UDK 528.14:512.25
681.306.004.13
Originalni znanstveni rad

USPOREDBA EFIKASNOSTI DVAJU ALGORITAMA ZA RJEŠAVANJE NORMALNIH JEDNADŽBI KOJI SE BAZIRAJU NA METODI CHOLESKOG

Damjan JOVIČIĆ, Miljenko LAPAINE, Bojan PETROVIĆ, Svetozar PETROVIĆ — Zagreb*

U radu [6] izvršili smo usporedbu našeg algoritma ** iz istog rada i Galićevog algoritma** [3], i to prebrajanjem računskih operacija, usporedbom veličine potrebne memorije itsl. i iz toga izveli odgovarajuće zaključke o njihovoj efikasnosti. U svom najnovijem radu [4] Galić je stavio pod sumnju da li će se prilikom stvarnog korištenja na računalu potvrditi ono što smo ustvrdili na temelju teoretskog razmatranja. Ovakve sumnje su u principu pozitivne jer potiču da se stvar i praktično provjeri. Međutim, Galić je to »zaboravio« napraviti i time nas potaknuo da tu provjeru izvršimo sami. U ovom radu dajemo najvažnije rezultate te provjere.

1. UVOD

a) U Geodetskom listu 1—3/ 1983 izlazi rad autora Z. Galića [3] u kome opisuje algoritam za rješavanje simetričnih sistema linearnih jednadžbi metodom Choleskog. Autor daje dijagram toka i program u FORTRAN-u.

b) U našem radu [6] (Geodetski list 1—3/1984) dali smo pregled problema uočenih proučavanjem niza radova (među njima i [3]) koji se bave rješavanjem normalnih jednadžbi i posebno metodom Choleskog. Pritom smo uočili niz nedostataka (ne samo kod Z. Galića), izveli odgovarajuće zaključke i na temelju njih dali prikladan algoritam koji nastoji iskoristiti prednosti što ih pruža ideja Choleskog. Na kraju smo priložili program u BASIC-u za stolno računalo HP 9845A. Budući da je i to važno za kasnije razmatranje, objasnimo zašto smo priložili upravo program u BASIC-u. Težište rada bilo je ne na programu nego na algoritmu, a program smo mogli i ne priložiti ili pak priložiti realizacije u nekoliko programskih jezika. S obzirom na jednostavnost algoritma svaki iole iskusniji potencijalni korisnik može ga lako i sam prevesti na pro-

* Adresa autora: Damjan Jovičić, dipl. mat., Miljenko Lapaine, dipl. mat., Svetozar Petrović, dipl. mat., Geodetski fakultet, Zagreb, Kačićeva 26, Bojan Petrović, dipl. mat., Institut Ruđer Bošković, Bijenička cesta 54.

** U ovom radu će se riječ »algoritam« upotrebljavati u užem značenju: detaljno razrađen i opisan postupak spreman za prevođenje na neki programski jezik, a koji se temelji na nekoj poznatoj metodi — u ovom slučaju metodi Choleskog

gramski jezik po vlastitom izboru, pa smo odlučili da bar manje iskusnim korisnicima serviramo gotov program. Prvenstveno smo imali u vidu one geodetske radne organizacije koje nemaju vlastite računske centre i odgovarajući kadar nego im na raspolaganju stoje samo manja stolna računala. Kako se u tim slučajevima u SRH uglavnom radi o stolnim računalima proizvodnje Hewlett Packard (vidi [8]), a vjerujemo da je i u drugim republikama situacija slična, odlučili smo priložiti program za stolno računalo HP 9845. Riječ je o računalu koje prihvaća jedino programe u BASIC-u, pa je time BASIC bio jedini (i prema tome najbolji) izbor. Uostalom, program koji smo priložili može se dosta lako, doslovno prevesti na FORTRAN.

c) Zdravko Galić objavljuje u Geodetskom listu 10—12/1984 članak »O efikasnosti računarskih programa« ([4]) u kome pokušava polemizirati na slijedeći način:

1. izbor programskog jezika BASIC proglašava najvećom mogućom pogreškom (s ovom tvrdnjom — možda ne u tako oštrom obliku — možemo se u pojedinim konkretnim slučajevima *kada postoji mogućnost izbora* složiti).

2. istrgnutim citatom želi prikazati da je težište našeg prethodnog rada [6] bilo upravo na izboru programskog jezika BASIC. Tako kod Galića (vidi [4], str. 276) stoji:

»Upuštajući se međutim, na određen način u ocjenu efikasnosti programa, pomenuti autori pored ostalog kažu da '*cijena koju korisnik plaća prilikom izvođenja programa ovisi o trajanju izvođenja, a ovo pak o broju računskih i drugih operacija*', i nude program za rješavanje sistema linearnih jednačina napisan u BASIC-u.«

Međutim, u našem radu [6] nakon citirane rečenice ne nudimo program u BASIC-u kao spasonosno rješenje, nego provodimo analizu raznih faktora koji utječu na efikasnost, kao i problema koji se pojavljuju, pa slijedi opis predloženog algoritma, a zatim i opis programa. Međutim, sva razmatranja o programiranju koja se u tom dijelu našeg rada pojavljuju ne odnose se na određeni programski jezik, tj. vrijede kako kod programiranja u BASIC-u tako i kod programiranja u FORTRAN-u ili nekom drugom programskom jeziku. Tek na kraju naveli smo moguću realizaciju predloženog algoritma u BASIC-u. (Razlog za izbor računala i jezika već je objašnjen.)

3. Nakon što nas je tako proglasio za propagatore BASIC-a (a čiji izbor je najveća moguća greška?!), Galić zaobilazi činjenicu da je u našem radu [6] prvenstveno bio razmatran algoritam, te svodi diskusiju na problem izbora jezika i računala*, pokušavajući pritom stvoriti utisak da se time može postići da početničke greške pri sastavljanju algoritma postanu njegove prednosti. Kod analize efikasnosti radi usput još jednu grešku izjednačavajući efikasnost s brzinom izvođenja na *različitim* računalima, a zaboravljajući pritom da je CPU-vrijeme na bržim računalima skuplje.

---

* Što jest važan problem, ali je taj izbor obično ograničen vanjskim faktorima i mogućnostima, dok je izbor dobrog ili lošeg algoritma stvar slobodnog opredjeljenja.

Da li je Z. Galić (vidi [4], str. 285, tabela 2) svoj izbor računala napravio tako kako jest zato što su IBM 1130 (prastaro) i IBM 4331 (ne najmodernije) najbolji izbor za rješavanje danog problema, ili zato što su mu ona bila dostupna?!

2. USPOREDBA EFIKASNOSTI GALIĆEVOG I NAŠEG ALGORITMA PROVEDENA POMOĆU RAČUNALA

Dva algoritma se *direktno* mogu uspoređivati jedino teoretskim razmatranjem. Pomoću računala moguća je jedino usporedba njihovih implementacija. Da bi se mogli izvoditi bilo kakvi *razumni zaključci o kvaliteti samih algoritama* potrebno je oba prevesti na *isti programski jezik* i implementirati na *isto računalo*. Što su veće utvrđene razlike, to su zaključci o relativnoj kvaliteti samih algoritama pouzdaniji. Naime, ako je jedan algoritam bitno lošiji od drugog, on će se kao takav manifestirati u svakoj implementaciji, iako će možda njegove loše strane jednom doći do izražaja u većoj, a drugi put u manjoj mjeri.

a) Izbor računala za testiranje

Da bi zaključci izvedeni o samim algoritmima bili što pouzdaniji, poželjno je testiranje ponoviti na više računala. Budući da će se u praktičnoj primjeni algoritmi koristiti na računalima koja su danas u upotrebi (u Jugoslaviji), treba *po mogućnosti* odabrati tipične predstavnike takvih računala.

Testiranje smo proveli na slijedećim računalima:

1. UNIVAC 1100/42 Sveučilišnog računskog centra u Zagrebu. Računalo ima 2 centralna procesora, 256 kW (k-riječi) primarne i 524 kW proširene centralne memorije. Operacioni sistem je OS 1100/EXEC 8. Korišteni su jezični procesori FTN9R1 i BASIC9R1.
2. IBM 4341-2 Sveučilišnog računskog centra u Zagrebu. To je računalo IBM-ove serije System/370. Centralna memorija iznosi 8 Mb. Operacioni sistem OS/VS2 MVS podržava rad s upotrebom virtuelne memorije. Korišten je jezični procesor VS FORTRAN.
3. HP 1000 Instituta »Ruđer Bošković« u Zagrebu. Radi se o mini-računalu centralne memorije 512 kb. Operacioni sistem: RTE IVB. Korišten je jezični procesor FTN4.

b) Izbor programskog jezika za testiranje

Za testiranje algoritama trebalo bi odabrati one programske jezike koji bi mogli doći u obzir pri njihovoj praktičnoj primjeni, što dobrim dijelom nije samo stvar slobodne odluke, kao što bi se moglo pomisliti na temelju [4], nego ovisi o računalima i jezičnim procesorima koji stoje na raspolaganju potencijalnim korisnicima. FORTRAN i BASIC bi svakako mogli doći u obzir.

Budući da su i Galićev ([3]) i naš ([6]) algoritam veoma jednostavni i ne sadrže elemente bitno vezane za svojstva nekog programskog jezika, najvjerojatnije je irelevantno koji će se jezik odabrati. *Budući da je Galić za implementaciju svog algoritma sam odabrao FORTRAN i budući da u svom najnovijem radu pomalo pokušava stvoriti utisak kako je njegov algoritam baš prilagođen FORTRAN-u, smatrali smo da je za testiranje najkorektnije odabrati FORTRAN.*

Sada ćemo navesti programe koje smo uspoređivali i uvesti kratice koje će se upotrebljavati u daljnjem tekstu.

1. GALIĆ-C: Galićev FORTRAN-ski program doslovno preuzet iz [3];
2. GALIĆ-R: isto kao GALIĆ-C, ali je operiranje s kompleksnim brojevima zamijenjeno realnim;
3. JLP-F: Naš program objavljen u [6], doslovno preveden s BASIC-a na FORTRAN (jedini spomena vrijedan zahvat: formatizaciju ispisa u trokutasti oblik malo je kompliciranije postići u FORTRAN-u, nego u BASIC-u);
4. JLP-B: Naš program objavljen u [6], doslovno preuzet kao BASIC program.

U skladu s principima navedenim na početku poglavlja trebalo bi uspoređivati samo programe GALIĆ-C i JLP-F. Zašto smo uključili preostala dva?

Korištenje kompleksnih brojeva ne donosi ništa kad matrica sistema sadrži realne koeficijente* (vidi [1], str. 643—645, [7], str. 11, 12, 27), a smanjuje efikasnost programa. Vrlo jednostavnim zahvatom može se iz programa GALIĆ-C dobiti GALIĆ-R, koji je ipak manje neefikasan, pa smo smatrali da i njega treba uključiti u testiranje.

Kako na UNIVAC-u 1100/42 instaliranom na SRC-u postoje i FORTRAN i BASIC procesor, zanimalo nas je da li je izbor BASIC-a uvijek tako velika pogreška kako to naglašava Galić u [4], pa smo zato testirali i program JLP-B.

c) Kriteriji za donošenje zaključaka

Značajnim se mogu smatrati slijedeći kriteriji:

1. utrošeno CPU-vrijeme
2. veličina programa i korištena memorija
3. maksimalni broj jednadžbi koje je moguće riješiti
4. postignuta numerička točnost
5. cijena koju korisnik plaća

Treba uočiti da kriteriji 1—5 nisu nezavisni: 3—5 su zapravo posljedice 1 i 2. Također, 3—5 je ono što ustvari interesira korisnika prilikom donošenja odluke.

Kriteriji 1 i 2 gledani svaki od njih za sebe (bez onog drugog) ne omogućavaju donošenje zaključaka o efikasnosti. Direktno uspoređivanje CPU-vremena moguće je tek ako je korištena memorija jednaka; isto tako direktno uspoređivanje korištene memorije moguće je tek ako su CPU-vremena jednaka. Korisniku je u principu svejedno da li je neka cijena nastala kao posljedica većeg angažiranja memorije i manjeg CPU-vremena, ili obratno; interesantnije mu je *kolika je ta cijena.* Isto tako, manje će ga zanimati korištena memorija sama po sebi, a više posljedice toga. U ovom slučaju o korištenoj memoriji osim cijene ovisi i koliko je najviše jednadžbi moguće riješiti. Naravno, slično

*Kompleksni brojevi se koriste kad se radi o sistemu jednadžbi s kompleksnim koeficijentima. Međutim, program GALIĆ-C ne rješava takve slučajeve. Kod njega ulazni podaci moraju biti realni, a samo dio računskih operacija se vrši u kompleksnoj aritmetici.

je i s utrošenim CPU-vremenom — ako je npr. vrijeme trošeno na izvršavanje suvišnih operacija, korisnik će to, osim u cijeni, možda osjetiti i kroz manju točnost dobivenih rješenja.

Zato je prilikom uspoređivanja efikasnosti jednostavnije i sigurnije *osloniti se uglavnom na kriterije 3—5*, koji direktnije govore o efikasnosti, a u sebi sadrže posljedice 1 i 2.

3. PROVEDENA TESTIRANJA I REZULTATI

Testiranje smo proveli tako da smo na svakom pojedinom računalu pomoću svakog od nabrojenih programa rješavali svaki od odabranih sistema normalnih jednadžbi. Rješavali smo uglavnom dobro uvjetovane sisteme od 10, 30, 50, 80, 120, 200 i 300 normalnih jednadžbi*, s time da se na nekom od nabrojenih računala moglo stići do više, a na drugom do manje jednadžbi. Testiranje smo proveli u običnoj preciznosti. (Korištenje dvostruke preciznosti mijenja apsolutne iznose parametara, ali ne i kvalitativne međusobne odnose.)

3.1. *Utrošeno CPU-vrijeme, veličina programa i korištena memorija*

Kao što smo već razjasnili, ove kriterije ne treba gledati izolirano, svakog za sebe, i nepogodniji su za ocjenu efikasnosti nego preostala tri. Iako se *Galićev algoritam pokazao kao bitno lošiji od našeg i u slučaju kada izolirano promatramo samo CPU-vrijeme, veličinu programa ili korištenu memoriju*, u skladu s prethodno rečenim suzdržat ćemo se od opterećivanja čitaoca eksplicitnim navođenjem tih parametara za isprobane primjere Detaljnije ćemo razmotriti preostala tri kriterija iz kojih su posljedice koje snosi korisnik mnogo bolje vidljive.

3.2. *Maksimalni broj jednadžbi koje je moguće riješiti*

UNIVAC 1100/42 standardno prihvaća programe veličine do 64 kW. Uz posebnu opciju prilikom kompajliranja i kolektiranja moguće je dobiti apsolutni program veličine do 262 kW. Izvođenje ovakvih programa u principu se dozvoljava samo u uvjetima neopterećenog sistema (u konkretnom slučaju samo u noćnoj smjeni). Zbog primjene koncepta virtuelne memorije ograničenja su kod IBM 4341-2 najblaža, i mogu se koristiti programi do oko 9 Mb. Kao mini-sistem HP 1000 nije namijenjen izvođenju glomaznih programa, pa je maksimalna dopuštena veličina 64 kb. Kod sva tri računala moguća su i dalja povećanja programa uz dodatne zahvate i modifikacije, što ovdje nije razmatrano.

Maksimalni broj jednadžbi koje je moguće riješiti pojedinim programom uz navedena ograničenja prikazan je u tabeli 1. *Uočava se da Galićev algoritam omogućava rješavanje znatno manjeg broja normalnih jednadžbi nego naš* — naravno *na istom računalu uz ista ograničenja.*

---

* Programi su posebno kompajlirani i kolektirani s odgovarajućim dimenzijama polja za pojedini broj jednadžbi. Kod BASIC-a nema kolektiranja.

MAKSIMALNI BROJ JEDNADŽBI KOJE JE MOGUĆE RIJEŠITI					
RAČUNALO		UNIVAC 1100/42		IBM 4341-2	HP 1000
OGRANIČENJE		64kW[1]	262kW[2]	9MB[1]	64kB[1]
PROGRAM	GALIĆ-C	82	186	560	40
	GALIĆ-R	97	220	665	48
	JLP-F	306	696	2100	150
	JLP-B	87[3]	NT	NT	NT

Tabela 1 Maksimalni broj jednadžbi koje je pojedinim programom moguće riješiti

NAPOMENE: 1 — standardno ograničenje
2 — blaže ograničenje na računalu UNIVAC 1100/42
3 — za BASIC postoji dodatno ograničenje da dimenzija polja mora biti manja od 4000, pa se ne koristi svih 64 kW
NT — testiranje nije provedeno

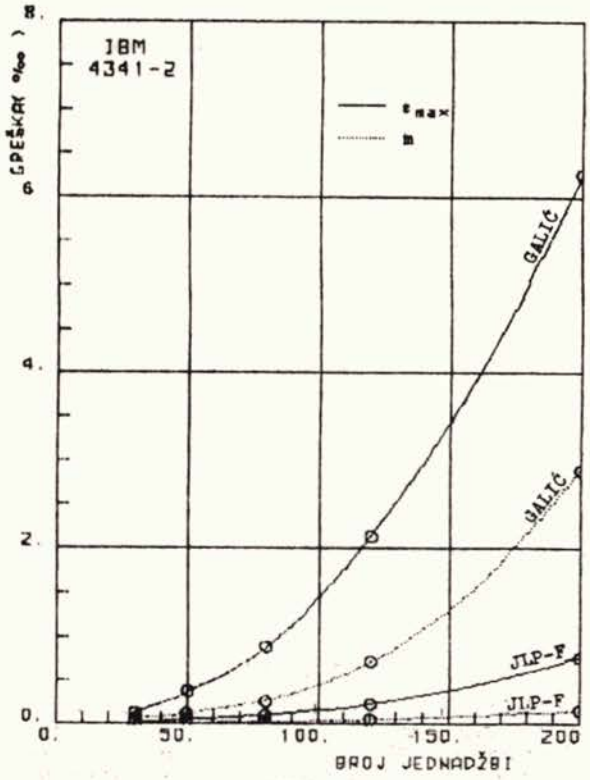
3.3. Točnost rješenja

Točnost rješenja je najteže testirati jer bi trebalo isprobati najrazličitije sisteme jednadžbi. Na svim računalima i svim isprobanim primjerima *naš algoritam je dao* zamjetno *točnija rješenja nego Galićev*. Kao ilustraciju, na slici 1 prilažemo jedan tipičan primjer izveden na računalu IBM 4341-2. U tom primjeru radi se o tako konstruiranom sistemu jednadžbi da su egzaktna rješenja bila unaprijed poznata. U pojedinom sistemu normalnih jednadžbi za sve koeficijente sistema, kao i za slobodne članove odabran je 1, osim dijagonalnih elemenata za koje je stavljeno 2. Interesantno je da relativne greške nisu tako male i da su približno 10 puta veće za rješenja dobivena Galićevim algoritmom nego za ona dobivena našim, iako se radi o sistemu jednadžbi vrlo jednostavne strukture koji je također i dobro uvjetovan.

3.4. Cijena koju korisnik plaća

Na slikama 2a i 2b prikazane su cijene koje korisnik plaća za rješavanje određenog broja normalnih jednadžbi na računalima UNIVAC 1100/42, odnosno IBM 4341-2. Na korištenom mini-sistemu HP 1000 nije definiran poseban obračun cijene, nego se ona obračunava proporcionalno utrošku CPU-vremena (posljedica činjenice da je centralna memorija i dopuštena veličina programa tog sistema za današnje pojmove mala). Prema tome slika 3 predstavlja istovremeno i CPU-vrijeme i cijenu koju korisnik plaća. Cijene prikazane na slikama 2a, 2b i 3 odnose se samo na izvođenje programa bez kompajliranja i kolektiranja*.

*Cijena kompajliranja i kolektiranja ne ovisi o broju jednadžbi koji će se rješavati (tj. o dimenziji) i praktički je jednaka za sva tri programa GALIĆ-C, GALIĆ-R i JLP-F (na istom računalu). Za malen broj jednadžbi ta cijena je znatno veća nego cijena samog izvođenja. Zato je u tom slučaju rasipnički svaki put iznova kompajlirati i kolektirati. Za velik broj jednadžbi cijena kompajliranja i kolektiranja postaje zanemariva prema cijeni izvođenja.

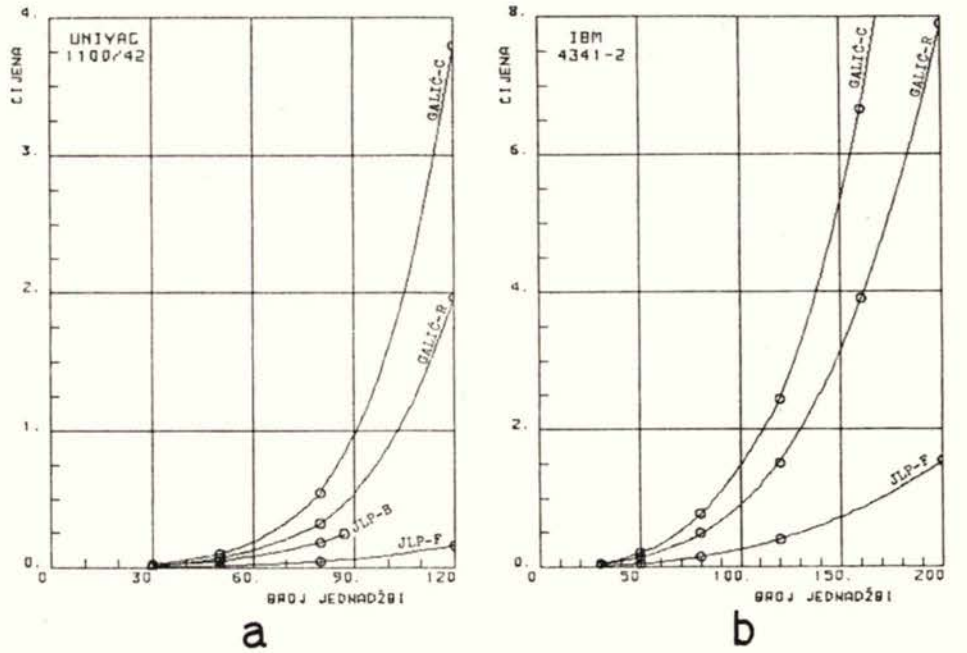


Slika 1

Točnost rješenja. Sve greške (razlike između egzaktnih i izračunatih rješenja) izražene su u promilima egzaktnog rješenja. Kružićima su naglašene vrijednosti koje su rezultat testa, dok je između njih učinjena interpolacija. ε_{max} su maksimalne, a m srednje kvadratne pogreške. Za programe GALIĆ-C i GALIĆ-R dobiju se isti grafovi, pa su na slici označeni samo sa GALIĆ.

U svim slučajevima radi se o ukupnim cijenama izvođenja programa (za FORTRAN, ranije pripremljenog load-modula) uključujući i ulaz i izlaz podataka, koji kod manjeg broja jednadžbi predstavljaju prilično značajan dio cijene. Ako se promatraju *samo cijene rada samih algoritama,* tj. samo rješavanja normalnih jednadžbi bez ulaza i izlaza, *relativni odnosi postaju još nepovoljniji po Galićev algoritam.* Međutim, pri primjeni razmatranih algoritama u praksi očito je da će morati postojati i ulaz i izlaz podataka, a pri testiranju smo željeli što vjernije oponašati stvarnu situaciju. *Sa slika 2a, 2b i 3 je vidljivo da je korištenje Galićevog algoritma mnogostruko skuplje nego korištenje našeg.*

Ako želimo napraviti međusobnu *usporedbu korištenih kompjuterskih sistema u odnosu na cijenu,* treba najprije uočiti da je razlika u veličini između HP 1000 i ostala dva tolika, da je ozbiljnija usporedba moguća samo između posljednja dva. Također, oni pripadaju istom računskom centru, pa je i međusobni odnos cijena usklađen tako da izvođenje prosječnog programa podjednako košta na oba. Treba primijetiti da je direktna usporedba cijena sigurno



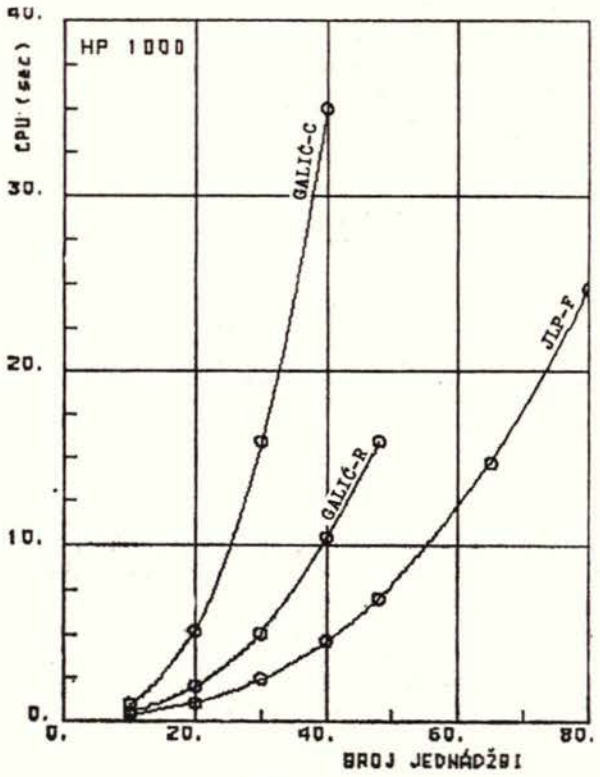
Slika 2

Cijena koju korisnik plaća (relativne jedinice) u ovisnosti o broju normalnih jednadžbi koje se rješavaju. Numeričke vrijednosti na osi cijena na slikama *a* i *b* su u istim jedinicama, a mjerilo slika je različito. Kružićima su naglašene eksperimentalno dobijene vrijednosti, a krivulje su dobivene interpolacijom.

barem malo nepravedna prema UNIVAC-u, koji daje veću točnost. Naime, u običnoj preciznosti IBM koristi 32-bitne riječi, a UNIVAC 36-bitne. Ipak, iz dobivenih rezultata može se zaključiti da je UNIVAC jeftiniji za manji broj jednadžbi, a IBM za veći (za program GALIĆ-C granica bi bila negdje oko N=100, a za JLP-F oko N=250). To je posljedica virtuelne memorije i dinamičke alokacije stranica programa kod IBM-a. Bitna razlika između sistema s virtuelnom memorijom (IBM) i bez nje (UNIVAC) uočava se i u omjeru cijena korištenja Galićevog algoritma u odnosu prema korištenju našeg. Iako je njegov uvijek višestruko skuplji, na IBM-u je taj omjer ipak značajno manji. IBM kao sistem s virtuelnom memorijom djelomično kompenzira rasipnički odnos prema memoriji, što je veoma značajno svojstvo takvih sistema.

4. BASIC KAO NAJLOŠIJI MOGUĆI IZBOR?!

Kao što smo već spomenuli, pitanje izbora programskog jezika ograničeno je izborom računala, a izbor računala nije samo stvar slobodne odluke. Na primjer, ako korisnik posjeduje vlastito računalo koje prihvaća samo programe u BASIC-u onda je za one probleme koji se na tom računalu mogu razumno riješiti, to računalo i BASIC bolji izbor nego korištenje npr. FORTRAN-a i tuđeg računala uz naplatu.



Slika 3

Cijena koju korisnik plaća. Kako se na ovom računalu cijena obračunava proporcionalno utrošenom CPU-vremenu, cijena je na slici izražena u vremenskim jedinicama. Kružićima su naglašene vrijednosti koje su rezultat testa, a između je izvršena interpolacija. Zbog objektivnih razloga testiranje je na ovom računalu urađeno sa samo djelomičnim ispisom izlaznih podataka na papir. Pri tome je ravnopravno postupljeno prema svim testiranim programima.

A što ako na raspolaganju stoji takvo računalo na kome postoji izbor između različitih programskih jezika? Ako se radi o izboru između FORTRAN-a i BASIC-a, svi imamo slične *predrasude* kao Galić — nećemo tražiti velika opravdanja da bismo odabrali FORTRAN.

Ipak, kad smo se upustili u ovakvo opsežno testiranje, željeli smo usput provjeriti: *da li je BASIC zaista uvijek loš?*

Jedno od loših svojstava BASIC-a vidljivo je iz tabele 1 : na korištenom sistemu UNIVAC 1100/42 ograničenja memorije pri korištenju BASIC-a mnogo su stroža nego pri korištenju FORTRAN-a. Pa ipak, FORTRAN-skim programom GALIĆ-C, doslovno onakvim kakav je objavljen u [3], moguće je na UNIVAC-u 1100/42 uz standardno ograničenje riješiti sistem od manje normalnih jednadžbi nego našim BASIC programom JLP-B, doslovno onakvim kakav je objavljen u [6] (tabela 1). I po cijeni JLP-B je bolji (jeftiniji) izbor na sistemu UNIVAC 1100/42 nego GALIĆ-C ili GALIĆ-R (slika 2a). Razlika je još i veća ako za FORTRAN-ske programe uračunamo kompajliranje i kolekti-

ranje. U takvoj situaciji, i do približno 70 normalnih jednadžbi, BASIC program je po cijeni jeftiniji (slika 2a), a po točnosti ravnopravan čak i s JLP-F, dakle ukupno gledano efikasniji i od njega.

Naglasimo da rezultati dobiveni na sistemu UNIVAC 1100/42 ne reprezentiraju globalni odnos FORTRAN-a i BASIC-a kao programskih jezika, nego se više odnose na spomenute procesore FTN9R1 i BASIC9R1. Naime, FORTRAN-procesori funkcioniraju na koliko-toliko sličan način, dok su među BASIC-procesorima razlike ogromne: postoje *interpreteri* (koje Galić u [4] na više mjesta apostrofira), ali i *kompajleri* (koje Galić u [4] ignorira), a možda i takvi koje je teško svrstati bilo u jednu, bilo u drugu grupu. Zato je moguće da bi na nekim računalima JLP-B prošao jako loše u usporedbi s JLP-F. Ali isto tako, iskustvo s UNIVAC-om pokazuje da ponekad BASIC može biti za tretiranu problematiku rješavanja normalnih jednadžbi u geodeziji vrlo dobar izbor, ravnopravan FORTRAN-u.

5. ZAKLJUČAK

1. Testiranje je potpuno potvrdilo ocjenu odnosa efikasnosti Galićevog i našeg algoritma koju smo dali u [6], jedino što se ovdje taj odnos ispoljio i kvantitatvno i to u vrlo drastičnom obliku. Iz priloženih slika i tabele je vidljivo da je *Galićev algoritam pokazao manju točnost rješenja nego naš, mogućnost rješavanja znatno manjeg broja normalnih jednadžbi na istom računalu uz isti režim rada računala, a ipak mnogo veću cijenu izvođenja na svim računalima uključenim u testiranje.*
2. *Galićev algoritam se na svim spomenutim računalima pokazao kao znatno manje efikasan od našeg, ali razlika ne dolazi uvijek jednako do izražaja.* Tako je npr. rješavanje 120 normalnih jednadžbi pomoću programa GALIĆ-C na UNIVAC-u 1100/42 približno *24 puta skuplje* nego pomoću JLP-F, dok je na IBM 4341-2 taj omjer *»svega« 6 puta.* (Za veći broj jednadžbi omjeri se povećavaju.)
3. Uočimo *»dobro držanje«* BASIC-a na UNIVAC-u 1100/42. Rješavanje 80 normalnih jednadžbi programom JLP-B je 4 puta skuplje nego programom JLP-F (ako ne uračunamo kompajliranje i kolektiranje ovog drugog), ali je još uvijek *3 puta jeftinije nego programom GALIĆ-C, što pokazuje da korištenje »boljeg« programskog jezika ne osigurava automatski efikasniji algoritam.*

6. NAPOMENE

a. Kao što ne propagiramo BASIC, isto tako ne propagiramo niti metodu Choleskog kao uvijek najpogodniju za rješavanje normalnih jednadžbi koje se pojavljuju u geodeziji. Ponekad bismo se radije opredijelili za jednu drugu metodu (vidi [5]). Ako se pak radi o normalnim jednadžbama, čija matrica sadrži relativno malo elemenata različitih od nule, onda je pogodniji nešto drugačiji pristup (vidi [2]). U ovom, kao i u prethodnom radu polazimo od toga da se radi o takvim slučajevima kada je izbor metode

Choleskog opravdan, a razmatra se pitanje kako tu metodu pretočiti u algoritam da bi njene dobre strane što više došle do izražaja. Iako su testiranja pokazala da nam je to uspjelo znatno bolje nego npr. Galiću, ne smatramo da se i našem programu, a možda čak i algoritmu, ne bi mogle uputiti razne primjedbe. Ipak, čini nam se da smo izbjegli barem osnovne pogreške, te da dalja poboljšanja ne bi *znatnije* utjecala na efikasnost. Mi sami nismo pretjerano zadovoljni onim što trenutno imamo i svjesni smo da u bliskoj budućnosti trebamo napisati slijedeće programe za računala koja koristimo:

— za rješavanje normalnih jednadžbi metodom Ckoleskog koji omogućuje istovremeno postojanje više stupaca slobodnih koeficijenata,

— modifikaciju postojećeg algoritma kod koje se skalarni produkti akumuliraju u dvostrukoj preciznosti, a sve ostale operacije rade u jednostrukoj, te vrši naknadno iterativno popravljanje rješenja u cilju postizanja mašinske točnosti rješenja.

b. Zainteresiranim čitaocima stavit ćemo na raspolaganje program JLP-F. Po želji ćemo također omogućiti uvid u detalje provedenog testiranja koji sadrže dodatne interesantne rezultate. Smatramo da oni nisu toliko zanimljivi širem krugu čitalaca, jer su uglavnom vezani za problematiku računarskih znanosti.

LITERATURA:

[1] Bunch J. R., Parlett B. N.: Direct methods for solving symmetric indefinite systems of linear equations, SIAM J. Numer. Anal., Vol. 8, No. 4, 1971, 639—655

[2] Cigrovski B., Lapaine M., Petrović S.: Operating with a sparse normal equations matrix, Zbornik radova naučnog skupa »Numerical methods and approximation theory«, Niš 1984, 67—72

[3] Galić Z.: Program za rješavanje sistema linearnih jednačina metodom Choleskog, Geodetski list, Zagreb 1983, 1—3, 31—37

[4] Galić Z.: O efikasnosti računarskih programa, Geodetski list, Zagreb 1984, 10—12, 275—287

[5] Jovičić D., Lapaine M., Petrović S.: Invertiranje matrice normalnih jednadžbi i stolno računalo, Zbornik radova Savjetovanja o automatizaciji u geodeziji, Bled 1983, 277—283

[6] Jovičić D., Lapaine M., Petrović S.: Rješavanje normalnih jednadžbi i metoda Choleskog, Geodetski list, Zagreb 1984, 1—3, 33—44

[7] Martin R. S., Peters G., Wilkinson J. H.: Symmetric decomposition of a positive definite matrix, u: Handbook for automatic computation, Volume II, J. H. Wilkinson-C. Reinsch: Linear algebra, Springer-Verlag, Berlin, Heidelberg, New York 1971, 9—30

[8] Solarić M., Visentin A.: Upotreba stolnih elektroničkih računala u geodetskoj praksi u SR Hrvatskoj, Zbornik radova IV susreta geodeta SRH, Osijek 1981, 1—11

SAŽETAK

U radu je izvršena usporedba efikasnosti dvaju algoritama za rješavanje normalnih jednadžbi koji se baziraju na metodi Choleskog. Oba algoritma bila su objavljena u ranijim brojevima Geodetskog lista. Autor jednog od njih

je Zdravko Galić iz Sarajeva, a drugog neki od autora ovog rada. Testiranje je provedeno korištenjem računala UNIVAC 1100/42, IBM 4341-2 i HP 1000. Oba algoritma implementirana su na sva tri računala u FORTRAN-u. Kao glavni kriteriji za uspoređivanje odabrani su: cijena koju korisnik plaća za rješavanje određenog broja jednadžbi, maksimalni broj jednadžbi koje je moguće riješiti i postignuta točnost. Algoritam autora ovog rada pokazao se na sva tri računala, po svakom od nabrojenih kriterija, kao mnogostruko bolji od Galićevog algoritma. Na računalu UNIVAC 1100/42 isprobana je također i implementacija algoritma autora ovog rada u BASIC-u. Pokazalo se da ponekad za tretiranu problematiku i BASIC može biti vrlo dobar izbor, ravnopravan FORTRAN-u.

ABSTRACT

In this paper a comparison has been made between efficiencies of two algorithms for solving normal equations. Both algorithms are based on the Cholesky's method and had been published in »Geodetski list«. The author of one of them was Zdravko Galić from Sarajevo, and the other was designed by some of the authors of the present paper. The test has been performed using computers UNIVAC 1100/42, IBM 4341-2 and HP 1000. Both algorithms have been implemented in FORTRAN to all the three computers. In comparing them the following main criteria have been used: the price paid by the user for solving a specific number of equations, the maximum number of equations which could be solved, and the achieved accuracy. The algorithm of the authors of the present paper proved to be far better than the other one, with respect to every of the adopted criteria and using any of the three computers. Additionally, the BASIC implementation of that better algorithm has been tested on UNIVAC 1100/42. It turned out that, for considered problems, BASIC can be a good choice under certain circumstances, not necessarily worse than FORTRAN.

Primljeno: 1985-04-12