ATC SERBIA
AUTOMOTIVE TRANING CENTER OF CENTRAL SERBIA

Primena Matlab-a u inženjerskim proračunima

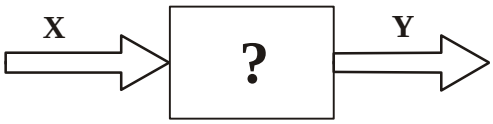
Dragan Pršić

MAŠINSKI FAKULTET KRALJEVO
KRALJEVO, 2011

1. Računarski alati

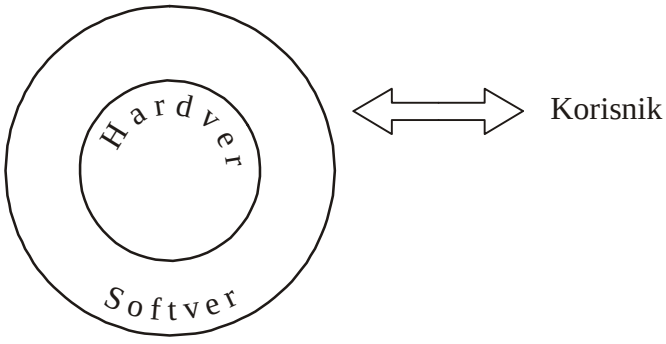
Računari se danas koriste u skoro svim oblastima života: učenju, proizvodnji, trgovini, administraciji, medicini, sportu, nauci, raznim oblicima umetnosti. Teško je pronaći neku oblast gde se ne koriste. Imaju i poslovnu i zabavnu upotrebu. Primena računara je danas stvar opšte kulture i deo svakodnevnog života.

Računar je mašina. Funkcionalnost te mašine simbolički se može prikazati kao na slici Sl.1.



Sl.1.1 Funkcionalni prikaz računara

Sa X su označene informacije koje ulaze u računar na obradu (ulazne informacije) a sa Y informacije koje se dobijaju kao rezultat obrade (izlazne informacije). Dakle, računar je mašina koja obradjuje informacije. Te informacije mogu biti u obliku brojeva koji predstavljaju rezultate nekog merenja, slova nekog teksta, tačaka neke slike, tonova neke muzike, itd. Ono što računar izdvaja u odnosu na druge mašine, zahvaljujući čemu i imaju tako široku primenu, je njegova fleksibilnost. Način na koji se ulazne informacije transformišu u izlazne možemo definisati sami. Otuda i znak pitanja na gornjoj slici. Pomoću računara možemo da kucamo tekst, obavljamo izračunavanja, simuliramo različite pojave, komuniciramo, pretražujemo podatke, igramo se, slušamo muziku, pratimo različita dešavanja, itd. Nijedna druga mašina (televizor, mobilni telefon, automobil, frižider) nema takvu fleksibilnost. Da bi smo razumeli odakle potiče takva fleksibilnost računara treba pogledati njegovu strukturu (Sl.2).



Sl.1.2 Simbolički prikaz strukture računara

Računar se sastoji od dva podsistema: hardvera i softvera. Hardver se odnosi na fizički opipljivu opremu računara: matičnu ploču, disk, memoriju, napajanje, tastaturu, miš, monitor, itd. Softver obavlja ulogu posrednika (interfejs) izmedju hardvera i korisnika i odnosi se na sve programe koji pokreću hardver. Programi su organizovani u slojevima zavisno od namene. Sloj najbliži hardveru pripada grupi programa poznatoj pod imenom operativni sistemi. U toj grupi su paketi: Windows, Linux, Unix, Apple Mac OS, itd. Programi iz svih ostalih slojeva koriste usluge ovih programa i jedan od programa iz ove grupe mora biti na svakom računaru. U slojevima iznad ovog osnovnog nalaze se korisnički programi različite namene. Oni nam omogućavaju da u grafičkom okruženju, na intuitivan način, bez potrebe da se udubljujemo u način funkcionisanja računara, obavimo različite poslove. Na primer različite kancelarijske zadatke možemo da obavimo u paketima MS Office, Open Office. Velike količine podataka možemo da čuvamo i pretražujemo pomoću baza SQL, MySQL ili Oracle. Internet možemo da pretražujemo pomoću Internet Explorer-a, Mozile, Opere, itd. Mogućnost da na bazi istog (ili sličnog) hardvera i operativnog sistema izaberemo korisnički program po želji objašnjava tako veliku primenu računara. Dakle, fleksibilnost računara potiče iz korisničkog sloja softvera. Uz malo poznavanje operativnog sistema, svo naše znanje o računaru može se završiti sa poznavanjem korisničkih alata. Na primer ako se samo bavimo pretraživanjem Interneta nije potrebno da budemo neki poseban stručnjak za računare, dovoljno je da poznajemo rano okruženje nekog od Internet pretraživača. Ako želimo da gledamo utakmicu na TV aparatu ne treba da poznajemo njegovu elektroniku i način rada. Dovoljno je da znamo da rukujemo sa daljinskim upravljačem. Najveći broj korisnika računara koristi gotove softverske pakete.

Posebnu grupu programa čine prevodioci za programske jezike kao što su Pascal, Visual Basic, C#, C++, C, Java, itd. To su programi pomoću kojih možemo da pravimo druge programe. Dakle, ako ne postoji neka softverska alatka koja nam treba, a ne mžemo je naći ili je skupa, onda možemo pomoću nekog od programskih prevodilaca da napišemo svoju aplikaciju. Ovo je potpuno drugačiji način upotrebe računara - programerski. Zahteva mnogo više poznavanja računarske tehnike ali pruža i više mogućnosti. Ako želite da popravite svoj TV aparat ili da mu dodate novu funkcionalnost moraćete da znate više od korišćenja daljinskog upravljača.

Gde se u svemu tome nalazi Matlab? Matlab je softverska alatka namenjena matematičkim izračunavanjima. To je program, namenjen krajnjem korisniku koji od računara očekuje moćnu alatku i pomoćnika u obradi i analizi numeričkih podataka a da pri tome ne mora da ulazi u tajne programiranja. Biblioteka ugradjenih matematičkih funkcija zaista je velika i malo je verovatno da će nam nekad nešto trebati što već ne postoji u Matlab-u. Ipak, za naprednije korisnike, u Matlab-u postoji ugradjen skript jezik koji nam omogućava da pišemo sopstvene funkcije kombinacijom već postojećih. To znači da je Matlab ne samo korisnički alat već i programski jezik namenjen proširivanju mogućnosti unutar ovog okruženja.

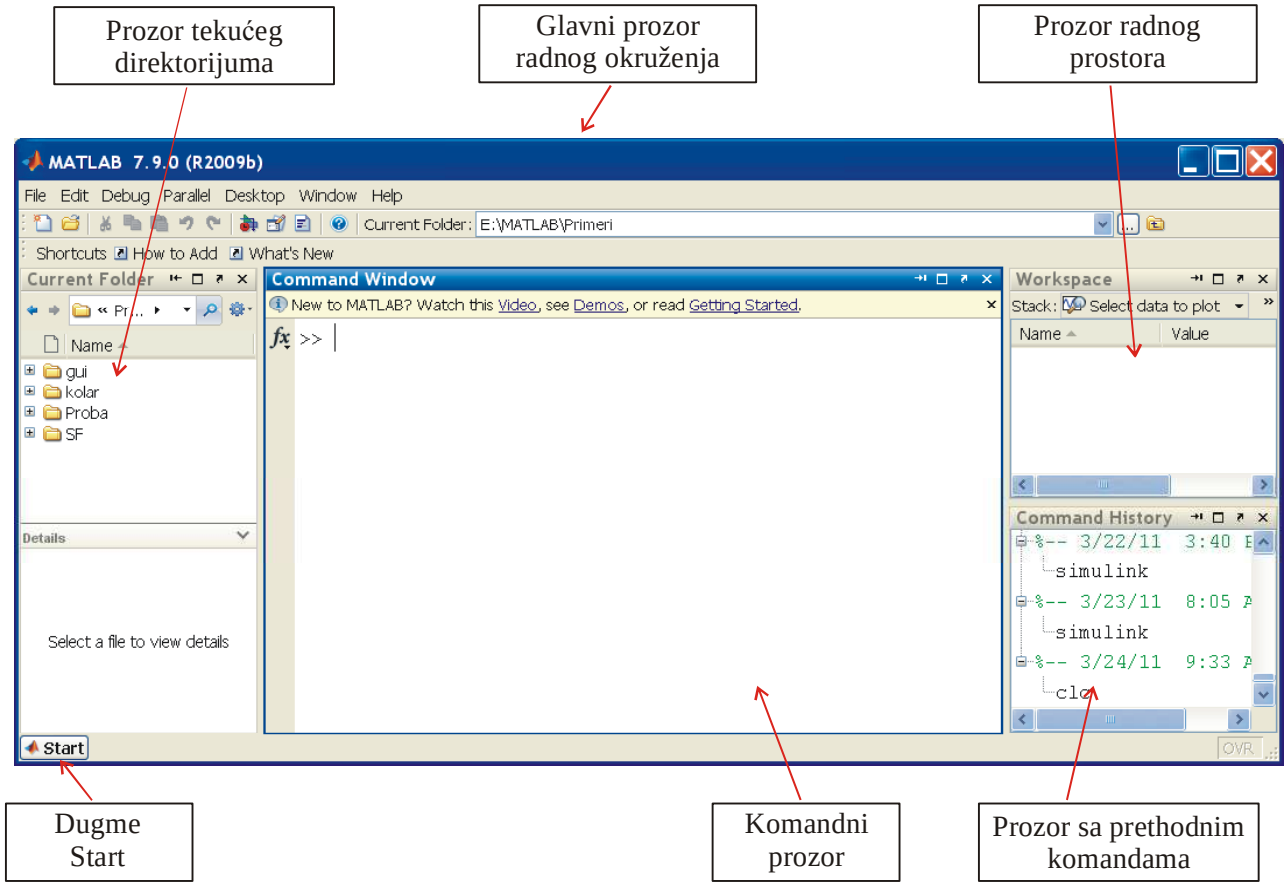
Ono što je npr. MS Word u obradi teksta to je Matlab® u obradi brojeva. Pored Matlab-a ovoj grupi alata pripadaju Mathematica®, Matchad®, Maple®, Scilab itd. Sve su to izvanredni alati, približno

sličnih mogućnosti makar na nekom prosečnom nivou korišćenja. Stvar je navike, ličnih afiniteta, intuicije, cene itd. koji ćemo od njih koristiti. Na ovom kursu nadalje, koristimo samo Matlab.
Pre nego što krenemo sa upotrebom Matlab-a treba se još jednom podsetiti da je računar samo mašina. Ona ne ume (još uvek) da pogadja vaše misli, očekivanja, osećanja. Neće uraditi ono što vi želite već ono što ste joj rekli da treba da uradi. U komunikaciji sa računarom morate biti jasni i precizni. Jedan zarez ili tačka mogu da dovedu do neočekivanih rezultata. Na sreću računar je dosledan - jednom usvojena pravila poštuje bez obzira na okolnosti.

1.1 Prvi koraci u Matlab-u

1.1.1 *Radni prozori u Matlab-u*

Kada se pokrene Matlab otvara se glavni prozor radnog okruženja kao što je prikazano na slici Sl.3.



Sl.1.3 Radno okruženje Matlab-a

Kao i svi drugi prozori i ovaj prozor ima standardni interfejs: naslovnu traku na kojoj se nalazi komandni meni (sakriven iza ikonice Matlab-a), tekst sa nazivom i verzijom vlasnika prozora (MATLAB 7.9.0 (R2009b)) i tri komandna dugmeta na desnom kraju (za minimiziranje,

maksimiziranje i zatvaranje prozora). Položaj i veličinu ovog (i svih drugih) prozora možemo i ručno menjati kao i kod svakog drugog Windows programa. Ispod naslovne trake nalazi se linija menija sa svim raspoloživim komandama iz Matlab radionice. Ispod menija nalazi se traka sa ikonicama najčešće korišćenih alatki. Osim toga, moguće je da formiramo sopstvenu paletu alata sa prečicama ka nama najpotrebnijim komandama. Sadržaj radnog prostora glavnog prozora čine drugi prozori iz Matlab-ovog radnog okruženja. Podrazumevani prikaz kod ove verzije (R2009b) sadrži četiri prozora (Sl.3): komandni prozor, prozor tekućeg direktorijuma, prozor radnog prostora i prozor sa prethodnim komandama. Tokom kursa upoznaćemo se sa još nekim prozorima koji nisu vidljivi u podrazumevanom prikazu. Kao i kod većine savremenih softverskih alata sadržaj glavnog prozora možete podešavati prema svojim potrebama. Prozore možete sakrivati/prikazivati, pomerati, menjati im veličinu shodno svojim potrebama i navikama. Podrazumevanom prikazu se uvek možete vratiti tako što iz menija izaberete: *Desktop* -> *Desktop Layout* -> *Default*. Alternativno, Matlab-ovim alatkama i svojstvima možemo pristupiti pomoću dugmeta *Start*.

Najveći deo vremena provešćemo u interaktivnom radu u komandnom prozoru[*] (Command Window). U interaktivnom radu naizmenično zadajemo komande i dobijamo odgovore od Matlab-a. To znači da nema formalnog prevodjenja, povezivanja, učitavanja i izvršavanja kao što moramo raditi kod programskih jezika višeg nivoa. Rad podseća na rad u Excel-u ili na rad sa kalkulatorom.

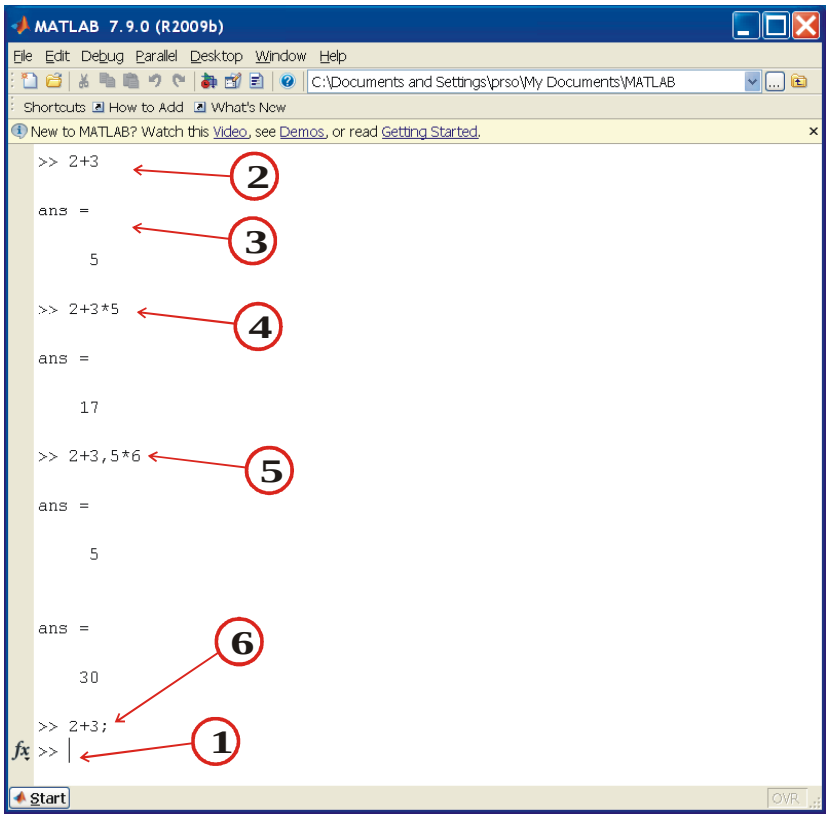
1.1.2 *Rad u komandnom prozoru*

Komandni prozor je najvažniji interfejs Matlab-ovog okruženja. U njemu kucamo komande i upravljamo radom Matlab-a. Pri unosu komandi važe sledeća pravila (Sl.4):

- Matlab je spreman za unos komande kada je kursor (|) neposredno iza komandnog odzivnika (>>)①. Red u kojem se kuca naredba naziva se komandna linija. Sadržaj komandne linije možemo uredjivati korišćenjem tastera kao što su npr. ←, →, *Del*, *Backspace*, itd.
- Nakon što se otkuca komanda, Matlab-u treba staviti do znanja da želimo njeno izvršenje. To se postiže pritiskom na taster *Enter* ②.
- Rezultat komande se dodeljuje promenljivoj *ans* ③.
- Narednu komandu 2+3*5 ④ možemo otkucati izpočetka ili tako što se nadovežemo na prethodnu komandu (2+3). U drugom slučaju, prvo pritisnemo taster sa strelicom nagore (↑) nakon čega se iza komandnog odzivnika pojavljuje prethodno uneta komanda 2+3. Zatim je dovoljno da dopišemo *5 i pritisnemo taster *Enter*. Izvršiče se komanda 2+3*5.
- Uzastopnim pritiskanjem tastera (↑) i (↓) možemo da se krećemo kroz listu prethodno unetih komandi i da ih naknadno uredjujemo.

[*] Ukoliko želimo da u glavnom prozoru vidimo samo komandni prozor možemo u meniju pokrenuti komandu: *Desktop* -> *Desktop Layout* -> *Command Window Only*.

- Prethodnu komandu ne možemo izmeniti tako što pokazivačem miša kliknemo na nju i onda pokušamo da unesemo izmenu. Prvo moramo kursorskim strelicama sa tastature da vratimo prethodnu komandu, unesemo izmenu, a onda sa tasterom *Enter* zahtevamo izvršenje komande. Ova mogućnosti može biti od velike koristi pri unosu složenih komandi. Dešava se da nakon izvršenja komande primetimo da smo napravili grešku u unosu. Nema potrebe da kucamo celu komadnu ispočetka. Dovoljno je da vratimo prethodnu komandu i samo ispravimo grešku. Ova mogućnost je korisna i kod testiranja složenih komandi. Recimo kada nam se javi greška u izvršavanju komande a ne znamo od kojeg dela izraza ta greška potiče. Na primer, više radi ilustracije, mi smo prethodnu komandu izvršili parcijalno. Prvo smo izvršili 2+3 a onda tako testiranu komandu samo dopunili sa *5.
- U istom redu možemo otkucati i više naredbi. Treba ih samo odvojiti zarezom ⑤. Naredbe će se izvršiti (nakon pritiska na *Enter*) redosledom kojim su i ukucane.
- Ukoliko se na kraju komandne linije (pre pritiska na *Enter*) otkuca ; (tačka-zarez) naredba će se izvršiti ali se neće prikazati rezultat ⑥. Ovo je podesno kod medjurezultata čije nas vrednosti ne interesuju. Na primer ako otkucamo u istom redu 2+3;5*6 rezultat prve komande biće 5 ali se neće prikazati. Rezultat druge komande biče 30 i on će se prikazati.
- Komanda **clc** samo briše sadržaj komandnog prozora. Sve prethodno unete promenljive i dalje postoje. Prethodno unete komande se i dalje mogu prelistavati pomoću kursorskih strelica. Kao kad se sundjerom obriše školska tabla. Medjutim, rezultati rada, ono što su deca prepisala u svoje sveske, ostaju.



Sl.1.4 Rad u komandnom prozoru

1.1.3 Prozor sa prethodnim komandama

Hronologiju prethodnih unetih komandi možemo pregledati, uredjivati, ponovo pokretati i u prozoru sa prethodnim komandama (*Command History*, Sl.3). Za listanje se mogu koristiti strelice (↑,↓) ili klizač prozora. Dvostrukim pritiskom (taster miša) na komandu u prozoru sa prethodnim komandama istu prebacujemo u komandni prozor odakle se pritiskom na taster *Enter* može ponovo izvršiti. Pri kucanju složenih komandi ova mogućnost nam može biti od velike koristi.

1.1.4 Dugme Start

Ovo dugme se nalazi u donjem levom uglu Matlab-ovog radnog okruženja. Pruža jos jednu mogucnost pristupa postojećim Matlab-ovim alatima, funkcijama, sistemu za pomoć, Internet servisima.

O preostala dva prozora (prozor tekućeg direktorijuma i prozor radnog prostora) standardnog Matlab-ovog interfejsa (Sl.3) reći ćemo nešto kasnije nakon upoznavanja sa novim funkcionalnostima ovog alata.

1.2 Promenljive u Matlab-u

Podaci i naredbe kojima se ti podaci obradjuju, sastavni su deo svakog rada u Matlab-u. Uvek nam je cilj da podatke sa ulaza na neki način obradimo da bi smo došli do željenih rezultata (Sl.1). Pri tome su nam na raspolaganju tri mogućnosti: da naredbe pišemo za konkretne podtake koje obradjujemo, za promenljive koje po potrebi mogu imati različite vrednosti ili kombinujući prethodne dve mogućnosti. Na primer ukoliko želimo da saberemo dva unapred poznata broja (3 i 8) možemo napisati:

$$3 + 8$$

Druga mogućnost je da uvedemo dve promenljive npr. x i y kojima dodelimo vrednosti (x=3, y=8) a onda napišemo:

$$x+y$$

Treba primetiti da u trenutku pisanja ove naredbe ne moramo znati vrednost promenljivih. Vrednost promenljivih moramo znati pre izvršavanja ove naredbe. Ovo je od suštinske važnosti kod programiranja u Matlab-u.

Treća mogućnost je da napravimo neku kombinaciju npr:

$$x+8$$

Koji ćemo način pisanja naredbi koristiti zavisi od konkretnog slučaja, ali pri malo ozbiljnijem radu drugin način je najefikasniji. Zato se javlja potreba uvodjenja promenljivih u Matlab. Promenljiva je simboličko ime za lokaciju u memoriji u kojoj se čuvaju podaci. Na primer kada napišemo

$$z = x + y$$

to praktično znači:

Sabrati podatak koji se nalazi na lokaciji čija je adresa *x* sa podatkom koji se nalazi na lokaciji čija adresa je *y* i sve to sačuvati na lokaciju čija je adresa *z*. Dakle pri pisanju naredbe mi se ne vezujemo za konkretan podatak već za lokaciju na kojoj se podatak čuva. Naravno pre pokretanja naredbe promenljivama *x* i *y* treba dodeliti vrednost. Ono što je bitno je da naredba ostaje ista bez obzira na dodeljene vrednosti.

Za razliku od vecine drugih programskih jezika u Matlab-u nije potrebno eksplicitno deklarisanje promenljivih. Da bi smo uveli novu promenljivu dovoljno je da imenu promenljive dodelimo vrednost. Kao operator dodele (*assignment operator*) koristi se znak "=". Naredba dodeljivanja se pise na sledeci nacin: pvo navedemo promenljivu ciju vrednost zelimo da definisemo, zatim dopisemo znak "=", a iza njega izraz koji odredjuje vrednost promenljive. Na primer, ako napišemo *x = 3* to ne zači da je *x* jednako 3 već da otvaramo novu memorijsku lokaciju čija je adresa *x* i u koju upisujemo broj 3. Ako zatim napišemo

x = x + 1

to znači da sadržaj lokacije x treba uvećati za 1 (bez obzira šta se na njoj nalazi) i rezultat ponovo upisati u promenljivu *x*. Operator dodele uvek vrednost izraza sa desne strane operatora dodeljuje memorijskoj lokaciji sa leve strane operatora. Npr. ispravno je napisati *x=3* ali nije ispravno *3=x*.

Kada želimo da saznamo vrednost neke promenljive (sadržaj neke memorijske lokacije), dovoljno je da u komandmom prozoru otkucamo ime te promenljive (adresu te lokacije). Na primer ako otkucamo:

```
>>x
```

dobićemo kao odgovor (ako su unete prethodne komande):

```
x =
    4
```

Ukoliko promenljiva *x* nije unapred definisana u memorijskom prostoru pojavice se poruka:

??? Undefined function or variable 'x'.

Spisak promenljivih koje su definisane u memorijskom prostoru Matlab-a i njihove trenutne vrednosti mogu se pregledati i uredjivati u prozoru radnog prostora (*Workspace*) (Sl.3). Pored toga, za rad sa promenljivama iz radnog prostora na raspolaganju su nam komande prikazane u tabeli T1.

Tabela T1 - Komande za rad sa promenljivama radnog prostora

Komanda	Rezultat
who	Prikazuje listu promenljivih iz memorije
whos	Prikazuje detaljnu listu promenljivih iz memorije
clear x y	Brise promenljive *x* i *y* iz memorije
clear	Brise sve promenljive iz memorije radnog prostora

Pri imenovanju promenljivih treba se pridrzavati pravila datih u tabeli T2.

Tabela T2 - Pravila za imenovanje promenljivih

R.b.	Pravilo
1.	Ime promenljive mora poceti slovom
2.	Ime promenljive moze sadrzati slova, brojeve i donju crtu (_). Ukupan broj znakova u imenu zavisi od verzije Matlab-a (U verziji R2009b Matlab koristi prva 63 znaka u imenu promenljive).
3.	U Matlabu postoji razlika izmedju malih i velikih slova. Promenljiva sa imenom *x* se razlikuje od promenljive sa imnom *X*. Slicno, promenljiva *sila* nije isto sto i promenljiva *Sila*.
4.	Rezervisane reci (npr. break, case, ...) se ne mogu koristiti za imena promenljivih. Lista rezervisanih reci Matlab-a se moze dobiti komandom *iskeyword*.
5.	Pri davanju imena promenljivama treba izbegavati imena ugradjenih funkcija i imena unapred definisanih promenljivih.

U tabeli T3 dato je nekoliko primera ispravnih i neispravnih imena.

Tabela T3 - Primeri ispravnih i neispravnih imena promenljivih u Matlab-u

Ime	Tip	Ime	Tip
x1	Ispravno	*vreme#*	Neispravno: Nije dozvoljena upotreba znaka # u imenu
brzina	Ispravno	*1x*	Neispravno: Ime mora poceti slovom
sila	Ispravno	*Normalna Sila*	Neispravno: Nije dozvoljen razmak u imenu
Temperatura1	Ispravno	*break*	Neispravno: break je rezervisana rec za Matlab
NormalnaSila	Ispravno	*sin*	Ispravno ali treba izbegavati jer je *sin* naziv za funkciju. U protivnom necemo moci da koristimo tu funkciju
ugaona_brzina	Ispravno	*pi*	Ispravno ali treba izbegavati jer je *pi* unapred definisana promenljiva (*pi*=3.14159265...) . U protivnom necemo moci da koristimo tu konstantu

U tabeli T4 data je lista unapred definisanih promenljivih.

Tabela T4 - Lista unapred definisanih promenljivih

Ime	Opis
ans	Tekuca promenljiva u Matlab-u. Njoj se dodeljuje vrednost poslednjeg izraza koji nije dodeljen nekoj promenljivoj. Npr. ako napisemo *y=2+3* vrednost izraza se dodeljuje promenljivoj *y*. Ako napisemo 2+3 vrednost izraza se dodeljuje promenljivoj *ans*.
pi	Broj π
i ili *j*	Imaginarna jedinica: $i = j = \sqrt{-1}$
eps	Najmanja razlika izmedju dva razlicita broja. Iznosi 2^(-52)

realmin	Najmanji realan broj 2^(-1022)
realmax	Najveci realan broj (2-eps)^(1023)
inf	Beskonacno velika vrednost. Npr. rezultat operacije 5/0.
NaN	Oznacava nedefinisanu vrednost (Not a Number). Npr. rezultat operacije 0/0 ili *inf-inf*.

Nakon izlaska iz radnog okruzenja Matlab-a (*File->ExitMATLAB, Ctrl+Q*) brise se sadrzaj memorijskog prostora dodeljenog komandnom prozoru. Drugim recima, promenljive i njihove vrednosti koje smo koristili tokom rada u Matlab-u bivaju nepovratno izgubljene. Kada sledeci put udjemo u Matlab prozor radnog prostora bice prazan[*].

Ipak, pre izlaska iz Matlab-a, uvedene promenljive i njihove trenutne vrednosti mozemo da sacuvamo komandom iz menija *File->Save Workspace As...*, naredbom *save* <Ime_Fajla> ili *Ctrl+S*. Program ce traziti da zadamo naziv fajla <Ime_fajla> u kojem treba da sacuva podatke. Podrazumevana ekstenzija fajla je *mat* a cuva se u tekucem direktorijumu (*Current Folder*). Tekuci direktorijum se moze uredjivati pomocu prozora tekuceg direktorijuma ili padajuce liste *Current Folder* na paleti sa alatkama ispod glavnog menija (Sl.3). Nakon toga, po potrebi, mozemo ucitavati fajl (*File->Import Data...*ili pomocu naredbe *load* <Ime_Fajla>) sa poslednjom verzijom sacuvanih podataka. Ove mogucnosti se treba setiti kada zelimo da nastavimo neki ranije zapocet rad.

1.3 Aritmeticki izrazi

Pri pisanju aritmetickih izraza pored brojeva i promenljivih koristimo i aritmeticke operatore. U tabeli T5 data je lista aritmetickih operatora.

Tabela T5 - Lista aritmetickih operatora

Operacija	Operator	Primer
Sabiranje	+	x+7
Oduzimanje	-	11-y
Mnozenje	*	x*y
Deljenje zdesna	/	8/3
Deljenje sleva	\	a\b = b/a (rezultat deljenja sleva je reciprocna vrednost deljenja sdesna)
Stepenovanje	^	2^3 (isto je sto i 2^3)

Treba imati u vidu da vrednost aritmetickih izraza zavisi ne samo od redosleda navodjenja operatora vec i od njihovog prioriteta. Drugim recima, aritmeticki operatori navedeni u gornjoj tabeli nemaju isti

[*] Treba reci da ce lista prethodno unetih komandi u interaktivnom radu (Prozor s prethodnim komandama) ostati sacuvana.

prioritet. Da bi se uocila vaznost prioriteta u tabeli T6 dato je nekoliko aritmetickih izraza sa rezulatatima.

Tabela T6 - Primeri aritmetickih izraza i njihove vrednosti

Aritmeticki izraz	Rezultat	Aritmeticki izraz	Rezultat
`7+2-3`	6	`2^2-1`	3
`2+3*2`	8	`2^(2-1)`	2
`(2+3)*2`	10	`18/6+3*2`	9
`15/3+2`	7	`18/(6+3)*2`	4
`15/(3+2)`	3	`24/((2+3)-1)/2`	3

U tabeli T7 data je lista prioriteta aritmetickih operatora. Broj 1 oznacava najveci prioritet.

Tabela T7 - Prioriteti aritmetickih operatora

Prioritet	Operacija
1	Zagrade. Ugnjezdene zagrad imaju veci prioritet.
2	Stepenovanje
3	Mnozenje i deljenje
4	Sabiranje i oduzimanje

Ukoliko niste sigurni u prioritet pojedinih operatora koristite zagrade. Sve ono sto je u zagradama ima veci prioritet. Sto su zagrade dublje, veci je prioritet.

U aritmetickim izrazima, osim osnovnih aritmetickih operatora, mozemo koristiti i funkcije. Snaga Matlab-a upravo lezi u bogatoj kolekciji ugradjenih funkcija. Vise nije pitanje da li neka funkcija postoji vec kako je naci. U tu svrhu se moze koristiti sistem za pomoc. Na primer ako otkucamo

help elfun

dobija se spisak elementarnih matematickih funkcija razvrstanih po kategoriji. Pritiskom na tastere *Shift+F1* otvara se pretrazivac funkcija. Do istog prozora mozemo doci i ako kliknemo na ikonicu f_x koja se nalazi neposredno ispred komandnog odzivnika (>>). Da bi smo dobili vise informacija o nekoj funkciji dovoljno je da u komandnom prozoru otkucamo:

help ImeFunkcije

gde je *ImeFunkcije* tacan naziv funkcije za koju trazimo objasnjenje.

Cak i ako nam lista ugradjenih funkcija nije dovoljna u Matlab-u mozemo praviti svoje funkcije (korisnicki definisane funkcije).

Sintaksa poziva funkcije ima ovaj oblik:

ImeFunkcije(lista_argumenata)

Dakle, posle naziva funkcije (*ImeFunkcije*) u zagradama navodimo opcionu listu argumenata. Na primer da bi smo odredili kvadratni koren broja 9 treba otkucati naredbu: *sqrt(9)*. *sqrt* je naziv funkcije a 9 je jedini argument te funkcije. Funkciju treba otkucati tacno kao sto je navedeno, nije dozvoljeno

Sqrt(9), sqrt 9, sqrt(9, ili *sqrt 9)*. Funkcija *date* vraca tekuci datum i nema argumenata koji joj se prosledjuju. Funkcija *max(a,b)* ima dva argumenta odvojena zarezom a kao rezultat daje veci od brojeva *a* i *b*.

Treba voditi racuna da ne koristimo naredbu oblika *sqrt=9*. Ova naredba ce od funkcije *sqrt* napraviti promenljivu *sqrt*, tako da nam funkcija nece biti vise na raspolaganju sve dok istoimenu promenljivu ne uklonimo iz memorije. Pri koriscenju funkcija u Matlabu treba biti obazriv cak i kada ih pravilno pozovemo jer se cesto dolazi do rezultata koje nismo navikli u drugim alatima. Na primer naredba *sqrt(-9)* kao rezultat vraca *0+3.0000i* (imaginarno 3) iako bi u vecini drugih alata rezultat bio greska ili neko upozorenje.

Argument funkcije moze biti broj, promenljiva ili neki drugi aritmeticki izraz. U tabeli T8 dato je nekoliko primera ispravnih poziva funkcije *sqrt* sa razlicitim argumentima.

Tabela T8 - Primeri poziva funkcije sqrt sa razlicitim argumentima

Primer
`sqrt(a/4)`
`sqrt(12+x^2)`
`sqrt(2-sqrt(3)*3+min(2,4))`

Pri pozivu ugnjezdenih funkcija voditi racuna da je broj otvorenih zagrada jednak broju zatvorenih.

U tabeli T9 data je lista par elementarnih matematickih funkcija.

Tabela T9 - Neke od ugradjenih matematickih funkcija u Matlabu

Funkcija	Opis	Primer
`sin(x)`	Sinus argumenta `x` datog u radijanima.	`sin(pi/3):` `ans=0.8660`
`sind(x)`	Sinus argumenta `x` datog u stepenima.	`sin(60):` `ans=0.8660`
`asin(x)`	Inverzna funkcija od `sin(x)`. Rezultat je u radijanima.	`asin(0.866):` `ans=1.0471`
`asind(x)`	Inverzna funkcija od `sin(x)`. Rezultat je u stepenima.	`asind(0.866):` `ans=59.9971`
`exp(x)`	Vrednost izraza e^x. e je osnova prirodnog algoritma.	`exp(1):` `ans= 2.7183`
`log(x)`	Prirodni logaritam broja `x`.	`log(2.7183)` `ans=1.0000`
`log10(x)`	Logaritam broja `x` za osnovu 10.	`log10(10):` `ans=1`
`sqrt(x)`	Kvadratni koren broja `x`.	`sqrt(9):` `ans=3`
`abs(x)`	Apsolutna vrednost broja `x`.	`abs(-3):` `ans=3`
`round(x)`	Zaokruzuje vrednost broja `x` na najblizi ceo broj.	`round(7.8):` `ans=8`
`fix(x)`	Zaokruzuje vrednost broja `x` na ceo broj blizi nuli.	`fix(7.8):` `ans=7`
`rem(x,y)`	Ostatak celobrojnog deljenja `x` sa `y`.	`rem(10,4):` `ans=2`
`sign(x)`	Funkcija znaka. Ako je `x>0` vrednost funkcije je 1. Kada je	`sign(2.5):`

	x<0 vrednost funkcije je -1. Funkcija vraca 0 kada je x=0.	ans=1

Pocetnici u Matlab-u treba da obrate paznju da se logaritam nekog broja za osnovu 10 ne racuna pomocu funkcije `log()` vec pomocu funkcije `log10()`. Logaritam broja `A` za ma koju osnovu `a` (`a>0`) moze se izracunati pomocu izraza: `log(A)/log(a)`.

Rezultate matematickih izraza Matlab moze da prikazuje u razlicitim formatima. Podrazumevano decimalni brojevi se prikazuju sa cetiri decimalna mesta. Na primer komanda

```
25/9
```

vraca kao rezultat

```
ans=2.7778
```

Ako zelimo da promenimo oblik prikaza broja koristimo naredbu `format`. Ako zelimo vise informacija o ovoj naredbi treba otkucati: `help format`. Na primer ako otkucamo: `format long` tada ce rezultat izraza

```
25/9
```

biti u obliku:

```
ans= 2.777777777777778
```

Treba reci da se naredbom `format` menja samo oblik prikaza broja ali ne i prava vrednost izraza sa kojom Matlab racuna. Dakle, ovom naredbom ne uticemo na tacnost izracunavanja vec samo na nacin prikaza broja. Ipak treba biti oprezan u interaktivnom radu kada koristimo podatke koje vidimo na ekranu. Na primer ako otkucamo:

```
sin(1.0472)
```

rezultat (sa podrazumevanim formatom) je oblika:

```
ans= 0.8660
```

ako sada otkucamo

```
asin(0.866)
```

rezultat je

```
ans=1.0471
```

a ocekivali bi smo da je `1.0472` jer je `asin()` inverzna funkcija od `sin()`. Uzrok neslaganja je u tome sto smo u naredbu `asin()` prosledili broj koji vidimo na ekranu (`0.866`) a ne pravu vrednost funkcije `sin(1.0472)`. Problem bi mogli da resimo uvodjenjem promenljive:

```
x=sin(1.0472)
```

daje rezultat oblika

```
x=0.8660
```

a naredba:

```
asin(x)
```

vraca

```
ans=1.0472
```

sto i ocekujemo.

Jos jednom, treba praviti razliku izmedju prave vrednosti promenljive i onoga sto se prikazuje kao njena vrednost kada se otkuca ime promenljive. Format prikaza ne utice direktno na vrednost podatka.
U finansijskim obracunima dovoljno je iznose prikazivati sa dve decimale sto se postize naredbom `format bank`. U tom slucaju, rezulta kolicnika `25/9` imao bi oblik `2.78`. Sto znaci da Matlab sam zaokruzuje prikaze rezultata na dve decimale.
U inzinjerskim proracunima mozemo da koristimo:
`format short eng` (`25/9 -> ans= 2.7778e+000`) ili
`format long eng` (`25/9 -> 2.77777777777778e+000`).

Obratiti paznju da slovo `e` u prethodnim prikazima (a i u svim narednim) oznacava eksponent za osnovu 10 a ne osnovu prirodnog algoritma. Na primer `2e2` nije $2*e^2$ vec $2*10^2$. Kao sto smo rekli (Tabela T9) e^2 se u Matlab-u pise kao `exp(2)`.
Podrazumevani format vracamo pomocu naredbe `format short`.

2. Podaci

Kao i kod drugih softverskih alata rad u Matlab-u se bazira na obradi podataka razlicitog tipa. Ono sto Matlab izdvaja u odnosu na druge alate je to sto je osnovna struktura podatka u Matlab-u niz (array). Cak i sam naziv paketa Matlab vodi poreklo od sintagme *Mat*rix *Lab*oratory (Laboratorija za matrice) jer je pre svega bio namenjen radu sa matricama. Skalare, vektore i matrice Matlab tretira na isti nacin - kao nizove razlicitih dimenzija. Drugim recima, sve promenljive u Matlab-u su nizovi. Elementi nizova mogu biti konstante (brojevi ili znakovi) i/ili izrazi.

Niz predstavlja skup podataka istog tipa sa zajednickim imenom. Za referenciranje pojedinih calnova u nizu koriste se indeksi. Na primer niz:

```
p=[-2 3 1 9]
```

ciji je naziv `p` ima cetri clana. Svi clanovi niza imaju isto ime ali je svaki clan niza jednoznacno odredjen indeksom koji se navodi u zagradama. Vrednost prvog clana niza je `p(1)=-2`, drugog `p(2)=3`, itd. Rad sa skupom podataka kao nizom moze biti znatno jednostavniji nego u slucaju da se svakom podatku dodeljuje nova promenljiva. Pri imenovanju nizova vaze ista pravila kao i kod skalarnih promenljivih. Nizovi mogu biti jednodimenzionalni, dvodimenzionalni i visedimenzionalni. U tehnici su jednodimenzionalni nizovi poznati pod imenom vektori (kolona ili vrsta) a dvodimenzionalni kao matrice.

Za rad u interaktivnom okruzenju potrebno je da znamo kako se podaci unose, kako im se pristupa i kako se azuriraju. U nastavku cemo pogledati kako se ove operacije realizuju u Matlab-u koristeci standardan ulaz i izlaz (tastaturu i ekran).

2.1 Unos podataka

Prvi korak u obradi podataka je njihov unos. Pod unosom podataka podrazumevamo njihovo smestanje u radni prostor Matlab-a. Svakom unetom podatku pridruzuje se po jedna promenljiva koja predstavlja simbolicku adresu memorijske lokacije na kojoj je podatak sacuvan. Pomocu nje, kasnije po potrebi, moze pristupati podacima. Posto postoje odredjene specificnosti, odvojeno cemo razmotriti unos skalara, vektora i matrica.

2.1.1 Unos skalarnih podataka

Skalar se u Matlab-u posmatra kao vektor sa jednim elementom ili kao matrica sa jednom vrstom i jednom kolonom (dimenzije 1x1). Za formiranje skalara, kao i za obicne promenljive, koristi se operator dodele "=". Na primer, mozemo napisati:

```
>> a(1)=3
a =
     3
```

ili

```
>> a(1,1)=3
a =
     3
```

Medjutim, kod skalara dovoljno je napisati:

```
>> a=3
a =
     3
```

sto je i najcesci nacin unosa skalarnih vrednosti.

Treba primetiti da u prethodnim primerima upotrebe naredbe dodele nismo koristili znak ";" pa se rezultat odmah prikazivao nakon njenog izvrsavanja. Medjutim, ako otkucamo:

```
>> b=7;
```

formirace se nova promenljiva u memoriji (mozemo proveriti u prozoru radnog prostora) ali se njena vrednost nece prikazati jer smo na kraju naredbe koristili znak ";". Naravno, uvek nam ostaje mogucnost da ako zelimo da vidimo rezultat promenljive kojoj smo dodelili vrednost naknadno otkucamo njeno ime. Na primer,

```
>> b
b =
     7
```

2.1.2 Unos vektora

Sintaksa naredbe za formiranje vektora vrste ima oblik:

```
ime_vektora=[lista_elemenata_vektora]
```

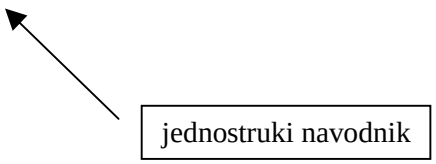
Prvo se navodi naziv vektora za kojim sledi operator dodele pracen listom vrednosti elemenata (izraza) nevedenom u uglastim zagradama. Elementi u listi se razdvajaju ili razmakom ili zarezom.

Na primer

```
>> brzina=[5 3 4*sin(3)]
brzina =
    5.0000    3.0000    0.5645
```

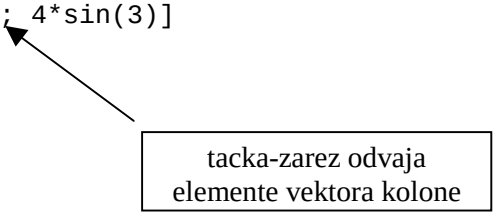
Vektor kolona se formira na slican nacin. Razlika je u nacinu navodjenja elemenata liste. Prva mogucnost je da posle desne uglaste zagrade (`]`) otkucamo jenostruki razvodnik (').

```
>> brzina=[5 3 4*sin(3)]'
brzina =
    5.0000
    3.0000
    0.5645
```



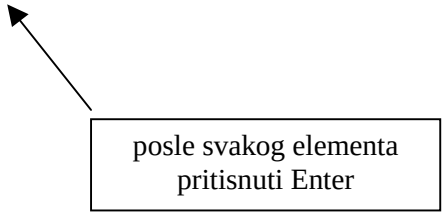
Druga mogucnost je da posle svakog elementa u listi otkucamo znak ; (tacka-zarez).

```
>> brzina=[5; 3; 4*sin(3)]
brzina =
    5.0000
    3.0000
    0.5645
```



I treca mogucnost je da posle svakog elementa u listi pritisnemo taster *Enter*. Nakon poslednjeg unetog elementa treba otkucati desnu uglastu zagradu].

```
>> brzina=[5
3
4*sin(3)
]
brzina =
    5.0000
    3.0000
    0.5645
```



Pri formiranju vektora sa istim razmakom izmedju elemenata mogu se koristiti dve komande koje znacajno ubrzavaju kreiranje vektora.

a) Prva mogucnost je da se koristi naredba oblika:

`ime_vektora=[p:k:q]` ili `ime_vektora=p:k:q`

gde je p - prvi element niza, q - poslednji element niza, k - korak izmedju elemenata niza. Ukoliko se korak izostavi podrazumevano ima vrednost 1. Uglaste zagrade nisu obavezne u formiranju vektora. Nekoliko primera formiranja vektora sa konstantnim korakom:

```
>> T=2:3:14
T =
     2     5     8    11    14

>> Q=-5:4:11
Q =
    -5    -1     3     7    11

>> P=2:7
P =
     2     3     4     5     6     7


>> F=24:-3:0
F =
```

```
    24    21    18    15    12     9     6     3     0
```

Ukoliko se poslednji clan niza ne moze dobiti kao algebarski zbir prvog elementa i celobrojnog umnoska koraka k, tada broj q u definiciji niza predstavlja gornju (za k>0) odnosno donju granicu (za k<0).

```
>> E=2:2.3:10
E =
    2.0000    4.3000    6.6000    8.9000

>> W=2:-2.4:-8
W =
    2.0000   -0.4000   -2.8000   -5.2000   -7.6000
```

Koriscenje funkcije linspace je druga mogucnost za kreiranje vektora sa konstantnim korakom izmedju elemenata. Sintaksa ove funkcije ima oblik:

```
ime_vektora=linspace(p,q,n)
```

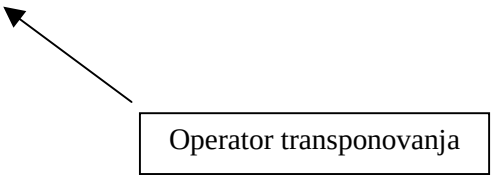
gde je p - prvi element vektora, q - poslednji element vektora, n - broj elemenata u vektoru. U nastavku su data dva primera.

```
A=linspace(-3,4,6)
A =
   -3.0000   -1.6000   -0.2000    1.2000    2.6000    4.0000

>> B=linspace(4,-3,6)
B =
    4.0000    2.6000    1.2000   -0.2000   -1.6000   -3.0000
```

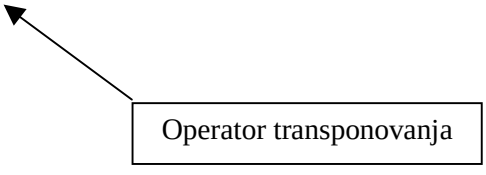
U prethodnim primerima kreirali smo vektore vrste. Vektore kolone sa istim rastojanjem izmedju elemenata mozemo dobiti na slican naci uz dodatak operatora za transponovanje ('). Na primer:

```
>> A=[2:3:14]'
A =
     2
     5
     8
    11
    14
```



ili:

```
>> B=linspace(-3,3,4)'
B =
    -3
    -1
     1
     3
```



2.1.3 Unos matrica

Matrica je pravougaona struktura podataka koja se sastoji od vrsta i kolona. Na primer, podaci u tabeli imaju matricnu strukturu. Broj vrsta i kolona definisu red matrice. Za matricu koja ima *m* vrsta i *n* kolona kaze se da je reda *m x n*. Ukoliko je *m = n* za matricu se kaze da je kvadratna reda *n*. U nastavku je dat primer matrice reda 3 x 4.

$$A = \begin{bmatrix} 2 & 5 & 1 & 2 \\ 6 & 2 & 4 & -7 \\ 3 & 0 & -4 & 1 \end{bmatrix}$$

Vidi se da je matrica dvodimenzionalni niz, odnosno jednodimenzionalni niz jednodimenzionalnih nizova. Matrica se u Matlab-u formira tako sto se promenljivoj dodeljuju elementi koji imaju pravougaonu strukturu. Kao i kod vektora elementi mogu biti konstante (brojevi ili znakovi) ili izrazi. Elemente matrice unosimo vrstu po vrstu unutar uglastih zagrada ([]). Sintaksa naredbe za generisanje matrice ima oblik:

```
ime_matrice=[elementi_prve_vrste; elementi_druge_vrste; ...;
             elementi_n_te_vrste]
```

Za unosenje elemenata vrste koriste se ista pravila kao i kod unosenja elemenata vektora vrste. Elementi vrste se razdvajaju razmakom ili zarezom a vrste se medjusobno razdvajaju tackom i zarezom ili pritiskom na taster *Enter*. Na primer, prethodna matrica se moze uneti na sledeci nacin:

```
>> A=[2 5 1 2;6 2 4 -7;3 0 -4 1]
```

Kada se pritisne taster *Enter* Matlab ce pokazati sadrzaj lokacije *A*:

```
A =
     2     5     1     2
     6     2     4    -7
     3     0    -4     1
```

Za definisanje elemenata matrice mogu se, pored brojeva, koristiti i promenljive i funkcije. Na primer:

```
>> x=2;
>> y=3+min(1,4);
>> B=[x -1 x^2+y^2;sin(y) 3 max(x,y);sqrt(y) x*y -7]
B =
    2.0000   -1.0000   20.0000
   -0.7568    3.0000    4.0000
    2.0000    8.0000   -7.0000
```

Za formiranje vrsta matrice mogu s koristiti naredbe za generisanje vektora sa istim razmakom izmedju elemenata.

```
>> x=[3 2 1];
>> C=[x;9:-2:5;linspace(4,15,3)]
C =
    3.0000    2.0000    1.0000
    9.0000    7.0000    5.0000
```

```
    4.0000    9.5000   15.0000
```

Prva vrsta je definisana pomocu promenljive *x* ciji su elementi rucno uneti. Elementi druge vrste su uneti pomocu naredbe za generisanje vektora sa jednakorazmaknutim elementima. Za trecu vrstu iskoriscena je funkcija *linspace( )*.

Treba samo voditi racuna da broj elemenata u svakoj vrsti bude isti bez obzira kako se oni unose.

Za brzo formiranje matrica sa elementima karakteisticnih vrednosti na raspolaganju nam je nekoliko ugradjenih funkcija prikazanih u tabeli T10.

Tabela T10 - Lista funkcija za kreiranje matrica

Funkcija	Opis
`rand(m,n)`	Kreira matricu reda *mxn* ciji su elementi slucajni brojevi izmedju 0 i 1
`zeros(m,n)`	Kreira matricu reda *mxn* ciji su svi elementi nule
`ones(m,n)`	Kreira matricu reda *mxn* ciji su svi elementi jedinice
`eye(n)`	Kreira jedinicnu matricu reda *n*

Pri testiranju pojedinih funkcija ili programa koje smo sami napisali ove funkcije nam mogu biti od koristi da bi smo brzo formirali proizvoljne matrice. U nastavku sledi nekoliko primera koriscenja ovih funkcija.

```
>> A=rand(2,3)
A =
    0.6787    0.7431    0.6555
    0.7577    0.3922    0.1712
```

Svaki put kada se pozove ova funkcija dobice se novi skup slucajnih brojeva sa uniformnom raspodelom izmedju 0 i 1.

Za formiranje matrice reda mxn ciji su elementi slucajni brojevi sa intervala [a,b] moze se koristiti sledeci izraz: *a+(b-a)*rand(m,n).* Na primer:

```
B=7+(15-7)*rand(2,3)
B =
   12.5586   14.6018   10.5100
    9.5368    7.2756   10.0525
```

Prethodna naredba generise matricu 2x3 sa slucajnim brojevima sa intervala 7-15.

Sledeca naredba generise kvadratnu matricu reda 3 sa svim elementima jednakim nuli.

```
>> C=zeros(3)
C =
     0     0     0
     0     0     0
     0     0     0
```

Matrica reda 3x4 sa svim elementima jednakim jedinici se moze dobiti naredbom:

```
>> D=ones(3,4)
```

```
D =
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

Ukoliko zelimo da su nam svi elementi matrice jednaki broju 7 treba napisati:

```
>> E=7*ones(3,4)
E =
     7     7     7     7
     7     7     7     7
     7     7     7     7
```

Jedinicna matrica reda 3 se moze dobiti jednostavnom naredbom:

```
>> I=eye(3)
I =
     1     0     0
     0     1     0
     0     0     1
```

Funkcija *eye( )* se moze se koristiti i u sledecem obliku:

```
>> F=eye(3,5)
F =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
```

Kao i kod vektora, za transponovanje matrica koristi se operator (').

```
>> A=round(15*rand(2,3))
A =
     2    14     9
     7     5     3
>> B=A'
B =
     2     7
    14     5
     9     3
```

2.2 Pristup podacima

Pod pristupom podacima podrazumevamo citanje sadrzaja memorijskih lokacija u kojima su sacuvani podaci. Mogucnost pristupa memorijskim lokacijama je neophodna da bi smo promenili njihov postojeci sadrzaj ili da bi smo postojeci sadrzaj upotrebili u matematickim izrazima. Videli smo, nizovi se formiraju tako sto se nekoj memorijskoj lokaciji pridruzuje simbolicko ime (adresa) a onda se u tu lokaciju upisuju vrednosti koje cine elemente niza. Cak i ako mi sami ne damo ime lokaciji u

koju upisujemo vrednosti, Matlab to radi umesto nas dodeljujuci joj naziv `ans`. Naziv je neophodan jer je to jedini nacin da se obratimo nizu odnosno njegovim clanovima. Dakle, sadrzaj neke memorijske lokacije dobijamo koristeci ime te lokacije. Na primer, ako smo ranije kreirali vektor pod imenom `a`, da bi smo videli vrednost njegovih elemenata dovoljno je da u komandnoj liniji otkucamo ime vektora (memorijske lokacije):

```
>> a
a =
     2     5     8     9     3
```

Pored obracanja celom nizu, nekada je potrebno da pristupimo pojedinim calanovima odnosno podgrupama u nizu. Iako svi clanovi niza imaju isti naziv svaki od njih je jednoznacno odredjen putem indeksa. Broj parametara potrebnih za identifikaciju svakog clana niza jednak je dimenziji niza. Kod vektora potreban je jedan indeks, kod matrice dva itd.

K-tom clanu u nizu se pristupa tako sto se prvo navede ime niza a onda u malim zagradama indeks datog clana. Na primer, ako otkucamo :

```
>> a(3)  odgovor ce biti:
ans =
     8
```

Svaki element niza mozemo koristiti kao zasebnu promenljivu u izrazima. Mozemo, na primer, napisati:

```
>> y=sin(a(2))+a(1)
```

Ako znamo da pristupimo odredjenom clanu niza (bez obzira na red) primenom operatora dodele lako mozemo da promenimo postojecu vrednost. Ako otkucamo:

```
>> a(5)=-1;
```

promenicemo vrednost petog clana niza *a*. U to se mozemo uveriti ako u komandnoj liniji otkucamo ime niza:

```
>> a
a =
     2     5     8     9    -1
```

Sledecom naredbom promenicemo vrednost prvog clana u nizu:

```
>> a(1)=a(1)^2+min(a(2),a(3))
a =
     9     5     8     9    -1
```

Kao i kod vektora sve elemente neke unapred definisane matrice mozemo prikazati tako sto u komandnoj liniji otkucamo ime te matrice. Na primer:

```
>> A
A =
    14    13     6    13
     8     9     6    14
```

```
    13    11    10     6
```

Elementima matrice pristupamo navodeci ime matrice i u malim zagradama dva indeksa odvojena zarezom. Prvi indeks se odnosi na vrstu a drugi na kolonu. Na primer, za prethodnu matrice, vrednost elementa iz druge vrste i cetvrte kolone mozemo dobiti tako sto otkucamo:

```
>> A(2,4)
ans =
    14
```

Elemente matrice mozemo koristiti kao samostalne promenljive u izrazima. Na primer:

```
>> y=3+min(A(1,1),A(3,4))
y =
     9
```

Sledecom naredbom promenicemo vrednost elementa iz trece vrste i cetvrte kolone:

```
A(3,4)=-1
A =
    14    13     6    13
     8     9     6    14
    13    11    10    -1
```

Treba primetiti da se pri formiranju nizova koriste uglaste zagrade [ ] dok se pri adresiranju clanova niza koriste male zagrade ( ).

Kao i kod generisanja nizova operator dvotacka (:) ima posebnu namenu. To je neka vrsta dzokera pomocu koga mozemo da izdvojimo podgrupu elemenata. Na primer neka je dat vektor a:

```
a =
     9     5     8     9    -1
```

Ako sada otkucamo:

```
b=a(2:5)
```

formiracemo novi vektor *b* od elemenata niza *a* ciji indeksi su u intervalu od 2 do 5. To jest:

```
b =
     5     8     9    -1
```

Pravilo je sledece: u malim zagradama generisemo niz indeksa koji adresiraju elemente izvornog niza. Na primer ako otkucamo:

```
>> c=a(1 4 5)
```

javice se greska:

```
??? c=a(1 4 5)
          |
Error: Unexpected MATLAB expression.
```

jer lista (`1 4 5`) u malim zagradama niza *a* ne predstavlja niz. Medjutim ako otkucamo:

```
>> c=a([1 4 5])
```

dobicemo ono sto smo i ocekivali. Pomocu `[ 1 4 5]` formirali smo jednodimenzionalni niz cije vrednosti odredjuju indekse elemenata niza *a* cije vrednosti dodeljujemo vektoru *c*. Drugim recima,

prvi, cetvrti i peti clan vektora *a* dodeljujemo vektoru *c*. Bice:

```
c =
     9     9    -1
```

Slicno ako otkucamo:

```
d=a(2,3:5)
```

rezultat ce biti greska jer izraz u zagradama ne generise niz. Ali ako otkucamo:

```
>> d=a([2,3:5])
```

dobicemo novi vektor *d* ciji su elementi 2.,3.,4. i 5. clan vektora *a*.

Pravilnim adresiranjem mozemo lako promeniti vrednost grupe elemenata u vektoru. Sledeca naredba:

```
>> a([2 5])=2
```

drugom i petom clanu niza dodeljuje vrednost 2.

Na slican nacin i kod matrice mozemo izdvajati grupe elemenata. Tipicni primeri upotrebe dvotacke u adresiranju elemenata matice dati su u tabeli T11. Dvotacka oznacava sve elemente iz nekog opsega.

Tabela T11 - Primeri adresiranja elemenata matrice pomocu operatora ":"

Sintaksa	Znacenje
`A(:,n)`	Oznacava sve elemente kolone *n* matrice *A.*
`A(m,:)`	Oznacava sve elemente vrste *m* matrice *A.*
`A(:,m:n)`	Oznacava sve elemente matrice A izmedju kolona *m* i *n.*
`A(m:n,:)`	Oznacava sve elemente matrice A izmedju vrsta *m* i *n.*
`A(m:n,p:q)`	Oznacava sve elemente matrice A izmedju vrsta *m* i *n* i kolona *p* i *q.*

Neka je data matrica A:

```
A =
    11     8     8    11    12
    10     7    10     8    12
     5    13     7    12    10
```

U nastavku je dato nekoliko primera upotrebe operatora ":" kod matrica.

```
>> A(:,4)
ans =
    11
     8
    12
```

Prethodna naredba izdvaja elemente svih vrsta iz cetvrte kolone. Ukoliko zelimo da izdvojimo sve elemente trece vrste pisemo:

```
>> A(3,:)
ans =
     5    13     7    12    10
```

Svi elementi izmedju trece i pete kolone se izdvajaju komandom:

```
>> A(:,3:5)
```

```
ans =
     8    11    12
    10     8    12
     7    12    10
```

Svi elementi prve i druge vrste:

```
>> A(1:2,:)
ans =
    11     8     8    11    12
    10     7    10     8    12
```

Elementi u preseku druge i trece vrste i cetvrte i pete kolone.

```
>> A(2:3,4:5)
ans =
     8    12
    12    10
```

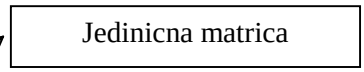
Ukoliko zelimo da izdvojimo nepovezane vrste i kolone mozemo ih pojedinacno navesti. Na primer, sledeca naredba

```
>> B=A([1 3],[2 5])
B =
     8    12
    13    10
```

formira novu matricu B od elemenata matrice A koji se nalzae u preseku prve i trece vrste i druge i pete kolone.

Ukoliko elementima koji se nalaze u preseku prve i trece vrste i druge i pete kolone zelimo da promenimo vrednost mozemo npr. otkucati:

```
>> A([1 3],[2 5])=eye(2)
A =
    11     1     8    11     0
    10     7    10     8    12
     5     0     7    12     1
```



Kao i ranije, kljucno je pravilno adresiranje elemenata niza kojima se menja vrednost.

2.3 Dodavanje i uklanjanje podataka

Do sada smo vidli kako se formiraju novi nizovi. Sada cemo videti kako se postojecim nizovima mogu dodavati novi elemente ili iz njih uklanjati postojeci. Novi clan u nizu se dodaje tako sto se prvo navede ime tog clana u nizu a onda mu se dodeli vrednost. Pri tome novi clanovi u nizu se mogu navoditi pojedinacno ili se moze koristiti operator ":", odnosno sintaksa za uzastopne clanove niza. Na

primer, neka je dat vektor a sa tri clana:

```
a =
     2     5     8
```

cetvrti clan nizu mozemo dodati na sledeci nacin:

```
>> a(4)=1
a =
     2     5     8     1
```

Da bi smo iz niza uklonili odredjeni element potrebno je tom elementu pridruziti prazan niz (prazne uglaste zagrade []). Na primer, drugi clan iz prethodnog niza uklanjamo naredbom:

```
>> a(2)=[]
a =
     2     8     1
```

Kap rezultat, dobijamo nov niz sa tri clana.

Ako nizu dodajemo novi clan koji nije neposredno iza poslednjeg clana tada se medjuclanovi popunjavaju nulama. Naredbom:

```
>> a(6)=-2
```

formiramo od postojeceg novi niz sa sest elemenata:

```
a =
     2     8     1     0     0    -2
```

Posto u trenutku unosenja sestog clana niza nismo definisali vrednosti za cetvrti i peti clan, Matlab im je sam dodelio vrednost nula.

Cetvrti i peti clan novog niza mozemo ukloniti naredbom:

```
>> a([4 5])=[]
a =
     2     8     1    -2
```

I kod dodavanja novih ili brisanja postojecih elemenata niza u malim zagradama mora da stoji izraz kojim se pravilno definise niz. Do istog rezultata kao i gore dosli bi smo naredbom:

```
>> a(4:5)=[]
```

Medjutim, naredba:

```
>> a(4 5)=[]
```

vraca gresku. Razlog je sto izrazi "[4 5]" i "4:5" definisu niz, dok to ne vazi za izraz "4 5".

Interesantno je da od postojeceg vektora lako mozemo formirati matricu dodavanjem novog clana. Naredbu:

```
>> a(2,4)=3
```

Matlab ce protumaciti kao nasu zelju da od postojeceg vektora a formiramo matricu a ciji je element druge vrste i cetvrte kolone jednak broju 3. Posto ostali elementi matrice nisu poznati bice popunjeni nulama:

```
a =
     2     8     1    -2
     0     0     0     3
```

Novoj matrici mozemo dodati treci red naredbom:

```
>> a(3,:)=2:5
a =
     2     8     1    -2
     0     0     0     3
     2     3     4     5
```

Pri koriscenju operatora ":" za formiranje vektora sa istim razmakom izmedju elemenata treba voditi racuna da broj elemenata vektora odgovara broju elemenata vrste ili kolone matrice. Na primer da smo u prethodnoj naredbi otkucali:

```
>> a(3,:)=2:5
```

javila bi se greska:

```
??? Subscripted assignment dimension mismatch.
```

Ako sada zelimo da prethodnoj matrici dodamo petu kolonu popunjenu jedinicama treba da otkucamo:

```
>> a(:,5)=1
```

sto daje rezultat:

```
a =
     2     8     1    -2     1
     0     0     0     3     1
     2     3     4     5     1
```

Matrici mozemo da dodajemo i pojedinacne elemente. Ostali elementi se popunjavaju nulama tako da se odrzi pravougaona struktura. Na primer naredba:

```
>> a(2,6)=9
```

prosiruje matricu a novom (sestom) kolonom tako da je drugi element te kolone jednak broju 9 dok su ostali elementi te kolone nule:

```
a =
     2     8     1    -2     1     0
     0     0     0     3     1     9
     2     3     4     5     1     0
```

Kod uklanjanja elemenata matrice vaze slicna pravila kao i kod vektora. Razlika je samo sto se elementi kojima se dodelju prazan niz ([]) definisu pomocu dva niza. Da bi smo uklonili cetvrtu i sestu kolonu prethodne matice treba otkucati:

```
>> a(:,[4 6])=[]
a =
     2     8     1     1
     0     0     0     1
     2     3     4     1
```

Naredba:

```
>> a(2,:)=[]
```

uklanja drugu vrst matrice:

```
a =
```

```
     2     8     1     1
     2     3     4     1
```

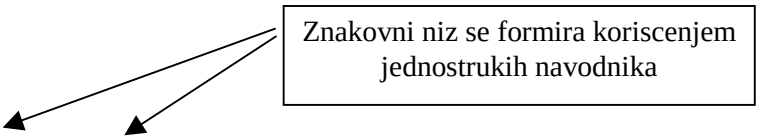
2.4 Znakovni nizovi

Posebnu grupu nizova u Matlab-u cine znakovni nizovi (nizovi znakova). Znakovni niz, cesto poznat pod imenom *string*, predstavlja konacan skup alfanumerickih simbola koji nosi neku informaciju ali nema numericku vrednost. Znakovni niz se formira pomocu simbola koji se unose izmedju jednostrukih navodnika (**'**). Na primer, naredba

```
>> a=[2 1 3]
a =
     2     1     3
```

formira vektor ciji elementi imaju numericku vrednost. Prvi element tog vektora je broj 2, drugi broj 1 i treci broj 3. Sa elementima takvog vektora mozemo obavljati razlicita matematicka izracunavanja. Na primer:

```
>> a(1)^2+3*a(2)/a(3)
ans =
     5
```



Medjutim ako napisemo:

```
>> a='[2 1 3]'
```

formira se niz ciji su elementi simboli koji nemaju numericku vrenost. Na primer broj koji predstavlja masu nekog tela ima svoju numericku vrednost. Medjutim, broj u telefonskom imeniku nema numericku vrednost. Za razliku od prethodnog niza koji je imao tri clana ovaj novi niz ima sedam clanova. Prvi clan je simbol "[', drugi je simbol "2', treci je " " (razmak), itd. U to se mozemo uveriti ako na primer otkucamo:

```
>> a(1)
ans =
[
```

Aritmeticki izrazi sa stringovima ne daju rezultate koje ocekujemo. Na primer naredba

```
>> a(2)^2
```

kao rezultat vraca:

```
ans =
        2500
```

sto nije ocekivani rezultat. Medjutim, stringovi nisu ni predvidjeni da se koriste u matematickim izrazima. Oni imaju drugaciju namenu u Matlab-ui za njih se koriste druge funkcije. Sve ono sto smo rekli kod keriranja, adresiranja, dodavanja i brosanja kod numerickih nizova vazi i kod stringova. Kao sto je vec pokazano znakovna promenljiva se formira pomocu naredbe dodeljivanja:

```
>> ulica='dositejeva 19'
```

```
ulica =
dositejeva 19
```

Elementima niza pristupamo pomocu indeksa:

```
>> ulica(1)
ans =
d
```

ili

```
>> ulica(2:6)
ans =
osite
```

Naredbom dodeljivanja menjamo vrednost clanova niza:

```
>> ulica(1)='D'
ulica =
Dositejeva 19
```

Znakovni niz moze biti predstavljen i u obliku matrice, kao dodimenzionalan niz. Kao i kod brojeva, svaki red znakovne matice predstavlja jednodimenzionalni znakovni niz. Pri unosenju, svaku vrstu zavrsavamo znakom tacka zarez (;). Treba voditi racuna da duzina znakovnog niza u svakoj vrsti bude ista.

```
>> Adresa=['Dositejeva 19 ';'Kraljevo 36000']
Adresa =
Dositejeva 19
Kraljevo 36000
```

Iza simbola "9" u prvoj vrsti namerno je dodat razmak " " da bi duzina stringova prve i druge vsrte bila ista. U suprotnom, da smo otkucali:

```
>> Adresa=['Dositejeva 19';'Kraljevo 36000']
```

rezultat bi bio greska:

```
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

Da nebi smo morali da rucno prebrojavamo znakove u stringovima na raspolaganju nam je funkcija `char` koja ce umesto nas ujednacavati duzinu stringova u matrici znakova. Prethodnu naredbu mozemo otkucati:

```
>> Adresa=char('Dositejeva 19','Kraljevo 36000')
Adresa =
Dositejeva 19
Kraljevo 36000
```

Elementima znakovne matrice mozemo pristupati koriscenjem indeksa:

```
>> Adresa(2,1)
```

```
ans =
K

>> Adresa(2,:)
ans =
Kraljevo 36000

>> Adresa(2,1:3)
ans =
Kra
```

Kao i ranije iz postojece matrice mozemo uklanjati elemente:

```
>> Adresa(2,:)=[]
Adresa =
Dositejeva 19
```

ili dodavati nove:

```
>> Adresa(2,:)='V. Banja 36210'
Adresa =
Dositejeva 19
V. Banja 36210
```

2.5 Funkcije za rad sa nizovima

U Matlab-u postoji dosta ugradjenih funkcija za rad sa nizovima. Ako u komandnom prozoru otkucamo *help elmat* dobicemo listu funkcija za generisanje matrica i njihovu obradu razvrstanih po kategorijama. U tabeli T12 date su neke od njih.

Tabela T12 - Funkcije za rad sa nizovima

Funkcija	Opis
`size(A)`	Vraca broj elemenata duz svake dimenzije u nizu. Kod vektora i matrica rezultat ce biti vektor ciji prvi element predstavlja broj vrsta a drugi element broj kolona u nizu.
`length(a)`	Vraca broj elemenata u vektoru.
`ndims(A)`	Vraca dimenziju niza.
`numel(A)`	Vraca broj elemenata u nizu.
`isempty(A)`	Vraca logicko 1 (True) ako je niz A prazan. Niz je prazan ako mu nisu definisani elementi.
`isequal(A,B)`	Vraca logicko 1 (True) ako su nizovi A i B numericki ekvivalentni. To znaci ako imaju istu dimenziju i jednake vrednosti odgovarajucih elemenata.
`reshape(A,m,n)`	Od postojeceg niza X dimenzije `pxq` formira niz dimenzije `mxn`. Preuredjivanje se vrsi po kolonama. Pri tome mora da vazi `pxq = mxn`.
`diag(a)`	Ako je `a` vektor formira kvadratnu matricu sa elementima na glavnoj

	dijagonali jednakim vektoru `a`.
`diag(A)`	Ako je `A` matrica formira vektor od elemenata njene glavne dijagonale.
`find(A)`	Vraca indekse elemenata niza `A` koji su razliciti od nule.
`end`	Koristi se kao oznaka poslednjeg elementa u nizu. Na primer: `a(3:end)` vraca sve elemente niza `a` osim prva dva.
`isscalar(a)`	Vraca 1 ukoliko je `a` skalar, u suprotnom vraca 0.
`isvector(a)`	Vraca 1 ukoliko je `a` vektor, u suprotnom vraca 0.

3. Matematičke operacije

Nakon sto smo definisali podatke potrebno ih je obraditi na zeljeni nacin. U prethodnom poglavlju vec smo koristili matematicke operacije sa skalarnim podacima. Medjutim, ovde nas pre svega interesuju osnovne matematicke operacije nad nizom kao celinom. Zbog specificnosti strukture podataka, i matematicke operacije u Matlab-u se donekle razlikuju od operacija u drugim alatima. Posto je osnovna struktura podatka niz, pri izvodjenju osnovnih matematickih operacija koriste se pravila linearne algebre.

3.1 Sabiranje i oduzimanje

Za sabiranje i oduzimanje nizova koriste se operatori "+" i "-" kao i kod skalara. Zbir (razlika) dva niza se dodobija tako sto se saberu (oduzmu) odgovarajuci elementi nizova. Time se, na krajnjem nivou, sabiranje i oduzimanje nizova svodi na sabiranje i oduzimanje odgovarajucih skalara. Na primer, neka su dati vektori a i b:

```
a =
     2     1    -3

b =
    -1     3     5
```

Tada je

```
>> a+b
ans =
     1     4     2
```

odnosno

```
>> a-b
ans =
     3    -2    -8
```

Slicno vazi i kod matrica. Na primer za dve matrice A i B:

```
A =
     2     3     0
     4    -2     3
B =
     5     1     2
```

```
     3     9     7
```

imamo

```
>> A+B
ans =
     7     4     2
     7     7    10
```

Pri sabiranju i oduzimanju dimenzije nizova moraju biti jednake. Na primer ako napisemo:

```
>> a+A
```

dobicemo poruku o gresci:

```
??? Error using ==> plus
Matrix dimensions must agree.
```

jer vektor a ima dimenziju 1x3 a matrica A dimenziju 2x3.

Medjutim, u Matlab-u je dozvoljeno sabiranje (oduzimanje) niza i skalara. Operacije se sprovode tako sto se skalar sabira (oduzima) sa svakim clanom niza. Na primer:

```
>> b+7
ans =
     6    10    12
```

ili

```
>> 5-A
ans =
     3     2     5
     1     7     2
```

3.2 Mnozenje

Pri mnozenju dva niza koriste se pravila linearne algebre. Pre svega, nizovi moraju biti saglasni za mnozenje. Ako matrica A ima dimenziju mxn a matrica B pxq, tada da bi matrice mogle da se pomnoze (A*B), mora da vazi n=p. Kao rezultat se dobija matrica C dimenzije mxq. Pri tome treba imati na umu A*B ≠ B*A (ne vazi zakon komutacije). Na primer, neka je matrica A dimenzije 2x2

```
 A =
     2     3
     4    -2
```

i matrica B dimenzije 2x3

```
B =
     5     1     2
     3     9     7
```

tada je rezultat operacije

```
>> C=A*B
```

matrica C dimenzije 2x3

```
C =
    19    29    25
    14   -14    -6
```

Medjutim, rezultat operacije

```
 >> D=B*A
```

je greska

```
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

jer matrice B i A nisu saglasne za mnozenje.

Rezultat proizvoda neke matrice i jedinicne matrice odgovarajuceg reda je sama ta matrica. Na primer

```
>> B*eye(3)
ans =
     5     1     2
     3     9     7
```

Isti rezultat bi smo dobili kao rezultat operacije:

```
>> eye(2)*B
```

Poznato je da je proizvod kvadratne matrice A i njene inverzne matrice (oznaka A^{-1}) jednak jedinicnoj matrici istog reda. Inverzna matrica se moze dobiti kao rezultat operacije `inv(A)` ili `A^-1`. Na primer, za gore definisanu matricu A imali bi smo

```
>> A*inv(A)
ans =
     1     0
     0     1
```

Pomocu operatora za mnozenje (*) mozemo naci stepen matrice. Naime, A^2=A*A ili A^3 = A*A*A. Za prethodno definisanu matricu A imali bi smo:

```
>> A*A
ans =
    16     0
     0    16
```

Naravno i ovde vazi ogranicenje da se operacija stepenovanja moze izvrsiti samo sa kvadratnom matricom.

```
>> B*B
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

I kod mnozenja vektora vazi uslov saglasnosti za mnozenje. To znaci da vektori moraju imati isti broj elemenata i da je jedan vektor vrsta a drugi vektor kolona. Neka je

```
a =
```

```
     2     1    -3
```

i

```
b' =
    -1     3     5
```

gde je `b'` je transponovani vektor vektora `b`. Tada je rezultat proizvoda

```
>> a*b
```

skalar (dimenzija 1x1):

```
ans =
   -14
```

Ovaj rezultat se naziva skalarni proizvod vektora `a` i `b`. U istu svrhu mozemo da koristimo ugradjenu funkciju `dot(a,b)`. Jedini uslov je da vektori a i b imaju isti broj elemenata.

```
>> dot(a,b)
ans =
   -14
```

Rezultat proizvoda (kolone i vrste)

```
>> b*a
```

je kvadratna matrica ciji je red jednak broju elemenata u vektoru.

```
ans =
    -2    -1     3
     6     3    -9
    10     5   -15
```

Pri mnozenju niza i skalara vazi isto pravilo kao kod sabiranja i oduzimanja. Svaki element niza se mnozi skalarom. Za gore definisani vektor `a` imali bi smo:

```
>> a*7
ans =
    14     7   -21
```

Pri mnozenju niza skalarom vazi zakon komutacije.

3.3 Deljenje

U Matlab-u postoje dve operacije deljenja nizova: deljenje sleva (left division) i deljenje zdesna (right division). I jednu i drugu operaciju najcesce koristimo kod resavanja sistema linearnih jednacina.
Matricnu jednacinu

$$AX=B$$

resavamo mnozenjem sleva inverznom matricom A^{-1}:

$$A^{-1}AX=A^{-1}B$$

odakle se dobija:

$$X= A^{-1}B$$

Prethodna jednacina se Matlab-u moze resiti koriscenjem funkcije `inv()`:

```
X=inv(A)*B
```

ili koriscenjem operatora "\" za deljenje sleva:

```
X=A\B
```

Kada su u pitanju veliki sistemi jednacina preporuka je da se koristi deljenje sleva jer daje tacnije rezultate. Kod manjih sistema jednacina rezultati su prakticno isti. Na primer, neka je dat sistem jednacina:

$$2x_1 + x_2 - 3x_3 = -9$$
$$-x_1 + 2x_2 + x_3 = 0$$
$$x_1 - 3x_2 + 4x_3 = 21$$

Sistem se u matricnom obliku moze zapisati:

```
AX=B
```

gde je:

$$A = \begin{bmatrix} 2 & 1 & -3 \\ -1 & 2 & 1 \\ 1 & -3 & 4 \end{bmatrix}; \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}; \qquad B = \begin{bmatrix} -9 \\ 0 \\ 21 \end{bmatrix}$$

Sada se u Matlab-u moze napisati:

```
>> X=A\B
```

sto daje resenje:

```
X =
     2
    -1
     4
```

ili

```
>> X=inv(A)*B
```

sa rezultatom u formatu `long`:

```
X =
   2.000000000000000
  -1.000000000000000
   4.000000000000001
```

Operacija deljenje zdesna (/) se koristi u resavanju matricne jednacine oblika XA=B, gde su X i B vektori vrste. Mnozenjem zdesna inverznom matricom A^{-1} dobijamo:

$$XA^{-1}A=BA^{-1}$$

sto daje

$$X= BA^{-1}$$

Prethodna jednacina se Matlab-u moze resiti koriscenjem funkcije `inv()`:

```
X=B*inv(A)
```

ili koriscenjem operatora "/" za deljenje zdesna:

```
X=A/B
```

Kao i kod deljenja sleva druga operacija daje tacnije rezultate kod sistema linearnih jednacina sa velikim brojem nepoznatih.

Prethodni sistem linearnih jednacina mozemo napisati u obliku:

```
XA=B
```

gde je:

$$A=\begin{bmatrix} 2 & -1 & 1 \\ 1 & 2 & -3 \\ -3 & 1 & 4 \end{bmatrix}; \qquad X=[x_1 \quad x_2 \quad x_3]; \qquad B=[-9 \quad 0 \quad 21]$$

Ako zadamo naredbu:

```
>> X=B/A
```

dobijamo rezultat:

```
X =
     2    -1     4
```

odnosno rezultat naredbe

```
>> X=B*inv(A)
```

daje rezultat (format `long`):

```
X =
   2.000000000000000  -1.000000000000000   3.999999999999999
```

3.4 Operacije nad pojedinacnim elementima nizova

Kao sto smo videli, kod sabiranja dva niza sabiraju se pojedinacno odgovarajuci elementi dva niza. Kod mnozenja to ne vazi vec se mnozenje izvodi prema pravilima linearne algebre. Medjutim, nekada je potrebno da se i mnozenje dva niza realizuje kroz pojedinacno mnozenje odgovarajucih elemenata. U tu svrhu u Matlab-u se koristi operator (.* - ispred operatora za mnozenje dopisana je tacka). Na primer, ako za definisane vektore:

```
a =
     2     1    -3
```

i

```
b =
    -1     3     5
```

otkucamo:

```
>> c=a*b
```

javice se greska jer vektori nisu saglasni za mnozenje

```
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Medjutim ako otkucamo

```
>> c=a.*b
```

kao rezultat dobicemo novi vektor

```
c =
    -2     3   -15
```

pri cemu vazi: `c(1)=a(1)*b(1);  c(2)=a(2)*b(2); c(3)=a(3)*b(3).`

Drugim recima obavili smo mnozenje nad pojedinacnim elementima nizova `a` i `b`.

Pored operatora mnozenja i operatori (^ - stepenovanja, / - deljenja zdesna i deljenja sleva \) imaju svoje modifikacije za primenu nad pojedinacnim clanovima niza. U tabeli T3.1 date su modifikacije ovih operatora.

Tabela T3.1 - Operatori nad pojedinacnim elementima

Operator	Opis
`.*`	Pojedinacno mnozenje
`.^`	Pojedinacno stepenovanje
`./`	Pojedinacno deljenje zdesna
`.\`	Pojedinacno deljenj sleva

Na primer za matrice

```
A =
     2     5     4
     3     9     8
```

i

```
B =
     1     2     4
     3     2     4
```

rezultat naredbe

```
>> A./B
```

je

```
ans =
    2.0000    2.5000    1.0000
    1.0000    4.5000    2.0000
```

Ili

```
>> A.^2
```

daje

```
ans =
     4    25    16
     9    81    64
```

Operacije nad pojedinacnim elementima mogu biti veoma korisne kod izracunavanja (crtanja) vrednosti funkcije u nizu tacaka. Na primer ako zelimo da odredimo vrednost funkcije:

$$y = \frac{4x}{x^2 + 1}$$

u nizu tacaka sa intervala [0:10] prvo definisemo vektor nezavisne promenljive:

```
>> x=1:10
x =
     1     2     3     4     5     6     7     8     9    10
```

Zatim otkucamo matematicki izraz koristeci operatore za operacije nad pojedinacnim elementima:

```
>> y=4*x./(x.^2+1)
y =
  Columns 1 through 6
    2.0000    1.6000    1.2000    0.9412    0.7692    0.6486
  Columns 7 through 10
    0.5600    0.4923    0.4390    0.3960
```

Treba primetiti da ispred operatora "*" i "+" u gornjem izrazu nismo koristili tacku jer se operacije sabiranja i mnozenja skalarne velicine sa nizom i inace primenjuju pojedinacno nad elementima niza.

Kada se ugradjenim funkcijama u Matlab-u kao argument prosledi niz tada se operacije definisane funkcijom izvode nad svakim clanom niza pojedinacno. Na primer ako matricu

```
A =
     4    25     9
    16    49     1
```

prosledimo kao argument funkciji `sqrt()` dobicemo kao rezultat:

```
>> sqrt(A)
ans =
     2     5     3
     4     7     1
```

sto pokazuje da je operacija korenovanja primenjena nad svakim clanom niza. Odnosno kao da smo za svaki element matrice A pozivali funkciju `sqrt()`sa vrednoscu tog elementa kao argumentom.

3.5 Ugradjene funkcije za rad sa nizovima

U Matlab-u postoji dosta ugradjenih funkcija za analiziranje nizova. U tabeli T3.2 date su neke od njih.

Tabela T3.2 - Ugradjene funkcije za rad s nizovima

Funkcija	Opis
`inv(A)`	Odredjuje inverznu matricu kvadratne matrice A
`det(A)`	Odredjuje determinantu kvadratne matrice A
`sum(A)`	Ako je A vektor vraca zbir njegovih elemenata. Ako je A matrica vraca vektor vrstu ciji elementi predstavljaju zbirove elemenata kolona matrice
`mean(A)`	Ako je A vektor vraca srednju vrednost njegovih elemenata. Ako je A matrica vraca vektor vrstu sa srednjim vrednostima kolona matrice.
`sort(A)`	Ako je A vektor sortira elemente vektora u rastucem redosledu. Ako je A matrica sortiranje primenjuje na svaku kolonu matrice pojedinacno.
`e=max(A)`	Ako je A vektor, e je najveci element vektora. Ako je A matrica e je vektor vrsta sa najvecim elementima kolona matrice A
`[e n]=max(A)`	Ako je A vektor, e je najveci element vektora a n je redni broj elementa. Ako je A matrica e je vektor vrsta sa najvecim elementima kolona matrice A a n je vektor vrsta sa rednim brojevima najvecih elemenata po kolonama.
`e=min(A)`	Kao i za `e=max(A)` ali za najmanji element
`[e n]=min(A)`	Kao i za `[e n]=max(A)` ali za najmanji element
`dot(a,b)`	Izracunava skalarni proizvod vektora a i b
`cross(a,b)`	Izracunava vektorski proizvod vektora a i b. Vektori moraju da imaju po tri elementa (koordinate).

4. Grafikoni

Analiza vece kolicine podataka i donosenje zakljucaka na osnovu njih, moze biti znatno olaksano ukoliko postoji mogucnost vizuelizacije tih podataka. Kaze se da jedna slika vredi vise od hiljadu reci. Matlab poseduje veoma mocan i fleksibilan sistem za graficko predstavljanje podataka.

4.1 Graficko prikazivanje podataka

U tehnici se za vizuelno predstavljanje podataka najcesce koriste ravanski, dvodimenzionalni (2D) grafikoni. Komanda za crtanje 2D grafikona u najjednostavnijem obliku ima sintaksu:

```
plot(x,y)
```

Argumenti `x` i `y` su unapred definisani vektori sa istim brojem elemenata. Elementi prvog vektora (`x`) u pozivu funkcije se nanose duz horizontalne ose (apcise) a elementi drugog vektora (`y`) duz vertikalne ose (ordinate) u pravouglom koordinatnom sistemu. Mnogo vise informacija o ovoj funkciji dobicemo ako u komandnom prozoru otkucamo `help plot`.

Neka je izvrseno 10 merenja nivoa i temperature tecnostu u dva rezervoara. Rezultati merenja prikazani su u tabeli T4.1.

T4.1 Rezultati merenja

R.b. merenja	1	2	3	4	5	6	7	8	9	10
Rezervoar I										
Visina [mm]	270	280	290	260	240	235	225	215	210	215
Temperatura [^{o}C]	18	17	16	19	21	21	22	22	23	22
Rezervoar II										
Visina [mm]	280	275	290	280	250	240	230	220	215	215
Temperatura [^{o}C]	18	18	17	19	21	20	20	22	21	23

Ako sada zelimo da graficki prikazemo ove rezultate merenja prvo cemo formirati dva vektora:

```
>> n=1:10
n =
     1     2     3     4     5     6     7     8     9    10
```

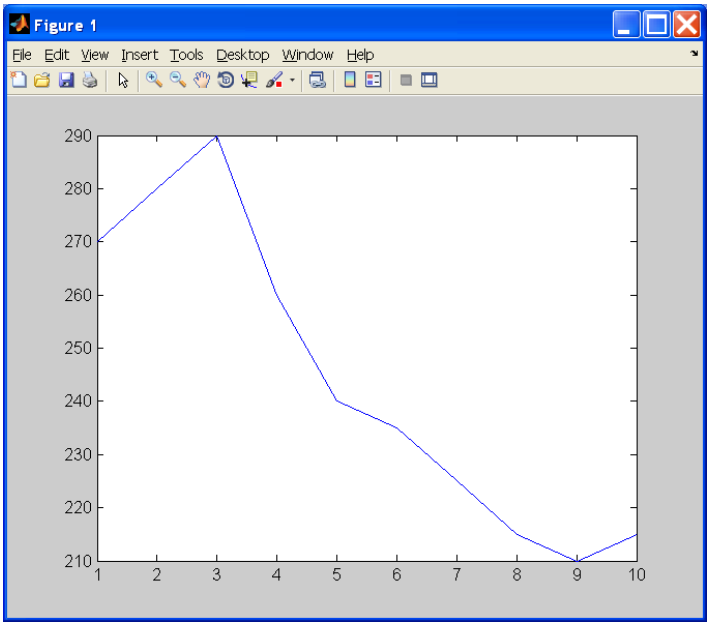
i

```
>> h1=[270  280   290   260   240   235   225   215   210   215];
>> T1=[18 17      16    19    21    21    22    22    23    22];
>> h2=[280 275    290   280   250   240   230   220   215   215];
>> T2=[18   18    17    19    21    20    20    22    21    23];
```

a onda se pozivom funkcije

```
>> plot(n,h1)
```

otvara graficki prozor pod nazivom `Figure 1` sa dijagramom promene nivoa tecnosti kao sto je to prikazano na slici Sl.4.1. Grafik se sastoji od pravolinijskih segmenata koji povezuju tacke cije su koordinate definisane elementima vektora `n` i `h1`.

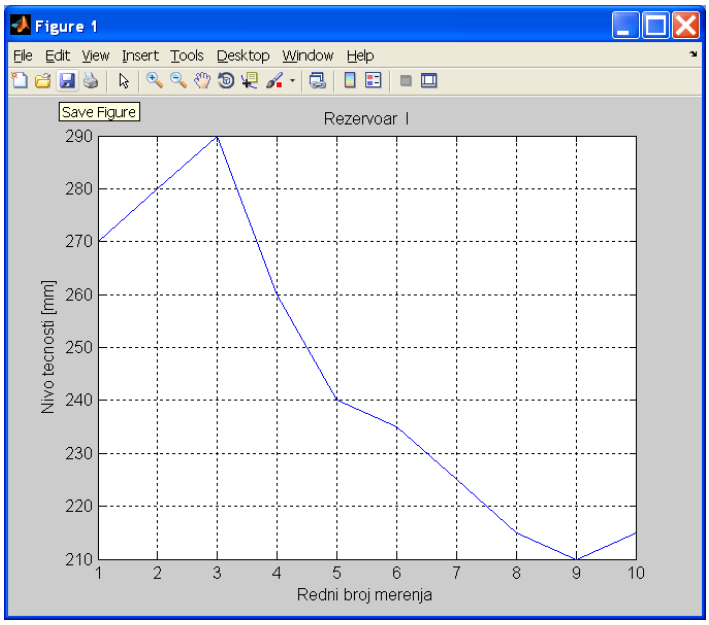


Sl.4.1 Osnovni grafik promene nivoa tecnosti u rezervoaru

Da bi smo malo blize objasnili sliku koju smo dobili dodacemo joj naslov, oznake za horizontalnu i vertikalnu osu, pozadinsku mrezu. U tu svrhu se koriste sledece naredbe:

`>> title('Rezervoar I')`	Dodaje naslov.
`>> xlabel('Redni broj merenja')`	Dodaje naziv za horizontalnu osu.
`>> ylabel('Nivo tecnosti [mm]')`	Dodaje naziv za vertikalnu osu.
`>> grid on`	Dodaje pozadinsku mrezu.

Rezultati su prikazani na slici Sl.4.2



Sl.4.2 Dodavanje oznaka na grafik

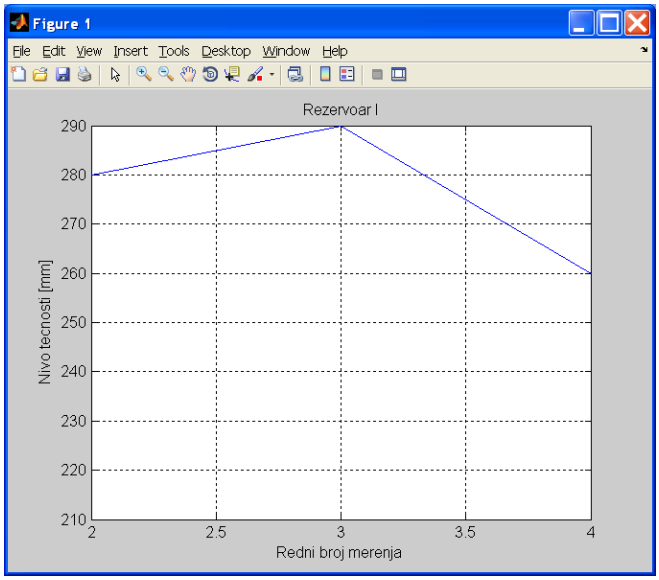
Program podrazumevano skalira horizontalnu i vertikalnu osu na osnovu minimalne i maksimalne vrednosti nezavisne i zavisne promenljive. Ukoliko zelimo da promenimo opseg prikazanih vrednosti mozemo da koristimo funkciju:

```
axis([xmin,xmax,ymin,ymax])
```

pomocu koje u uglastim zagradama zadajemo granice za horizontalnu (xmin, xmax) i vertikalnu osu (ymin, ymax). Na primer, komanda:

```
>> axis([2,4,210,290])
```

umesto prethodnog, kao rezultat daje grafik prikazan na slici Sl.4.3.



Sl.4.3 Promena opsega prikazanih vrednosti

Ako sada zelimo da prikazemo promenu nivoa u Rezervoaru II, posto smo podatke vec definisali,

dovoljno je da otkucamo naredbu:

```
>> plot(n,h2)
```

Medjutim, Matlab ce prvo izbrisati prethodni dijagram i u istom prozoru (`Figure 1`) nacrtati novi. Posle toga morali bi smo ponovo da kucamo komande za dodavanje oznaka na dijagram. Ako zelimo da novi dijagram nacrtamo u novom prozoru, ne brisuci prethodni, pre naredbe za crtanje treba otkucati naredbu:

```
>> figure
```

Ova naredba otvara novi graficki prozor pod imenom `Figure 2`, odnosno `Figure n+1` ako je prethodno vec otvoreno `n` prozora. Novootvoreni prozor postaje aktivan i sve naredbe koje se ticu grafike odnose se na ovaj prozor. Aktivan prozor mozemo da menjamo tokom rada komandom `figure(n)` gde je `n` redni broj prozora kojeg aktiviramo. Graficke prozore zatvaramo naredbom `close` odnosno `close(n)`.

Dakle, tek posto smo aktivirali novi prozor kucamo komandu:

```
>> plot(n,h2)
```

da bi smo nacrtali novi grafik u novom prozoru a sacuvali prethodni u postojecem.

Za graficko prikazivanje podataka mozemo da koristimo i funkciju `line` koja u osnovnoj verziji ima oblik:

```
>> line(x,y)
```

Naredba `line` ima slicnu sintaksu kao i naredba `plot`. Osnovna razlika je sto naredba `plot` pre iscrtavanja novog grafika brise prethodni sadrzaj aktivnog grafikona dok naredba `line` samo dodaje novi grafik na postojeci.

4.2 Graficko prikazivanje funkcija

Crtanje grafika funkcije u osnovi se svodi na graficko prikazivanje podataka. Potrebno je prethodno za ciljne funkcije formirati skupove podataka a onda koriscenjem naredbi `plot` ili `line` dobiti njihove grafike. Na primer, neka je data funkcija:

$$y(x) = e^{-0.5x}\sin(3x)$$

za koju je potrebno nacrtati grafik na intervalu [0-10].

Prvo cemo formirati vektor nezavisne velicine `x`:

```
>> x=0:0.01:10;
```

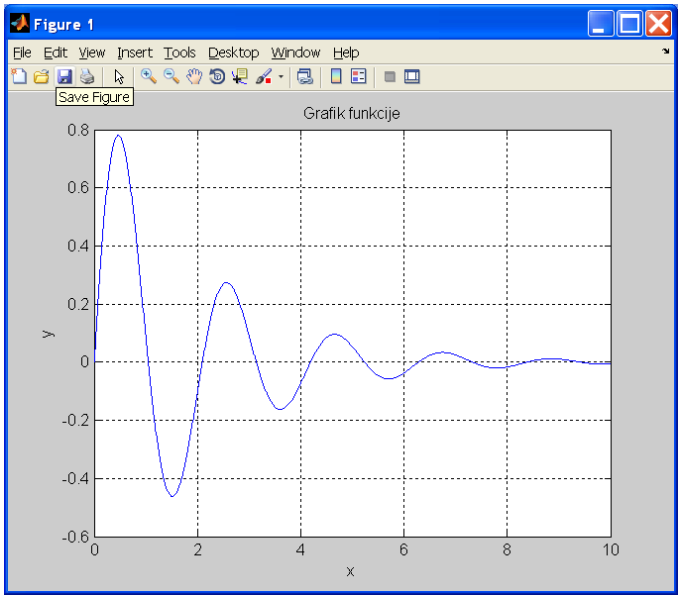
To je vektor od 1001 clana sa konstantnim rastojanjem od 0.01. Sada formiramo vektor `y` za sve elemente vektora `x`:

```
>> y=exp(-0.5*x).*sin(3*x);
```

Primenom naredbe

```
>> plot(x,y)
```

i naredbi za dodavanje oznaka mozemo dobiti grafik funkcije prikazan na slici Sl.4.4



Sl.4.4 Graficki prikaz funkcije

Pri crtanju funkcija treba voditi racuna o rezoluciji vektora nezavisne velicine. Prethodni grafik je dobijen za vektore `x` i `y` koji imaju po 1001 element. Ako sada smanjimo rezoluciju vektora `x`:

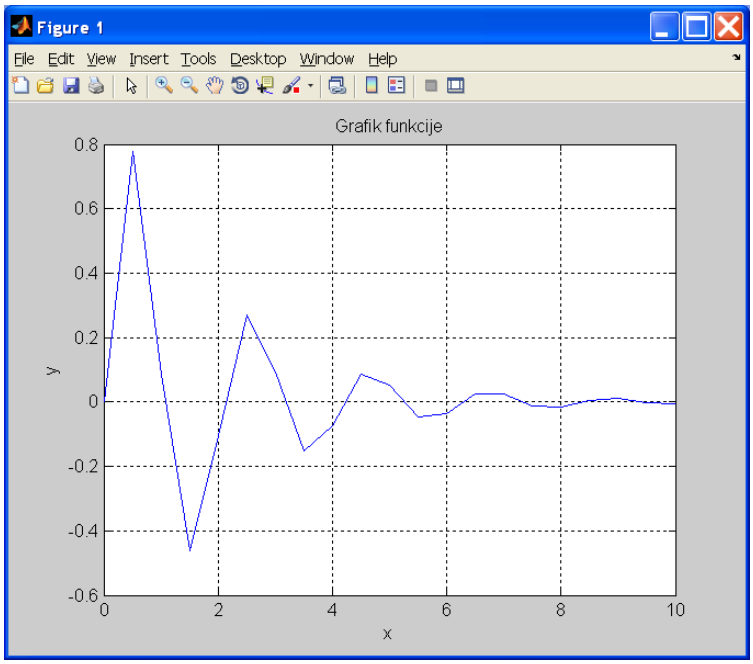
```
>> x=0:0.5:10;
```

odnosno povecamo korak izmedju elemenata na 0.5, duzina vektora bice 21. Ponovo pokrenuta naredba:

```
>> y=exp(-0.5*x).*sin(3*x);
```

ce formirati novi vektor `y` iste duzine.

Grafik iste funkcije smanjene rezolucije prikazan je na slici Sl.4.5.



Sl.4.5 Graficki prikaz funkcije smanjene rezolucije

Zbog smanjenog broja tacaka (21 par umesto 1001) pomocu kojih se crta funkcije grafik sada predstavlja manje tacnu aproksimaciju pravog izgleda funkcije.

Za prikazivanje grafika funkcija mozemo da koristimo i naredbu `fplot` cija je osnovna sintaksa:

```
fplot('funkcija',granice)
```

Prvi argument ove naredbe je znakovni niz koji predstavlja funkciju koju zelimo da nacrtamo. Drugi argument definise granice nezavisne promenljive. Na primer, prethodnu funkciju mozemo da nacrtamo naredbom:

```
>> fplot('exp(-0.5*x).*sin(3*x)',[0 10]);
```

To znaci da ne moramo prvo formirati vektore x i y. Dovoljno je da izraz za funkciju otkucamo unutar jednostrukih navodnika a u uglastim zagradma definisemo granice za nrzavisnu promenljivu.

4.3 Grafikoni sa vise grafika

Ukoliko zelimo da u istom prozoru, na istom grafikonu, istovremeno prikazemo promenu dve ili vise zavisne velicine na raspolaganju su nam tri mogucnosti.

Prva mogucnost je da prosirimo listu argumenata u naredbi `plot` tako da svakom grafiku odgovara po jedan par vektora. Na primer, da bi smo na istom grafikonu prikazali promenu nivoa u u oba rezervoar mozemo da otkucamo naredbu:

```
>> plot(n,h1,n,h2)
```

Sada naredba `plot` ima cetiri argumenta. Prva dva se odnose na prvu krivu a treci i cetvrti na drugu. Rezultat ove naredbe prikazan je na slici Sl.4.6. Listu argumenata i dalje mozemo prosirivati ali moramo uvek imati paran broj vektora i svaki par uzastopnih vektora mora imati isti broj elemenata.

Ukoliko krive koje zelimo da nacrtamo na istom grafikuonu nemaju istu skalu vrednosti na raspolaganju nam je funkcija:

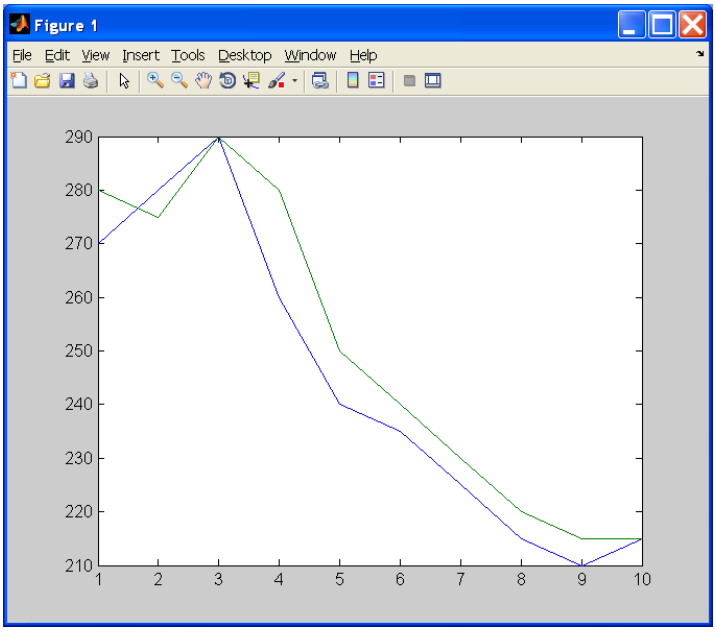
```
>> plotyy(x1,y1,x2,y2)
```

Ova funkcija u istom koordinatnom sistemu crta dve y ose - jednu sa leve strane a drugu sa desne strane grafikona. Prva dva argumenta u pozivu funkcije se odnose na prvu krivu a duga dva na drugu. Ukoliko je `x1=x2` funkcije se crtaju nad istim skupom vrednosti nezavisne promenljive. U opstem slucaju $x1 \neq x2$. Jedino ogranicenje je da je broj elemenata para `x1,y1` odnosno `x2,y2` jednak.

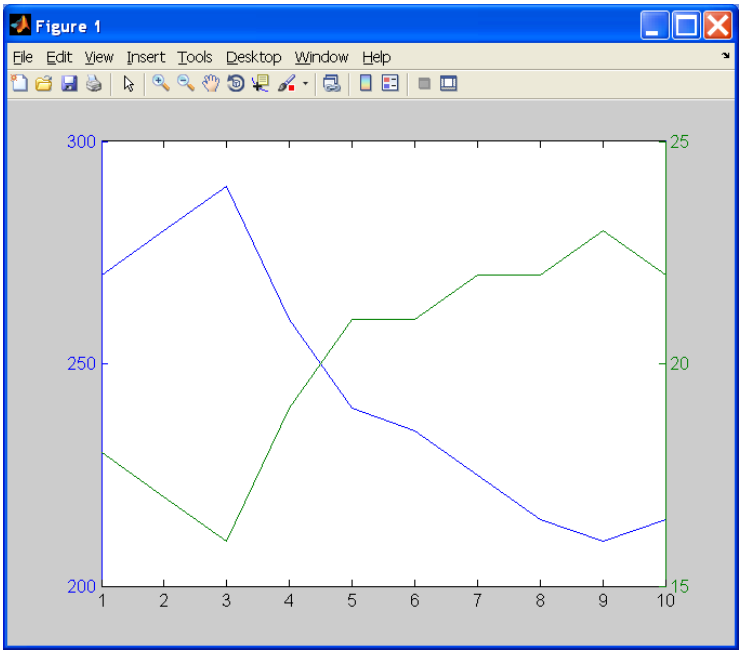
Ako bi smo u nasem primeru na istom grafikonu nacrtali promenu nivoa i temperature jednog rezervoara pomocu naredbe `plot` prikaz ne bi bio jasan jer su intervali vrednosti ovih velicina razliciti. Medjutim, pomocu naredbe `plotyy` mozemo prikazati grafikone sa dva opsega vrednosti zavisnih promenljivih. Jedan opseg vrednosti na levom kraju a drugi na desnom kraju grafikona. Na primer, ako otkucamo

```
>> plotyy(n,h1,n,T1)
```

dobicemo grafik kao onaj prikazan na slici Sl.4.7.



Sl.4.6 Crtanje dve krive na istom grafikonu



Sl.4.7 Grafikoni krivih sa razlicitim opsegom vrednosti zavisnih promenljivih

Time postizemo da na istom grafikonu prikazemo promenu nivoa tecnosti u intervalu 200-300 mm i promenu temperature u interlvalu 15-25 ^{o}C. Grafici i odgovarajuce vertikalne ose prikazani su istom bojom.

Druga mogucnost je da prvo nacrtamo jednu krivu a onda, pre crtanje druge, otkucamo naredbu:

```
>> hold on
```

Posle ove naredbe novi grafik ce biti pridodat postojecem na istom grafikonu. I u ovom slucaju rezultat ce biti isti kao na slici Sl.4.6.

Dejstvo naredbe `hold on` ostaje da vazi i za buduce krive sve dok se naredba ne iskljuci sa `hold off` ili dok se ne zatvori prozor koji je bio aktivan u trenutku kada smo otkucali naredbu `hold on`.
Treca mogucnost istovremenog prikaza vise grafika na istom grafikonu zasniva se na koriscenju naredbe `line` koju smo prethodno pomenuli.
Ukoliko zelimo da u istom grafickom prozoru prikazemo vise grafikona koristimo naredbu `subplot(m,n,p)`. Ova naredba formira nov (deli aktivan) graficki prozor na pravougaonu matricu od `mxn` grafikona u kojoj je grafikon sa rednim brojem `p` aktivan. Redni brojevi rastu duz vrste od prve do poslednje. Nakon ovoga naredba `plot` crta grafik u aktivnom grafikonu. Aktivni grafikon menjamo ponovnim pozivanjem funkcije `subplot` sa novim rednim brojem. Na primer ukoliko zelimo da sve cetiri velicine oba rezervoara (`h1, T1, h2, T2`) prikazemo u istom grafickom prozoru treba otkucati:

prvi grafikon

```
>> subplot(2,2,1)
>> plot(n,h1)
>> grid on
>> title('Nivo R-I')
```

drugi grafikon

```
>> subplot(2,2,2)
>> plot(n,T1)
>> grid on
>> title('Temperatura R-I')
```

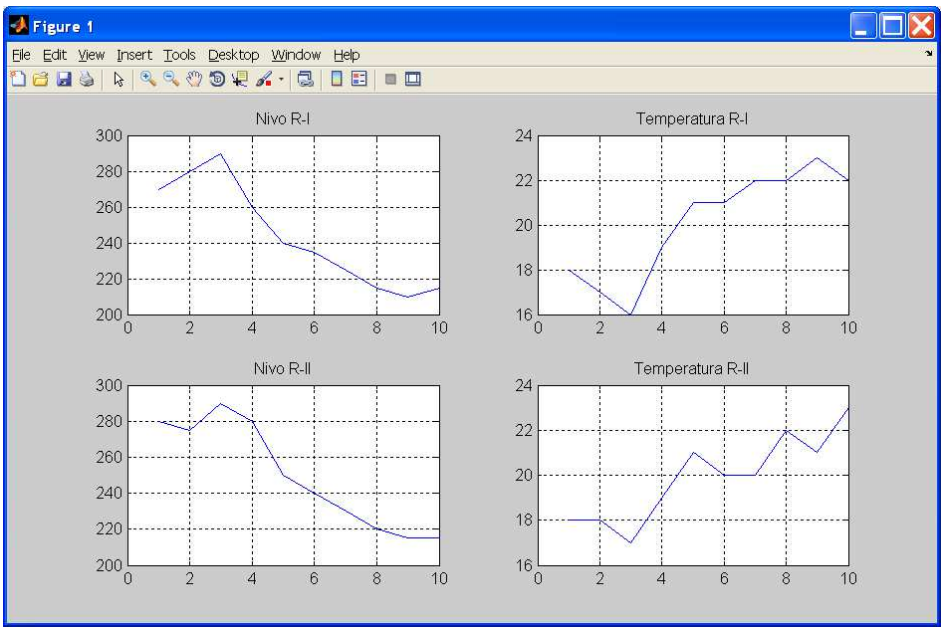
treci grafikon

```
>> subplot(2,2,3)
>> plot(n,h2)
>> grid on
>> title('Nivo R-II')
```

cetvrti grafikon

```
>> subplot(2,2,4)
>> plot(n,T2)
>> grid on
>> title('Temperatura R-II')
```

Rezultati ova cetiri skupa naredbi prikazan je na slici Sl.4.8.



Sl.4.8 Prikazivanje vise grafikona u istom grafickom prozoru

4.4 Formatiranje grafikona

Pri prikazivanju grafikona sa vise grafika pozeljno je za razlicite krive koristiti razlicite formate (boje, debljine, tipove, markere) kako bi se povecala citljivost i razumljivost prikazanih podataka. Iako smo do sada samo koristili osnovni oblik naredbe `plot` u opstem slucaju njena sintaksa glasi:

```
plot(x,y,'OznakaLinije','ImeSvojstva', VrednostSvojstva)
```

Svi argumenti osim prva dva (vektori podataka) su opcioni. `OznakaLinije` je znakovni niz kojim se definise tip i boja linije i/ili markera. U tabeli T4.1 date su oznake za nekoliko tipova linija a u tabeli T4.2 oznake za vrstu markera.

Tabela T4.1 Oznake za tip linije

Oznaka	Opis	Oznaka	Opis
-	Puna linija (podrazumevani tip).	:	Tackasta linija
- -	Isprekidana linija.	- .	Crta tacka linija.

Tabela T4.2 Oznake za tip markera

Oznaka	Opis	Oznaka	Opis
.	Tacka	d	romb
o	Kruzic	v	trougao (dole)
x	Znak x	^	trougao (gore)
+	Znak +	<	trougao (levo)
*	Zvezda	>	trougao (desno)
s	Kvadrat	p	petokraka

Za podesavanje boja linija i markera koriste se oznake prikazane u tabeli T4.3.

Tabela T4.3 Oznake za boju

Oznaka	Opis	Oznaka	Opis
b	Plava	m	Magenta
g	Zelena	y	Zuta
r	Crvena	k	Crna
c	Cijan	w	Bela

Kao sto smo ranije rekli, graficki prikaz podataka se sastoji od pravolinijskih segmenata koji povezuju tacke cije su koordinate definisane vektorima - argumentima funkcije `plot`. U podrazumevanom prikazu (Sl.4.1) tacke koje predstavljaju podatke su utopljene u pravolinijske segmente i ne mogu se raspoznati u grafikonu. Ukoliko zelimo da se na grafikonu vide samo tacke koje predstavljaju podatke treba da koristimo markere. Na primer, naredba:

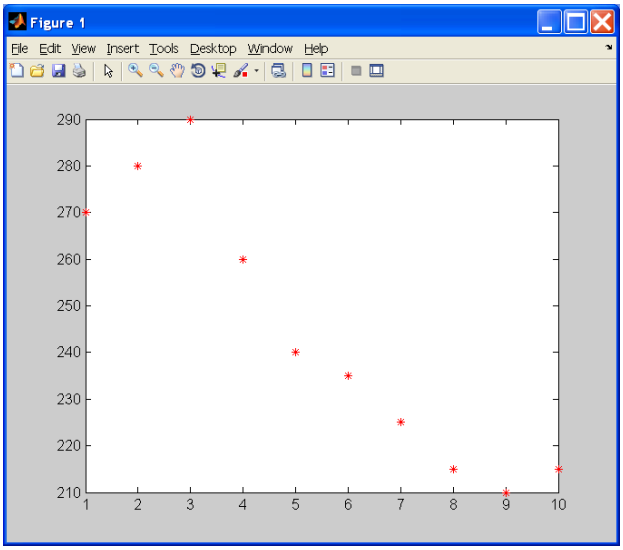
```
>> plot(n,h1,'r *')
```

prikazuje pojedinacno nivoe tecnosti pri svakom merenju oznacene zvezdicom crvene boje (Sl.4.9).

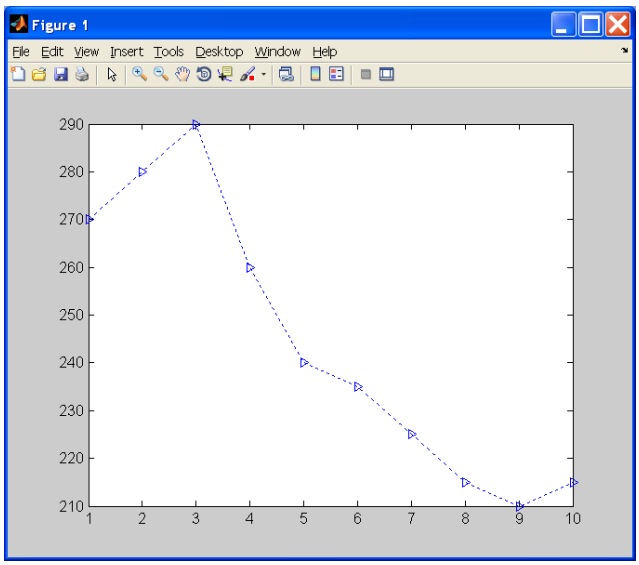
Ukoliko zelimo da se vide i podaci i segmenti koji ih povezuju mozemo da, na primer, koristimo naredbu oblika:

```
>> plot(n,h1,'b: >')
```

Rezultat je grafik na kojem su izmerene vrednosti prikazane markerima u obliku trougla a trend je prikazan tackastim linijama (Sl.4.10)



Sl.4.9 Graficko prikazivanje podataka pomocu markera



Sl.4.10 Graficko prikazivanje podataka pomocu markera
i pravolinijskih segmenata

Treci argument u prethodnoj naredbi plot je znakovni niz koji predstavlja neko imenovano svojstvo koje se odnosi na linije i markere. U tabeli T4.4 data ja lista imenovanih svojstava.

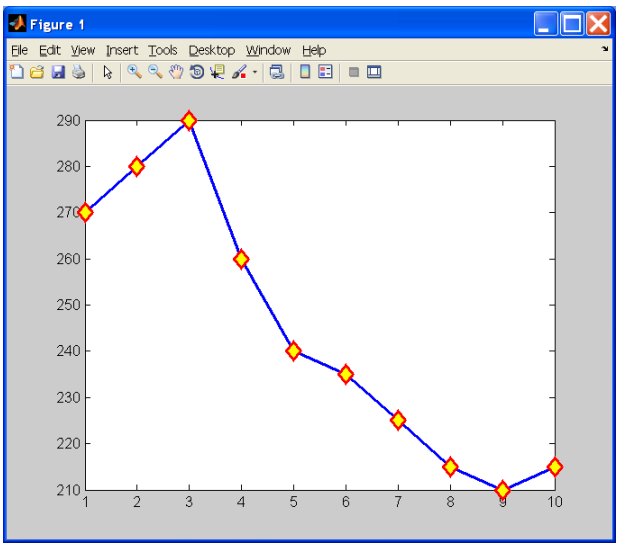
Tabela T4.3 Oznake za boju

Oznaka	Opis
`LineWidth (ili linewidth)`	Debljina linije
`MarkerSize (ili markersize)`	Velicina markera
`MarkerEdgeColor (ili markeredgecolor)`	Boja markera ili boja ivice za popunjene markere
`MarkerFaceColor (ili markerfacecolor)`	Boja popune za markere

Na primer, naredba:

```
>>plot(n,h1,'b-d', 'LineWidth', 2,'markersize',10,
'MarkerFaceColor','y','markeredgecolor','r')
```

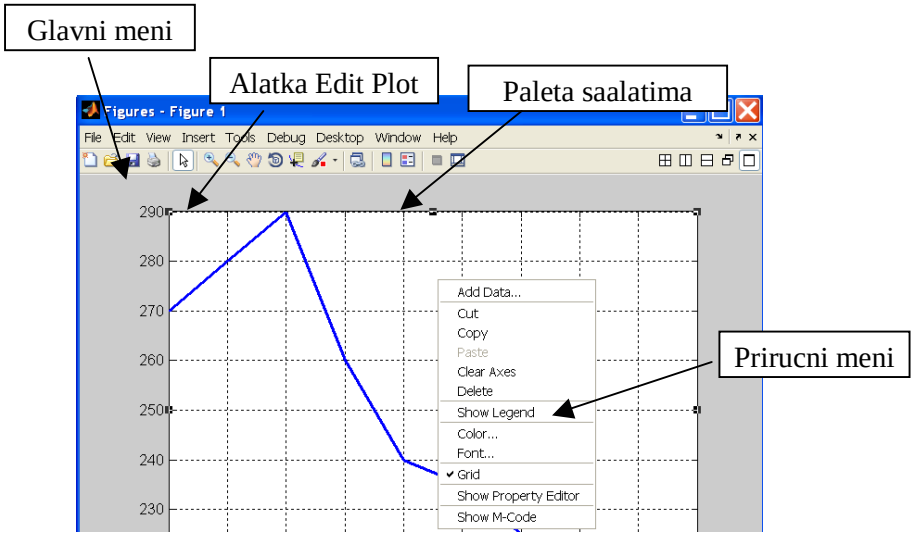
crta grafik u kojem su mereni podaci prikazani markerima u obliku romba, velicine 10, sa ivicama crvene boje i bojom popune zute boje. Linijski segmenti su prikazani punom linijom plave boje (Sl.4.11).



Sl.4.11 Grafikon formatiran imenovanim svojstvima

Vise informacija o opcijama naredbe `plot` moze se dobiti ako se u komandnom prozoru otkuca `help plot`.

Pored komandnog prozora za rad sa grafikonom mozemo da koristimo i graficki korisnicki interfejs (GUI) vezan za graficki prozor. Ovaj GUI nam omogucava da brzo i jednostavno uredjujemo grafik koriscenjem komandi iz glavnog menija, palete sa alatkama ili iz prirucnog menija (Sl.4.12). Pritiskom na alatku `Edit Plot` (strelica na paleti sa alatima) ulazimo u rezim za editovanje grafikona. Posle toga pokazivacem misa biramo objekat koji zelimo da uredjujemo. Ponovnim pritiskom na alatku `Edit Plot` izlazimo iz rezima za editovanje grafikona.



Sl.4.12 Uredjivanje grafikona pomocu GUI-a

4.5 Promena koordinatnog sistema

Sve dosadasnje grafike crtali smo u pravouglim koordinatnim sistemima sa linearnom podelom na osama. U praksi je cesto potrebno, zbog velikog opsega prikazanih podataka, uvesti logaritamsku podelu (sa osnovom 10) po jednoj i/ili drugoj osi. U tabeli T4.4 data je lista funkcija za crtanje grafika u koordinatnim sistemima sa logaritamskom podelom.

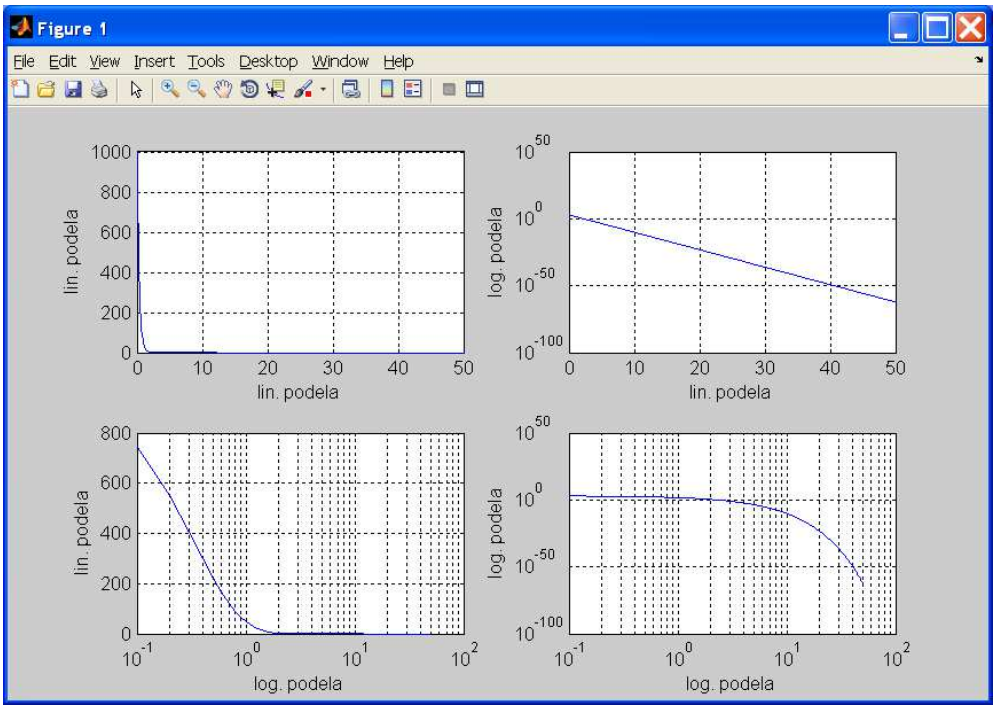
Tabela T4.4 Funkcije za crtanje grafika u logaritamskim koordinatnim sistemima

Funkcija	Opis
`semilogy(x,y)`	Crta grafik sa linearnom podelom horizontalne ose i logaritamskom podelom vertikalne ose.
`semilogx(x,y)`	Crta grafik sa logaritamskom podelom horizontalne ose i linearnom podelom vertikalne ose.
`loglog(x,y)`	Crta grafik sa logaritamskom podelom duz obe ose.

I na ove funkcije se mogu primeniti formatiranja kao i kod funkcije `plot`. Na slici Sl.4.13 prikazani su grafici iste funkcije:

```
>> y=1e3*exp(-3*x);
```

na intervalu [0:0.1:50] u koordinatnim sistemima sa razlicitom podelom duz osa. Treba primetiti da na osi sa logaritamskom podelom nije moguce prikazati vrednosti koje su manje ili jednake nuli (funkcija `log` nije definisana za te vrednosti).



Sl.4.13 Prikazi grafika u koordinatnim sistemima sa razlicitom podelom

Ponekad su grafici ilustrativniji ukoliko se podaci prikazuju u polarnim koordinarnim sistemima. Polozaj tacke u ravni je odredjen uglom koji poteg izmedju koordinatnog pocetka i te tacke zaklapa sa pozitivnim krajem horizontalne ose i rastojanjem izmedju koordinatnog pocetka i te tacke. U tom

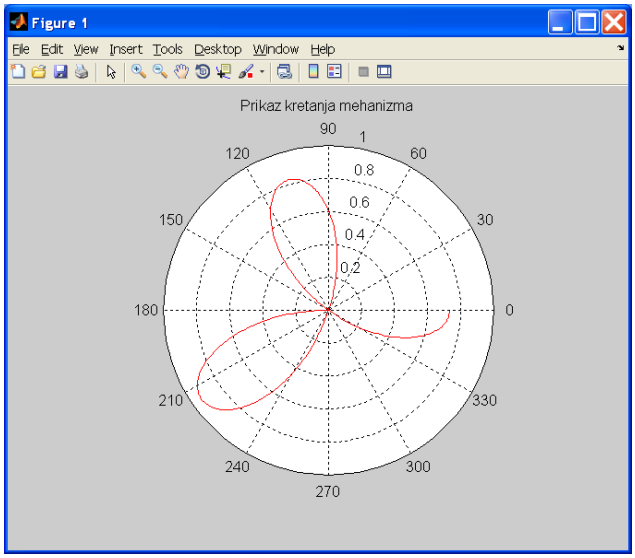
slucaju se umesto funkcije `plot` koristi funkcija `polar` oblika:

```
polar(t,r,'OznakaLinije')
```

Argumenti `t` i `r` su vektori podataka koji definisu polozaj (ugao, rastojanje) svake tacke u ravni. Treci argument je opcioni i ima isto znacenje kao i kod naredbe `plot`. Na primer blok naredbi:

```
>> t=[0:0.01:pi];
>> r=exp(-0.1*t).*cos(2.5*t+pi/2);
>> polar(t,r,'r-')
>> title('Prikaz kretanja mehanizma')
```

daje grafik prikazan na slici Sl.4.14. Dodatna podesavanja se mogu sprovesti u grafickom editoru. Kao i kod funkcije `plot`, pre poziva funkcije za crtanje moraju se definisati vektori ulaznih podataka(`t`, `r`).



Sl.4.14 Prikaz podataka u polarnom koordinatnom sistemu

4.6 Promena tipa grafikona

U svim dosadasnjim primerima koristili smo linijske grafikone. Tacke koje su definisane vektorima podataka u funkcijama za crtanje medjusobno su povezane linijskim segmentima. U nekim slucajevima (na primer pri statistickoj obradi podataka) podaci se bolje reprezentuju koriscenjem drugih tipova grafikona. Medju njima su trakasti, kruzni, konturni, dijagrami za opis raspodele podataka, dijagrami sa animacijom, itd. Kompletan spisak sa objasnjenjem osnovnih i specijalizovanih grafikona moze se naci u sistemu za pomoc pod stavkom Graphics.

5. Skript datoteke

Komandni prozor predstavlja jedan nacin izvrsavanja naredbi u Matlab-u. Drugi nacin je da neredbe koje zelimo da izvrsimo prvo snimimo u datoteku a onda pokretanjem datoteke izvrsimo sve naredbe. Rad u komandnom prozoru nije podesan kada treba izvrsiti niz naredbi koje su medjusobno povezane. Na primer ako zelimo da nacrtamo grafik neke funkcije sekvenca naredbi bi mogla da izgleda:

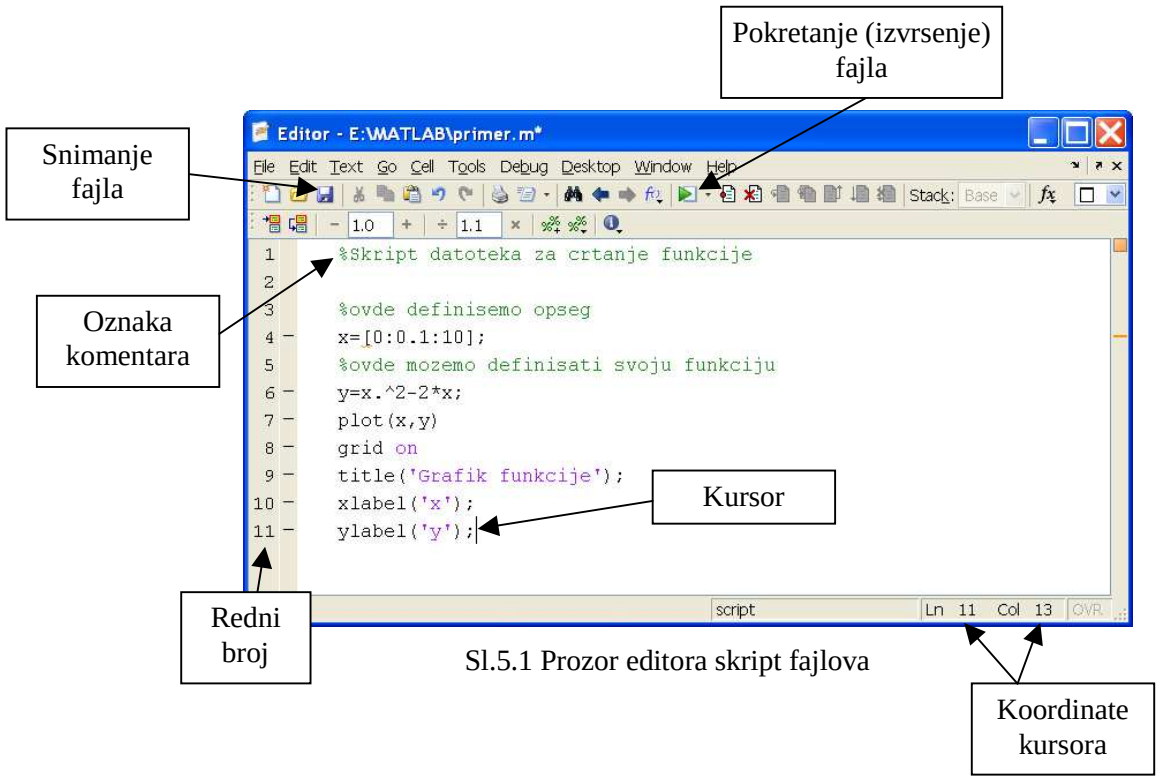
```
>> x=[0:0.1:10];
>> y=x.^2-2*x;
>> plot(x,y)
>> grid on
>> title('Grafik funkcije');
>> xlabel('x');
>> ylabel('y');
```

Ako sada zelimo da promenimo opseg vrednosti za nezavisnu promenljivu `x` morali bi smo da prvu naredbu otkucamo ponovo. Medjutim, to povlaci i ponovno izvrsavanje svih naredbi nakon nje. Promena skupa vrednosti za vektor `x` ne znaci i automatsku promenu vrednosti vektora `y`. To moramo sami da uradimo. Iako je posao je donekle olaksan koriscenjem editora prethodno unetih komandi, ipak celu sekvencu naredbi moramo pojedinacno da izvrsimo. Ako sve to moramo vise puta da ponovimo u toku testiranja nekog skupa podataka rad moze biti zamoran i podlozan greskama. Jedno resenje ovog problema je da skup naredbi koje cine jedan povezan blok snimimo u datoteku i onda po potrebu pokrecemo (izvrsavamo) datoteku. Kada pokrenemo datoteku sve komande koje su u njoj bice izvrsene redosledom kojim su unete u datoteku. Ako je potrebno mozemo praviti izmene u datoteci, snimiti te promene i ponovo pokrenutu datoteku. Time se oslobadjamo potrebe da svaku naredbu pojedinacno pokrecemo. Ovakve datoteke zovemo skript datoteke (script files).

5.1 Kreiranje skript datoteka

Skript datoteke se cesto zovu i M datoteke jer datoteke ovog tipa imaju ekstenziju *m*. Nova skript datoteka se kreira tako sto se u meniju *File* Matlab-ovog radnog okruzenja izabere *New* a onda *Blank M-File* (Precica sa tastature je *Ctrl+N*). Nakon toga otvara se prozor (Sl.5.1) koji sluzi za uredjivanje M fajlova. Rad u editoru je slican radu u komandnom prozoru. Naredbe se kucaju u redu oznacenom kursorom (|), jedna za drugom, redosledom kojim zelimo da se izvrsavaju. Posle svakog unetog reda pritiska se taster *Enter*. Razlika u odnosu na komandni prozor je sto posle *Enter*-a ne sledi izvrsavanje naredbe vec prelazak u novi red. Svaki zapoceti red oznacen je rednim brojem. Polozaj kursora definise parametare *Ln* (Line) i *Col* (Column) cije se trenutne vrednosti mogu procitati na statusnoj liniji prozora za editovanje (Sl.5.1). Nakon unosenja sekvence naredbi za crtanje grafika sadrzaj skript datoteke prikazan je na slici Sl.5.1. Obicno se na pocetku skripta nalaze redovi koji pocinju znakom

procenta (%). Takvi redovi predstavljaju komentare i imaju za cilj blize objasnjenje datoteke. Komentari nisu obavezni i nemaju uticaja na izvrsenje programa ali mogu biti od pomoci pri kasnijem tumacenju fajla. Naredbe mozemo direktno kucati ili ih kopirati iz drugih fajlova. Ceo program mozemo otkucati i u nekom obicnom tekstualnom editoru. Medjutim, rad u editoru M fajlova je laksi jer pruza niz dodatnih mogucnosti.



Sl.5.1 Prozor editora skript fajlova

Tokom rada sadrzaja datoteke treba povremeno snimati na disk. Samo pri prvom snimanju program ce traziti da definisemo lokaciju i ime fajla pod kojim zelimo da sacuvamo datoteku. Pri svakom narednom snimanju dovoljno je da pritisnemo ikonicu diskete na paleti sa alatkama (Sl.5.1). Fajl se podrazumevano cuva u tekucem direktorijumu (*Current Folder*). Podrazumevani direktorijum mozemo promeniti pomocu prozora tekuceg direktorijuma ili pomocu prozora za snimanje fajlova. Kod imenovanja datoteka vaze ista pravila kao i kod promenljivih. Naziv fajla ne treba da sadrzi ekstenziju (.m) jer je Matlab sam dodaje.

Da bi smo pokrenuli otvorenu datoteku dovoljno je da na paleti sa alatima pritisnemo dugme *Run*, pokrenemo istoimenu naredbu iz menija *Debug* ili jednostavno pritisnemo funkcijski taster *F5*. Ukoliko je skript datoteka zatvorena nije potrebno da je prethodno otvaramo da bi smo je pokrenuli. Dovoljno je da u komandnom prozoru otkucamo ime fajla da bi smo je pokrenuli (pod uslovom da se fajl nalazi u tekucem direktorijumu ili na putanji za pretrazivanje). Zato treba voditi racuna da ime M datoteke ne preklopi neko vec postojece ime u Matlab-u.

5.2 Ulazni podaci skript datoteka

Skript datoteka predstavlja samo jedan oblik organizovanja i izvrsavanja naredbi ali ne uvodi poseban memorijski prostor. Komandni prozor i skript datoteke dele isti radni prostor. To znaci da promenljive

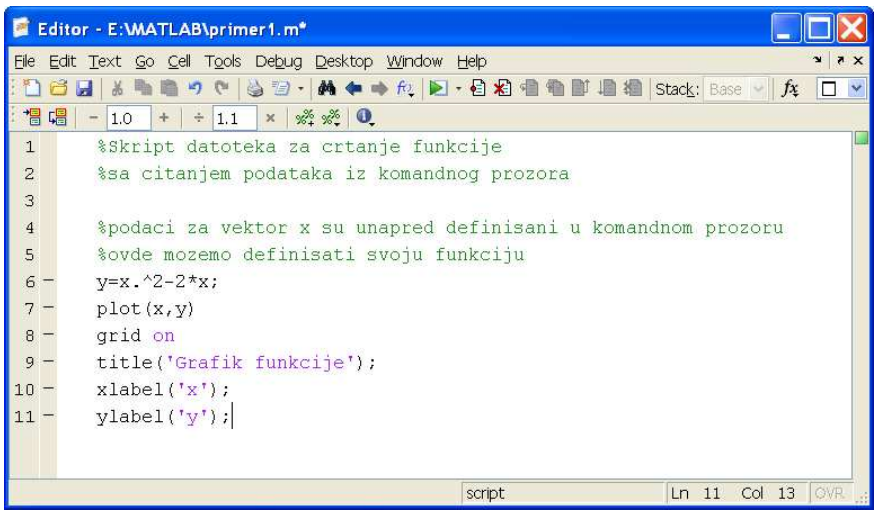
koje su vidljive u komandnom prozoru istovremeno su vidljive i u skript datoteci i obratno. U vezi sa tim postoje tri nacina za definisanje promenljivih koje se koriste u skript datotekama.

Prvi nacin je da se u samom M fajlu definisu promenljive kao sto je to uradjeno u primeru sa slike Sl.5.1. Pri definisanju promenljivih vaze ista pravila kao i kod definisanja promenljivih u komandnom prozoru.

Drugi nacin bi bio da se prvo u komandnom prozoru definise promenljiva a da se zatim pozove skript datoteka koja ce obaviti preostali deo posla. Na primer, u komandnom prozoru bi smo otkucali:

```
>> x=[0:0.1:10];
```

a onda bi smo pozvali M fajl sledeceg sadrzaja:



Sl.5.2 Skript datoteka sa citanjem podataka iz komandnog prozora

Treca mogucnost je da se podaci definisu u skript datoteci ali da se konkretne vrednosti unose nakon pokretanja skript datoteke. Ova mogucnost se zasniva na koriscenju funkcije `input` cija sintaksa ima oblik:

```
ime_promenljive=input('tekst poruke')
```

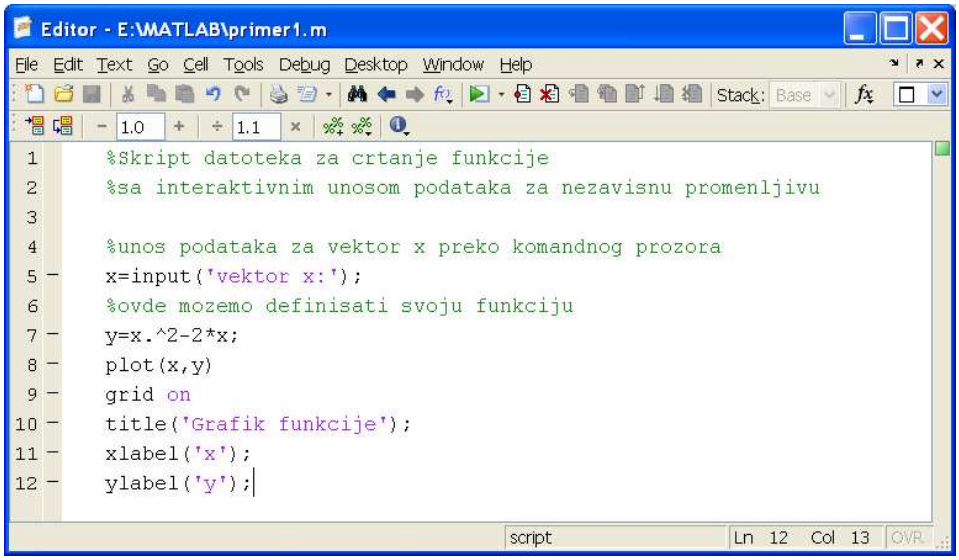
Kada se skript datoteka pokrene i izvrsavanje stigne do funkcije `input` u komandnom prozoru se ispisuje `tekst poruke` koja korisniku blize objasnjava sta treba da otkuca u odzivu na kursor. Ono sto korisnik otkuca, nakon pritiska na taster `Enter` biva dodeljeno promenljivoj `ime_promenljive`. Nakon toga skript datoteka nastavlja sa svojim izvrsavanjem. Prethodna skript datoteka preuredjena za interaktivan unos podataka bi izgledala kao na slici Sl.5.3. Kada se pozove funkcija `input` u komandnom prozoru se prikazuje poruka

```
vektor x:
```

u cijem nastavku treba otkucati:

```
[0:0.1;10]
```

i pritisnuti `Enter`. Nakon toga pojavice (azurirace) se grafik sa iscrtanom funkcijom.



Sl.5.3 Skript datoteka sa interaktivnim unosom podataka

Izlazni podaci skript datoteke se mogu generisati na razlicite nacine.

5.3 Izlazni podaci skript datoteka

Skript datoteka predstavlja organizovani nacin cuvanja i izvrsavanja skupa naredbi. U svemu ostalom rad odgovara radu u komandnom prozoru. Ukoliko se posle nekog izraza u skript datoteci izostavi znak ";" vrednost ce biti ispisana neposredno nakon njegovog izvrsenja. Za prikazivanje prethodno definisane promenljive dovoljno je da u komandnoj datoteci otkucamo njeno ime. Sem toga, za prikazivanje izlaznih podataka mozemo da koristimo i funkcije `disp` i `fprintf`. Obe ove komande su nam na raspolaganju u komandnom prozoru, skript ili funkcijskoj datoteci. Sintaksa funkcije `disp` ima oblik:

`disp(ImePromenljive)` ili `disp('Znakovni Niz')`

Na primer, ako smo unapred definisali promenljive

```
>> t=0:0.1:0.5;
>> sila=10*exp(-t).*sin(2*t);
```

njihove vrednosti mozemo prikazati naredbom

```
>> disp(t)
         0    0.1000    0.2000    0.3000    0.4000    0.5000
```

i

```
>> disp(sila)
         0    1.7976    3.1883    4.1830    4.8086    5.1038
```

Funkcija `disp` dozvoljava samo jedan argument u pozivu. Na primer nije dozvoljeno

```
>> disp(t,sila)
```

ali je moguce prethodno formirati niz pa onda prikazati njegovu vrednost

```
>> disp([t;sila])
```

sto kao rezultat daje

```
0     0.1000     0.2000     0.3000     0.4000     0.5000
0     1.7976     3.1883     4.1830     4.8086     5.1038
```

Ukoliko se u pozivu funkcije koristi argument sa jednostrukim navodnicima prikazuje se tekst uokviren tim navodnicima. Naredba

```
>> disp('sila')
```

u komandnom prozoru prikazuje tekst

```
sila
```

Treba primetiti da se podaci mogu prikazati i direktnim kucanjem imena promenljive ili stringa a bez pozivanja funkcije `disp`. Na primer, da bi smo prikazali vrednost promenljive `sila` dovoljno je otkucati samo `sila`. Slicno vazi i za string.

5.3.1 Primena naredbe `fprintf`

Naredba `fprintf` pruza vise mogucnosti oko prikazivanja podataka. Izlazne podatke je moguce formatirati i prikazati na ekranu ili ih sacuvati u fajlu. Sintaksa funkcije `fprintf` u opstem slucaju ima oblik:

```
broj=fprintf(fid,format,x,y,z,...)
```

`fid` je identifikator fajla u koji se vrsi upis podataka. Ukoliko se ovaj identifikator izostavi ili ima vrednost 1 rezultati se prikazuju na ekranu. `format` je string kojim se definise tekst i nacin prikazivanja podataka. `x,y,z` su numericki podaci cije vrednosti zelimo da prikazemo. Promenljiva `broj` sadrzi broj uspesno upisanih bajtova.

Na primer, da bi smo na ekranu prikazali samo tekstualnu poruku koristimo naredbu oblika

```
>> fprintf('Nivo tecnosti u rezervoaru Rezervoar I Rezervoar II')
```

Rezultat ove naredbe bice tekstalna poruka ispisana u jednom redu:

```
Nivo tecnosti u rezervoaru Rezervoar I Rezervoar II>>
```

Ukoliko zelimo da deo teksta bude ispisan u novom redu mozemo u tekstu da koristimo oznaku (izlaznu sekvencu) `\n`. Ova oznaka nece biti stampana vec ce samo ukazati Matlab-u da treba preci u novi red. Na primer naredba:

```
>> fprintf('Nivo tecnosti u rezervoaru \nRezervoar I Rezervoar II\n')
```

formira prikaz:

```
Nivo tecnosti u rezervoaru
Rezervoar I Rezervoar II
```

Da bi prikaz naredbe `fprintf` poceo u novom redu tekst treba zapoceti sa `\n`.

Postoje i druge izlazne sekvence. Na primer `\t` ima isti efekat kao da smo pritisnuli taster `Tab` na tastaturi. Naredba

```
>> fprintf('\t\tNivo tecnosti u rezervoaru \nRezervoar I
```

```
\t\t\tRezervoar II\n')
```

formira sledeci ispis:

```
                Nivo tecnosti u rezervoaru
     Rezervoar I                        Rezervoar II
```

Za dodatno pomeranje teksta mozemo da koristimo blanko znake u samom tekstu.

Ukoliko zelimo da prikazemo numericki podatak funkcija `fprintf` mora da ima makar dva argumenta. Prvim argumentom, tekstualnog tipa, definisemo format prikaza broja a drugim sam taj broj. Format prikaza u opstem slucaju ima oblik:

```
% i m.n p
```

Znak "%" je obavezan i ukazuje da zelimo da prikazemo numericki podatak. Indikator `i` je opcioni. Njime se definisu vodeci znaci u prikazu broja. Parametar `m` definise ukupan broj polja rezervisanih za prikaz broja a `n` broj polja za decimalne cifre. Parametar `p` je obavezan i definise nacin prikaza broja. Vrednosti i znacenje ovog parametra dati su u tabeli T.5.1.

Tabela T5.1 - Parametar `p` u formatu prikaza numerickog podatka naredbe `fprintf`

Vrednost	Znacenje
`e` ili `E`	Prikaz broja u eksponencijalnoj notaciji
`f`	Prikaz broja u decimalnoj notaciji
`d` ili `i`	Prikaz celih oznacenih brojeva
`u, o, x`	Prikaz celih neoznacenih brojeva u sistemu sa osnovom 10, 8 i 16.

Na primer, neka je:

```
>> a=215.457;
```

u nastavku je dato nekoliko primera naredbe `fprintf` i njenih rezultata.

```
>> fprintf('%f\n',a)
215.457000
>> fprintf('%7.3f\n',a)
215.457
>> fprintf('%7.2f\n',a)
 215.46
```

Ukoliko je u pitanju ceo neoznacen broj:

```
>> b=25;
```

mozemo ga prikazati u sistemu sa osnovom 10:

```
>> fprintf('%u\n',b)
25
```

sa osnovom 8:

```
>> fprintf('%o\n',b)
31
```

ili osnovom 16:

```
>> fprintf('%x\n',b)
19
```

Pomocu naredbe `fprintf` mozemo da dajemo kombinovane prikaze teksta i brojeva. Na primer:

```
fprintf('Vrednost broja pi je:%5.3f (prikaz sa tri decimale)\n',pi)
```

kao rezultat daje:

```
Vrednost broja pi je:3.142 (prikaz sa tri decimale)
```

Pri prikazivanju elemenata niza naredba `fprintf` se ponavlja sve dok se ne prikazu svi elementi niza. Na primer, neka su u tabeli T5.2 dati rezultati merenja nivoa tecnosti u dva rezervoara.

T5.2 Rezultati merenja nivoa tecnosti u dva rezervoara.

R.b. merenja	1	2	3	4	5
Rezervoar I: Visina [mm]	270	280	290	260	240
Rezervoar II:Visina [mm]	215	275	230	245	250

Pomocu skript datoteke:

```
fprintf('\nNivo tecnosti u rezervoaru')
fprintf('\nRezervoar I\n')
fprintf('%i   ',h1)
fprintf('\nRezervoar II\n')
fprintf('%i   ',h2)
fprintf('\n')
```

Formiramo prikaz oblika:

```
Nivo tecnosti u rezervoaru
Rezervoar I
270   280   290   260   240
Rezervoar II
215   275   230   245   250
```

Pri prikazivanju elemenata matrice elementi se prikazuju kolonu po kolonu. Na primer, skript datoteka:

```
fprintf('\tNivo tecnosti u rezervoaru')
fprintf('\nRezervoar I \t\tRezervoarII\n')
fprintf('\t%i \t\t\t\t %i\n',[h1;h2])
```

formira prikaz:

```
      Nivo tecnosti u rezervoaru
Rezervoar I           RezervoarII
      270                   215
```

```
      280                   275
      290                   230
      260                   245
      240                   250
>>
```

Dodatne informacije o funkciji `fprintf` mogu se dobiti od sistema za pomoc.

5.3.2 Upis podataka u datoteku

Rezultate razlicitih izracunavanja, osim prikazivanja u komandnom prozoru, mozemo upisivati u datoteke. U tu svrhu mozemo da koristimo naredbu `fprintf`. Upis podataka u datoteku se sastoji od tri koraka:

1. Otvaranje datoteke za upis.
2. Upis podataka u otvorenu datoteku.
3. Zatvaranje otvorene datoteke.

Pre upisa podataka datoteku moramo otvoriti. Za otvaranje se koristi naredba `fopen`. Sintaksa ove naredbe ima oblik:

```
fid=fopen('ime_datoteke', 'dozvola')
```

6. Upis i čitanje podataka iz datoteka

Povremeno je, tokom rada, potrebno trajno sacuvati podatke na cvrsti disk odnosno ucitati ranije sacuvane podatke sa cvrstog sika. Kao i za vecinu drugih aktivnosti u Matlab-u postoji vise nacina da se to uradi. U ovom poglavlju cemo nesto vise reci o naredbama `save` i `load` kojima se, iz komandnog prozora, ove aktivnosti realizuju na najjednostavniji nacin. Kao sto smo ranije vec pomenuli ove naredbe se najcesce koriste za cuvanje (ucitavanje) podataka iz radnog prostora Matlab-a. Kada izadjemo iz programa sve sto je bilo u radnom prostoru biva nepovratno izgubljeno. Svaka nova sesija u Matlab-u pocinje sa praznim radnim prostorom.

6.1 Naredba `save`

Komanda `save` se koristi za upisivanje podataka u datoteku. U najednostavnijem slucaju sintaksa ove naredbe ima oblik:

```
save ime_fajla
```

Argument `ime_fajla` je naziv datoteke u kojoj zelimo da sacuvamo podatke. Naziv datoteke ne treba da sadrzi ekstenziju, Matlab podrazumevano pridruzuje ekstenziju *mat*. Rezultat naredbe je datoteka punog naziva `ime_fajla.mat` u kojoj su sacuvane sve promenljive iz radnog prostora. Za svaku promenljivu se cuva ime, tip, velicinu i vrednost. Format ove datoteke je specifican za Matlab pa se ona ne moze se otvarati iz drugih programa.

Pri koriscenju naredbe `save` treba biti pazljiv. Ako u tekucem direktorijumu vec postoji datoteka istog imena ona ce biti izbrisana a umesto nje formirace se nova istog imena sa novim sadrzajem. Razlog za obazrivost je tim veca sto nas Matlab, pri upotrebi naredbe `save`, nece upozoriti da datoteka istog imena vec postoji. Ukoliko datoteku zelimo da sacuvamo u drugom direktorijumu, pre upotreba naredbe `save` treba da pomocu prozora tekuceg direktorijuma (*Current Folder*) predjemo u zeljeni direktorijum. Druga mogucnost je da u argumentu ove naredbe pored imena fajla navedemo i putanju do tog fajla. Na primer, naredba

```
>> save d:\zadaci\podaci
```

upisuje podatke radnog prostora u datoteku `podaci.mat` koja se nalazi na disku `d:` u direktorijumu `zadaci` bez obzira na tekuci direktorijum. Naravno, uslov je da direktorijum `zadaci` postoji na navedenoj putanji.

Pomocu naredbe `save` mozemo da snimimo samo neke (ne sve) promenljive iz radnog prostora. Dovoljno je da posle imena fajla navedemo imena promenljivih (bez razdvajanja zarezom) koje zelimo da snimimo. Na primer, neka su u radnom prostoru definisane dve promenljive:

```
>> t=0:0.1:10;
>> y=sin(2*t+0.5);
```

Tada se naredbom

```
>> save podaci1 t y
```

u tekuci direktorijum, u datoteku `podaci1.mat` upisuju samo promenljive `t` i `y`.

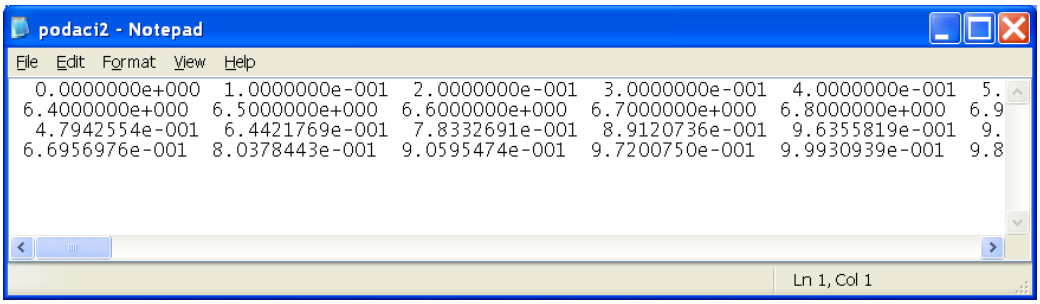
Ukoliko zelimo da se podaci u fajl snime u ASCII formatu takodje mozemo koristiti naredbu `save` sa dodatnim parametrom `-ascii`:

```
save -ascii ime_fajla
```

U ovom slucaju naziv fajla treba da sadrzi i ekstenziju. Na primer, da bi smo u tekuci direktorijum snimili fajl pod imenom `podaci2.dat` sa vrednostima promenljivih `t` i `y` yreba otkucati naredbu:

```
>> save -ascii podaci2.dat t y
```

Rezultat ce biti tekstualna datoteka u koju su prvo upisane vrednosti promenljive `t` a onda i vrednosti promenljive `y`. Pri snimanju promenljivih u ASCII formatu snimaju se samo njihove vrednosti kao stringova bez ocuvanja imena, tipova i velicina tih promenljivih. Medjutim, prednost je sto se datoteke u ASCII formatu mogu otvarati i pomocu vecine drugih programa. Drugim recima, ovaj tip datoteke mozemo da koristimo za razmenu podataka sa drugim programima. Na primer, sadrzaj datoteke `podaci2.dat` otvorene u Notepad-u izgleda kao na slici Sl.6.1.



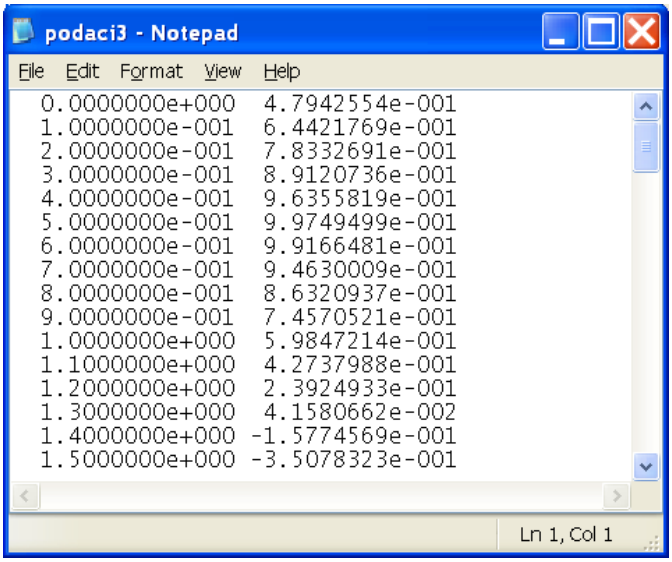
Sl.6.1 Prikaz sadrzaja tekstualne datoteke formirane u Matlab-u

Kao sto se vidi, u datoteci se nalaze samo vrednosti promenljivih ispisane kao tekst. Ove datoteke mozemo dodatno da obradjujemo u ma kojem editoru teksta.

Pre upisa podataka u datoteku mozemo im prilagoditi formu po zelji. Na primer, ukoliko vrednosti promenljivih `t` i `y` zelimo da snimimo kao dve kolone podataka mozemo da koristimo sledece naredbe:

```
>> z=[t;y]';
>> save -ascii podaci3.dat z
```

Sadrzaj datoteke `podaci3.dat` prikazan je na slici Sl.6.2



Sl.6.2 Delimican prikaz datoteke sa dve kolone podataka

Ukoliko zelimo da se pri snimanju podataka stari podaci ne brisu, vec da se novi samo pridodaju na postojece u naredbi `save` treba koristiti opcioni parametar `-append`. Na primer, naredba

```
>> save -append -ascii podaci4.dat z
```

otvara postojecu datoteku `podaci4.dat` i nju dopunjuje novim vrednostima za `z`.

6.2 Naredba `load`

Naredba `load` radi suprotno od naredbe `save`, podatke iz datoteke prepisuje u radni prostor. U najednostavnijem slucaju sintaksa ove naredbe ima oblik:

```
load ime_fajla
```

Argument `ime_fajla` predstavlja ime fajla koji zelimo da otvorimo. Ako se ne navede ekstenzija podrazumevano se otvara fajl punog naziva `ime_fajla.mat` iz tekuceg direktorijuma. Ako se zeljeni fajl ne nalazi u tekucem direktorijumu prvo moramo da promenimo tekuci direktorijum ili da ispred imena fajla otkucamo putanju do tog fajla. Na primer, naredba

```
>> load d:\zadaci\podaci
```

cita podtke iz datoteke `podaci.mat` koja se nalazi na disku `d:` u direktorijumu `zadaci` bez obzira na tekuci direktorijum.

Pri citanju podataka iz fajlova sa nastavkom *mat* ucitavaju se ne samo njihove vrednosti vec i imena, tipovi i velicine promenljivih. Na primer, posle naredbe

```
>> load podaci1
```

u radni prostor se ucitavaju podaci o prethodno upisanim promenljivama `t` i `y`. Naredba `whos` daje sledeci rezultat:

```
>> whos
  Name      Size              Bytes  Class     Attributes

  t         1x101               808  double
  y         1x101               808  double
```

Drugim recima, vrsi se potpuna rekonstrukcija ranije snimljenog radnog prostora.

Ukoliko zelimo da procitamo samo odredjene promenljive potrebno je da u nastavku naziva fajla otkucamo imena tih promenljivih. Na prime, naredba

```
>> load podaci1 t
```

u radni prostor iz datoteke `podaci1.mat` ucitava samo promenljivu naziva `t`. Ova sintaksa nam ukazuje na dve stvari. Prvo, moramo znati tacan naziv promenljive koju zelimo da uvezemo i drugo, promenljive se po imenu mogu ucitavati samo iz `mat` datoteka jer samo datoteke ovog tipa cuvaju imena promenljivih.

Za citanje ASCII datoteka koristi se isti oblik naredbe load kao i kod `mat` datoteka. Matlab ce sam prepoznati koji tip datoteke je u pitanju. Pri navodjenju imena datoteke moramo otkucati i ekstenziju fajla kojeg zelimo da ucitamo. Na primer, za prethodno snimljenu datoteku `podaci2.dat` naredba

```
>> load podaci2
```

vraca gresku jer Matlab podrazumevano trazi fajl `proba2.mat`. Zato moramo otkucati:

```
>> load podaci2.dat
```

Svakoj promenljivoj koja se unosi u radni prostor mora biti pridruzeno ime jer u suprotnom ne bi mogli da joj pristupimo. Kako se kod ASCII datoteka ne cuva podatak o imenu promenljive, postavlja se pitanje kojoj se promenljivi dodeljuju podaci pri njihovom ucitavanju? Za razliku od komandnog prozora gde se neimenovani podaci pridruzuju promenljivoj `ans` u ovom slucaju kreira se nova promenljiva istog imena kao i ime fajla. Dakle, u prethodnom primeru formirala bi se promenljiva pod imenom `podaci2` sa kojom mozemo raditi kao i sa ostalim promenljivama radnog prostora. Zato, ASCII datoteka koju zelimo da otvorimo u Matlab-u mora da ima jedinstvenu strukturu podataka jer se svi podaci pridruzuju samo jednoj promenljivoj. Ne mozemo citati datoteku u kojoj su kombinovano zapisani skalari, vektori ili matrice. Svi podaci moraju biti snimljeni u obliku jedinstvene Matlab promenljive.

Ukoliko zelimo da se pri citanju ASCII datoteke podaci pridruze promenljivoj proizvoljnog imena mozemo da koristimo naredbu `load` sledeceg oblika:

```
>> ime_promenljive=load('ime_fajla');
```

Naredba u ovom slucaju ima oblik funkcije ciji je argument ime fajla kojeg otvaramo. Podaci se pridruzuju promenljivoj sa imenom `ime_promenljive`. Na primer, sledeca skript datoteka otvara datoteku imena `podaci3.dat` i na osnovu podataka koji imaju strukturu matrice sa dve kolone (Sl.6.2) crta grafik.

```
>> z=load('podaci3.dat');
>> t=z(:,1);
>> y=z(:,2);
>> plot(t,y);
```

U narednom primeru skript datoteka ucitava i obradjuje podatke iz fajla `podaci2.dat`. Ucitani podaci imaju strukturu matrice sa dve vrste. Rezultati obrade se upisuju u novu datoteku pod imenom `podaci4.dat`. Snimljeni podaci imaju strukturu matrice sa dve kolone.

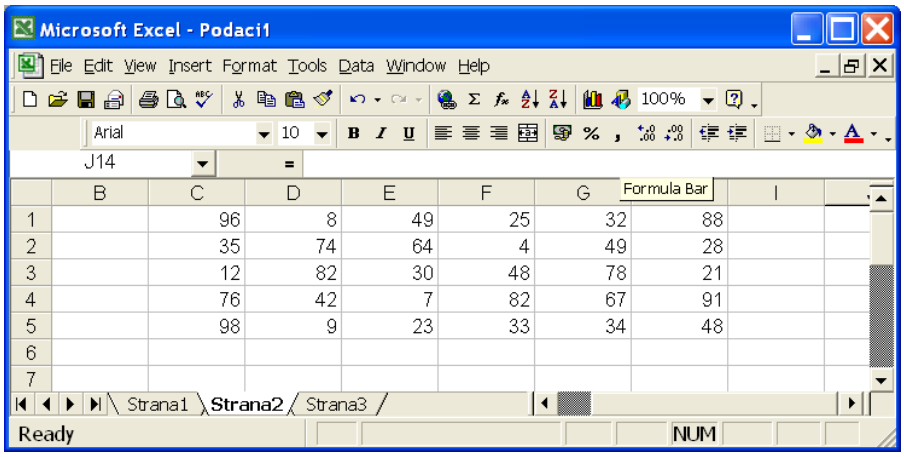
```
>> load podaci2.dat;
>> t=podaci2(1,:);
>> y=podaci2(2,:);
>> y=exp(-0.8*t).*y;
>> z=[t;y]';
>> save -ascii podaci4.dat z
```

6.3 Razmena podataka sa MS Excel-om

Datoteke snimljene u ASCII formatu pruzaju mogucnost indirektne razmene podataka izmedju Matlab-a i drugih programa. Podaci ne moraju biti samo numericki. Mogu se razmenjivati i tekstualni, zvucni, graficki ili drugi podaci. Za neke programe koji imaju siroku primenu u obradi podataka postoji mogucnost i direktne razmene podataka. Jedan od takvih programa je i MS Excel koji je postao standard u obradi tabelarnih (matricnih) podataka. U Matlab-u postoje ugradjene funkcije za direktno citanje i upisivanje u Excel-ove datoteke. Za citanje podataka koristi se funkcija `xlsread` cija sintaksa ima oblik:

```
ime_promenljive=xlsread('ime_datoteke','ime_lista','opseg')
```

Funkcija ima tri argumenta tekstualnog tipa. Prvi argument `'ime_datoteke'` definise datoteku iz koje se citaju podaci. Pri navodjenju imena ne treba kucati ekstenziju datoteke. Fajl se mora nalaziti ili u tekucem direktorijumu ili naveden u putanji za pretrazivanje. Opciono, prvi argument pored imena moze sadrzati i putanju do datoteke. Drugi argument definise ime radnog lista u Excel-ovoj radnoj svesci sa kojeg se citaju podaci. Ovaj argument je opcioni i ako se ne navede podaci se citaju sa prvog lista. Treci argument je opcioni i definise opseg radnog lista iz kojeg se citaju podaci. Opseg je pravougaona oblast koja se definise krajnjom gornjom levom i krajnjom donjom desnom celijom. Ukoliko se ne navede citaju se svi podaci sa radnog lista.
Na primer neka je data Excel-ova radna sveska prikazana na slici Sl.6.3.



Sl.6.3 Excel-ova radna sveska sa podacima

Da bi smo Matlab-ovoj promenljivoj A dodelili sve podatke iz kolona D i E radnog lista pod imenom "Strana2" koristimo naredbu:

```
>> A=xlsread('podaci1','Strana2','D1:E5')
A =
     8    49
    74    64
    82    30
    42     7
     9    23
```

Naravno, uvezene podatke mozemo da obradjujemo po zelji. Na primer, matrici A dodacemo trecu kolonu:

```
>> A(:,3)=linspace(7,15,5)
A =
     8    49     7
    74    64     9
    82    30    11
    42     7    13
     9    23    15
```

Ukoliko zelimo da podatke iz radnog prostora Matlab-a prebacimo u Excel-ov fajl koristimo naredbu `xlswrite` cija sintaksa ima oblik:

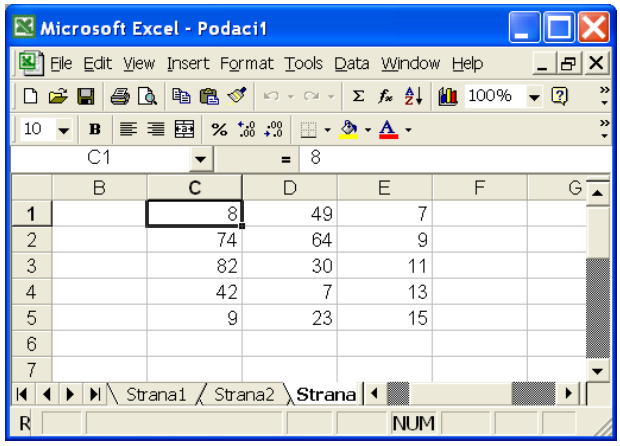
```
xlswrite('ime_datoteke',ime_promenljive,'ime_lista','opseg')
```

Tekstualni argumenti `'ime_datoteke','ime_lista'` i `'opseg'`imaju isto znacenje kao i kod naredbe `xlsread`. Argument `ime_promenljive` (navodi se bez apostrofa) oznacava promenljivu cije vrednosti zelimo da upisemo u fajl.

Na primer naredba

```
>> xlswrite('podaci1',A,'Strana3','c1:e5')
```

prethodno formiranu matricu `A` upisuje u fajl `podaci1.xls` tekuceg direktorijuma, na stranu `'Strana3'` u pravougaonu oblast izmedju celija `c1` i `e5`. Rezultat naredbe prikazan je na slici Sl.6.4.



Sl.6.4 Rezultat upisa podataka u Excel-ovu datoteku