ELEKTROTEHNIČKI FAKULTET UNIVERZITETA U BEOGRADU

IMPLEMENTACIJA ALGORITMA ZA DEBLOKADU VEZE U KLOSOVOM KOMUTATORU

–Diplomski rad –

Kandidat:

Marko Vuković 2006/0094

Mentor:

doc. dr Zoran Čiča

Beograd, Oktobar 2014.

SADRŽAJ

1. UVOD

Trokaskadni Klosovi komutatori predstavljaju popularna rešenja za komutaciju kako u komutaciji kola tako i u komutaciji paketa. Uslovno blokirajuće strukture predstavljaju ekonomično rešenje koje omogućava da se sve željene veze uspostave kroz komutator. Međutim, može doći do potrebe da se pojedine veze preurede da bi se neka nova željena veza kroz komutator mogla uspostaviti. U tu svrhu se koristi Paulov algoritam. Deblokada veze Paulovim algoritmom se koristi u mrežama baziranim na komutaciji kola (fiksna telefonska mreža, pojedine optičke mreže), dok u paketskim mrežama nema upotrebnu vrednost pošto se komutator rekonfiguriše u tzv. vremenskim slotovima koji su veoma mali u slučaju visokih protoka linkova, odnosno kapaciteta koji komutator mora da ostvari.

U ovoj tezi će biti realizovan Paulov algoritam u okviru Matlab softverskog paketa. Realizacija će podržavati proizvoljnu veličinu komutatora, i moći će da se koristi u nastavi kao demonstracija rada Paulovog algoritma.

Ostatak rada je organizovan na sledeći način. U poglavlju 2 biće izložen opis Paulovog algoritma i uslovi pod kojima ga je moguće koristiti tako da ima željene rezultate. U poglavlju 3 opisuje se sama implementacija algoritma u Matlabu. U poglavlju 4 su prikazani primeri koji su korišćeni za verifikaciju, ali isto tako i demonstriraju rad razvijene aplikacije. Na kraju je dat zaključak u kome su izložena završna razmatranja teze.

2. OPIS PAULOVOG ALGORITMA

2.1. KOMUTACIONA POLJA BAZIRANA NA KOMUTACIJI KOLA

Komutacija kola je povezivanje različitih vodova kroz telekomunikacionu mrežu u cilju povezivanja dva korisnika. Šta je konkretno vod zavisi od samog tipa (tehnologije) telekomunikacione mreže. Kada se jednom uspostavi veza između dva korisnika zauzeti resursi se koriste samo za njihovu komunikaciju dok se ta veza ne raskine. Povezivanje se vrši preko komutacionog polja koje povezuje određeni ulaz i izlaz jednosmernom vezom.

Postoji nekoliko karakteristika koje su bitne kod projektovanja komutacionog polja. To su dostupnost, skalabilnost, cena, kompleksnost, blokada itd.

Po pitanju blokade razlikujemo tri tipa komutacionih polja:

- Komutaciona polja sa potpunom blokadom
- Komutaciona polja sa uslovnom blokadom
- Komutaciona polja bez blokade

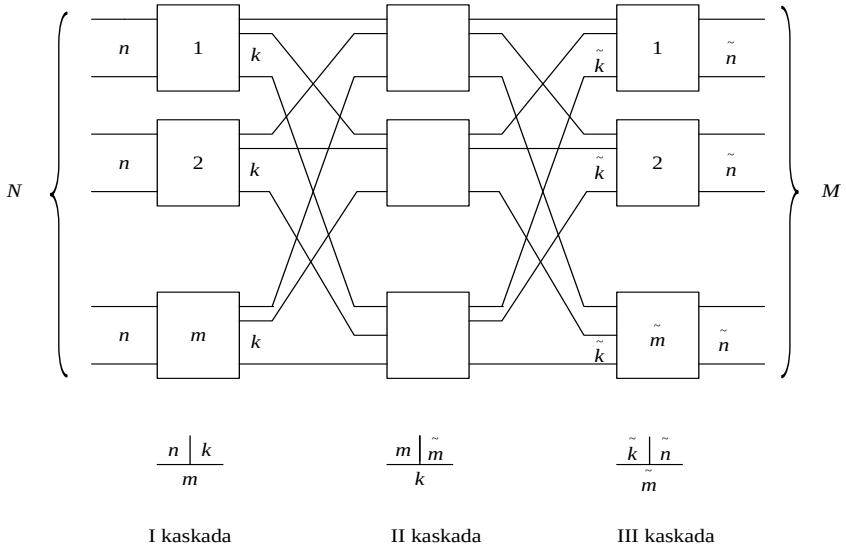
Kod komutacionih polja sa potpunom blokadom nemoguće je uspostaviti blokiranu vezu između određenog ulaza i izlaza bez obzira na preraspodelu već postojećih veza. U slučaju komutacionih polja sa uslovnom blokadom, uz preraspodelu postojećih veza moguće je deblokirati i uspostaviti traženu vezu. I konačno, kod komutacionih polja bez blokade moguće je uspostaviti bilo koju vezu između slobodnog ulaza i izlaza bez obzira na već postojeće veze (ali takva polja zahtevaju najviše fizičkih resursa).

Posebno bitno komutaciono polje u praksi koje je i korišćeno u okviru ovog rada je trokaskadno Klosovo polje.

2.2. TROKASKADNO KLOSOVO KOMUTACIONO POLJE

Trokaskadno komutaciono polje, kao što se vidi na slici 2.1.1 se sastoji od tri kaskade komutatora kod kojih je korišćeno Klosovo povezivanje. Klosovo povezivanje podrazumeva: povezivanje *i*-tog izlaza komutatora posmatrane kaskade sa *i*-tim komutatorom sledeće kaskade. Oznaka za Klosov komutator je C(n,m,k,m_1,n_1) gde je n oznaka za broj u jedan komutator prve kaskade, m oznaka za broj komutatora u prvoj kaskadi, k broj komutatora u drugoj kaskadi, $m_1$broj komutatora u trećoj kaskadi i n_1 broj izlaza iz komutatora treće kaskade. U implementaciji algoritma iz ovog rada se koristi simetrično Klosovo trokaskadno polje sa oznakom C(n,m,k,m_1,n_1).

Kao što je prethodno navedeno, postoje polja sa potpunom blokadom, uslovnom blokadom i bez blokade. Polja bez blokade su najbolja jer nije potrebno vršiti preraspodelu veze, ali nije optimalno rešenje jer koristi veliki broj komutatora čime je i cena veća. Korišćenjem polja sa uslovnom blokadom je moguće postići potpunu dostupnost uz dosta nižu cenu u odnosu na polja bez blokade.



Slika 2.1.1 Simetrično trokaskadno komutaciono polje [1]

Određivanje kakvo je Klosovo trokaskadno polje po pitanju blokade se vrši pomoću Klosovog uslova. Klosov uslov glasi:

$$k \geq n+n_1-1 \tag{2.1.1}$$

Ako je ispunjen Klosov uslov komutaciono polje je neblokirajuće, u suprotnom je blokirajuće. U slučaju da je ispunjena jednakost onda kažemo da je u pitanju Klosovo rešenje.

Slepianov uslov predstavlja uslov da bi trokaskadno komutaciono Klosovo polje bilo uslovno blokirajuće i glasi:

$$k \geq \max(n,n_1) \tag{2.1.2}$$

Ako je ispunjen Slepianov uslov onda je trokaskadno Klosovo polje uslovno blokirajuće i tada je u slučaju blokade maksimalan broj preuređenja već postojećih veza $m+m_1-2$.

Paulov rezultat daje korekciju Slepianovog uslova i kaže da je minimalan broj preuređenja veza u cilju deblokiranja komutacionog polja manja od $\min(m,m_1)$. Paul je razvio Paulov algoritam za preuređivanje postojećih veza da bi se odblokirala blokirana veza.

Implementacija Paulovog algoritma je predstavljena u ovom radu pa ga je potrebno detaljno opisati.

2.3. PAULOV ALGORITAM

Paul je u svom radu kojim je predstavio svoj algoritam pokazao da je moguće deblokirati komutaciono polje sa mnogo manjim brojem preuređenja u odnosu na do tada važeći Slepianov uslov. On je predstavio simetrično trokaskadno Klosovo komutaciono polje C(n,m,k,m,n) Paulovom matricom. Paulova matrica je imala m vrsta koje predstavljaju komutatore prve kaskade i m kolona koje predstavljaju komutatore treće kaskade. U matricu se upisuju brojevi od 1 do k koji predstavljaju komutatore druge kaskade i to tako da se u preseku vrste i kolone koji odgovaraju komutatorima koji se koriste u vezi upisuje broj koji odgovara broju drugog komutatora. Za neku vezu kažemo da je blokirana kada unija skupova odgovarajuće vrste i kolone sadrži sve komutatore druge kaskade.

U slučaju da je veza blokirana primenjuje se Paulov algoritam.

Prvo je potrebno naći razlike skupova oznaka komutatora druge kaskade iz odgovarajuće vrste i kolone kojem pripada polje gde se nalazi veza koja je blokirana. Drugi korak je zatim formiranje lanaca komutatora druge kaskade. Lanac se sastoji od oznaka komutatora koji su dobijeni kao razlika vrste i kolone i razlika kolone i vrste. Svaki lanac ima svog alternativnog para gde je samo redosled obrnut. Ukupan broj lanaca je $2xy$ gde je x broj elemenata skupa dobijenog razlikom vrste i kolone, a y broj elemenata skupa dobijenog kao razlika kolone i vrste. Ukoliko se dobije kao razlika prazan skup tada je nemoguće formirati lanac tj. deblokirati vezu.

Kada se lanci formiraju onda se vrši njegova inverzija tj. od lanca A-B-A... se dobija B-A-B... Inverzijom lanca se vrši deblokada veze i moguće ju je uspostaviti. Teoretski najmanja dužina lanca je 1. Uvek je bolje u praksi uzeti što kraći lanac jer se tako veza deblokira sa manjim brojem preuređivanja postojećih veza.

Algoritam preuređivanja veza ima dve varijante koje se razlikuju po vremenu aktivacije:

- Pre uspostave svake nove veze se aktivira algoritam preuređivanja
- Nakon raskida svake veze se vrši preuređivanje veza da bi bilo što lakše primiti nove veze

Takođe, koristi se Benešovo pravilo pri kreiranju veza kroz komutaciono polje. Ovo pravilo se zasniva u tome da se sistematski iskorišćavaju komutatori srednje kaskade. Prvo se maksimalno koristi prvi komutator iz srednje kaskade, pa onda drugi i tako redom. Na taj način ostaje najviše mogućih slobodnih veza.

3. OPIS IMPLEMENTACIJE

Za implementaciju je korišćen programski paket Matlab. Ceo algoritam je urađen kroz četiri funkcije koje izvršavaju određene delove algoritma.

Prva funkcija je *kreiranje.m* koja kreira Paulovu matricu iz ulaznih argumenata *n, m, k* koje označavaju redom broj ulaza u komutatore prve kaskade, broj komutatora u prvoj kaskadi i broj komutatora u drugoj kaskadi. Pošto je polje simetrično *m* i *n* označavaju takođe i broj komutatora u trećoj kaskadi i broj izlaza iz komutatora treće kaskade. Sama matrica se pravi funkcijom *cell* koja pravi matricu sa ćelijama koja omogućava da se kao elementi matrice pojavljuju skupovi od više elemenata.

Sledeća funkcija je *dodavanje.m* koja se koristi za dodavanje veza u matricu. Kao ulazne argumente koristi Paulovu matricu koja je kreirana u prethodno opisanoj funkciji kreiranja, *broj_ulaza*, *broj_izlaza* i *broj_2kom*, kao i *n*, *m* i *k* koje je koristila funkcija kreiranja Paulove matrice. Argumenti *broj_ulaza*, *broj_izlaza* i *broj_2kom* specificiraju broj ulaza i izlaza koji se povezuju, kao i komutator srednje kaskade koji se koristi za vezu.

Prvo je potrebno odrediti broj komutatora prve i treće matrice kako bi znali u koju ćeliju je potrebno upisati komutator srednje kaskade preko kog se veza ostvaruje. Ti brojevi se dobijaju deljenjem broja ulaza ili izlaza sa ukupnim brojem komutatora prve odnosno treće kaskade i zaokruživanjem na veću vrednost. Dalje na osnovu *broj_2kom* određujemo po kom principu će se dodeljivati komutatori druge kaskade: ako je *broj_2kom* jednak *k*+1 onda se signalizira funkciji da se koristi Benešovo pravilo koje je opisano u prethodnom tekstu. Ako je *broj_2kom* veći od *k+1* dodavanje veze se neće izvršiti, a ako je manje od *k*+1 onda time signaliziramo funkciji dodavanja koji tačno komutator druge kaskade želimo da koristimo. Nakon odabira načina dodeljivanja vrši se provera blokiranosti veze u oba slučaja na sličan način, korišćenjem funkcije *ismember* odnosno *~ismember*. Ako se utvrdi da je veza blokirana vrši se funkcija *deblokiranje.m* pa se po izvršavanju deblokiranja upisuje odgovarajuća vrednost u polje matrice koje je već ranije određeno.

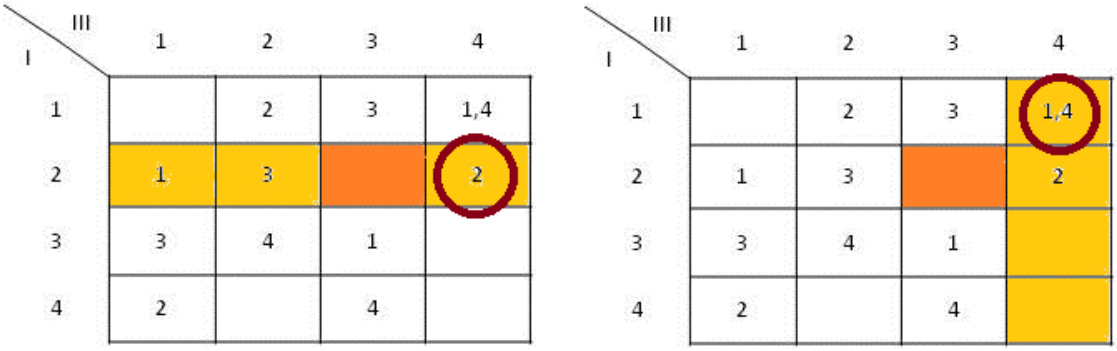
Funkcija koja vrši najveći deo posla i koja je centralni deo ove implementacije je *deblokada.m*. Funkcija *deblokada.m* za ulazne parametre ima promenjive *Paullova_matrica*, *br_ulaza*, *br_izlaza*, *m*, *n*, *k* koje imaju isto značenje kao u prethodnim funkcijama. Izlazni argument je deblokirana matrica.

Prvo je, kao u prethodnoj funkciji, potrebno odrediti vrstu i kolonu da bi se našla ćelija matrice koju je potrebno deblokirati. Zatim se funkcijom *setdiff* traže dve razlike koje su potrebne. Nakon toga se *for* petljama prolazi kroz svaku kombinaciju od dva elementa dobijena razlikom skupova i traže se lanci podfunkcijom *nadji_lanac.m*. U podfunkciji *nadji_lanac.m* se počevši od polja koje je potrebno deblokirati traže elementi lanca. U skladu sa teorijom svaka kombinacija od dva broja koji označavaju komutatore druge kaskade, a koji su dobijeni operacijom razlika skupova u prethodnom delu funkcije *deblokada.m* formira dva lanca. Ta dva lanca se razlikuju po redosledu brojeva koji označavaju komutatore druge kaskade.

Pošto postoje dva lanca postoje i dva dela potprograma kojima odgovaraju dva slučaja formiranja lanca. Kada je vrednost promenjive *slucaj* 1 to znači da se, u skladu sa teorijom, prvo traži broj koje pravi konflikt u vrsti u kojoj se nalazi polje koje je potrebno deblokirati. Kada se

nađe taj broj potrebno je naći broj koji označava u kojoj se koloni nalazi. Kada se nađe broj kolone, promenjiva *smer*, koja je jednaka 1 ako se prvo traži u vrsti pa u koloni, menja vrednost u 2. Pošto je vrednost promenjive smer jednaka 2 to znači da će se u sledećem prolazu kroz petlju sledeći element u lancu tražiti po koloni. Na sličan način se nastavlja dok se naiđe na vrstu/kolonu u kojoj se ne nalazi sledeći element u nizu.

Na slici 3.1. je prikazana ilustracija algoritma pretrage prvo vrste a zatim i kolone matrice. Na levom delu slike 3.1 je osenčena vrsta u kojoj se nalazi element matrice (2,3) koji označava vezu koju treba odblokirati. Ova slika odgovara situaciji kada je promenjiva *smer* jednaka 1. Zaokruženi broj predstavlja element vrste 2 matrice koji je dobijen kao razlika 2. vrste i 3. kolone i koji označava prvi element lanca (2). U promenjivu *sledeci* u podfunkcije *nadji_lanac.m* se smešta vrednost 4 koja predstavlja broj kolone u kojoj se nalazi prvi element lanca. Na desnoj strani slike se vidi sledeći korak funkcije pri kome je vrednost promenjive *smer* jednaka 2. Sada se pretražuju svi elementi 4. kolone tražeći sledeći (drugi) element lanca dobijen kao razlika 3. kolone i 2. vrste. Ta razlika je 4 i ta vrednost je i zaokružena na desnoj strani slike 3.1. Kada je nađen drugi element onda se vrednost promenjive *sledeci* postavlja na 1 (što je broj vrste u kojoj treba tražiti sledeći element lanca) i ponavlja se postupak dok se ne dođe do kraja lanca.



Slika 3.1 Ilustracija traženja elementa lanca

Tada se komandom *break* izlazi iz petlje i pamti se promenjiva *brojac* koja broji elemente u lancu i promenjive prvi i drugi koje pokazuju prvi i drugi element u lancu. Time imamo sve podatke potrebne za određivanje lanca.

Funkcija *deblokada.m* uzima te podatke i, pomoću promenjive *brojac* gleda najbolju opciju odnosno koji je lanac najkraći, samim ti je on i optimalan. Kada se najbolji slučaj pronađe i odredi onda se izvršava potrebna modifikacija matrice tako da novodobijena matrica obezbeđuje da se uspostavi veza između dva korisnika sa datim brojevima ulaza/izlaza. Proces modifikacije je veoma sličan podfunkciji *nadji_lanac.m* uz razliku da se vrši prolaz kroz samo jedan lanac (najkraći lanac) i pri tome se dodatno vrše preuređivanja veza (zamena komutatora srednje kaskade u lancu).

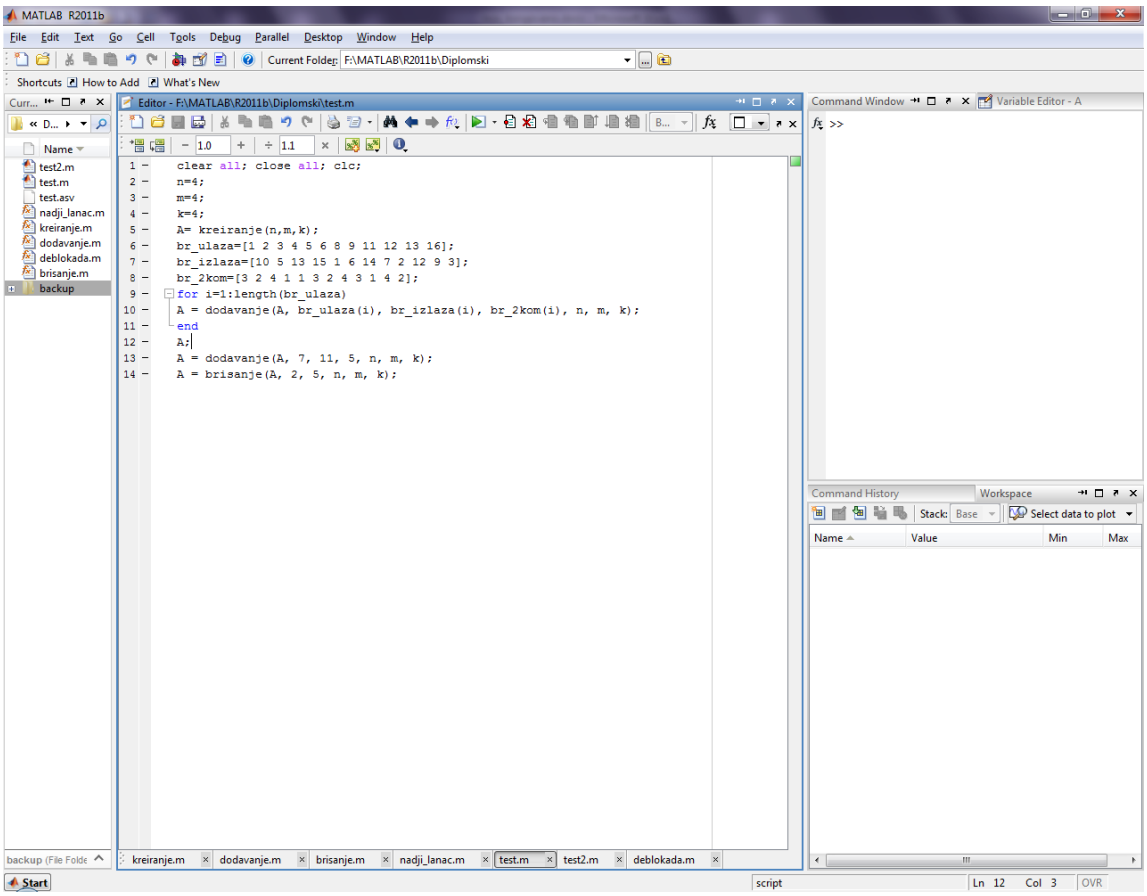
Poslednja funkcija je funkcija *brisanje.m* kojom se briše veza iz matrice. Veza se iz matrice briše tako što se izbriše broj komutatora u preseku vrste i kolone koji označavaju o kojoj se vezi radi. Umesto broja se stavi prazan skup ili se izbriše jedan od komutatora iz skupa u toj ćeliji.

4. PRIMERI RADA

Kao primere rada uzećemo dva primera simetričnog trokaskadnog Klosovog komutatora iz materijala sa predmeta Komutacione sistemi [2].

Prvi primer: Dato je trokaskadno komutaciono polje (KP) $C_3(4,4,4,4,4)$. Za dato KP su uspostavljene sledeće veze: (1,C,10), (2,B,5), (3,D,13), (4,A,15), (5,A,1), (6,C,6), (8,B,14), (9,D,7), (11,C,2), (12,A,12), (13,D,9), (16,B,3). Koristeći Paulov algoritam izvršićemo deblokadu veze (7,11).

Da bi se demonstrirao rad algoritma napravljena je skripta *test.m* koja izvršava sve potrebne funkcije navedene u prethodnom delu rada.



Slika 4.1. Prikaz skripte *test.m*

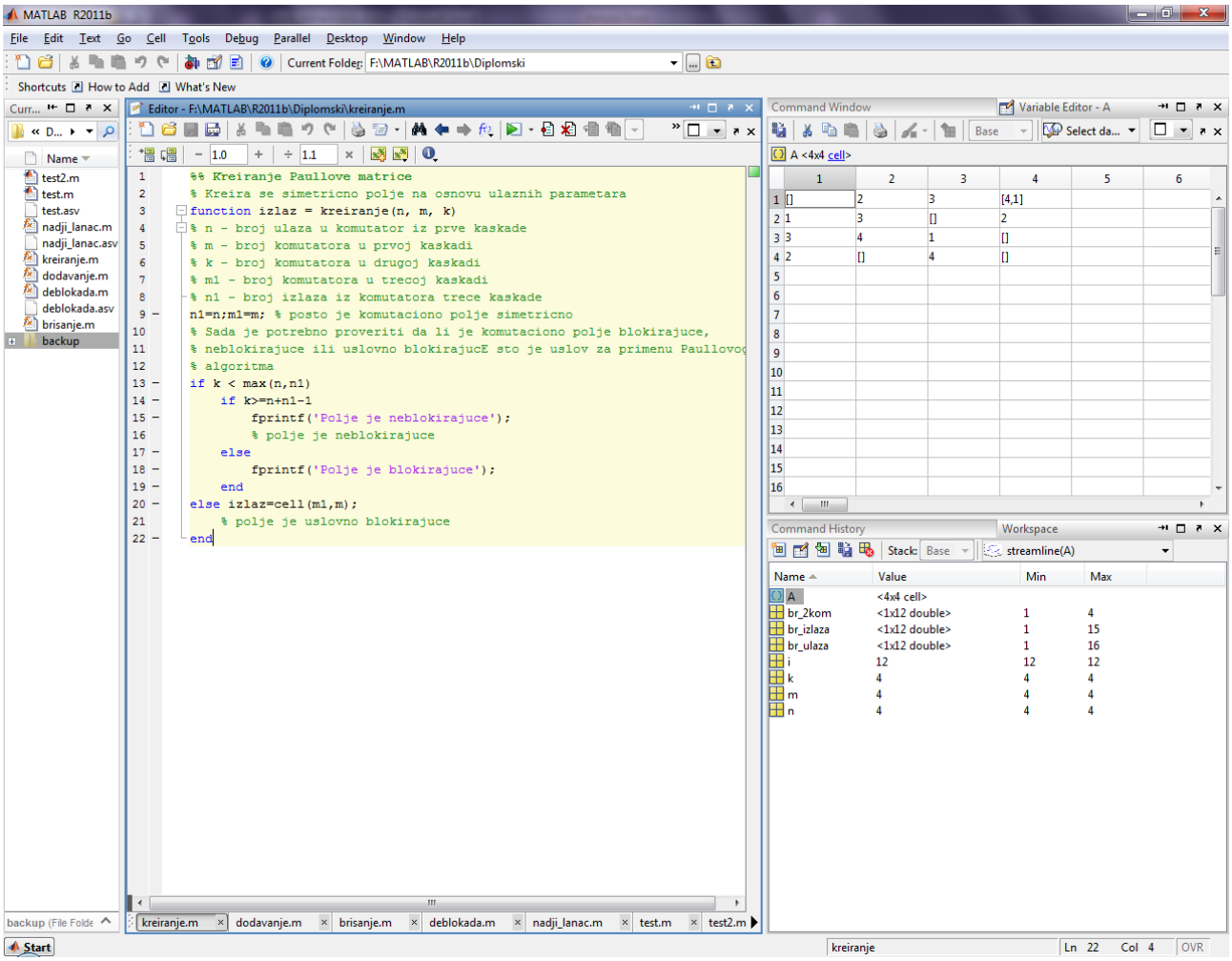
U ovoj skripti je jednostavno izvršeno pozivanje svake od funkcija prethodno opisanih u poglavlju 3 radi demonstriranja njihovog rada.

Prvo se definišu promenjive n, m i k koje određuju broj komutatora i izlaza/ulaza komutacionog polja. Zatim se vrši pozivanje funkcije *kreiranje.m* linijom:

A= kreiranje(n,m,k);

koja formira praznu Paulovu matricu dimenzija mxm.

Na slici 4.2 se može videti prikaz funkcije $kreiranje.m$ sa formiranom praznom Paullovom matricom.

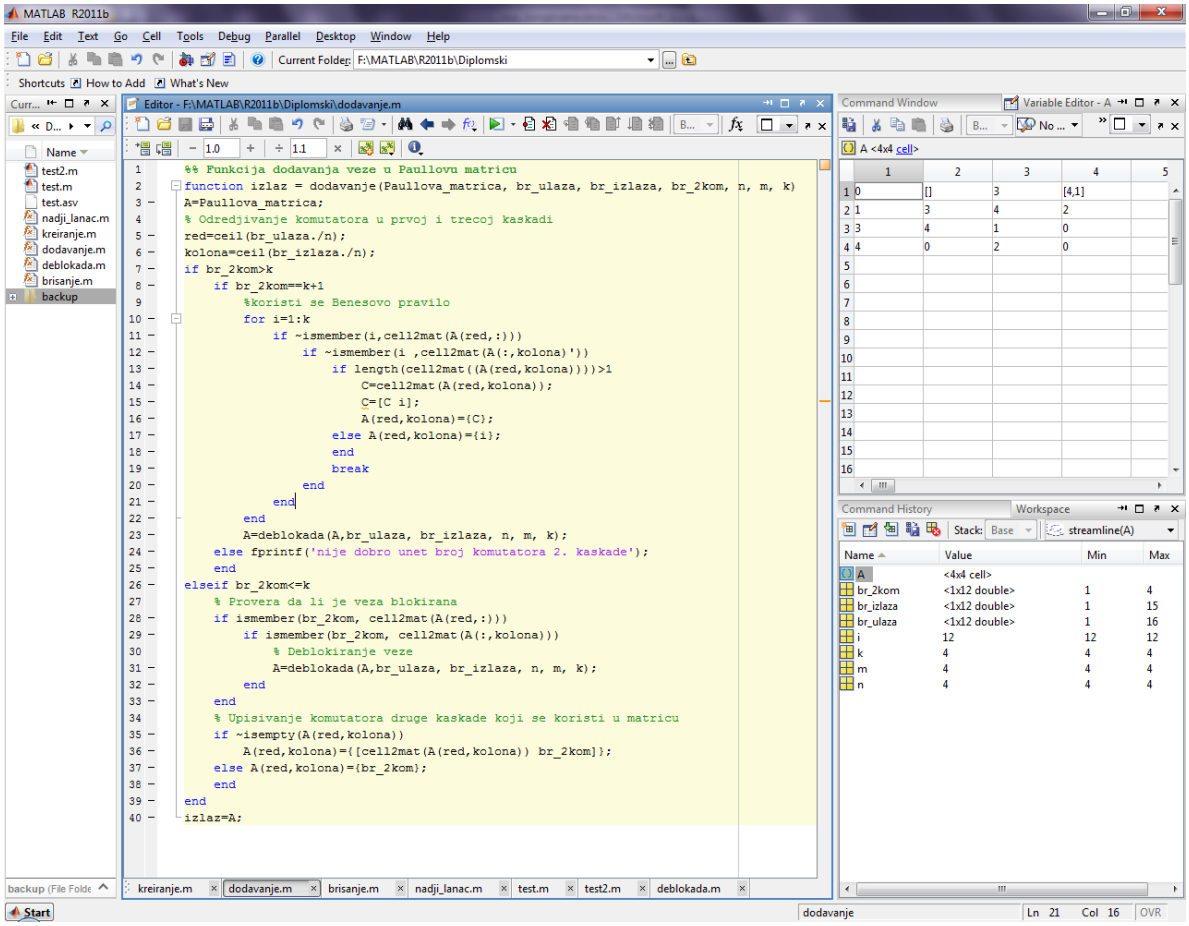


Slika 4.2. Prikaz funkcije *kreiranje.m*

Dalje u skripti se učitavaju vrednosti promenljivih br_ulaza, br_izlaza i $broj\_2kom$ za koje smo u prethodnom tekstu objasnili značenje. Ove promenljive su ubačene kao nizovi gde isti nizovi označavaju jednu konekciju i ubacuju se u matricu funkcijom $dodavanje.m$ uz pomoć *for* petlje.

Na slici 4.3 se vidi prikaz funkcije $dodavanje.m$ sa Paullovom matricom u koju su ubačene sve veze iz primera. Takođe, može se primetiti da je umesto slovnih oznaka u tekstu zadatka za komutatore druge kaskade (A,B,C,D) korišćena numerička oznaka za iste (1,2,3,4). Sadržaj Paulove matrice nakon dodavanja svih veza je prikazan na slici 4.4.

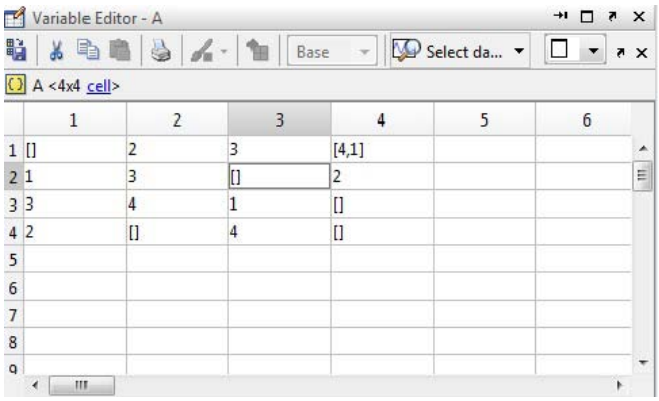
Sama funkcija daje mogućnost da se učitavanje vrši po Benešovom pravilu iako to nije specificirano u našem primeru. Ako korisnik želi da koristi Benešovo pravilo onda je potrebno uneti vrednost $k+1$ za promenjivu $broj\_2kom$ pri pozivanju funkcije $dodavanje.m$.



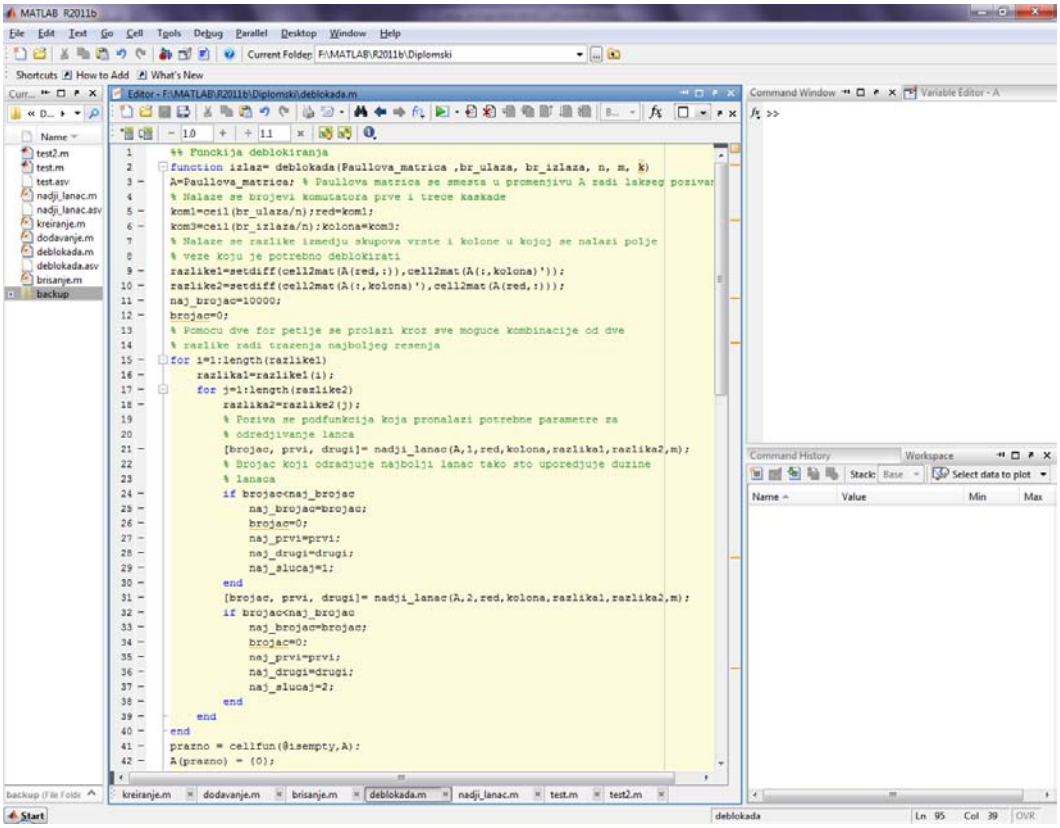
Slika 4.3. Prikaz funkcije *dodavanje.m*

Dalje, idući kroz skriptu *test.m* se demonstrira rad funkcije *deblokada.m*. Po tekstu iz primera potrebno je deblokirati vezu (7,11).

U prvom delu funkcije potrebno je prvo odrediti sve promenjive potrebne pri izvršavanju same funkcije. To su *kom1* koji označava broj komutatora prve kaskade, *kom3* - broj komutatora treće kaskade, *razlike1* i *razlike2* koji označavaju razliku skupa brojeva koji predstavljaju komutatore druge kaskade koji se nalaze u vrsti i brojeva u koloni (*razlike1*) i razliku skupa brojeva u koloni u odnosu na one u vrsti (*razlike2*).



Slika 4.4. Prikaz tabele pošto su dodate veze

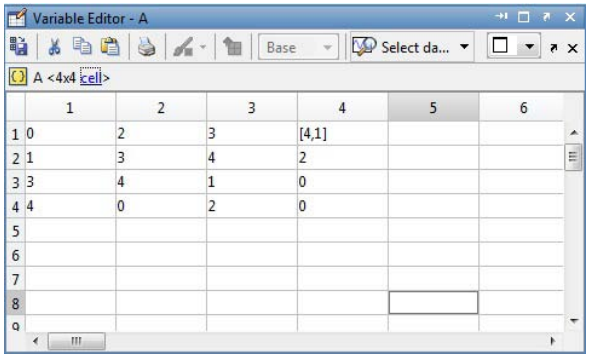


```
%% Funckija deblokiranja
function izlaz= deblokada(Paullova_matrica ,br_ulaza, br_izlaza, n, m, k)
A=Paullova_matrica; % Paullova matrica se smesta u promenjivu A radi lakseg pozivan
% Nalaze se brojevi komutatora prve i trece kaskade
kom1=ceil(br_ulaza/n);red=kom1;
kom3=ceil(br_izlaza/n);kolona=kom3;
% Nalaze se razlike izmedju skupova vrste i kolone u kojoj se nalazi polje
% veze koju je potrebno deblokirati
razlike1=setdiff(cell2mat(A(red,:)),cell2mat(A(:,kolona)'));
razlike2=setdiff(cell2mat(A(:,kolona)'),cell2mat(A(red,:)));
naj_brojac=10000;
brojac=0;
% Pomocu dve for petlje se prolazi kroz sve moguce kombinacije od dve
% razlike radi trazenja najboljeg resenja
for i=1:length(razlike1)
    razlika1=razlike1(i);
    for j=1:length(razlike2)
        razlika2=razlike2(j);
        % Poziva se podfunkcija koja pronalazi potrebne parametre za
        % odredjivanje lanca
        [brojac, prvi, drugi]= nadji_lanac(A,1,red,kolona,razlika1,razlika2,m);
        % Brojac koji odradjuje najbolji lanac tako sto uporedjuje duzine
        % lanaca
        if brojac<naj_brojac
            naj_brojac=brojac;
            brojac=0;
            naj_prvi=prvi;
            naj_drugi=drugi;
            naj_slucaj=1;
        end
        [brojac, prvi, drugi]= nadji_lanac(A,2,red,kolona,razlika1,razlika2,m);
        if brojac<naj_brojac
            naj_brojac=brojac;
            brojac=0;
            naj_prvi=prvi;
            naj_drugi=drugi;
            naj_slucaj=2;
        end
    end
end
prazno = cellfun(@isempty,A);
A(prazno) = {0};
```

Slika 4.5. Prikaz prvog dela funkcije *deblokada.m*

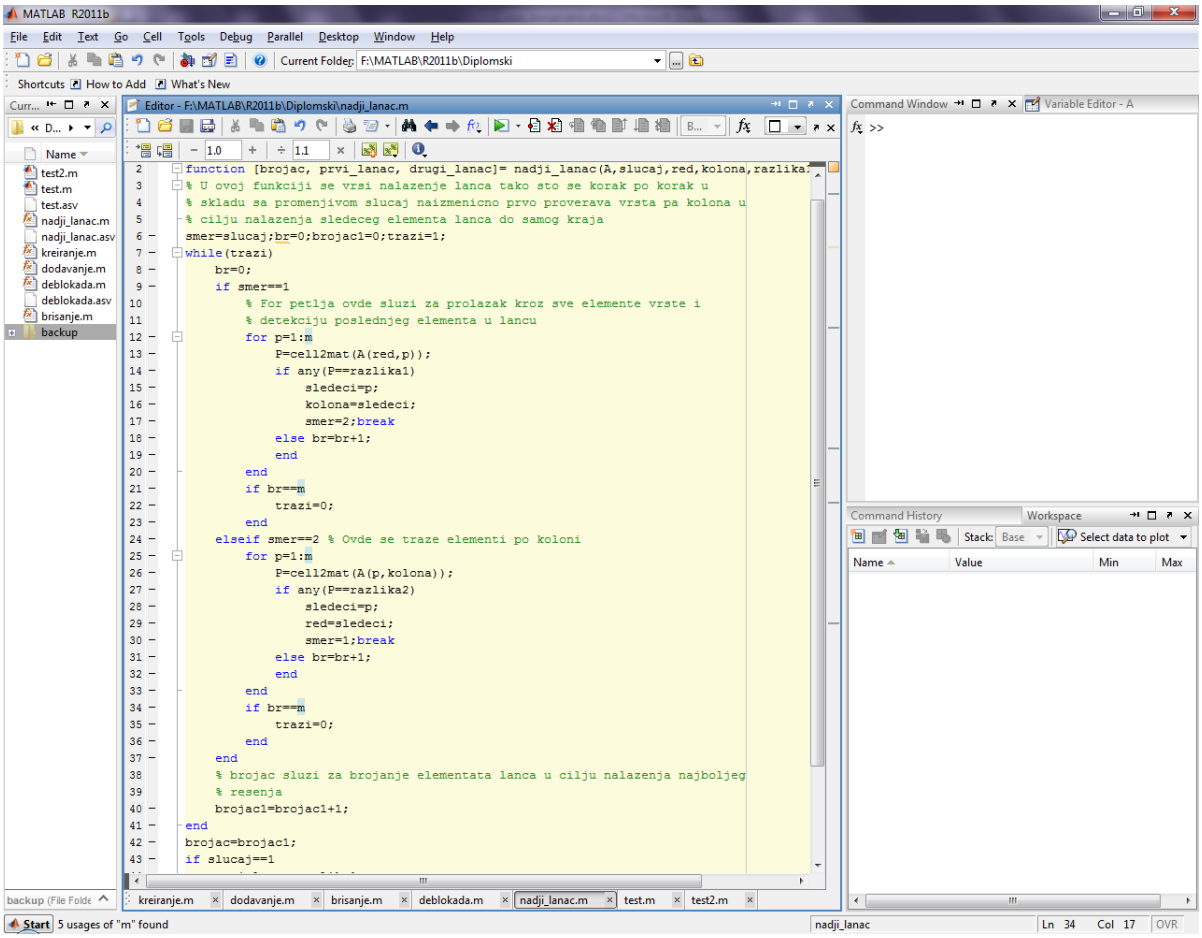
Kada se odrede ove vrednosti imamo sve potrebne podatke za podfunkciju *nadji_lanac.m* (slika 4.6.) koja određuje, u odnosu na početne uslove, sve potrebne podatke da bi lanac mogao biti tačno rekonstruisan. To su *brojac* koji određuje koliko elemenata ima u lancu, *prvi* koji određuje prvi element lanca i *drugi* koji određuje drugi element lanca. Nakon toga vršimo proveru da li je dobijeni lanac najbolji u odnosu na sve prethodno pronađene tako što se upoređuje sa najboljim do sada po pitanju dužine lanca tj. ako je promenjiva *brojac* manja od do sada najmanje onda se nova vrednost pamti. Naravno, pamte se i promenljive *prvi* i *drugi* zbog dalje upotrebe u funkciji.

U našem primeru vrednosti *razlika1* i *razlika2* su 2 i 4, respektivno pa se formiraju lanci 2-4-2-4 (ako se krene pretraga prvo po vrsti) i 4-2 (ako se krene prvo po koloni). Jasno je da je povoljniji drugi lanac što se i vidi po završetku deblokade na slici 4.6. Takođe se vidi da je po deblokiranju matrice oslobođeno mesto za vezu (7,11) na mestu (2,3) gde je upisana vrednost komutatora druge kaskade koji se koristi (4).



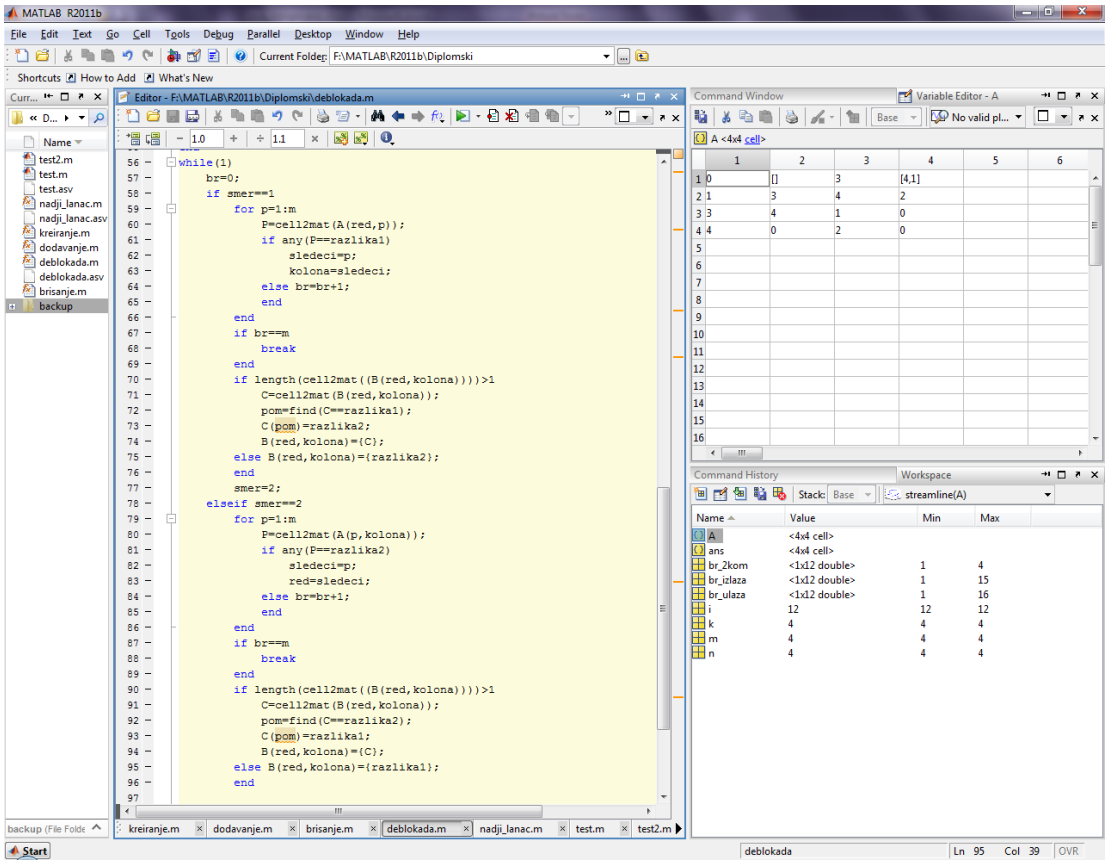
	1	2	3	4	5	6
1	0	2	3	[4,1]		
2	1	3	4	2		
3	3	4	1	0		
4	4	0	2	0		
5						
6						
7						
8						
9						

Slika 4.6. Prikaz matrice nakon izvršene deblokade



Slika 4.7. Prikaz podfunkcije *nadji_lanac.m*

U drugom delu funkcije *deblokada.m* se prolazi još jednom kroz sličan deo koda kao u podfunkciji *nadji_lanac.m* sa tim što se menjaju vrednosti u matrici da odgovaraju novom, deblokiranom stanju. Zatim se upisuje vrednost koja označava da je konekcija koja je trebala da se uspostavi uspostavljena. Time je dobijena konačna matrica sa novom vezom (7,11). Konačna vrednost matrice je, uz odgovarajući deo funkcije *deblokada.m* prikazana na slici 4.8, odnosno na slici 4.6 gde je dat krupniji prikaz sadržaja matrice radi bolje preglednosti.



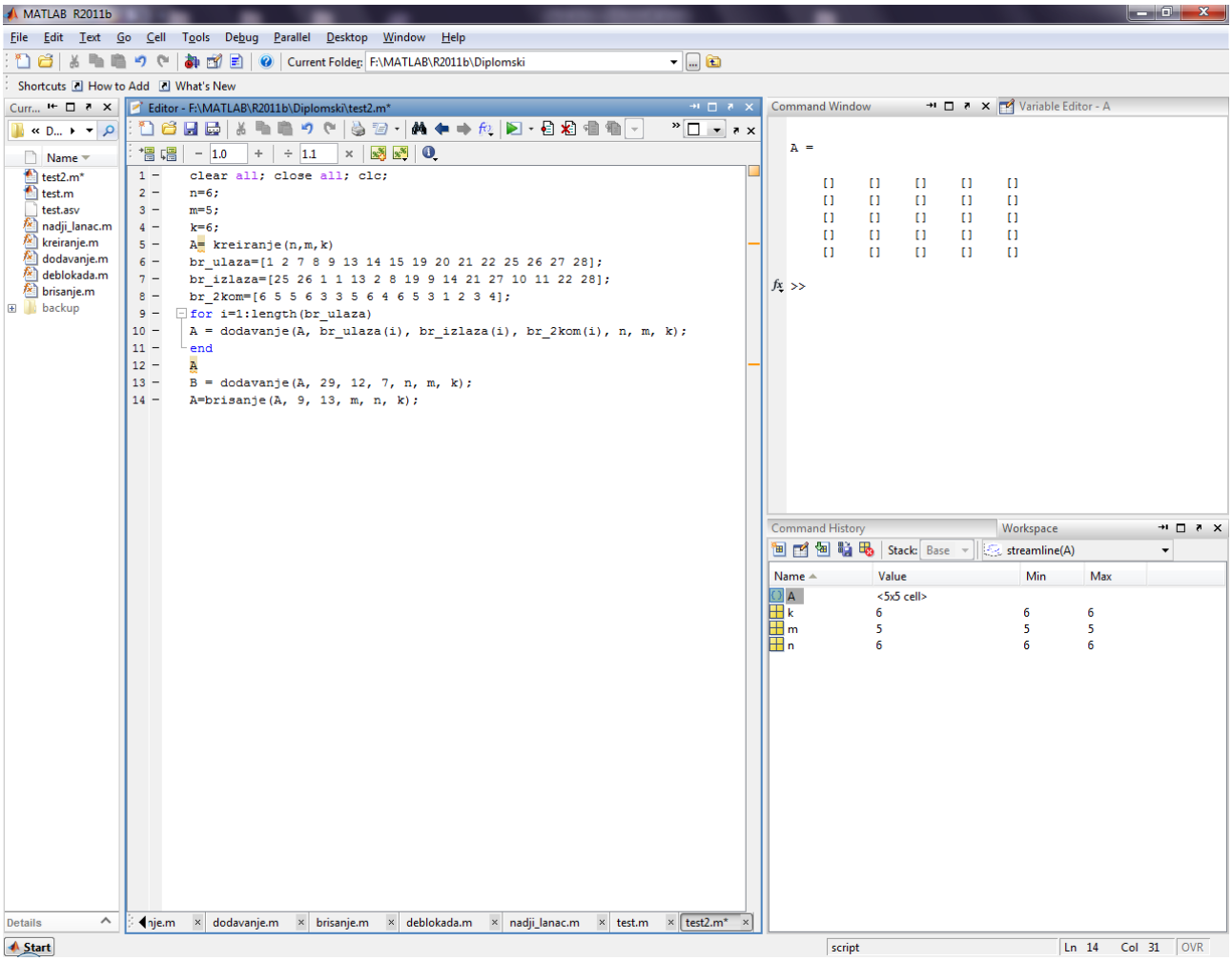
Slika 4.8. Drugi deo funkcije *deblokada.m* i deblokirana matrica

Poslednji deo skripte *test.m* je demonstracija funkcije *brisanje.m* koja raskida konekciju između dva korisnika. Funkcija na osnovu početnih argumenata *br_ulaza* i *br_izlaza* nalazi element matrice na koji se odnosi konekcija i postavlja umesto njega ili prazan skup ili samo eliminiše element iz već postojećeg skupa.

Drugi primer: Dato je trokaskadno prostorno KP (komutaciono polje) $C_3(6,5,6,5,6)$. Komutatori u srednjoj kaskadi su redom obeleženi sa A,B,C,D,E,F. Kroz komutaciono polje su uspostavljene sledeće veze: (1,F,25), (2,E,26), (7,E,1), (8,F,7), (9,C,13), (13,C,2), (14,E,8), (15,F,19), (19,D,9), (20,F,14), (21,E,21), (22,C,27), (25,A,10), (26,B,11), (27,C,22), (28,D,28). Prikazati sadržaj Paullove matrice. Koristeći Paulov algoritam izvršiti deblokadu veze (29,12). Ulazi/izlazi komutacionog polja se numerišu počev od broja 1.

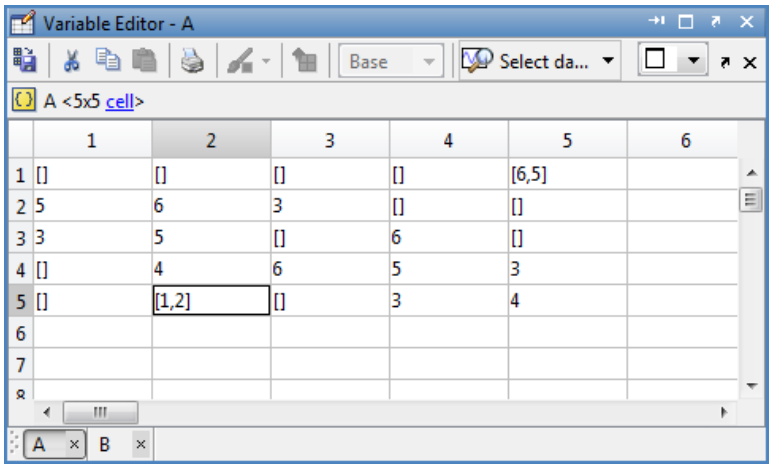
Da bi se prikazale funkcije u drugom primeru korišćena je skripta *test2.m* (slika 4.9) koja je slična kao i *test.m* sa tim što se odnosi na drugi primer. Ono što je drugačije u odnosu na prvi primer osim dimenzija je to što se prilikom formiranja razlike skupova komutatora srednje kaskade, u jednoj razlici nalazi više od jednog elementa skupa pa ovim primerom testiramo i taj kompleksniji slučaj u odnosu na prvi primer.

Na slici 4.9. se takođe vidi i matrica formirana funkcijom *kreiranje.m* i u ovom slučaju je formirana matrica dimenzija 5x5.



Slika 4.9. Prikaz skripte *test2.m*

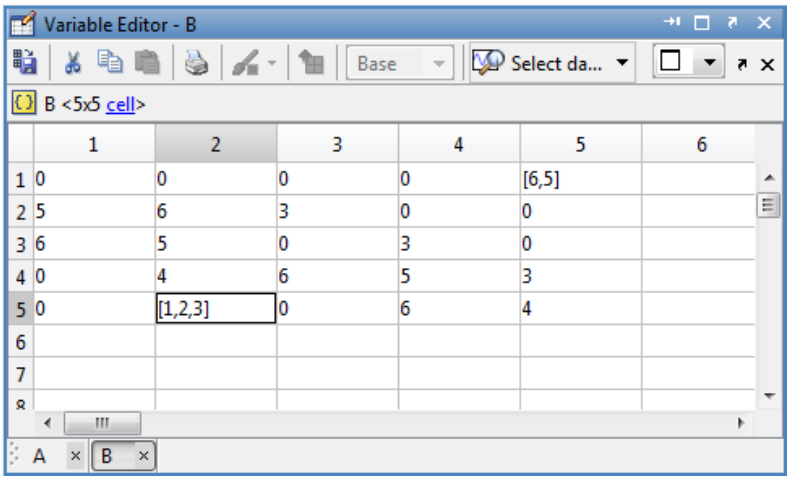
Posle formiranja matrice se vrši funkcija *dodavanje.m* koja dodaje veze u matricu. Posle dodavanja svih veza iz postavke primera 2, Paulova matrica izgleda kao na slici 4.10.



Slika 4.10 Matrica uspostavljenih veza

Slično kao u prvom primeru da bi se uspostavila veza (29,12) potrebno je prvo odblokirati polje u matrici (5,2). Sama deblokada se vrši analogno prvom primeru uz druge vrednosti u

lancima. U ovom primeru vrednosti *razlike1* i *razlike2* su 3 i {5,6} respektabilno. Formiraju se četiri lanca: 3-5-3-5, 3-6-3, 5-3-5-3, 6-3-6-3-6. Jasno je da je najkraći lanac 3-6-3 i u tom slučaj vrednosti *prvi* i *drugi* su 3 i 6. I kada se izvrši deblokada, Paulova matrica koja se dobije je prikazana na slici 4.11. Kao što se vidi, izabran je najkraći lanac što potvrđuje ispravan rad realizovane aplikacije.



Slika 4.11. Prikaz deblokirane matrice

Poslednji deo skripte *test2.m* je funkcija *brisanje.m* koja briše element iz matrice time označavajući raskid veze.

5. ZAKLJUČAK

Uslovno blokirajuća trokaskadna Klosova komutaciona polja predstavljaju značajnu uštedu resursa u odnosu na komutaciona polja bez blokade. Slepian je dao uslov kada je polje uslovno blokirajuće i koliko je maksimalno potrebno preuređivanja da bi se veza oslobodila. Paulov algoritam predstavlja dodatnu uštedu u broju preuređivanja čime doprinosi bržoj preraspodeli i samim tim bržoj uspostavi veze.

Ova implementacija pokazuje na praktičnom primeru neke karakteristike Paulovog algoritma i bliže objašnjava sam proces deblokiranja veze. Imajući u vidu korisnost samog algoritma i danas u optičkim mrežama i mrežama koje se još uvek baziraju na komutaciji kola ova se implementacija može koristiti i u nekim praktičnim primerima pri deblokiranju veze. Takođe, postoji i primena kao nastavno sredstvo u tome što omogućava studentima da na konkretnom primeru mogu da vide implementaciju algoritma bilo kao pokazna vežba, bilo kao sredstvo za rešavanje problema.

Ovu implementaciju je moguće unaprediti dodavanjem grafičkog prikaza same matrice u okviru Matlabovog Graphical User Interface (GUI) kako bi se efikasnije mogla koristiti kao nastavno sredstvo. Naravno, Paul je predvideo korišćenje algoritma ne samo za trokaskadna polja već i za komutaciona polja sa više kaskada (pet, devet...) gde se vrednostima u matrici mogu predstavljati cela trokaskadna komutaciona polja pa se ponavljanjem istog algoritma može doći do rešenja. Takođe, pošto se ova implementacija uglavnom fokusira na lance od dva elementa bez obzira na ostatak matrice moguće je uključivanjem drugih komutatora postići efikasniju preraspodelu. Takođe, Paul je u svom radu gde je predstavio algoritam nagovestio mogućnost primene algoritma u pravougaonim matricama koje bi odgovarale mrežama sa funkcijom koncentracije pa čak i sa trougaonim mrežama pa je možda moguće ovu implementaciju proširiti tako da obuhvata i te slučajeve.

LITERATURA

[1] Paull M.C. Reswitching of Connection Networks B.S.T.J 41 pp. 833-855

[2] Čiča Zoran Materijali sa predmeta Komutacioni sistemi

A. KOD REALIZOVANIH FUNKCIJA U MATLABU

A.1. Funkcija *kreiranje.m*

```
%% Kreiranje Paullove matrice
% Kreira se simetricno polje na osnovu ulaznih parametara
function izlaz = kreiranje(n, m, k)
% n - broj ulaza u komutator iz prve kaskade
% m - broj komutatora u prvoj kaskadi
% k - broj komutatora u drugoj kaskadi
% m1 - broj komutatora u trecoj kaskadi
% n1 - broj izlaza iz komutatora trece kaskade
n1=n;m1=m; % posto je komutaciono polje simetricno
% Sada je potrebno proveriti da li je komutaciono polje blokirajuce,
% neblokirajuce ili uslovno blokirajucE sto je uslov za primenu Paullovog
% algoritma
if k < max(n,n1)
    if k>=n+n1-1
        fprintf('Polje je neblokirajuce');
        % polje je neblokirajuce
    else
        fprintf('Polje je blokirajuce');
    end
else izlaz=cell(m1,m);
    % polje je uslovno blokirajuce
end
```

A.2. Funkcija *dodavanje.m*

```
%% Funkcija dodavanja veze u Paullovu matricu
function izlaz = dodavanje(Paullova_matrica, br_ulaza, br_izlaza, br_2kom, n, m,
k)
A=Paullova_matrica;
% Odredjivanje komutatora u prvoj i trecoj kaskadi
red=ceil(br_ulaza./n);
kolona=ceil(br_izlaza./n);
if br_2kom>k
    if br_2kom==k+1
        %koristi se Benesovo pravilo
        for i=1:k
            if ~ismember(i,cell2mat(A(red,:)))
                if ~ismember(i ,cell2mat(A(:,kolona)'))
                    if length(cell2mat((A(red,kolona))))>1
                        C=cell2mat(A(red,kolona));
                        C=[C i];
                        A(red,kolona)={C};
                    else A(red,kolona)={i};
```

```
                    end
                    break
                end
            end
        end
        A=deblokada(A,br_ulaza, br_izlaza, n, m, k);
    else fprintf('nije dobro unet broj komutatora 2. kaskade');
    end
elseif br_2kom<=k
    % Provera da li je veza blokirana
    if ismember(br_2kom, cell2mat(A(red,:)))
        if ismember(br_2kom, cell2mat(A(:,kolona)))
            % Deblokiranje veze
            A=deblokada(A,br_ulaza, br_izlaza, n, m, k);
        end
    end
    % Upisivanje komutatora druge kaskade koji se koristi u matricu
    if ~isempty(A(red,kolona))
        A(red,kolona)={[cell2mat(A(red,kolona)) br_2kom]};
    else A(red,kolona)={br_2kom};
    end
end
izlaz=A;
```

A.3. Funkcija *deblokada.m*

```
%% Funckija deblokiranja
function izlaz= deblokada(Paullova_matrica ,br_ulaza, br_izlaza, n, m, k)
A=Paullova_matrica; % Paullova matrica se smesta u promenjivu A radi lakseg
pozivanja
% Nalaze se brojevi komutatora prve i trece kaskade
kom1=ceil(br_ulaza/n);red=kom1;
kom3=ceil(br_izlaza/n);kolona=kom3;
% Nalaze se razlike izmedju skupova vrste i kolone u kojoj se nalazi polje
% veze koju je potrebno deblokirati
razlike1=setdiff(cell2mat(A(red,:)),cell2mat(A(:,kolona)'));
razlike2=setdiff(cell2mat(A(:,kolona)'),cell2mat(A(red,:)));
naj_brojac=10000;
brojac=0;
% Pomocu dve for petlje se prolazi kroz sve moguce kombinacije od dve
% razlike radi trazenja najboljeg resenja
for i=1:length(razlike1)
    razlika1=razlike1(i);
    for j=1:length(razlike2)
        razlika2=razlike2(j);
        % Poziva se podfunkcija koja pronalazi potrebne parametre za
        % odredjivanje lanca
        [brojac, prvi, drugi]= nadji_lanac(A,1,red,kolona,razlika1,razlika2,m);
        % Brojac koji odradjuje najbolji lanac tako sto uporedjuje duzine
        % lanaca
        if brojac<naj_brojac
            naj_brojac=brojac;
            brojac=0;
            naj_prvi=prvi;
            naj_drugi=drugi;
            naj_slucaj=1;
```

```
        end
        [brojac, prvi, drugi]= nadji_lanac(A,2,red,kolona,razlika1,razlika2,m);
        if brojac<naj_brojac
            naj_brojac=brojac;
            brojac=0;
            naj_prvi=prvi;
            naj_drugi=drugi;
            naj_slucaj=2;
        end
    end
end
prazno = cellfun(@isempty,A);
A(prazno) = {0};
B=A;
% U sledecem delu funkcije se za najbolji slucaj prolazi jos jednom kroz
% slican postupak kao u podfunkciji nadji_lanac osim sto se vrednosti u
% matrici menjaju u skladu sa algoritmom
if naj_slucaj==1
    razlika1=naj_prvi;
    razlika2=naj_drugi;
    smer=naj_slucaj;
else
    razlika2=naj_prvi;
    razlika1=naj_drugi;
    smer=naj_slucaj;
end
while(1)
    br=0;
    if smer==1
        for p=1:m
            P=cell2mat(A(red,p));
            if any(P==razlika1)
                sledeci=p;
                kolona=sledeci;
            else br=br+1;
            end
        end
        if br==m
            break
        end
        if length(cell2mat((B(red,kolona))))>1
            C=cell2mat(B(red,kolona));
            pom=find(C==razlika1);
            C(pom)=razlika2;
            B(red,kolona)={C};
        else B(red,kolona)={razlika2};
        end
        smer=2;
    elseif smer==2
        for p=1:m
            P=cell2mat(A(p,kolona));
            if any(P==razlika2)
                sledeci=p;
                red=sledeci;
            else br=br+1;
            end
        end
        if br==m
            break
```

```
        end
        if length(cell2mat((B(red,kolona))))>1
            C=cell2mat(B(red,kolona));
            pom=find(C==razlika2);
            C(pom)=razlika1;
            B(red,kolona)={C};
        else B(red,kolona)={razlika1};
        end

        smer=1;
    end
end
if length(cell2mat((B(kom1,kom3))))>1
    C=cell2mat(B(kom1,kom3));
    C=[C naj_prvi];
    B(kom1,kom3)={C};
else B(kom1,kom3)={naj_prvi};
end
izlaz=B;
```

A.3.1. Podfunkcija nadji_lanac.m

```
%% Funkcija nadji lanac
function [brojac, prvi_lanac, drugi_lanac]=
nadji_lanac(A,slucaj,red,kolona,razlika1,razlika2,m)
% U ovoj funkciji se vrsi nalazenje lanca tako sto se korak po korak u
% skladu sa promenjivom slucaj naizmenicno prvo proverava vrsta pa kolona u
% cilju nalazenja sledeceg elementa lanca do samog kraja
smer=slucaj;br=0;brojac1=0;trazi=1;
while(trazi)
    br=0;
    if smer==1
        % For petlja ovde sluzi za prolazak kroz sve elemente vrste i
        % detekciju poslednjeg elementa u lancu
        for p=1:m
            P=cell2mat(A(red,p));
            if any(P==razlika1)
                sledeci=p;
                kolona=sledeci;
                smer=2;break
            else br=br+1;
            end
        end
        if br==m
            trazi=0;
        end
    elseif smer==2 % Ovde se traze elementi po koloni
        for p=1:m
            P=cell2mat(A(p,kolona));
            if any(P==razlika2)
                sledeci=p;
                red=sledeci;
                smer=1;break
            else br=br+1;
            end
        end
```

```
        if br==m
            trazi=0;
        end
    end
    % brojac sluzi za brojanje elementata lanca u cilju nalazenja najboljeg
    % resenja
    brojac1=brojac1+1;
end
brojac=brojac1;
if slucaj==1
    prvi_lanac=razlika1;
    drugi_lanac=razlika2;
else
    prvi_lanac=razlika2;
    drugi_lanac=razlika1;
end
end
```

A.4. Funkcija *brisanje.m*

```
%% Funkcija brisanja
function izlaz=brisanje(Paullova_matrica, br_ulaza, br_izlaza, n, m, k)
A=Paullova_matrica;
kom1=ceil(br_ulaza/n);red=kom1;
kom3=ceil(br_izlaza/n);kolona=kom3;
if length(cell2mat((A(red,kolona))))>1
    C=cell2mat(A(red,kolona));
    C(1)=[];
    A(red,kolona)={C};
else A{kom1,kom3}=[];
end
izlaz=A;
```