v raziskovanju tega problema je bil, ko je leta 1957 H. Markowitz objavil članek o tej problematiki (glej |8|). V njem je postavil nekaj trditev, ki so spodbudile številne poznejše raziskave:

1) Postopek invertiranja je smiselno razdeliti v dve fazi. V prvi fazi, imenujemo jo analitična faza, se vnaprej določi zaporedje, po katerem se bo vršilo pivotiranje. Pri tem se uporabljajo le informacije o tem, kateri elementi so neničelni, ne oziraje se na njihovo vrednost. V drugi fazi, imenujemo jo numerična faza, se na podlagi vnaprej določenega zaporedja pivotov izvrši dejanski izračun nove ETA-faktorizacije.

2) Dobre rezultate pri minimizaciji neničelnih elementov v ETA-faktorizaciji lahko dosežemo tako, da na posameznem koraku izbiramo tisti pivot, pri katerem ima "Markowitzevo število"

   $(v_i - 1)(s_j - 1)$

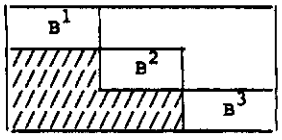
   minimalno vrednost. Z v_i ("vrstični števec") in s_j ("stolpični števec") sta označeni števili neničelnih elementov v i-ti vrstici in j-tem stolpcu, pri čemer upoštevamo le tiste vrstice in stolpce, v katerih še ni bilo izvršeno pivotiranje.

Markowitz je bil tudi prvi, ki je ugotovil primernost eliminacijske oblike za predstavitev bazne matrike pri linearnem programiranju.

ANALITIČNA FAZA REINVERZIJE

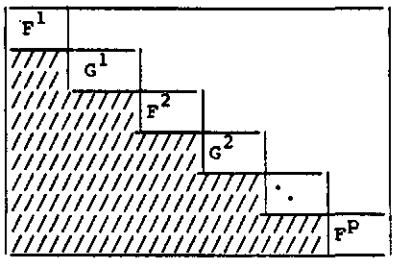
Ideja o ločenem izvajanju analitične faze se ni uveljavila le pri reinverziji bazne matrike, temveč tudi pri vseh drugih problemih numerične matematike, ki zahtevajo reševanje razpršenih sistemov linearnih enačb. Praksa je potrdila tudi to, da uporaba Markowitzevega pravila pri izbiri pivota vodi k zelo razpršeni ETA-faktorizaciji inverzne matrike. Vseeno je direktna uporaba tega pravila otežkočena, ker zahteva hkraten dostop do vrstic in stolpcev matrike. Kolikor nam je znano, uporablja Markowitzevo pravilo v čisti obliki le podprogram za reinverzijo pri IBM-ovem programskem paketu MPSX/370 (glej |2|). Izkazalo se je, da je možno uporabiti različne kriterije, ki na posreden način merijo Markowitzevo število, hkrati pa zahtevajo veliko manj računanja.

Sodobni programski paketi v analitični fazi reinverzije večinoma uporabljajo algoritem Hellermana in Raricka (glej |6| in |7|). Ta algoritem je prirejen strukturi matrik, ki so značilne za praktične probleme linearnega programiranja, s čimer lahko pojasnimo njegovo učinkovitost. Cilj tega algoritma je, da se z zamenjavo vrstic in stolpcev matrika B prevede v t.i. HR matriko, ki ima strukturo kot na sliki 1,



Slika 1

kjer so B^1 in B^3 spodnje trikotne matrike z neničelnimi elementi na diagonali. Lahko privzamemo, da ima B^2, če ni prazna, v vsaki vrstici in stolpcu vsaj dva neničelna elementa, ker bi v nasprotnem primeru lahko razširili B^1 ali B^3. Matriko B^2 imenujemo jedro ali izboklina ("bump"). Algoritem Hellermana in Raricka se nadaljuje z ustreznimi operacijami na jedru, dokler jedro ne postane strukturirano kot na sliki 2, kjer so G^k ali prazne ali spodnje trikotne matrike z neničelnimi elementi na diagonali:



Slika 2

Matrike F^p imenujemo zunanje izbokline ("external bumps"). Imajo vsaj po en stolpec z neničelnimi elementi nad diagonalo. Take stolpce imenujemo konice ("spikes"). Zunanje izbokline se morajo odlikovati z naslednjimi lastnostmi:

i) zadnji stolpec v izboklini je konica z neničelnim elementom v najvišji vrstici,

ii) stolpci, ki niso konice, morajo imeti neničelne diagonalne elemente.

Algoritma ne bomo podrobneje opisovali. Omenimo le, da se konice izbirajo na temelju stolpične številske funkcije $t_k(j)$, kjer

$t_k(j)$ = število neničelnih elementov j-tega stolpca v vrsticah z vrstičnim številom manjšim ali enakim k

Algoritem Hellermana in Raricka pokaže tudi zanimivo povezavo med teorijo grafov in linearnim programiranjem. Binarno matriko, s katero delamo v analitični fazi reinverzije, lahko interpretiramo kot matriko sosednosti usmerjenega grafa. V jeziku teorije grafov pomeni iskanje bločne trikotne oblike matrike (to je oblika matrike na sliki 2) določanje močno povezanih komponent usmerjenega grafa. To je problem, ki je spodbudil precejšnjo pozornost raziskovalcev s področja teorije grafov. Mi smo za reševanje te naloge sprogramirali eno varianto t. i. Tarjanovega algoritma, ki smo jo povzeli po |3|. Zanimivo je, da je uporaba metod iz teorije grafov pripeljala do izboljšane variante algoritma Hellermana in Raricka, ki je v literaturi dobila ime P4 ("Partitioned Preassigned Pivot Procedure"). S to varianto se praviloma dobi večje število zunanjih izboklin in manjše število konic kot s starejšo varianto algoritma, ki je znana pod nazivom P3 ("Preassigned Pivot Procedure").

Na koncu naj omenimo, da so bili na temelju algoritma Hellermana in Raricka razviti še nekateri zanimivi algoritmi, kot je npr. algoritem opisan v |9|.

NUMERIČNA FAZA REINVERZIJE

V numerični fazi reinverzije se, na temelju zaporedja pivotov določenega v analitični fazi, računa nova ETA-faktorizacija inverzne bazne matrike. V tej fazi moramo različno obravnavati produktno in eliminacijsko obliko inverzne matrike, kar ni bilo potrebno v analitični fazi. V |4| in |5| je podan zelo izčrpen pregled različnih algoritmov za izvajanje numerične faze reinverzije. V nadaljevanju podajamo opis algoritma za produktno obliko inverzne bazne matrike, ki smo ga spogramirali v |11|.

Predhodno določeno zaporedje pivotov je podano z vektorjem C, ki vsebuje indekse stolpcev, in z vektorjem R, ki vsebuje indekse vrstic. Razen tega potrebujemo še indekse konic in podatke o tem, kje se začnejo posamezne zunanje izbokline. Zaradi prihranka pomnilniškega prostora je te informacije koristno vključiti v vektorja R in C. To lahko storimo tako, da z negativnim predznakom shranimo v C indekse stolpcev, ki so konice, in v R indekse vrstic, kjer se začnejo posamezne zunanje izbokline.

N1: |Inicializacija|

a) Vpeljemo naslednje parametre:
   (i) i=1 za indeksiranje pivotov,
   (ii) $l=C_i$ za stolpični indeks pivota,
   (iii) $r=|R_i|$ za vrstični indeks pivota,
   (iv) p=1 za indeksiranje začetka zunanje izbokline.

b) Rezerviramo prostor za
   (i) y - realni vektor dimenzije m,
   (ii) ETA-datoteko (predstavitev matrik E_i za $1 \le i \le m$).

c) Definiramo $y = B_{*l}$ (l-ti stolpec matrike B).

N2: |Določanje novega ETA-vektorja|

$$z_k = \begin{cases} 1/y_r & \text{za } k=r \\ y_k & \text{za } k \neq r \end{cases}$$

Vektor z je r-ti stolpec elementarne matrike E_i.

N3: |Testiranje konca algoritma|

a) če je i=m nadaljujemo na N7 (konec algoritma)

b) če je i<m definiramo
   (i) i=i+1
   (ii) $l=|C_i|$
   (iii) $r=|R_i|$

c) če je $R_i<0$ definiramo p=i (začetek nove izbokline)

N4: |Opredelitev konice|

če je $C_i > 0$ definiramo $y = B_{*l}$ in nadaljujemo na N2

N5: |Transformacija konice|

Izračunamo

$y = E_{i-1} \cdots E_p B_{*l}$

(E_p ustreza prvemu stolpcu v izboklini, ki pripada B_{*l})

N6: |Zamenjava konice, če je pivot enak nič|

a) če je $y_r \neq 0$ postopek nadaljujemo na koraku N2

b) Preiščemo vse konice s stolpičnimi indeksi $l = |C_j|$ za j>i, ki se nahajajo

v isti zunanji izboklini, dokler ne najdemo takšne za katero je r-ta komponenta $E_{i-1}\cdots E_k B_{*1}$ neničelna

c) Izračunamo $y = E_{i-1}\cdots E_p B_{*j}$

d) Zamenjamo C_i in C_j

e) Postopek nadaljujemo na koraku N2

N7: |Konec|

Določanje ETA-datoteke je končano.

Kot vidimo, je korak N5 potreben le za tiste stolpce, ki so konice. Edino v teh stolpcih se lahko pri reinverziji poveča število neničelnih elementov. To je razlog, da je algoritem Hellermana in Raricka tako pomemben. Preverjanje ali je pivot enak nič, ki je potrebno na koraku N6, se v praksi zamenjuje z testi, ki zagotavljajo numerično stabilnost postopka.

ZAKLJUČEK

Uporaba matematike v praksi često zahteva poznavanje metod, ki jih ni možno naučiti iz standardnih učbenikov. Ta ugotovitev velja tudi za metode, ki smo jih morali vključiti v podprogram za invertiranje bažne matrike linearnega programa. Naša raziskovalna skupina nadaljuje s spremljanjem tega raziskovalnega področja in razvojem ustreznih rešitev v sklopu programske opreme za linearno programiranje.

LITERATURA

1) Bastian M., Lineare Optimierung grosser Systeme: Compact Inverse Verfahren und Basisfaktoriesirung, Konigstain: Verlag Anton Hain, 1980.

2) Benichou M., J.M. Gauthier, G. Hentges, G. Ribiere, "The Efficient Solution of Large-Scale Linear Programming Problems - Some Algorithmic Techniques and Computational Results", Mathematical Programming, 13, pp. 280-322, 1977.

3) Gustavson F., "Finding the Block Lower Triangular Form of a Sparse Matrix", v: Bunch J.R., D.J. Rose (eds.), Sparse Matrix Computations, New York: Academic Press, 1976.

4) Helgason R.V., J.L. Kennington, "Spike Swapping in Basis Reinversion", Naval Research Logistic Quarterly, 27, pp. 697-701, 1980.

5) Helgason R.V., J.L. Kennington, "A note on Splitting the Bump in an Elimination Factorisation", Naval Research Logistics Quarterly, 29, pp. 169-178, 1982.

6) Hellerman E., D. Rarick, "Reinversion with the Preassigned Pivot Procedure", Mathematical Programming, 1, pp. 195-216, 1971.

7) Hellerman E., D. Rarick, "The Partitioned Preassigned Pivot Procedure", v: Rose D.J., R.A. Willoughby (eds.), Sparse Matrices and Their Applications, New York-London: Plenum Press, 1972.

8) Markowitz H., "The Elimination Form of the Inverse and its Application to Linear Programming", Management Science, 3, pp. 255-269, 1957.

9) Lin T.D., R.S.H. Mah, "Hierarchical Partition - a New Optimal Pivoting Algorithm", Mathematical Programming, 12, pp. 260-278, 1977.

10) Petkovšek M., "NP-polni problemi", v: M. Petkovšek, Pisanski T., Izbrana poglavja iz računalništva 1. del, Ljubljana: DMFA SRS, 1982.

11) Barle J., J. Grad, V. Rupnik, "Programska oprema za rešitev linearnega programa z razširitvami", Raziskovalni projekt po naročilu Iskre-Delta, Ljubljana: RCEF, Ekonomska fakulteta Borisa Kidriča v Ljubljani, 1984.