Programiranje u Octaveu

Domović, Daniel; Rolich, Tomislav

Sveučilište u Zagrebu

Tekstilno-tehnološki fakultet

Daniel Domović

Tomislav Rolich

Programiranje u Octaveu

Zagreb, 2022.

MANUALIA UNIVERSITATIS STUDIORUM ZAGREBIENSIS
UDŽBENICI SVEUČILIŠTA U ZAGREBU
TEKSTILNO-TEHNOLOŠKI FAKULTET

Nakladnik:
Sveučilište u Zagrebu Tekstilno-tehnološki fakultet

Urednik:
Prof. dr. sc. Tanja Pušić
Sveučilište u Zagrebu Tekstilno-tehnološki fakultet

Recenzenti:
Prof. dr. sc. Nenad Bolf
Sveučilište u Zagrebu Fakultet kemijskog inženjerstva i tehnologije
Prof. dr. sc. Darko Grundler (u mirovini)
Sveučilište u Zagrebu Tekstilno-tehnološki fakultet
Prof. dr. sc. Željko Penava
Sveučilište u Zagrebu Tekstilno-tehnološki fakultet

Lektor:
Ružica Barać, prof.

Odlukom Senata Sveučilišta u Zagrebu klasa: 032-01/21-02/14 urbroj: 380-061/36-22-5 od 24. svibnja 2022. odobrava se rukopis autora dr. sc. Daniela Domovića i prof. dr. sc. Tomislava Rolicha **Programiranje u Octaveu** korištenje naziva sveučilišni udžbenik (Manualia Universitatis Studiorum Zagrebiensis).

ISBN 978-953-7105-83-9

SADRŽAJ

1. UVOD

Matematičke je probleme moguće rješavati analitičkim ili numeričkim pristupom. Analitičkim se pristupom primjenjuje skup logičkih koraka za rješavanje problema kojima se dobiva egzaktno rješenje problema. Analitičkim se postupcima ponekad ne može naći rješenje ili njegovo pronalaženje predugo traje.

Postoje zadaci iz stvarnog svijeta za koje je približni numerički rezultat dovoljno dobar. Inženjeri općenito ne trebaju savršeno točan odgovor, već samo rješenje koje je dovoljno dobro kako bi bilo iskoristivo.

Numeričko računanje je matematička disciplina čija je svrha naći približno rješenje. Numeričkim se računanjem rješavaju problemi koje nije moguće riješiti analitičkim postupcima. Ciljevi numeričkog računanja su dobiti što točnije rješenje, odnosno rješenje koje ima što manju pogrešku i doći do njega u što manjem broju operacija.

Numeričko računanje je po prirodi iterativan postupak. Složen analitički pristup pronalaska rješenja može se zamijeniti s više jednostavnijih izračuna. Pristupi programiranju temeljeni na numeričkom računanju zahtijevaju izvršavanje jednostavnih operacija koje se ponavljaju. Takvi se pristupi najčešće temelje na praćenju kvalitete rješenja dobivenih zadanim ulaznim parametrima, prilagođavanjem ulaznih parametara na osnovu dobivenih podataka o kvaliteti rješavanja i ponavljanjem navedenog postupka do zadovoljavajuće razine pogreške.

Kada se program pokrene ti se koraci izvršavaju sekvencijalno, odnosno jedan za drugim bez dodatne interakcije s korisnikom. To znači da se program mora napraviti tako da se operacije izvršavaju uzastopno i da program sam donosi odluke u ovisnosti o dobivenim rješenjima. Programi zasnovani na numeričkoj analizi najčešće se sastoje od: metoda za inicijalizaciju varijabli i unos ulaznih podataka, petlji kojima se ponavljaju operacije, struktura za donošenje odluka i metoda za prikaz rješenja.

Inženjerske i znanstvene zadatke često je lakše rješavati ako se podaci predoče vektorima i matricama, a rezultate je često potrebno prikazati grafički. Programski jezici opće namjene (npr. Fortran, Cobol, C i dr.) nemaju ugrađene naredbe i funkcije za rad s matricama i grafičke prikaze.

U inženjerskoj praksi postojala je potreba za što jednostavnijim načinom računanja s vektorima i matricama te grafičkim prikazom rješenja kako bi se lakše usredotočili na problem koji se rješava, a ne na programsko rješavanje operacija s matricama i grafičkim prikazima. Programski jezik koji bi zadovoljio ove uvjete morao je biti što jednostavniji s gledišta učenja i primjene. Primjerice, poželjno je podatke grafički prikazati jednom naredbom.

S tim težnjama razvijen je Octave. Octave je interaktivni sustav otvorenog koda za numeričko računanje i grafiku. Octave je dizajniran za izračune temeljene na matricama (npr. rješavanje sustava jednadžbi). Podaci u mnogim inženjerskim problemima iz stvarnog svijeta mogu se svesti na ovakav prikaz podataka. Octaveom je moguće prikazati podatke na više načina, a dolazi i s vlastitim programskim jezikom. Može ga se, primjerice, smatrati snažnim programibilnim, grafičkim kalkulatorom. Upravo su to i prednosti korištenja Octavea..

Knjiga se sastoji od četrnaest poglavlja. Prvo je poglavlje uvod (ovo poglavlje). U drugom je poglavlju prikazan povijesni razvoj Octave programa. U trećem su poglavlju navedeni dijelovi Octave programa, objašnjeno je grafičko sučelje te glavni prozori Octave programa. Četvrto poglavlje opisuje vrste varijabli u Octave programu, te način njihova definiranja, spremanja i dobave. U petom je poglavlju opisano stvaranje matrica i vektora. U šestom su poglavlju opisani aritmetički, relacijski i logički operatori. Naredbe odluke i ponavljanja opisane su u sedmom poglavlju. U osmom poglavlju opisane su vrste funkcija, njihovo definiranje i uporaba. U devetom poglavlju opisane su transcedentne funkcije. U desetom poglavlju opisane su često rabljene statističke funkcije, u jedanaestom funkcije za rad s polinomima, a u dvanaestom neke od ostalih funkcija. U trinaestom poglavlju opisan je 2D prikaz

podataka, vrste 2D grafova i pripadajuće naredbe, a u četrnaestom 3D prikaz podataka, vrste 3D grafova i pripadajuće naredbe.

2. OCTAVE

Octave se pojednostavljeno može opisati kao program namijenjen rješavanju različitih, ponajprije inženjerskih zadataka, koji zahtijevaju numeričko računanje. Za razliku od drugih računalnih programa, Octave omogućuje rješavanje takvih zadataka uporabom ugrađenih funkcija za širok raspon zadataka. Tako je, npr. sustav dviju jednadžbi s dvije nepoznanice u Octave programu moguće riješiti tako da se napišu te dvije jednadžbe (na prikladan način) i pokrene program koji će ispisati rješenja.

Octave je računalni programski jezik visoke razine namijenjen razvoju algoritama, vizualizaciji podataka, analizi podataka i numeričkom računanju. Uporabom Octave programa različiti zadatci inženjerskog računanja mogu se riješiti brže nego uporabom ostalih računalnih programskih jezika, primjerice C-a, C++-a ili FORTRAN-a.

Octave ima interaktivno sučelje koje dodatno olakšava oblikovanje i rješavanje zadataka. Program je primjenjiv u brojnim područjima uključujući, npr. obradu slike, komunikacije, upravljanje, mjerenja, financijska modeliranja i analize. Različiti dodatni programski alati namijenjeni su zadatcima određenog užeg područja i dodatno olakšavaju rješavanje zadataka tog područja.

Značajke Octave programa su:

- Riječ je o programskom jeziku visoke razine za numeričko računanje.
- Ima grafički razvojni okoliš za rukovanje programima, datotekama i podatcima.
- Ima ugrađene interaktivne alate za interaktivno rješavanje zadataka.
- Razvijene su funkcije za linearnu algebru, statistiku, Fourierovu analizu, filtre, optimiranje i numeričku integraciju.
- Postoje 2D i 3D funkcije za vizualni prikaz podataka.
- Postoje ugrađeni alati za izradu grafičkog korisničkog sučelja.

Octave se temelji na programskom jeziku koji se naziva engl. *M-code* ili skraćeno M. Najjednostavniji je način izvršenja M programa upisati ga u prozor Octave programa i pokrenuti izvršenje. Program se može pohraniti u tekstualnu datoteku koja se naziva M-datoteka (engl. *M-file*).

Octave se sastoji od više dijelova:

- Razvojnih alata koji su povezani prikladnim grafičkim korisničkim sučeljem. U te alate ubrajaju se primjerice: naredbeni prozor (engl. *command window*), koji omogućuje pisanje i izvršavanje naredbi, skripti i funkcija, program za pisanje M-datoteka (engl. *M-file editor*), program za provjeru programa (engl. *debugger*), sustav pomoći (engl. *help*) i dr.
- Octave programskog jezika za izradu programa koji se mogu izvršavati u Octave radnom okolišu.
- Biblioteke gotovih funkcija (engl. *built-in functions*). Riječ je o mnogo provjerenih i optimiranih funkcija za različita računanja, posebice za računanja s matricama.
- Biblioteke za grafički prikaz podataka. Riječ je o dvodimenzionalnim i trodimenzionalnim prikazima koje je moguće oblikovati prema potrebama. Uz to postoje i funkcije za izradu grafičkog korisničkog sučelja.
- Biblioteke programskog sučelja (engl. *API*, *application program interface*) za razvoj C i FORTRAN programa koji se mogu povezati s Octave programima.

Jedno od važnih svojstava Octave programa mogućnost je izrade korisničkih programa koji se spremaju u M-datoteke. Uobičajeno je skup takvih korisničkih programa koji se odnose na pojedino područje objediniti i nazvati alatima (engl. *toolbox*). Postoji mnogo alata, od kojih se neki isporučuju s Octave programom dok se drugi mogu kupiti ili besplatno dobaviti od drugih dobavljača. Prednost je gotovih alata u tome što su oni redovito plod rada stručnjaka iz tog područja te što su provjereni i optimirani (glede

brzine izvršavanja u Octaveu, iskorištenja memorije i sl.). Ako ne postoje gotovi alati za područje kojim se bavi, korisnik može napisati svoje alate.

2.1 Povijesni razvoj Octave programa

U znanstvenoj se zajednici dugo osjećala potreba za specijaliziranim programom za numeričko računanje, gdje se unaprijed definirane funkcije mogu jednostavno i po potrebi iskoristiti. Iz tog razloga nastali su specijalizirani programski paketi poput MATLAB-a, Scilaba, Mathematice i Octavea.

Preteča Octavea je MATLAB kojeg je razvio Cleve Moler koji je bio profesor matematike na Sveučilištu u New Mexicu i predavao je numeričku analizu i teoriju matrica. U okviru svog doktorskog studija, napisao je kod u FORTRAN-u kako bi riješio sustave linearnih jednadžbi koje uključuju matričnu algebru. Tom programu dao je ime MATrixLABoratory (MATLAB). Kao profesor želio je da njegovi studenti mogu koristiti napisane programske pakete bez pisanja programa u FORTRAN-u. Stoga je krajem 70-ih g. 20. st. izašla prva verzija MATLAB-a (napisana u FORTRAN-u).

U okviru prve verzije bilo je dostupno 80 funkcija za rješavanje problema linearne algebre. Jack Little i Steve Bangert reprogramirali su MATLAB na C-u s dodatnim značajkama za komercijalnu verziju programa. Sva trojica zajedno su 1984. u Kaliforniji osnovala The MathWorks, koji razvija, održava i distribuira MATLAB i njegove proizvode širom svijeta. Osnovnom paketu MATLAB-a dodan je velik broj alata i značajki koji čine bogat skup knjižnica čija se veličina mjeri u gigabajtima podataka.

U znanstvenoj je zajednici MATLAB postao iznimno popularan. Ponekad se s pravom naziva i „tehnički jezik“. Jeftina dostupnost digitalnih računalnih resursa ubrzala je njegovu uporabu u industriji i akademiji do te mjere da gotovo svaki laboratorij rabi MATLAB.

Prigrlili su ga znanstvena zajednica, ali i industrija. Inženjer koji nije bio osposobljen za rad u MATLAB-u bio je manje konkurentan na tržištu rada od drugih zbog čega su ga mnoga sveučilišta usvojila u svom nastavnom programu.

No, MATLAB ima dva nedostatka: cijenu i potrebu za licenciranjem. Naime, iako je u početku bio besplatan jer su ga pisali znanstvenici, postavši komercijalni proizvod na tržištu se nalazi pod komercijalno restriktivnom licencom i visokom cijenom. Licencom se ograničilo dijeljenje programa, što je osobito smetalo znanstvenicima, a dodatno je utjecalo i na trošak za obavljanje istraživačkog rada.

Iako cijena licence nije predstavljala problem sveučilištima na zapadu, pokazalo se da je licenca za MATLAB preskupa za ostatak svijeta. Velika znanstvena zajednica drugih zemalja trebala je otvorenu alternativu MATLAB-u. Tako su nastali Octave i Scilab.

Iako je Scilab izuzetno moćan, nije bio kompatibilan sa MATLAB sintaksom. Podrijetlo je dobio iz istih dijelova koda iz kojih je rođen MATLAB, no poslije se razvijao u drugom smjeru i imao je drukčiju sintaksu od MATLAB-a. Tako se MATLAB datoteka nije mogla izravno pokrenuti koristeći Scilab. S druge je strane razvijan Octave koje je zamišljen tako da se datoteke MATLAB-a s ekstenzijom .m mogu u njemu izravno pokrenuti.

Octave je osmišljen 1988. godine. U početku su ga zamislili samo kao popratni program Jamesa B. Rawlingsa sa Sveučilišta Wisconsin-Madison i Johna G. Ekerdta sa Sveučilišta u Teksasu, za sveučilišni udžbenik o dizajnu kemijskih reaktora. Shvatili su da je studentima kemijskog inženjerstva teško kodirati u FORTRAN-u. Umjesto toga, željeli su rješenje u kojem bi se studenti mogli koncentrirati na rješavanje problema kemijskog inženjerstva.

Sljedećih pet godina razvojem Octavea težilo se da postane toliko dobar koliko i MATLAB. Godine 1993. objavljena je verzija 1.0. Suprotno uvriježenom mišljenju, ime Octave nije povezano s glazbom. Ime je dobio po dr. Octaveu Levenspielu, profesoru koji je napisao poznati udžbenik o kemijskom reakcijskom inženjerstvu.

Octave je zaštičen licencom GNU General Public, stoga ga je moguće slobodno mijenjati i redistribuirati prema načelima definiranima licencom. Octave je brzo postao jedan od najpopularnijih projekata otvorenog koda, gdje su programeri (uglavnom studenti) iz cijelog svijeta pridonijeli razvitku programa. Time se obogatio glavni program, a razvili su se i razni specijalizirani paketi.

Njegova velika baza knjižničnih funkcija čini ga dobrim izborom za definiranje inženjerskih problema. Postojeći korisnici MATLAB-a mogu se brzo prilagoditi na novi sustav, a na sličan način, novi korisnici mogu naučiti programirati u Octaveu i zatim se prebaciti u MATLAB okruženje prema potrebi. Od inačice 4.0.3 GNU Octave ima i grafičko korisničko sučelje (GUI).

Osim Scilaba i Octavea za rješavanje numeričkih problema moguće je koristiti i programske jezike poput Pythona, C-a, C++-a i Jave.

Na slici 2.1 prikazan je logotip programa Octave.

Slika 2.1 Logotip programa Octave

2.2 Brzi početak rada s programom Octave

Nakon pokretanja programa Octave, otvara se prozor radnog okružja programa. Program je obično nakon instalacije namješten tako da je prikazan naredbeni prozor (engl. *command window*) u kojem je moguće pisati i izvršavati naredbe Octave programa. Napiše li se u radnoj plohi 2*3 i pritisne tipka Enter, prikazat će se rezultat:

```
>> 2*3
ans =
     6
```

U boji je označeno ono što unosi korisnik, a crno ono što ispisuje program. Očigledno je program izračunao zadanu računsku operaciju.

Upiše li se naredba **`help`**, prikazat će se popis parametara koje treba napisati nakon naredbe **`help`** kako bi se dobio prikaz objašnjenja naredbi. Npr. napiše li se:

```
>> help exp
```

prikazat će se popis osnovnih matematičkih funkcija programa Octave (engl. *elementary functions*, skraćeno elfun). Upisom naredbe **`help`** i naredbe Octave programa, prikazat će se opis te naredbe, npr.

```
>> help exp
 EXP    Exponential.
    EXP(X) is the exponential of the elements of X, e to the X.
    For complex Z=X+i*Y, EXP(Z) = EXP(X)*(COS(Y)+i*SIN(Y)).

    See also expm1, log, log10, expm, expint.

    Overloaded functions or methods (ones with the same name in other
directories)
       help fints/exp.m
       help xregcovariance/exp.m
       help sym/exp.m

    Reference page in Help browser
       doc exp
```

Na temelju opisa može se rabiti funkcija **`exp`** tako da se, npr. izračuna e^3.

```
>> exp(3)
ans =
   20.0855
```

Radna se ploha može obrisati tako da se klikne na izbornik **Edit** i zatim na padajućem izborniku izabere **Clear Command Window**.

Opisani su postupci dovoljni za početak rada u programu Octave. Podrobniji je opis u poglavlju Sučelje programa Octave, koji se nalazi u trećem poglavlju.

Osim opisanog upisa naredbe **`help`** u naredbenom prozoru, opsežna pomoć može se dobiti klikom na izbornik **Help**. U padajućem izborniku zatim treba izabrati željeno područje.

2.3 Dodatni izvori informacija

Postoje desetci knjiga u kojima je opisan program Octave te mnoštvo web stranica posvećenih programu. Ako se u pretraživač Google napiše riječ Octave, dobit će se popis s više od petsto šezdeset milijuna web stranica! Ovdje su zato navedeni najvažniji dodatni izvori informacija:

- GNU Octave
  Web adresa programa je: www.gnu.org./software/octave/index
- Octave forge
  Centralno mjesto na kojem se objavljuju razvojni paketi za Octave. Web adresa sjedišta je: https://octave.sourceforge.io/
- Octave dokumentacija
  Web mjesto na kojem se nalaze upute za korištenje najnovije verzije programa Octave: www.ocatve.org/doc/latest
- John W. Eaton, David Bateman, Søren hauberg, Rik Wehbring. Free Your Numbers (http://octave.org/octave.pdf)
- Jesper Schmidt Hansen. GNU Octave Beginner's Guide (https://jordi.platinum.edu.pl/octave/Jesper%20Schmidt%20Hansen%20-%20GNU%20Octave%20for%20Beginners.pdf)

3. OCTAVE PROGRAM

Octave je program za numeričko računanje koji se sastoji od više cjelina međusobno objedinjenih u Octaveovu radnom okolišu koji olakšava pisanje i provjeru programa.

3.1 Sučelje programa Octave

Octave se može pokrenuti na 2 načina:

1. dvostrukim klikom miša na ikonu Octave programa koja se nalazi na radnoj površini okruženja Windows ili
2. klikom na izbornik **Start/All Programs/Octave-6.4.0.0. (GUI).**

Nakon pokretanja Octavea na zaslonu se pojavljuje prozor programa (slika 3.1).

Kao što je prikazano na slici 3.1, lijeva strana sučelja sastoji se od tri prozora:

- Preglednik datoteka (engl. *File Browser*)
- Radni prostor (engl. *Workspace*)
- Povijest naredbi (engl. *Command History*).

U pregledniku datoteka mogu se pregledavati datoteke u radnom direktoriju i mijenjati njihova imena. Datoteka se može otvoriti klikom na datoteku s ekstenzijom .m. Datoteka se otvara u prozoru za uređivanje (engl. *Editor*) i odatle se može pokrenuti.

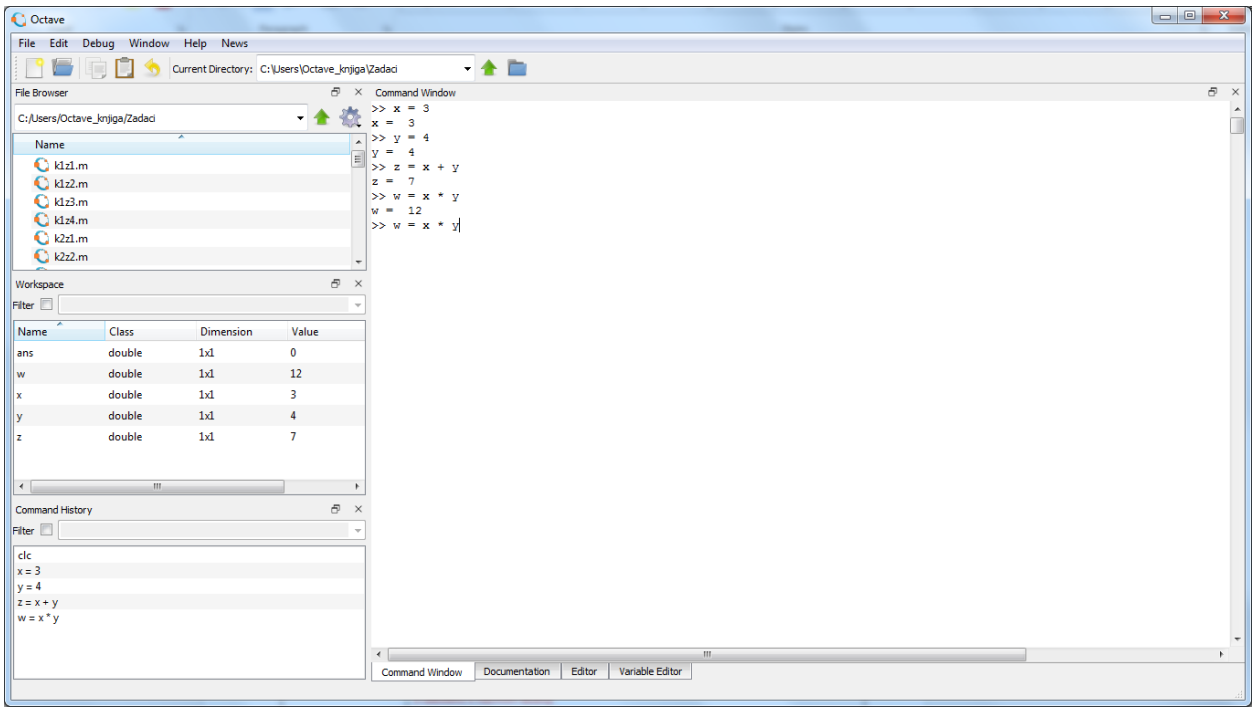
U radnom prostoru prikazana su imena varijabli, vrijednosti i njihova svojstva kao što su vrsta i veličina.

U prozoru Povijest naredbi prikazane su pohranjene naredbe koje su se koristile u prethodnim pokretanjima programa Octave. Naredba se može pokrenuti klikom na nju u naredbenom prozoru, a potom se izvršava u naredbenom retku Octave. Sva tri prozora se mogu uključiti ili isključiti.

S desne strane sučelja nalazi se prozor koji se sastoji od četiri kartice:

- Naredbeni prozor (engl. *Command Window*)
- Uređivač (engl. *Editor*)
- Dokumentacija (engl. *Documantation*)
- Uređivač varijabli (engl. *Variable Editor*)

U naredbeni se prozor naredbe unose u retke. Uređivač se koristi za pisanje .m skripti koje se mogu izvršiti. U prozoru Dokumentacija može se čitati dokumentacija i potražiti pomoć kako bi se saznalo više o naredbama. Octave ima opsežnu dokumentaciju koja početniku omogućuje učenje Octavea samo s naredbenim retkom. Također pomaže iskusnom korisniku koji može potražiti pomoć u korištenju manje uobičajenih naredbi.



Slika 3.1 Prikaz radnog okruženja (prozor Octave programa)

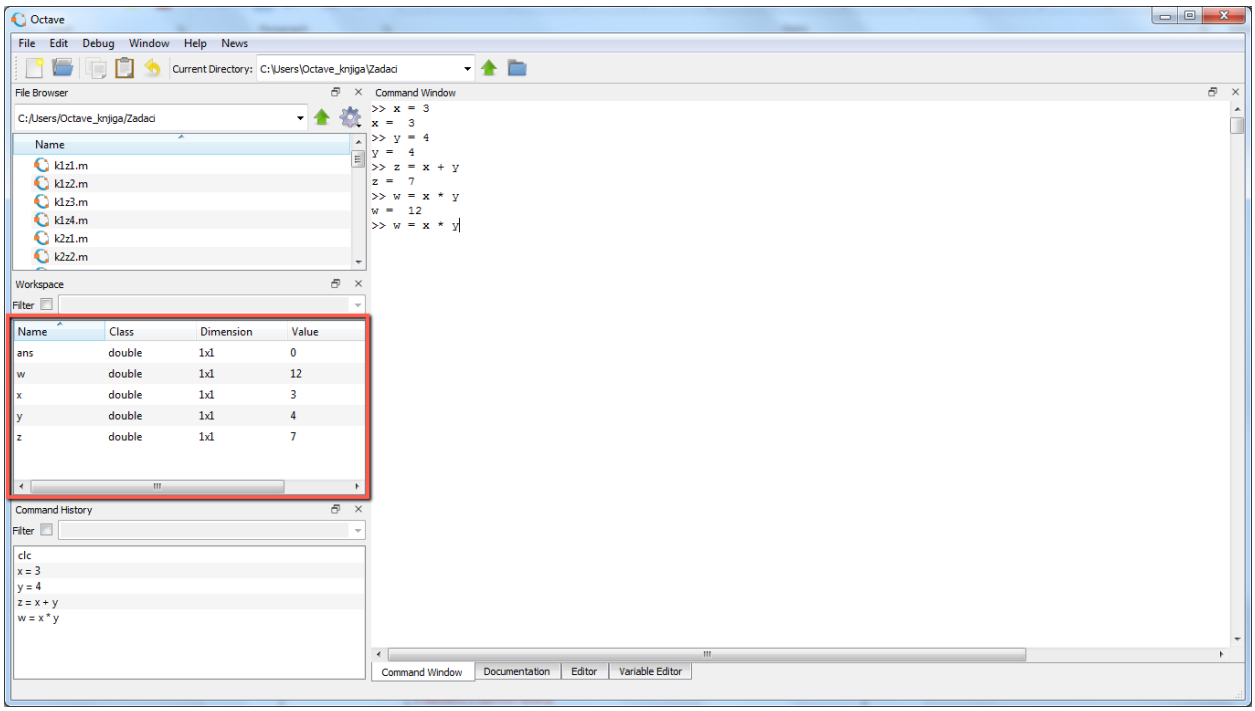
Nakon pokretanja programa Octave, kazalo se nalazi u naredbenom prostoru iza znaka >> gdje se očekuje upis naredbi, a zatim i ispis rezultata.

Radni prostor programa Octave (engl. *Workspace*) mjesto je gdje se spremaju sve varijable koje se definiraju tijekom rada u programu (slika 3.2). U prozoru radnog prostora prikazano je ime varijable ili objekta, vrijednost varijable za skalar, odnosno dimenzija ako se radi o vektoru ili matrici (za objekte su prikazani njihovi detalji) te tip (vrsta) varijable, odnosno objekta.

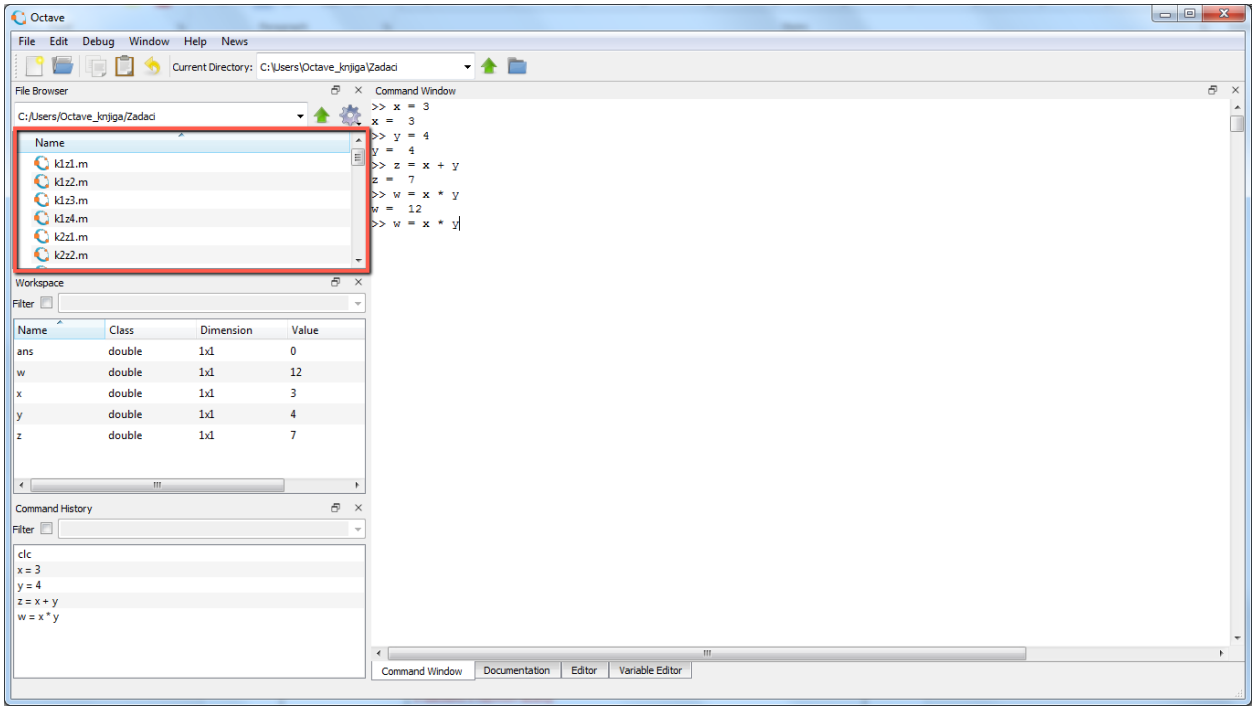
U prozoru radnog imenika (engl. *Current Directory*) nalazi se popis svih datoteka i podimenika koji se nalaze u tom imeniku (slika 3.3).

U prozoru povijest naredbi (engl. *Command History*) može se vidjeti popis svih naredbi koje su bile napisane (od zadnjeg brisanja povijesti naredbi) u naredbenom prozoru (slika 3.4).

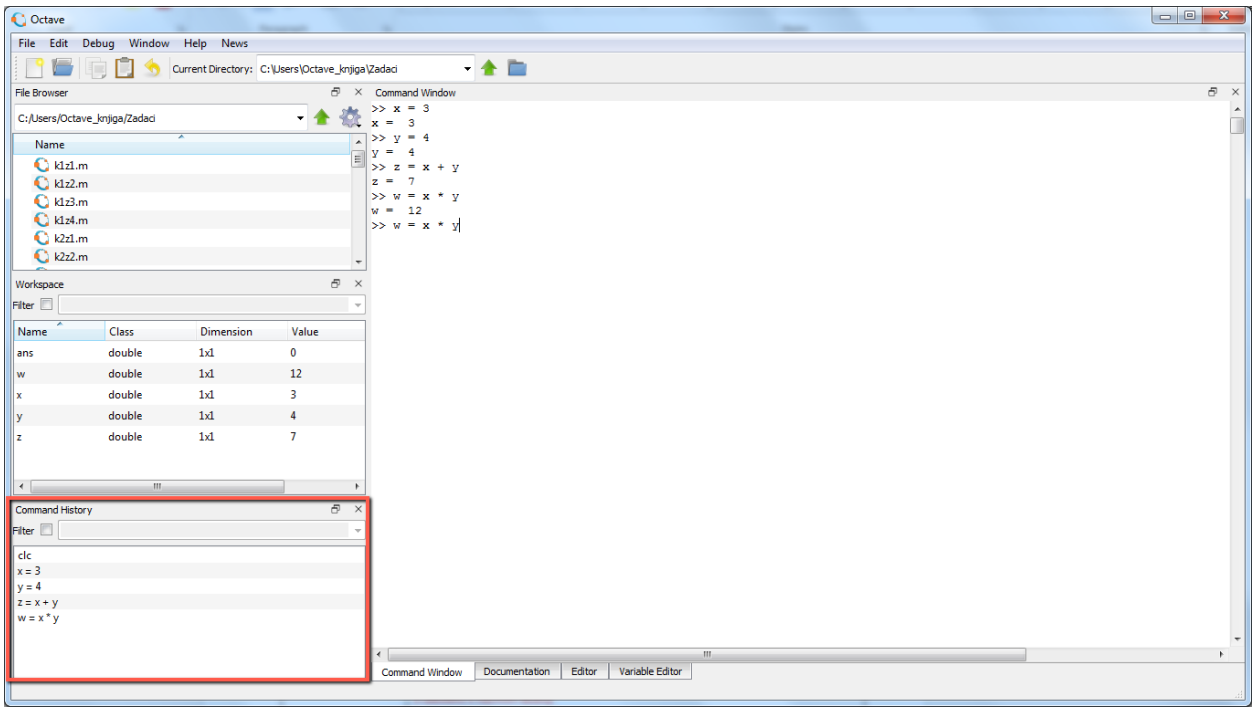
Osim opisanih prozora programa Octave, često se koristi prozor pomoći (engl. *Help*). Na slici 3.5 prikazan je prozor pomoći koji se otvara izbornikom **Help/Octave Help** ili pritiskom na tipku F1.



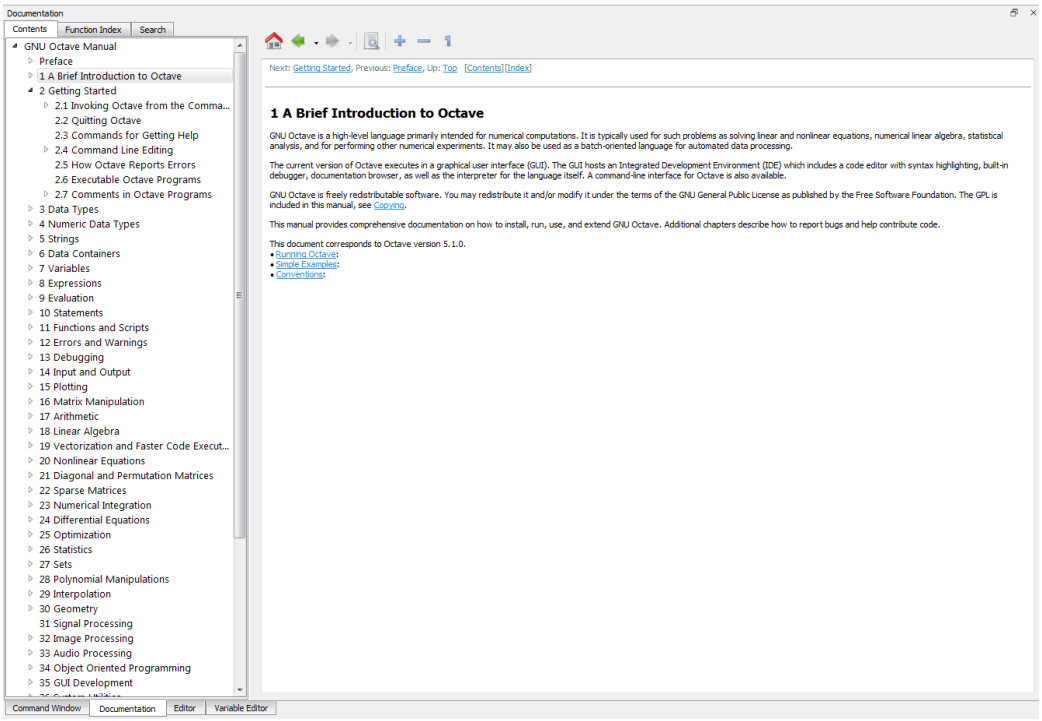
Slika 3.2 Prikaz radnog prostora (engl. *Workspace*)



Slika 3.3 Prikaz radnog imenika (engl. *Current Directory*)



Slika 3.4 Prikaz povijesti naredbi (engl. *Command History*)



Slika 3.5 Prozor pomoći (**Help/Documentation/On disk**)

Iz programa Octave može se izaći:

- izbornikom **File/Exit Octave** ili
- upisivanjem naredbe `quit` ili `exit` u naredbenom prozoru i pritiskom tipke **Enter** (**Return**).

3.2 Promjena radnog imenika

Radni imenik (engl. *current directory*) koji je prikazan u vrpci s alatima može se promijeniti na tri načina:

- otvaranjem izbornika pritiskom na strelicu (slika 3.6) i izborom prethodno otvaranih imenika
- izborom ikone engl. *Browse Directories* (označena zelenom kružnicom na slici 3.6), izborom imenika na željenom disku (slika 3.7) te izborom gumba OK te
- pomoću naredbe **`cd naziv_imenika`** (npr. cd C:\Users\Octave_knjiga) iz naredbenog prozora.

pwd

Ako se želi ispisati putanja trenutačnog radnog imenika u naredbenom prozoru potrebno je upisati naredbu **`pwd`** i pritisnuti tipku Enter, npr.

```
>> pwd
ans =
C:\Users\Octave_knjiga\Documents\Zadatci;
```

Promjena radnog imenika moguća je uporabom naredbe **`cd newpath`**, gdje je **`newpath`** puni put novog radnog imenika, npr.:

```
Cd C:\Users\Octave_knjiga
```

Ako se Octave želi namjestiti tako da se pri pokretanju programa automatski određeni imenik proglašava radnim imenikom, treba to učiniti prije pokretanja Octave programa. Desnom tipkom treba kliknuti na ikonu za pokretanje Octave programa i iz izbornika koji se pojavi birati **Properties**. U prozoru koji se zatim otvori birati jahač **Shortcut**. U okvir **Start in:** treba upisati puni put željenog radnog imenika.

Opisani je postupak potrebno učiniti samo jednom. Nakon toga će se pri svakom pokretanju Octave programa automatski namjestiti radni imenik koji je naveden u okviru **Start in:**.

addpath

Sve skripte i funkcije koje poziva Octave moraju biti dostupne, tj. moraju se nalaziti u putanji Octavea. Kad poziva funkcije Octave pretražuje popis direktorija za datoteku koja sadrži deklaraciju funkcije. Ovaj popis direktorija poznat je kao put učitavanja (engl. *load path*). Prema zadanim postavkama put učitavanja sadrži popis direktorija distribuiranih s Octaveom i trenutačni radni direktorij. Kako biste vidjeli svoj trenutačni put učitavanja, potrebno je u naredbenom prozoru pozvati funkciju **path** bez ikakvih ulaznih ili izlaznih argumenata.

Dodavati ili ukloniti direktorije na ili s putanje učitavanja moguće je ostvariti pomoću funkcija **addpath** i **rmpath**.

Naredbom **`addpath('dir','dir2','dir3' ...)`** mogu se dodati putevi (dir1, dir2 itd.) gdje se nalaze potrebne skripte i funkcije. Relativno se često događa da pri pozivu funkcije ili skipte Octave javi pogrešku da skripta ili funkcija ne postoji, iako je korisnik siguran da postoji. Redovito je riječ o tome da skripta ili funkcija nije dostupna jer nije navedena putanja gdje se ona nalazi. Tada putanju treba dodati naredbom **`addpath`**.

Ako se želi dodati putanju koja uključuje određeni imenik i sve podimenike, treba to učiniti naredbom **`addpath(genpath('directory'))`**, pri čemu je **directory** putanja imenika kojeg se želi dodati u putanju. Naredba **genpath** proslijedit će funkciji **addpath** putanju direktorija i svih njegovih poddirektorija.

rmpath

Naredbom **`rmpath('directory')`** odstranjuje se put **`directory`** iz puteva dostupnih datoteka.

path

Naredba **`path`** prikazuje sve imenike dostupne Octaveu.

restoredefaultpath

Ako zbog nekog razloga korisnik želi postavke Octaveu dostupnih datoteka vratiti na postavke kakve su bile pri prvoj instalaciji programa, treba izvršiti naredbu **`restoredefaultpath`**.

M-datoteke navedene u prozoru radnog imenika mogu se otvoriti:

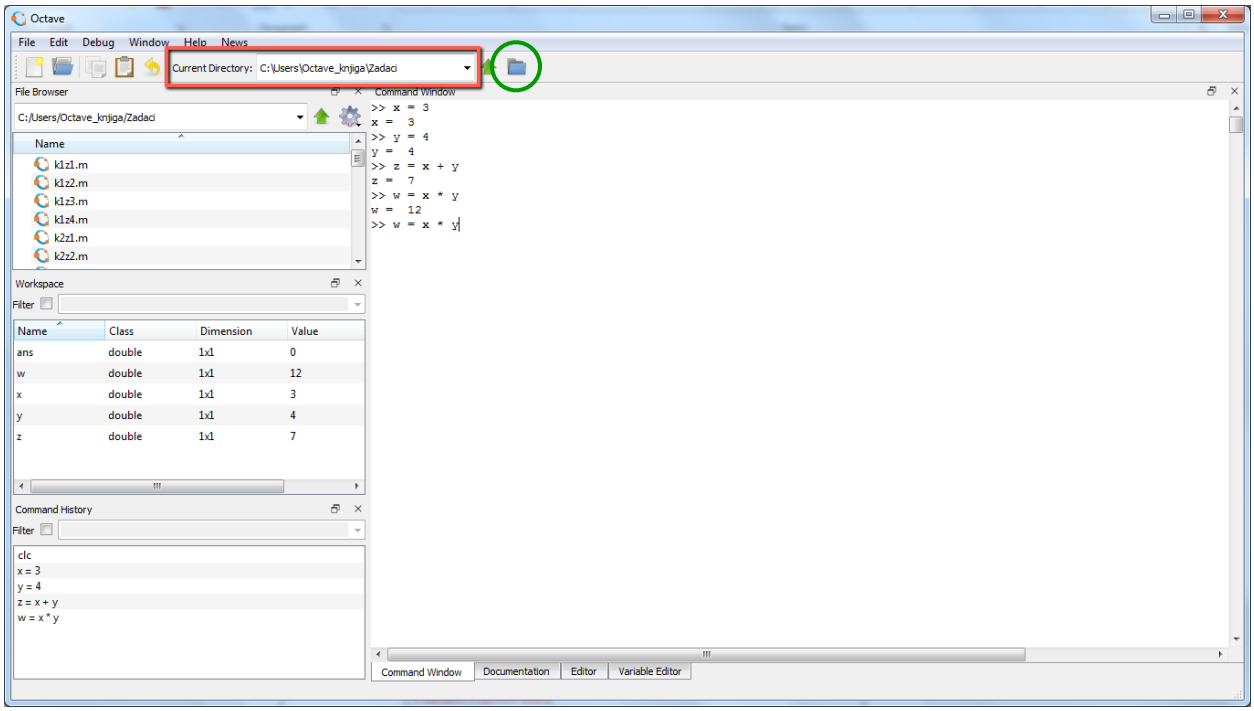
- dvostrukim klikom miša na izabranu M-datoteku
- pritiskom desne tipke miša na željenu datoteku otvara se izbornik u kojem je potrebno lijevom tipkom miša izabrati stavku engl. *Open* (slika 3.8)
- pritiskom desne tipke miša na željenu datoteku otvara se izbornik u kojem je potrebno lijevom tipkom miša izabrati stavku engl. *Open in Text Editor* (slika 3.8).

Desnim klikom miša na M-datoteku u prozoru radnog imenika otvara se izbornik (slika 3.8).

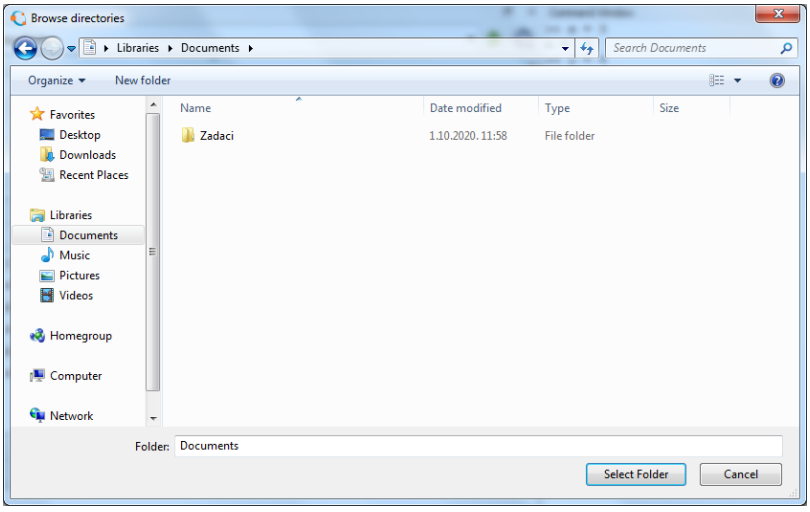
Izborom stavke *Open* otvara se M-datoteka.

Izborom stavke *Run* izvršavaju se naredbe pohranjene u M-datoteci.

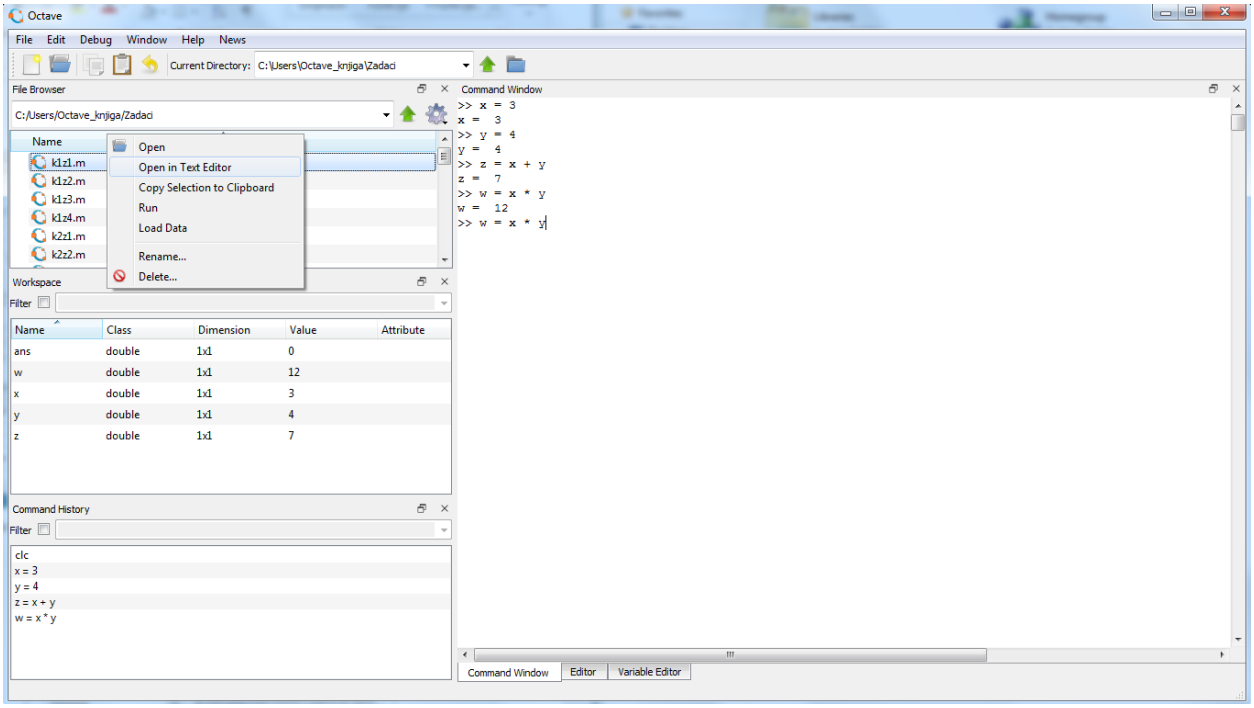
Izborom stavke *Open as Text* otvara se M-datoteka.



Slika 3.6 Izbor imenika koji su već prethodno otvarani



Slika 3.7 Izbor željenog imenika pomoću ikone *Browse directories*



Slika 3.8 Prozor radnog imenika

Izborom stavke *Rename,* može se preimenovati M-datoteka.

Izborom stavke *Delete*, M-datoteka se može izbrisati.

M-datoteka može se izabrati i pokrenuti njezino izvršavanje samo ako je u prozoru *Current Directory* naveden imenik u kojem se nalazi datoteka ili ako je put do datoteke naveden u Octaveovu putu datoteka.

3.3 Naredbeni prozor

Glavni dio Octaveova radnog okoliša naredbeni je prozor (**Command Window**). U njemu je moguće pisati i izvršavati Octaveove naredbe. Pri pokretanju Octave programa otvorit će se i naredbeni prozor s kazalom na mjestu gdje se očekuje upis naredbe.

3.4 Radni prostor

Octave radni prostor (engl. *workspace*) memorijsko je područje u kojem se spremaju sve varijable koje nastaju tijekom rada u naredbenom prozoru. Te su varijable vidljive i dostupne u naredbenom prozoru. Varijable nastaju u radnom prostoru tako da se rabe funkcije, operatori, pozivaju M-datoteke ili da se dobavi pohranjeno stanje radnog prostora iz datoteke. Npr. operatorima pridjeljivanja:

```
>> x = 13;
>> y = 5.3;
```

u radnom prostoru nastaju varijable **`x`** i **`y`**, te im se pridjeljuju odgovarajuće vrijednosti.

Varijable se iz radnog prostora mogu obrisati (ukloniti) naredbom **`clear varname`** (brisanje varijable **`varname`**) ili **`clear`** (brisanje svih varijabli iz radnog prostora). Npr. naredbom:

```
>> clear x
```

briše se varijabla **x** iz radnog prostora. Na zaslonu se neće vidjeti nikakav odziv, ali će varijabla u memoriji biti izbrisana. Naredbom

```
>> clear
```

brišu se sve varijable radnog prostora (nema odziva na zaslonu).

3.5 Izvršavanje naredbi iz naredbenog prozora

Octave naredbe mogu se pisati izravno u naredbenom prozoru (engl. *command window*). Nakon što se napiše jedna naredba i pritisne tipka **Enter**, naredba će se izvršiti. Korisnik mora nakon svake naredbe pritisnuti tipku **Enter**, pa je to prikladan način izvršavanja naredbi i provjere manjih dijelova programa. Ako se naredbe izvršavaju iz naredbenog prozora pisanjem jedne po jedne, korisni mogu biti niže navedeni savjeti.

Često je pri pisanju naredbi u naredbenom prozoru nakon određenog broja naredbi, potrebno napisati naredbu koja je već bila napisana. Umjesto ponovnog pisanja cjelokupne naredbe, brže je to učiniti ovako:

- Pritiskom strelice na tipkovnici prema gore ili prema dolje pokazat će se prije upisane naredbe (pretraživanje prije napisanih naredbi).
- Pisanjem jednog ili nekoliko početnih slova prije napisane naredbe i zatim pritiskom strelice na tipkovnici prema gore i prema dolje pojavit će se prije napisane naredbe koje počinju napisanim slovima (pretraživanje prije napisanih naredbi koje počinju napisanim slovima).
- Pisanjem jednog ili nekoliko početnih slova prije napisane naredbe i zatim pritiskom tipke **tab** na tipkovnici ispisat će se prije napisana naredba koja počinje napisanim slovima.
- Izvršenjem naredbe **commandhistory** kojom se otvara prozor **command history** (ako već nije otvoren) s kronološkim popisom izvršenih naredbi, u kojem se može izabrati naredba koja će se ponovo izvršiti.

Za prikaz ispravne sintakse pojedinih naredbi može se rabiti naredba **help commandname** (**commandname** je ime naredbe).

exit, quit

Naredba **exit** ili **quit** napisana u naredbenom prozoru zatvara Octave prozor i završava rad u Octave okolišu (izlaz iz programa).

ver

Naredba **ver** prikazuje podatke o inačicama Octavea. Isto se može vidjeti pomoću izbornika **Help/About Octave**.

version

Naredba **version** prikazuje podatke o inačici Octave programa. Isto se može vidjeti pomoću izbornika **Help/About Octave**.

clc

Sadržaj naredbenog prozora može se izbrisati (isprazniti) naredbom **clc**. Tom se naredbom briše samo sadržaj prozora, ali ne i vrijednost varijabli radnog prostora (memorija radnog prostora u kojoj se čuvaju varijable ostaje netaknuta naredbom **clc**).

Razlika između naredbe **clc** i **clear** je što se naredbom **clc** briše tekst u naredbenom prozoru, dok sve prethodno definirane varijable ostaju pohranjene u radnom prostoru i mogu se ponovno prikazati u naredbenom prozoru. Naredbom **clear** varijabla se briše u radnom prostoru, a stanje naredbenog prozora se ne mijenja. Pozivanjem obrisane varijable u naredbenom prozoru javlja se upozorenje da varijabla nije definirana.

format

Naredba **format** koristi se za oblikovanje prikaza numeričkih podataka. Zadana postavka prikaza u Octaveu je tip *short*. Tablica 3.1 prikazuje mogućnosti oblikovanja prikaza numeričkih podataka.

Tablica 3.1 Oblikovanje prikaza numeričkih podataka

Tip	Rezultat	Primjer
short	Zapis s nepomičnom decimalnom točkom s 5 značajnih znamenki	0.4000
long	Zapis s nepomičnom decimalnom točkom s 15 značajnih znamenki	0.400000000000000
short e	Eksponencijalni zapis pri kojem se broj prikazuje mantisom i eksponentom, pri čemu mantisa ima 5 značajnih znamenki	4.0000e-01
long e	Eksponencijalni zapis pri kojem se broj prikazuje mantisom i eksponentom, pri čemu mantisa ima 15 značajnih znamenki	4.000000000000000e-01
short g	Optimalno se izabire između prikaza s nepomičnom decimalnom točkom i eksponencijalnog formata do 5 značajnih znamenki	0.4
long g	Optimalno se izabire između prikaza s nepomičnom decimalnom točkom i eksponencijalnog formata do 15 značajnih znamenki	0.4
short eng	Inženjerski zapis (prikaz tipa short s eksponentom koji je višekratnik broja 3)	400.0000e-003
long eng	Inženjerski zapis (prikaz tipa long s eksponentom koji je višekratnik broja 3)	399.999999999999943e-003
bank	Zapis prikladan za prikaz valuta (dva decimalna mjesta)	0.40

Primjer različito oblikovanog ispisa broja π.

```
>> format short
>> pi
ans = 3.1416
>> format long
>> pi
ans = 3.141592653589793
>> format short e
>> pi
ans = 3.1416e+00
>> format long e
>> pi
ans = 3.141592653589793e+00
>> format short g
>> pi
ans = 3.1416
>> format long g
>> pi
ans = 3.141592653589793
>> format short eng
>> pi
ans = 3.1416e+000
>> format long eng
>> pi
ans = 3.141592653589793e+000
```

more

Naredbom **`more`** moguće je odrediti koliko će se ispisa teksta prikazati na zaslonu monitora.

Naredbom **`more on`** određuje se prikaz ispisa koji zauzima jedan prozor. Ako se želi nastaviti ispis, potrebno je pritisnuti odgovarajuću tipku (vidjeti tablicu 3.2). Nakon nastavka ispisa ponovo će se ispisati jedan prozor, i tako redom.

Naredba **`more(n)`** određuje prikaz n redaka. Za nastavak ispisa treba pritisnuti odgovarajuću tipku (tablica 3.2). Nakon nastavka ispisa ponovo će se ispisati jedan prozor s n redaka, i tako redom.

Tablica 3.2 Rezultat naredbe **`more`**

Pritisnuti tipku	Rezultat
Return (Enter)	Prikaz sljedećeg retka.
Razmaknicu (engl. *space*)	Prikaz sljedeće stranice.
Q	Zaustavlja prikaz. Nije dobro rabiti Ctrl+C za prekid ispisa jer može doći do ispisa pogreške u naredbenom prozoru.

Naredba **`more off`** isključuje prikaz ispisa koji zauzima jedan prozor. Uporabom te naredbe prikazat će se bez zaustavljanja cjelokupan ispis koji je rezultat izvršavanja programa.

home

Naredba **`home`** postavlja kazalo u gornji lijevi dio naredbenog prozora. Sadržaj naredbenog prozora pritom se briše.

diary

Naredbe i rezultati naredbenog prozora mogu se pohraniti u datoteku. U naredbenom prozoru potrebno je napisati naredbu **`diary filename`**. Nakon izvršenja te naredbe, sve što se prikaže u naredbenom prozoru pohranjivat će se u datoteku **`filename`**. **`Filename`** je tekstualna datoteka koja se može čitati i uređivati bilo kojim programom za uređivanje teksta. Kad se želi prekinuti pohrana, treba napisati naredbu **`diary off`**. U tablici 3.3 prikazani su oblici naredbe **`diary`**.

Tablica 3.3 Oblici naredbe **`diary`**

`diary('filename')` **`diary filename`**	Uključuje pohranu u datoteku **`filename`** u tekućem Octaveovu imeniku (**Current Directory**). Ako datoteka **`filename`** već postoji, podatci će se dodavati na kraj datoteke. Ako se navede naredba **`diary`** bez imena datoteke, podatci se spremaju u datoteku s nazivom **`diary`**.
`diary off`	Isključuje pohranu u datoteku.
`diary on`	Uključuje pohranu u datoteku s nazivom koji je određen naredbom **`diary('filename')`**. Podatci će se dodavati na kraj datoteke. Ako takva datoteka ne postoji, podatci se spremaju u datoteku s nazivom **`diary`**.
`diary`	Prebacuje iz jednog stanja u drugo (isključeno/uključeno). Stanje se može vidjeti pomoću naredbe **`get(0,'diary')`**. Ako datoteka ne postoji, podatci se spremaju u datoteku s nazivom **`diary`**.

3.6 M-datoteke

Za veće programe prikladnije je napisati pojedine dijelove programa ili cjelokupan program, pohraniti ga u datoteku i zatim ga izvršiti unutar Octaveova radnog prostora (engl. *base workspace*) pozivom pohranjene datoteke. To se može učiniti slično kao i u bilo kojem drugom programskom jeziku. Treba napisati Octaveov program u txt formatu i pohraniti ga s produžetkom (engl. *extension*) **`m`** (npr. **`test.m`**). Takve se datoteke nazivaju M-datoteke (engl. *M-files*). M-datoteke su dakle datoteke koje sadrže Octaveov program.

Pretpostavimo da je napisan program za izračunavanje površine kruga i da je taj program spremljen kao M-datoteka **`zadatak.m`** (naziv može biti bilo koji naziv kojeg prihvaća operacijski sustav, a produžetak mora biti m). Nakon toga je u Octaveovu naredbenom prozoru dovoljno napisati:

```
>> zadatak
```

i izvršit će se redom sve naredbe napisane u tom programu, tj. pohranjene u M-datoteci **`zadatak.m`**.

Octave ima dio za pisanje M-datoteka pod nazivom **Editor** koji je integriran u Octaveov okoliš pa je najpraktičnije M-datoteke pisati u tom dijelu programa. **Editor** se može pokrenuti iz glavnog Octaveova izbornika na više načina.

Ako se želi stvoriti nova M-datoteka, treba u izborniku izabrati **File/New/New script**. Ako se želi otvoriti postojeća M-datoteka, treba u izborniku izabrati **File/Open…** i zatim izabrati M-datoteku (datoteku s produžetkom m). U oba slučaja otvorit će se program **Editor** za pisanje i obradu M-datoteka.

M-datoteka sprema se tako da se u izborniku programa za pisanje M-datoteka **M-file editor** izabere **File/Save As…**.

what

Izvršenjem naredbe **`what`** prikazat će se sve m, mat, mex, oct, mdl, slx i p-datoteke te imenici s Octaveovim klasama koji se nalaze u tekućem imeniku.

Popis svih datoteka u tekućem imeniku moguće je vidjeti u prozoru **Current Directory** koji se otvara izbornikom **Desktop/Current Directory**.

```
>> what

M-files in directory C:\Users\Octave_knjiga\Zadatci:

   k1z1.m   k1z4.m   k2z3.m
   k2z6.m   k3z2.m   k3z5.m
   k4z1.m   k4z4.m   k4z7.m
```

which

Naredba **`which filename`** prikazuje puni put datoteke **`filename`**. Služi za pronalaženje u kojem se imeniku nalazi funkcija ili datoteka. Potrebno je navesti točan naziv funkcije ili datoteke. Npr. naredba:

```
>> which k1z1.m
'k1z1.m' is the file C:\Users\Octave_knjiga\Zadatci\k1z1.m
```

prikazuje puni put (engl. *path*) datoteke **`zadatak.m`**, a naredba:

```
>> which sqrt
'sqrt' is a built-in function from the file libinterp/corefcn/mappers.cc
```

prikazuje puni put do ugrađene funkcije **`sqrt`**.

lookfor

Naredba **`lookfor keyword`** pretražuje prvi redak komentara svih M-datoteka i prikazuje popis onih u čijem je zaglavlju pronađena riječ **`keyword`**. Stoga je pri pisanju M-datoteka važno u prvi red komentara napisati smisleni sadržaj kako bi se mogao pronaći naredbom **`lookfor`**.

Često je traženu funkciju lakše pronaći naredbom **`lookfor`** nego naredbom **`which`** jer je kod naredbe **`which`** potrebno navesti točan naziv funkcije, npr.:

```
>> which sqr
'sqr' not found.
```

U gornjem primjeru nije pronađena nijedna datoteka.

Naredba **`lookfor`** pronaći će sve M-datoteke kod kojih se u prvom retku komentara nalazi riječ **`keyword`**, npr.:

```
>> lookfor sqr
I            Return a scalar, matrix, or N-dimensional array whose
             elements are all equal to the pure imaginary unit, definned
             as 'sqrt (-1)'.
sqrt         Compute the square root of each element of X.
sqrtm        Compute the matrix square root of the square matrix A.
realsqrt     Return the real-valued square root of each element of X
```

U ovom su primjeru pronađene sve datoteke koje u prvom retku imaju riječ sqr u bilo kojem obliku (može biti i dio neke riječi). Popis pronađenih datoteka ovisi o instalaciji Octavea i o datotekama koje postoje na računalu na kojem se naredba **`lookfor`** izvršava.

3.7 Skripte

Postoje dvije vrste M-datoteka. Najjednostavnije vrste M-datoteka nazivaju se skripte (engl. *script*). Riječ je o M-datotekama u kojima se nalazi jedna ili više Octaveovih naredbi, ali prva naredba NIJE definicija funkcije.

Octave izvršava skriptu redak po redak, rabeći pritom trenutačne vrijednosti varijabli koje se nalaze u radnom prostoru (engl. *base workspace*). To je isto kao da se naredbe zapisane u skripti upisuju jedna po jedna u naredbenom prozoru. Rezultat će biti isti, samo što je mnogo jednostavnije i brže činiti to pomoću skripte. Ukratko, skripte jednostavno prepisuju i izvršavaju svoj sadržaj u naredbeni prozor i pritom rabe radni prostor za varijable.

Skripte se rabe kad se program sastoji od više naredbi ili kad se program želi pohraniti, mijenjati i dopunjavati.

Varijable i sve ostale vrijednosti moraju biti definirane u skripti ili radnom prostoru. Skripte nemaju mogućnost prihvaćanja ulaznih argumenata. Po završetku skripte, novostvorene vrijednosti nalaze se u radnom prostoru i na raspolaganju su (npr. mogu ih rabiti druge skripte ili funkcije).

Neka je, npr. napisan i u M-datoteku (skriptu) pod nazivom **`rctngl.m`** pohranjen program za izračunavanje površine pravokutnika stranica x i y.

```
>> z = x * y
```

Ako u naredbenom prozoru upišemo:

```
>> rctngl
```

kao rezultat prikazat će se poruka:

```
error: 'x' undefined near line 1 column 3
error: called from
    rctngl at line 1 column 2
```

Poruka govori o tome da u radnom prostoru ne postoji varijabla **`x`** (nije određena, nije definirana). Pogreška je nastala zato što u radnom prostoru prije pozivanja skripte **`rctngl.m`** nije određena vrijednost varijable **`x`**. Prije izvođenja skripte treba u radnom prostoru ili u samoj skripti odrediti sve varijable koje skripta rabi, npr.:

```
>> x = 3
x =
     3
>> y = 4
y =
     4
>> rctngl
z =
    12
```

Ovaj se put program izvršio bez pogreške. Rezultat je pohranjen u varijablu **`z`** koja se nalazi u radnom prostoru. Njezina se vrijednost može vidjeti tako da se u naredbeni prozor napiše **`z`** i pritisne tipka **Enter**:

```
>> z
z =
```

```
    12
```

Varijable **x**, **y** i **z** ostat će u radnom prostoru sve dok ih se ne promijeni, izbriše ili izađe iz Octave programa.

Skripta se može izvršiti i izravno iz programa za pisanje M-datoteka (**Editor**) izborom naredbe iz izbornika **Save File and Run** ili pritiskom na tipku **F5**.

Radni prostor i skripte rabe zajedničke varijable. To znači da se promjena vrijednosti varijable u radnom prostoru odražava na istoimenoj varijabli u skripti i obrnuto. Istoimena varijabla u radnom prostoru i skripti je ista.

Varijabla definirana u radnom prostoru ili skripti nije dostupna istoimenoj varijabli u funkciji (funkcije su opisane u nastavku). Istoimene varijable radnog prostora (a time i skripti) i funkcija odvojene su i različite. Promjena vrijednosti varijable radnog prostora (a time i skripti) ne utječe na vrijednost istoimene varijable u funkcijama.

Poseban su slučaj istoimene globalne varijable koje su iste (dostupne, vidljive) u radnom prostoru, skriptama i funkcijama (globalne varijable opisane su u nastavku).

Treba uočiti ove značajke skripti:

- Vrijednosti svih podataka koji se rabe u skripti moraju biti određene u radnom prostoru ili unutar skripte.
- Rezultat skripte pohranjen je u varijabli koja je dostupna (vidljiva) u radnom prostoru.
- Skripte se ne mogu prevesti u drugi programski jezik ili u izvršni oblik (kompajlirati, engl. *compile*), ali se mogu prevesti u pseudokod (engl. *P-code*).
- Skripta može promijeniti vrijednosti varijabli u radnom prostoru.

Zadnje navedeno svojstvo često nije poželjno i može izazvati pogreške koje nije lako otkriti. Kad god je to moguće, preporučuje se rabiti funkcije umjesto skripti.

type

Izvršenjem naredbe **type filename** u naredbenom prozoru ispisuje se sadržaj M-datoteke naziva **filename.m**. To je prikladno za brzi prikaz relativno kratkih programa. Za veće programe bolje je rabiti program za pisanje M-datoteka (**M-file editor**). Npr. naredba **type sinplot** prikazuje sadržaj M-datoteke (skripte) **sinplot.m**:

```
>> type sinplot

x = [0:0.1:10];
y = sin(x);
plot(x,y)
xlabel('x')
ylabel('y')
title('Graf sin(x)')
```

echo

Ako se pri izvršenju skripte želi prikazati sve naredbe skripte, prije izvršenja skripte treba izvršiti naredbu **echo on**. Nakon toga će se prikazivati sve naredbe svih skripti koje će se izvršavati. Ako se želi spriječiti prikaz naredbi skripte, pri izvršenju treba izvršiti naredbu **echo off**. Ako se izvrši naredba **echo**, promijenit će se trenutačno stanje prikaza naredbi skripte (ako je prikaz uključen, isključit će se i obrnuto).

Kad je riječ o funkcijama (opisane u nastavku), naredba **echo** ima nekoliko oblika prikazanih u tablici 3.4., pri čemu **fcnname** predstavlja korisničku funkciju.

Tablica 3.4 Oblici naredbe **echo** za funkcije

Oblik naredbe	Rezultat
echo fcnname on	Uključuje prikaz naredbi za funkciju **fcnname**

`echo fcnname off`	Isključuje prikaz naredbi za funkciju **`fcnname`**
`echo fcnname`	Mijenja stanje prikaza naredbi za funkciju **`fcnname`**
`echo on all`	Uključuje prikaz naredbi za sve funkcije
`echo off all`	Isključuje prikaz naredbi za sve funkcije

input

Prije izvršenja skripti treba definirati sve varijable koje se nalaze u skripti. Za korisnika je razumljivije definiranje varijabli pomoću naredbe **`input`** nego izravno unošenje u radni prostor. Naredba **`input`** ispisuje poruku i čeka unos podatka od korisnika. Primjer skripte **`circarea.m`** koja rabi naredbu **`input`**:

```
r = input('Unesi polumjer: ');
y = r^2*pi;
y
```

Izvršavanjem te skripte pojavit će se poruka:

```
>> circarea
Unesi polumjer:
```

i tek nakon što korisnik unese podatak i pritisne tipku **Enter**, nastavit će se izvršenje programa:

```
Unesi polumjer: 4
y =
   50.2655
```

Takav način definiranja varijabli mnogo je razumljiviji za korisnika jer se može opisati o kakvoj je varijabli riječ.

disp

Pri prikazu rezultata često je korisno, osim samog numeričkog rezultata, napisati i tekst koji objašnjava o čemu je riječ. Naredbom **`disp`** prikazuje se poruka korisniku bez zaustavljanja programa. Npr. prepravi li se skripta **`circarea.m`**, pri ispisu rezultata prikazat će se i tekstualna poruka.

```
r = input('Unesi polumjer: ');
y = r^2*pi;
disp (['Površina kruga je: ', num2str(y)])
```

```
>> circarea
Unesi polumjer: 3
Površina kruga je: 28.2743
```

Argumenti funkcije **`disp`** su nizovi (engl. *string*), pa numerički podatak treba pretvoriti u niz ako ga se želi proslijediti kao argument naredbi **`disp`**. To u gornjem primjeru čini funkcija **`num2str`**.

error

Naredba **`error`** ima oblik **`error('message')`**, a prikazuje poruku o pogrešci, npr.:

```
>> error('Ovo je pogreška')
Error: Ovo je pogreška
```

Naredba **error** ima smisla samo ako je napisana unutar skripte ili funkcije. Napisana u naredbenom prozoru nema smisla.

lasterr

Naredba **lasterr** prikazuje zadnju Octaveovu poruku o pogrešci.

warning

Naredba **warning** ima oblik **warning ('message')**, a prikazuje poruku upozorenja, npr.:

```
>> warning('Ovo je upozorenje')
Warning: Ovo je upozorenje
```

lastwarn

Naredba **lastwarn** prikazuje zadnju Octaveovu poruku upozorenja.

eval

Nareba **eval** izvršava niz (engl. *string*) koji joj je argument. Npr. naredba:

```
z = eval('sin(x)')
```

isto je što i naredba:

```
z = sin(x)
```

Primjer uporabe naredbe **eval** stvaranje je 12 matrica uporabom petlje i naredbe **eval**:

```
for n = 1:12
      magic_str = ['m', int2str(n),' = magic(n)'];
      eval(magic_str)
end
```

evalc

Nareba **evalc** ima oblik **x = evalc(w)** izvršava niz (engl. *string*) koji joj je argument, ali rezultat sprema u znakovni niz **x**.

```
for n = 1:12
      magic_str = ['m', int2str(n),' = magic(n)'];
      x = evalc(magic_str)
end
```

Izvršenjem skripte u radnom prostoru postoji varijabla **x** čiji je sadržaj znakovni niz koji je rezultat naredbe **evalc**. To se može provjeriti naredbom **whos x**:

```
>> whos x
   Attr Name        Size                     Bytes  Class
   ==== ====        ====                     =====  =====
        x           1x3897                    3897  char
```

U ovom prikazu izostavljene su ostale varijable; prikazana je samo varijabla **x**.

Kad se rabi naredba **evalc**, naredbe **diary**, **more** i **input** isključuju se.

evalin

Naredba **evalin** izvršava niz koji (engl. *string*) joj je argument u kontekstu navedenog radnog prostora. Radni prostor može biti **'base'** (osnovni radni prostor) ili **'caller'** (radni prostor iz kojeg se funkcija poziva). Naredba ima oblik:

```
[a1, a2, a3, ...] = evalin(ws, expression)
```

Gdje su **a1, a2, a3, ...** izlazne vrijednosti (rezultat), **ws** radni prostor (**'base'** ili **'caller'**) i **expression** niz koji je bilo koji važeći Octave izraz. Npr. neka je definirana i u funkcijsku M-datoteku pohranjena funkcija:

```
function x = exam()
evalin('caller','sin(x)');
endfunction
```

Neka je definirana i druga i u funkcijsku M-datoteku pohranjena funkcija:

```
function examcall()
exam;
endfunction
```

Ako se pokuša izvršiti skripta:

```
>> x=1
x =
     1
>> examcall
error: 'x' undefined near line 1 column 5
```

prikazat će se poruka o pogrešci jer varijabla **x** definirana u osnovnom radnom prostoru nije vidljiva u radnom prostoru funkcije **examcall** koja poziva funkciju **exam**. Varijabla **x** mora se definirati u radnom prostoru funkcije **examcall**, npr.:

```
function examcall(z)
x = z;
exam;
endfunction
```

pa je nakon toga moguće pozvati i izvršiti funkciju **exam**:

```
>> examcall(1)
ans =
    0.8415
```

jer je varijabla **x** definirana u radnom prostoru funkcije **examcall** i vidljiva je pri pozivu funkcije **exam**.

Ako se funkcija **exam** prepravi da izgleda ovako:

```
function x = exam()
evalin('base','sin(x)');
endfunction
```

tada će varijabla **x** iz osnovnog radnog prostora biti vidljiva i unutar funkcije **examcall** koja poziva funkciju **exam**:

```
>> x=1
x =
     1
>> examcall
ans =
    0.8415
```

4. VARIJABLE

Pretpostavimo da u radnom prostoru postoji varijabla pod nazivom **xname**, da postoji i funkcija istog imena **xname** pohranjena u datoteku **xname.m**, da postoji podfunkcija istog imena unutar primarne funkcije **xname**, te da postoji i privatna funkcija naziva **xname**. Ako se u naredbenom prozoru napiše:

```
>> xname
```

što će se dogoditi? Koje će od gore navedenog Octave smatrati pozvanim?

Octave raspoznaje poziv ovim redoslijedom (od važnijeg prema manje važnom):

1. Provjerava je li riječ o varijabli.
2. Provjerava je li riječ o funkciji (ili podfunkciji ako se poziva iz funkcije).
3. Provjerava je li riječ o privatnoj funkciji.
4. Provjerava je li riječ o funkciji u trenutačno namještenom putu traženja Octaveova okoliša (bira prvu funkciju na koju naiđe pri pretraživanju puta).

Primjerice, ako u radnom prostoru postoji varijabla istog naziva kao i funkcija pri navođenju tog naziva u naredbenom prozoru (ili unutar druge funkcije), smatrat će se da se traži vrijednost varijable, a ne da se poziva funkcija. Npr. ako se definira varijabla **clc**:

```
>> clc = 5
clc =
     5
```

i nakon toga pokuša izvršiti Octaveova naredba **clc**:

```
>> clc
```

neće se izbrisati radna ploha naredbenog prozora (što je uobičajen rezultat izvršenja naredbe **clc**), već će se prikazati vrijednost varijable **clc**:

```
>> clc
clc =
     5
```

Varijabla radnog prostora **clc** "sakrila" je ugrađenu naredbu **clc** jer je prema navedenom prvenstvu traženja varijabla ispred funkcije ili podfunkcije.

4.1 Lokalne i globalne varijable

Za razumijevanje i ispravnu uporabu Octave programa treba znati razliku između globalnih i lokalnih varijabli.

Varijabla definirana u radnom prostoru (engl. *base workspace*) dostupna je u radnom prostoru i skriptama, ali nije dostupna funkcijama. Varijabla definirana unutar funkcije dostupna je samo unutar funkcije u kojoj je definirana i nije dostupna u radnom prostoru, skriptama ili drugim funkcijama. Svaka funkcija ima svoj memorijski prostor (engl. *function workspace*, *local workspace*, *private workspace*) u kojem su pohranjene varijable samo te funkcije. Podfunkcije imaju svoj memorijski prostor, pa varijable definirane unutar podfunkcije nisu vidljive u nadređenoj funkciji.

Želi li se neku varijablu dijeliti između radnog prostora (a time i skripti) i funkcija, treba je i u radnom prostoru (ili skripti) i svakoj funkciji definirati kao globalnu. U tom će se slučaju promjena vrijednosti globalne varijable odraziti istodobno i u radnom prostoru (a time i u svim skriptama) i u svim funkcijama gdje je varijabla navedena kao globalna. Ako se želi varijablu dijeliti između funkcija, ali ne i radnog prostora, potrebno ju je definirati kao globalnu u svim funkcijama među kojima se želi dijeliti, ali ne i u radnom prostoru (ili skripti).

Za globalne varijable vrijedi sljedeće:

- Kako bi globalna varijabla bila prepoznata u radnom prostoru, mora se u naredbenom prozoru (ili skript datoteci) izvršiti naredba **`global VAR_NAME`**.
- Varijabla se mora deklarirati kao globalna u svakoj funkciji koja je treba prepoznati. Globalna varijabla dostupna je (vidljiva) samo funkcijama u kojima je definirana.
- Naredba **`global`** mora prethoditi prvoj uporabi varijable. Stoga se preporučuje da naredba **`global`** bude pri vrhu M-datoteke.
- Varijabli se može dodjeljivati vrijednost (proizvoljan broj puta) na svim mjestima gdje je prepoznata. Svaka promjena globalne varijable odrazit će se na svim mjestima gdje je ta varijabla definirana kao globalna.
- Za globalne varijable preporučuje se uporaba dugačkih opisnih imena i samo velikih slova, kako bi se lakše uočila razlika između njih i lokalnih varijabli.
- Globalne se varijable iz radnog prostora brišu naredbom **`clear global`**.

Ako to nije nužno, treba izbjegavati uporabu globalnih varijabli. Preporučuje se između funkcija te između funkcija i radnog prostora (i skripti) podatke razmjenjivati argumentima (ulaznim varijablama) i izlaznim varijablama (rezultatom funkcije).

Neka postoji skripta **`area.m`**:

```
w = x*y
```

Ako u radnom prostoru definiramo varijable **`x`** i **`y`**:

```
>> x = 3
x =
     3
>> y = 4
y =
     4
```

i zatim izvršimo skriptu **`area.m`**:

```
>> area
```

prikazat će se rezultat:

```
w =
    12
```

Varijable **x** i **y** definirane u radnom prostoru vidljive su u skripti.

Neka postoji funkcija **area** pohranjena u M-datoteci **area.m**:

```
function [w] = area(x,y);
w = x*y;
```

Izvršimo li funkciju **area** pozivanjem u naredbenom prozoru, prikazat će se poruka o pogrešci:

```
>> area(x,y)
??? Input argument "x" is undefined.
Error in ==> area at 2
w = x*y;
```

Varijable **x** i **y** koje su definirane u radnom prostoru nisu vidljive unutar funkcije.

Preradimo funkciju **area** u oblik:

```
function [w] = area(x);
x = 2;
w = x;
```

Unutar funkcije izmijenili smo vrijednost varijable **x**. Izvršimo li tu funkciju pozivanjem u naredbenom prozoru, prikazat će se rezultat:

```
>> area(x)
ans =
     2
```

Provjerimo li vrijednost varijable **x** u radnom prostoru:

```
>> x
x =
     3
```

Iako se vrijednost varijable **x** promijenila unutar funkcije, nije se promijenila vrijednost varijable **x** u radnom prostoru. To je stoga što su to dvije odvojene i različite varijable unatoč tomu što imaju isti naziv.

global

Varijabla se može učiniti dostupnom (vidljivom) i u funkciji i u radnom prostoru naredbom **global**. Ako se želi varijablu učiniti dostupnom i u radnom prostoru (skripti) i u funkciji, nije ju dovoljno definirati samo u funkciji. Preporučuje se globalne varijable pisati velikim slovima kako bi se razlikovale od lokalnih varijabli. Npr. neka preradimo funkciju **area** u oblik:

Varijable

```
function [w] = area(x);
global X
X = 2;
w = X;
```

Pridijeli li se vrijednost varijabli **x** u radnom prostoru, npr.:

```
>> X = 3
X =
     3
```

i zatim izvrši funkcija **area** pozivanjem u naredbenom prozoru, prikazat će rezultat:

```
>> area(1)
ans =
     2
```

Provjeri li se vrijednost varijable **x** u radnom prostoru:

```
>> X
X =
     3
```

Unatoč definiranju globalne varijable unutar funkcije, ona još uvijek nije dostupna (vidljiva) u radnom prostoru. Da bi bila vidljiva i u radnom prostoru i u funkciji, potrebno ju je definirati na oba mjesta. Dakle:

```
>> global X
```

Izvršimo li sad funkciju **area** pozivanjem u naredbenom prozoru, prikazat će rezultat:

```
>> area(1)
ans =
     2
```

Provjerimo li vrijednost varijable **x** u radnom prostoru:

```
>> X
X =
     2
```

Očigledno se ovaj put promjena vrijednosti varijable **x** unutar funkcije odrazila na vrijednost varijable **x** u radnom prostoru. Te su dvije varijable postale ista varijabla nakon što su definirane kao globalne na oba mjesta.

persistent

Nakon izvršenja funkcije sve varijable definirane unutar funkcije brišu se, pa ih je pri ponovnom pozivanju funkcije potrebno definirati. Ponekad je potrebno zadržati vrijednost određene unutarnje varijable funkcije. To je moguće pomoću naredbe **persistent**. Npr.:

```
function findfile(file)
persistent lastDir

if isempty(lastDir)
    prompt = 'Enter directory:  ';
else
    prompt = ['Enter directory[' lastDir ']:  '];
end
response = input(prompt, 's');

if ~isempty(response)
    dirName = response;
else
    dirName = lastDir;
end

dir(strcat(dirName, file))
lastDir = dirName;
```

Pri prvom izvršenju funkcije **findfile** varijabla **lastDir** je prazna i popunit će se tek nakon što korisnik unese njezinu vrijednost. Pri drugom izvršenju funkcije **findfile** korisniku će se ponuditi zadnja vrijednost varijable **lastDir** koja je, zbog toga što je varijabla **lastDir** definirana kao **persistent**, ostala sačuvana. Varijable definirane kao **persistent** ostaju sačuvane između poziva funkcije.

Pohranjena vrijednost varijable koja je definirana kao **persistent** može se izbrisati naredbom **clear filename**, gdje je **filename** ime M-datoteke u kojoj je pohranjena funkcija.

mlock, munlock, mislocked

Ako se želi osigurati da se varijabla neće zabunom izbrisati naredbom **clear filename**, može se M-datoteka zaključati naredbom **mlock** navedenom unutar M-datoteke. Varijable koje su definirane kao **persistent** unutar zaključane M-datoteke neće se izbrisati naredbom **clear**. Datoteku je moguće otključati naredbom **munlock(filename)**. Naredbom **mislocked(filename)** moguće je provjeriti je li M-datoteka **filename** zaključana.

4.2 Definiranje, dobava i brisanje varijabli radnog prostora

Varijable radnog prostora (engl. *base workspace*) mogu se definirati u naredbenom retku ili učitati iz postojeće datoteke.

save

Varijable radnog prostora mogu se pohraniti u datoteku naredbom:

```
>> save filename
```

Izvršenjem te naredbe u datoteku **filename.mat** pohranit će se definicije i vrijednosti varijabli radnog prostora (nastavak **mat** će Octave sam dodati datoteci). Ta je datoteka binarnog formata i nije čitljiva u programima za obradu teksta. Naredba **save** ima više oblika koji se mogu vidjeti pomoću naredbe **help save**. U tablici 4.1 prikazani su oblici naredbe **save**.

Tablica 4.1 Oblici naredbe **save**

save	Pohrana svih varijabli radnog prostora u datoteku s nazivom **Octave.mat**.
save filename **save('filename')**	Pohrana svih varijabli radnog prostora u datoteku s nazivom **filename.mat**.
save filename x y **save('filename','x','y')**	Pohrana varijabli **x** i **y** radnog prostora u datoteku s nazivom **filename.mat**.

csvwrite

Vrijednost varijable koja je matrica može se pohraniti i u tekstualnu datoteku naredbom **csvwrite**. Tom se naredbom elementi matrice pohranjuju u datoteku tako da su međusobno odvojeni zarezom (engl. *comma separated values*, *csv*). Npr.

```
>> m = [3,4,5; 6,7,8]
m =
     3     4     5
     6     7     8
>> csvwrite('matrixm.csv',m)
```

Stvorena je datoteka **matrixm.csv** koja u programu za obradu teksta izgleda ovako:

```
3,4,5
6,7,8
```

Svaki redak matrice pohranjen je kao posebni redak tekstualne datoteke. Preporuka je da se tekstualnim datotekama daje produžetak **csv** ili **dat** kako bi se razlikovale od binarnih datoteka s **mat** produžetkom.

Ako se elemente matrice u tekstualnoj datoteci umjesto zarezom želi odvojiti nekim drugim znakom, treba rabiti naredbu **dlmwrite** (sintaksu i primjenu vidjeti pomoću naredbe **help dlmwrite**).

load

Stanje radnog prostora iz datoteke **filename.mat** moguće je dobaviti u radni prostor naredbom:

```
>> load filename
```

Naredba **load** ima više oblika koji se mogu vidjeti pomoću naredbe **help load**. U tablici 4.2 prikazani su oblici naredbe **load**.

Tablica 4.2 Oblici naredbe **load**

load	Dobava svih varijabli iz datoteke s nazivom **Octave.mat** u radni prostor.
load filename **load('filename')**	Dobava svih varijabli iz datoteke s nazivom **filename.mat** u radni prostor.
load filename x y **load('filename','x','y')**	Dobava varijabli **x** i **y** iz datoteke s nazivom **filename.mat** u radni prostor.

csvread

Naredbom **csvread** moguće je u radni prostor učitati matricu koja je pohranjena u tekstualnoj datoteci u kojoj su elementi matrice odvojeni zarezom (engl. *comma separated values*, *csv*). Svaki redak matrice mora biti u novom redu tekstualne datoteke. Ako su, npr. podatci pohranjeni u ovom obliku u tekstualnoj datoteci **vectorx.csv**:

1,2,3

4,5,6,

može se naredbom:

```
>> x = csvread('vectorx')
x  =
     1     2     3
     4     5     6
```

učitati matrica **x** u radni prostor.

Ako su umjesto zareza u tekstualnoj datoteci elementi matrice odvojeni nekim drugim znakom, treba rabiti naredbu **dlmread** (sintaksu i primjenu vidjeti pomoću naredbe **help dlmread**).

Pomoću naredbi **File/Import Data…** na izborniku moguće je u Octaveov radni prostor dobaviti podatke iz različitih formata datoteka koji nisu ovdje spomenuti.

who, whos

Stanje varijabli u radnom prostoru može se saznati pomoću Octaveovih naredbi **who** i **whos**. Naredba **who** prikazuje samo popis varijabli (skraćeni prikaz), a naredba **whos** opsežniji prikaz varijabli radnog prostora (engl. *base workspace*). Npr.:

```
>> x = 3
x =
     3
>> y = 4
y =
     4
>> z = x*y
z =
    12

>> who
Your variables are:
x  y  z

>> whos
  Name      Size                     Bytes  Class
  x         1x1                          8  double array
  y         1x1                          8  double array
  z         1x1                          8  double array
```

Druga mogućnost uvida u radni prostor je otvaranjem prozora **Workspace** pomoću izbornika **Desktop/Workspace**.

Definirane varijable postoje u radnom prostoru sve dok ih se ne promijeni u radnom prostoru, skripti ili funkciji (ako je riječ o globalnim varijablama). Ponekad je potrebno obrisati varijable u radnom prostoru. Obrisati ne znači pripisati im vrijednost nula, već znači da one više ne postoje i nisu dostupne ni vidljive u radnom prostoru.

clear

Sve varijable moguće je izbrisati iz radnog prostora naredbom **clear**. Treba uočiti razliku između naredbi **clc** (brisanje radne plohe u naredbenom prozoru) i naredbe **clear** kojom se brišu varijable u radnom prostoru (u memoriji).

Naredbom **clc** briše se samo radna ploha naredbenog prozora, dok varijable u radnom prostoru (engl. *base workspace*) ostaju netaknute. Izvršenjem naredbe **clc** u naredbenom prozoru briše se radna ploha, ali time nisu izbrisane varijable u radnom prostoru (memoriji). Npr.:

```
>> x = 2
x =
     2
>> y = 3
y =
     3
>> clc
```

Nakon izvršenja naredbe **clc**, briše se radna ploha. Provjeri li se nakon toga stanje varijabli u radnom prostoru:

```
>> x
x =
     2
>> y
y =
     3
```

očigledno je da je vrijednost varijabli ostala nepromijenjena nakon korištenja naredbe **clc**.

Naredbom **clear** brišu se varijable iz radnog prostora te više ne postoje. Tom se naredbom ne briše radna ploha u naredbenom prozoru.

```
>> x = 2
x =
     2
>> y = 3
y =
     3
>> clear
>> x
error: 'x' undefined near line 1, column 1
>> y
error: 'y' undefined near line 1, column 1
```

Naredbom **clear** izbrisane su sve varijable iz radnog prostora pa program javlja da ne postoje varijable **x** i **y**.

Ako se želi iz radnog prostora izbrisati samo neke varijable, potrebno je iza naredbe **clear** navesti imena varijabli koje se žele izbrisati, npr.:

```
>> clear x
```

čime će se iz radnog prostora izbrisati samo varijabla **x**.

U tablici 4.3 su prikazani češće rabljeni oblici naredbe **clear**. Za podrobniji opis svih oblika naredbe **clear** pogledati Octaveovu dokumentaciju ili **help**.

Tablica 4.3 Oblici naredbe **`clear`**

Naredba	Rezultat
`clear all`	Briše sve varijable, funkcije i mex datoteke iz memorije (radni prostor ostaje prazan). Briše prekidne točke u M-datotekama.
`clear functions`	Briše sve trenutačno prevedene (engl. *compiled*) M-datoteke i mex datoteke iz memorije. Briše prekidne točke u M-datotekama.
`clear global`	Briše sve globalne varijable iz radnog prostora.
`clear variables`	Briše sve varijable iz radnog prostora.

workspace

Naredba **`workspace`** otvara prozor **Workspace** za pregled stanja radnog prostora (engl. *workspace browser*). U tom je prozoru moguće vidjeti stanje varijabli u radnom prostoru. Pomoću odgovarajućih izbornika moguće je u tom prozoru obaviti radnje koje odgovaraju naredbama: **`clear`**, **`load`**, **`open`**, i **`save`** napisanim u naredbenom prozoru. U načelu je lakše i jednostavnije rukovati radnim prostorom u prozoru **Workspace** nego pišući naredbe.

openvar

Naredba **`openvar('varnm')`** otvara prozor **Array Editor** za uređivanje nizova gdje se u grafičkom okolišu mogu uređivati varijable radnog prostora. **`varnm`** je naziv varijable koja mora biti tipa brojčanog niza (engl. *numeric array*), znakovnog niza (engl. *string*) ili niz znakovnih ćelija (engl. *cell array of strings*).

Ako je riječ o nizovima s više elemenata, pregled i uređivanje varijabli pomoću prozora za uređivanje nizova pregledniji su i lakši od uređivanja u naredbenom prozoru.

5. VEKTORI I MATRICE

Najčešći zapis podataka u Octaveu je skalar, vektor i matrica. Skalar je u Octaveu definiran kao matrica dimenzija 1*1, a vektor kao matrica dimenzija m*1 (stupčani vektori) ili 1*n (redni vektor).

Primjer 5.1: Definirane su varijable **x**, **y** i **z** kao skalari, odnosno matrice dimenzija 1*1.

```
>> x = 2
x =  2
>> y = 3
y =  3
>> z = x + y
z =  5
>>
```

Vektor se u Octaveu definira kao uređeni niz brojeva koji se nalaze u uglatim zagradama. Vektor može biti redni ili stupčani. Brojevi u vektoru mogu se odvajati razmakom ili zarezom te na taj način nastaje redni vektor, a ako se brojevi u vektoru odvajaju točka-zarezom u tom slučaju nastaje stupčani vektor.

Primjer 5.2: Stvoren je redni vektor **x** dimenzija 1*3, redni vektor **y** dimenzija 1*4, redni vektor **w** dimenzija 1*4 i stupčani vektor **z** dimenzija 3*1.

```
>> x = [1 5 8]
x =
     1     5     8
>> y = [8 9 -3 2]
y =
     8     9    -3     2
>> w = [5, 4, 7, 2]
w =
     5     4     7     2
>> z = [3; 8; 5]
z =
     3
     8
     5
>>
```

Ponekad je potrebno stvoriti elemente vektora od neke početne do neke krajnje vrijednosti s određenim korakom. To je moguće napraviti na sljedeći način:

x = [xmin:xstep:xmax] – gdje je **xmin** početna vrijednost elementa vektora, **xstep** korak povećavanja vrijednosti elemenata vektora, a **xmax** je završna vrijednost elementa vektora.

Potrebno je paziti da **xmin** bude manji od **xmax**, a također i da korak **xstep** ne bude veći od raspona između **xmin** i **xmax**. Ako je **xmin** veći od **xmax**, korak **xstep** mora biti negativan.

Primjer 5.3: Definiran je redni vektor **x** od 1 do 10 s korakom 2, redni vektor **y** od -8 do -2 s korakom 3, redni vektor **w** od -8 do 10 s korakom 4, redni vektor **z** od 5 do 50 s korakom 5, redni vektor **p** od 2 do 3 s korakom 0,1 te redni vektor **q** od 10 do 8 s korakom -0,2.

```
>> x = [1:2:10]
x =
     1.00     3.00     5.00     7.00     9.00
>> y = [-8:3:-2]
y =
    -8.00    -5.00    -2.00
>> w = [-8:4:10]
w =
    -8.00    -4.00     0.00     4.00     8.00
>> z = [5:5:50]
z =
    5.00   10.00   15.00   20.00   25.00   30.00   35.00   40.00   45.00
50.00
>> p = [2:0.1:3]
p =
    2.00   2.10   2.20   2.30   2.40   2.50   2.60   2.70   2.80   2.90
3.00
>> q = [10:-0.2:8]
q =
   10.00   9.80   9.60   9.40   9.20   9.00   8.80   8.60   8.40   8.20
8.00
>>
```

Matrica je uređena pravokutna ili kvadratna tablica brojeva. Matrice se sastoje od redaka i stupaca uređenih brojeva. Matrice u Octaveu stvaraju se tako da se elementi redaka odvajaju razmakom ili zarezom, a svaki redak je odvojen točka-zarezom.

Primjer 5.4: Stvorena je matrica **x** dimenzija 2*2, matrica **y** dimenzija 3*3 te matrica **z** dimenzija 4*2.

```
>> x = [2 5; 9 7]
x =
     2     5
     9     7
>> y = [1 2 3; -5 -4 -3; 5 5 5]
y =
     1     2     3
    -5    -4    -3
     5     5     5
>> z = [2, 4; 3, 6; 4, 8; 5, 10]
z =
     2     4
     3     6
     4     8
     5    10
>>
```

5.1 Adresiranje elemenata vektora i matrice

Često je potrebno obaviti neku operaciju, naredbu ili funkciju nad nekim elementom ili elementima vektora ili matrice, ali ne nad svim. Elementi vektora ili matrice adresiraju se slično kao što se to radi u matematici, pomoću indeksa elemenata. U Octaveu se navodi ime matrice ili vektora te redni broj elementa u okruglim zagradama (kod vektora) ili broj retka i stupca odvojen zarezom u okruglim zagradama (kod matrica), npr. **x(2)**, **x(4)**, **y(2,3)**, **y(5,8)** itd.

Primjer 5.5: Definiran je redni vektor **x** dimenzija 1*6. Zatim je adresiran drugi element vektora **x** koji je jednak 10, peti element vektora **x** koji je jednak 2, te šesti element vektora **x** koji je jednak -7.

```
>> x = [5 10 -4 3 2 -7]
x =
     5    10    -4     3     2    -7
>> x(2)
ans =  10
>> x(5)
ans =   2
>> x(6)
ans =  -7
>>
```

Primjer 5.6: Definirana je matrica **y** dimenzija 3*4. Zatim je adresiran element matrice **y** u drugom retku i trećem stupcu koji je jednak -3, nakon toga element matrice **y** u trećem retku i prvom stupcu koji je jednak 5, te element matrice **y** u prvom retku i trećem stupcu koji je jednak 3.

```
>> y = [1 2 3 4; -5 -4 -3 9; 5 8 7 2]
y =
     1     2     3     4
    -5    -4    -3     9
     5     8     7     2
>> y(2,3)
ans = -3
>> y(3,1)
ans =  5
>> y(1,3)
ans =  3
>>
```

5.2 Adresiranje retka ili stupca matrice

Ponekad je potrebno obaviti operaciju nad elementima nekog stupca ili retka matrice. Za to se koristi operator **:**, npr.:

x(i,:) – u ovom slučaju adresiraju se svi elementi i-tog retka matrice **x**

y(:,j) – u ovom slučaju adresiraju se svi elementi j-tog stupca matrice **y**.

Moguće je navesti i raspon redaka, odnosno stupaca:

x(2:4,:) – u ovom slučaju adresiraju se svi elementi drugog, trećeg i četvrtog retka matrice **x**

y(:,5:8) – u ovom slučaju adresiraju se svi elementi petog, šestog, sedmog i osmog stupca matrice **y**

w(1:3,4:6) – u ovom slučaju adresiraju se elementi četvrtog, petog i šestog stupca koji se nalaze u prvom, drugom i trećem retku matrice **w**

z(2,3:8) – u ovom slučaju adresiraju se elementi od trećeg do osmog stupca koji se nalaze u drugom retku matrice **z**.

Primjer 5.7: Definirana je matrica **x** dimenzija 8*5 koja sadrži slučajne cijele brojeve u rasponu [10,15]. Zatim su adresirani svi elementi drugog retka matrice **x**, nakon toga svi elementi trećeg stupca matrice **x**, zatim svi elementi od 5 do 8 retka matrice **x**, te svi elementi od 2 do 4 stupca matrice **x**.

```
>> x = randi([10 15],8,5)
x =
   11   15   10   13   14
   14   13   14   11   14
   14   13   12   14   10
   13   10   13   10   11
   13   12   14   11   13
```

```
    15    12    14    14    15
    12    14    10    13    11
    10    13    11    15    14
>> x(2,:)
ans =
    14    13    14    11    14
>> x(:,3)
ans =
    10
    14
    12
    13
    14
    14
    10
    11
>> x(5:8,:)
ans =
    13    12    14    11    13
    15    12    14    14    15
    12    14    10    13    11
    10    13    11    15    14
>> x(:,2:4)
ans =
    15    10    13
    13    14    11
    13    12    14
    10    13    10
    12    14    11
    12    14    14
    14    10    13
    13    11    15
>>
```

Primjer 5.8: Definirana je matrica **x** dimenzija 8*5 koja sadrži slučajne cijele brojeve u rasponu [10,15]. Zatim su adresirani elementi koji se nalaze u drugom retku trećeg i četvrtog stupca matrice **x**, nakon toga su adresirani elementi koji se nalaze u trećem, četvrtom i petom retku trećeg stupca matrice **x**, nakon toga su adresirani elementi koji se nalaze u petom, šestom, sedmom i osmom retku prvog, drugog i trećeg stupca matrice **x**, te na kraju adresirani su elementi koji se nalaze u četvrtom, petom i šestom retku drugog, trećeg i četvrtog stupca matrice **x**.

```
>> x = randi([10 15],8,5)
x =
    13    12    13    12    11
    12    14    13    14    11
    12    13    11    11    12
    14    11    10    11    10
    14    14    11    10    12
    15    11    14    14    10
    12    14    13    14    10
    11    11    12    13    15
>> x(2,3:4)
ans =
    13    14
>> x(3:5,3)
ans =
    11
    10
    11
>> x(5:8,1:3)
ans =
    14    14    11
    15    11    14
    12    14    13
    11    11    12
>> x(4:6,2:4)
ans =
```

```
    11    10    11
    14    11    10
    11    14    14
>>
```

5.3 Promjena elemenata vektora i matrice

Ponekad je potrebno izmijeniti određeni element postojeće matrice ili vektora. To se radi tako da se adresira taj element i dodijeli mu se nova vrijednost. Moguće je odjednom promijeniti i više elemenata postojeće matrice ili vektora koristeći operator `:`.

Primjer 5.9: Definiran je redni vektor **x** dimenzija 1*8. Promijenjena je vrijednost petog elementa vektora **x** iz 0 u 1. Zatim je definiran redni vektor **y** dimenzija 1*3. Promijenjena je vrijednost trećeg elementa vektora **y** iz 5 u 3.

```
>> x = [2 3 8 10 0 5 7 6]
x =
     2     3     8    10     0     5     7     6
>> x(5) = 1
x =
     2     3     8    10     1     5     7     6
>> y = [1 2 5]
y =
     1     2     5
>> y(3) = 3
y =
     1     2     3
>>
```

Primjer 5.10: Definiran je redni vektor **x** dimenzija 1*8. Promijenjene su vrijednosti šestog, sedmog i osmog elementa vektora **x** čije su vrijednosti bile 5, 7 i 6 u 0. Zatim je definiran redni vektor **y** dimenzija 1*3. Promijenjene su vrijednosti drugog i trećeg elementa vektora **y** čije su vrijednosti bile 2 i 5 u 1.

```
>> x = [2 3 8 10 0 5 7 6]
x =
     2     3     8    10     0     5     7     6
>> x(6:8) = 0
x =
     2     3     8    10     0     0     0     0
>> y = [1 2 5]
y =
     1     2     5
>> y(2:3) = 1
y =
     1     1     1
>>
```

Primjer 5.11: Definirana je matrica **x** dimenzija 3*3. Promijenjena je vrijednost elementa prvog retka i drugog stupca matrice **x** iz 3 u -3. Zatim je promijenjena vrijednost elementa trećeg retka i trećeg stupca matrice **x** iz 1 u 9. Na kraju su promijenjeni svi elementi trećeg retka matrice **x** iz 7, 6, 9 u 1, 2, 3.

```
>> x = [2 3 8; 10 0 5; 7 6 1]
x =
     2     3     8
    10     0     5
     7     6     1
```

```
>> x(1,2) = -3
x =
     2    -3     8
    10     0     5
     7     6     1
>> x(3,3) = 9
x =
     2    -3     8
    10     0     5
     7     6     9
>> x(3,:) = [1 2 3]
x =
     2    -3     8
    10     0     5
     1     2     3
>>
```

5.4 Dodavanje elemenata vektoru i matrici

Ponekad je potrebno dodati novi element ili elemente vektoru ili matrici. Na taj se način povećava dimenzija vektora ili matrice. Ako se dodaje element koji nije sljedeći po redu (u smislu indeksa ili adresiranja), svi elementi do tog novog elementa imaju vrijednost nula.

Primjer 5.12: Definiran je redni vektor **x** dimenzija 1*3. Dodan je četvrti element vektoru **x** čija je vrijednost 8. Zatim je definiran redni vektor **y** dimenzija 1*3. Dodan je šesti element vektoru y čija je vrijednost 10. Budući da je vektor **y** prije bio dimenzija 1*3, a vektoru **y** dodan je šesti element, Octave je automatski dodao četvrti i peti element vektoru **y** te im dodijelio vrijednost 0. Zatim je definiran redni vektor **z** dimenzija 1*3. Dodani su od osmog do dvanaestog elementa vektoru **z** čije su vrijednosti 15, 14, 13, 12 i 11. Budući da je vektor **z** prije dodavanja elemenata bio dimenzija 1*3, a vektoru **z** su dodani od osmog do dvanaestog elementa, Octave je automatski dodao četvrti, peti, šesti i sedmi element vektoru **z** te im dodijelio vrijednost 0.

```
>> x = [7 -5 3]
x =
     7    -5     3
>> x(4) = 8
x =
     7    -5     3     8
>> y = [1 2 3]
y =
     1     2     3
>> y(6) = 10
y =
     1     2     3     0     0    10
>> z = [1 2 3]
z =
     1     2     3
>> z(8:12) = [15:-1:11]
z =
     1     2     3     0     0     0     0    15    14    13    12    11
>>
```

Primjer 5.13: Definirana je matrica **x** dimenzija 2*3. Dodan je element prvog retka i četvrtog stupca matrici **x** koji je jednak 9. Budući da je matrica pravokutna ili kvadratna struktura, Octave je automatski dodao matrici **x** element drugog retka i četvrtog stupca i dodijelio vrijednost 0. Definirana je matrica **y** dimenzija 2*3. Dodan je element trećeg retka i četvrtog stupca matrici **y** koji je jednak 10. Octave je automatski dodao sve druge potrebne elemente matrici **y** i dodijelio vrijednost 0. Definirana je matrica **z** dimenzija 2*3. Dodan je treći redak matrici **z** čiji su svi elementi jednaki 10. Zatim je dodan šesti stupac matrici **z** čiji su svi elementi jednaki 8. Octave je automatski dodao četvrti i peti stupac matrici **z** i dodijelio vrijednost 0.

```
>> x = [7 -5 3; 1 2 8]
x =
     7    -5     3
     1     2     8
>> x(1,4) = 9
x =
     7    -5     3     9
     1     2     8     0
>> y = [1 2 3; 9 8 7]
y =
     1     2     3
     9     8     7
>> y(3,4) = 10
y =
     1     2     3     0
     9     8     7     0
     0     0     0    10
>> z = [5 4 3; 1 5 2]
z =
     5     4     3
     1     5     2
>> z(3,:) = 10
z =
     5     4     3
     1     5     2
    10    10    10
>> z(:,6) = 8
z =
     5     4     3     0     0     8
     1     5     2     0     0     8
    10    10    10     0     0     8
>>
```

5.5 Brisanje elemenata vektora i matrice

Ponekad je potrebno neke elemente vektora, odnosno neke retke i/ili stupce matrice obrisati. To se radi tako da se tim elementima dodijeli prazna vrijednost (ništa), za što se u Octaveu koristi `[]`.

Primjer 5.14: Definiran je redni vektor **x** dimenzija 1*9 te je nakon toga obrisan peti element čija je vrijednost -5. Nakon toga redni vektor **x** ima dimenziju 1*8. Definiran je i redni vektor **y** dimenzija 1*9 te su nakon toga obrisani treći i četvrti element čije su vrijednosti 3 i 9. Nakon toga redni vektor **y** ima dimenziju 1*7.

```
>> x = [7 -5 3 4 -5 -7 8 1 2]
x =
     7    -5     3     4    -5    -7     8     1     2
>> x(5) = []
x =
     7    -5     3     4    -7     8     1     2
>> y = [1 2 3 9 8 7 5 4 3]
y =
     1     2     3     9     8     7     5     4     3
>> y(3:4) = []
y =
     1     2     8     7     5     4     3
>>
```

Primjer 5.15: Definirana je matrica **x** dimenzija 2*3. Nakon toga obrisan je treći stupac matrice **x** čime matrica **x** dobiva nove dimenzije 2*2. Definirana je i matrica **y** dimenzija 2*3 te je zatim obrisan prvi redak matrice **y**, čime matrica **y** dobiva nove dimenzije 1*3 (redni vektor). Definirana je i matrica **z** dimenzija 3*3 te je zatim obrisan drugi stupac matrice **z**, čime matrica **z** dobiva nove dimenzije 3*2.

```
>> x = [7 -5 3; 1 2 8]
x =
     7    -5     3
     1     2     8
>> x(:,3) = []
x =
     7    -5
     1     2
>> y = [1 2 3; 9 8 7]
y =
     1     2     3
     9     8     7
>> y(1,:) = []
y =
     9     8     7
>> z = [5 4 3; 1 5 2; 10 8 2]
z =
     5     4     3
     1     5     2
    10     8     2
>> z(:,2) = []
z =
     5     3
     1     2
    10     2
>>
```

linspace

Vektore je moguće definirati i pomoću funkcije **linspace**. Oblik funkcije je:

linspace(xmin,xmax,n) – gdje je **xmin** početna vrijednost elemenata vektora, **xmax** krajnja vrijednost elemenata vektora, a **n** je broj elemenata vektora.

Funkcija **linspace** stvara elemente vektora prema linearnoj podjeli.

Primjer 5.16: Stvoren je redni vektor **x** dimenzija 1*5 u rasponu [-10,10] prema linearnoj podjeli, redni vektor **y** dimenzija 1*8 u rasponu [0,1] prema linearnoj podjeli te redni vektor **z** dimenzija 1*10 u rasponu [-10,-5] prema linearnoj podjeli.

```
>> x = linspace(-10,10,5)
x =
   -10.00    -5.00     0.00     5.00    10.00
>> y = linspace(0,1,8)
y =
     0.00     0.14     0.29     0.43     0.57     0.71     0.86     1.00
>> z = linspace(-10,-5,10)
z =
  -10.00   -9.44   -8.89   -8.33   -7.78   -7.22   -6.67   -6.11   -5.56   -5.00
>>
```

logspace

Ponekad je potrebno definirati vektore čija je razdioba elemenata prema logaritamskoj podjeli. To se radi pomoću funkcije **logspace**. Oblik funkcije je:

logspace(p1,p2,n) – gdje je **p1** eksponent baze 10 za početnu vrijednost elemenata vektora, **p2** je eksponent baze 10 za krajnju vrijednost elemenata vektora, a **n** je broj elemenata vektora.

Primjer 5.17: Stvoren je redni vektor **x** dimenzija 1*5 u rasponu od 10 do 100 prema logaritamskoj podjeli, redni vektor **y** dimenzija 1*8 u rasponu od 0,1 do 10 prema logaritamskoj podjeli i redni vektor **z** dimenzija 1*4 u rasponu od 100 do 10000 prema logaritamskoj podjeli.

```
>> x = logspace(1,2,5)
x =
    10.00    17.78    31.62    56.23   100.00
>> y = logspace(-1,1,8)
y =
    0.10    0.19    0.37    0.72    1.39    2.68    5.18   10.00
>> z = logspace(2,4,4)
z =
     100.00     464.16    2154.43   10000.00
>>
```

length

Pomoću funkcije **length** moguće je saznati duljinu vektora, odnosno broj elemenata vektora. Oblik funkcije je:

length(x) – gdje je **x** vektor kojem se želi odrediti broj elemenata.

Primjer 5.18: Definiran je redni vektor **x**. Pomoću funkcije **length** dobiva se podatak o duljini vektora koja iznosi 10. Definiran je redni vektor **y** te se pomoću funkcije **length** dobiva podatak o duljini vektora koja iznosi 4. Definiran je redni vektor **z** te se pomoću funkcije **length** dobiva podatak o duljini vektora koja iznosi 8.

```
>> x = [1:2:20]
x =
    1.00  3.00  5.00  7.00  9.00  11.00  13.00  15.00  17.00  19.00
>> length(x)
ans =    10.00
>> y = [-5:3:5]
y =
   -5.00  -2.00   1.00   4.00
>> length(y)
ans =    4.00
>> z = linspace(2,4,8)
z =
    2.00   2.29   2.57   2.86   3.14   3.43   3.71   4.00
>> length(z)
ans =    8.00
>>
```

Ako se naredba **length** primijeni na matricu dobit će se podatak o većoj dimenziji matrice. Ako je matrica dimenzija **m*n** i **m>n** naredba **length** će prikazati vrijednost **m**, a ako je matrica dimenzija **m*n** i **m<n** naredba **length** će prikazati vrijednost **n**.

size

Pomoću funkcije **size** može se saznati podatak o dimenziji matrice. Oblik funkcije je:

size(x) – gdje je **x** matrica kojoj se želi odrediti dimenzija,

[m n] = size(x) – gdje je **x** matrica kojoj se želi odrediti dimenzija, **m** je varijabla u koju se pohranjuje broj redaka, a **n** je varijabla u koju se pohranjuje broj stupaca matrice **x**.

Primjer 5.19: Definirana je matrica **x**. Pomoću funkcije **size** dobiva se podatak o dimenziji matrice koja je 2*3. Definirana je matrica **y** te se pomoću funkcije **size** dobiva podatak o dimenziji matrice koja je 3*2. Definirana je matrica **z** te se pomoću funkcije **size** dobiva podatak o dimenziji matrice koja je 3*4.

```
>> x = [2 3 4; 5 6 7]
x =
   2   3   4
```

```
    5    6    7
>> size(x)
m =    2
n =    3
>> y = [2 4; 1 3; -1 1]
y =
    2    4
    1    3
   -1    1
>> [m n] = size(y)
m =    3
n =    2
>> z = randi([1 10],3,4)
z =
    9    7    6    9
    9    5    9    1
    5    3    2    9
>> [m n] = size(z)
m =    3
n =    4
>>
```

numel

Pomoću funkcije **numel** moguće je saznati ukupan broj elemenata matrice. Oblik funkcije je:

numel(x) – gdje je **x** matrica kojoj se želi odrediti broj elemenata.

Primjer 5.20: Definirane su matrice **x**, **y** i **z**. Pomoću funkcije **numel** dobiva se podatak o broju elemenata matrica.

```
>> x = [2 3 4; 5 6 7]
x =
    2    3    4
    5    6    7
>> numel(x)
ans =  6
>> y = [2 4; 1 3; -1 1]
y =
    2    4
    1    3
   -1    1
>> numel(y)
ans =  6
>> z = randi([1 10],3,4)
z =
    4    1    1    3
    1    6    1    6
    2    3    2    2
>> numel(z)
ans =  12
>>
```

5.6 Funkcije za definiranje matrica i vektora

U Octaveu postoje funkcije za definiranje posebnih matrica ili vektora.

zeros

Funkcija **zeros** u Octaveu definira matricu ili vektor (ili općenito višedimenzionalno polje) čiji su svi elementi jednaki nuli. Takva je, npr. matrica:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

Funkcija **zeros** ima nekoliko različitih sintaksi:

zeros(m) ili **zeros(m,m)**

zeros(m,n) ili **zeros([m n])**

zeros(d1,d2,...,dn) ili **zeros([d1 d2 ... dn])**

Napomena: Oblik naredbe **zeros(d1,d2,...,dn)** i **zeros([d1 d2 ... dn])** daje identičan rezultat samo je sintaksa različita.

Primjer 5.21: Različite sintakse funkcije **zeros**. Matrica **x** je dimenzija 2*4, matrica **y** je dimenzija 3*3, a matrica **z** je dimenzija 2*2.

```
>> x = zeros(2,4)
x =
     0     0     0     0
     0     0     0     0
>> y = zeros(3,3)
y =
     0     0     0
     0     0     0
     0     0     0
>> z = zeros(2)
z =
     0     0
     0     0
>>
```

ones

Funkcija **ones** u Octaveu definira matricu ili vektor (općenito višedimenzionalno polje) čiji su svi elementi jednaki jedan. Takva je, npr. matrica:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

Funkcija **ones** ima nekoliko različitih sintaksi:

ones(m) ili **ones(m,m)**

ones(m,n) ili **ones([m n])**

ones(d1,d2,...,dn) ili **ones([d1 d2 ... dn])**

Napomena: Oblik naredbe **ones(d1,d2,...,dn)** i **ones([d1 d2 ... dn])** daje identičan rezultat samo je sintaksa različita.

Primjer 5.22: Različite sintakse funkcije **ones**. Matrica **x** je dimenzija 2*4, matrica **y** je dimenzija 3*3, matrica **z** je dimenzija 2*2, a matrica **w** je dimenzija 3*4. U zadnjem primjeru je funkcija **ones** pomnožena sa skalarom (u ovom slučaju brojem 5) tako da je dobivena matrica čiji su svi elementi jednaki tom skalaru.

```
>> x = ones(2,4)
x =
```

```
     1     1     1     1
     1     1     1     1
>> y = ones(3,3)
y =
     1     1     1
     1     1     1
     1     1     1
>> z = ones(2)
z =
     1     1
     1     1
>> w = 5*ones(3,4)
w =
     5     5     5     5
     5     5     5     5
     5     5     5     5
>>
```

eye

Funkcija **eye** u Octaveu definira jediničnu matricu, odnosno matricu koja na glavnoj dijagonali ima jedinice, a svi ostali elementi matrice jednaki su nuli. Takva je, npr. matrica:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Uvjet je da je m=n, odnosno da je matrica kvadratna. U Octaveu funkcija **eye** stvara pseudo jediničnu matricu ako vrijedi m≠n. Funkcija **eye** ima nekoliko različitih sintaksi:

eye(m) ili **eye(m,m)**

eye(m,n) ili **eye([m n])**

Primjer 5.23: Različite sintakse funkcije **eye**. Matrica **x** je dimenzija 3*3, matrica **y** je dimenzija 4*4, matrica **z** je dimenzija 2*2, matrica **w** je dimenzija 3*5, a matrica **q** je dimenzija 3*3. U ovom je primjeru matrica **w** dimenzija 3*5 pseudo jedinična matrica zbog toga što je m≠n. Matrica **q** dimenzija 3*3 pomnožena je sa skalarom (u ovom slučaju brojem 8) tako da su elementi na glavnoj dijagonali jednaki 8.

```
>> x = eye(3,3)
x =
Diagonal Matrix
   1   0   0
   0   1   0
   0   0   1
>> y = eye(4,4)
y =
Diagonal Matrix
   1   0   0   0
   0   1   0   0
   0   0   1   0
   0   0   0   1
>> z = eye(2)
z =
Diagonal Matrix
   1   0
   0   1
>> w = eye(3,5)
w =
Diagonal Matrix
   1   0   0   0   0
   0   1   0   0   0
   0   0   1   0   0
>> q = 8*eye(3,3)
```

```
q =
Diagonal Matrix
   8   0   0
   0   8   0
   0   0   8
>>
```

rand

Funkcija **rand** u Octaveu definira matricu ili vektor (općenito višedimenzionalno polje) čiji su elementi slučajni realni brojevi iz jednolike razdiobe u rasponu [0,1]. Funkcija **rand** ima nekoliko različitih sintaksi:

rand(m) ili **rand(m,m)**

rand(m,n) ili **rand([m n])**

rand(d1,d2,...,dn) ili **rand([d1 d2 ... dn])**

Napomena: Oblik naredbe **rand(d1,d2,...,dn)** i **rand([d1 d2 ... dn])** daje identičan rezultat samo je sintaksa različita.

Primjer 5.24: Različite sintakse funkcije **rand**. Matrica **x** je dimenzija 2*4, matrica **y** je dimenzija 3*3, matrica **z** je dimenzija 2*2, a matrica **w** je dimenzija 4*4. U ovom je primjeru matrica **w** dimenzija 4*4 pomnožena sa skalarom (u ovom slučaju brojem 10) tako da su stvoreni slučajni realni brojevi iz jednolike razdiobe u rasponu [0,1] još pomnoženi sa skalarom 10.

```
>> x = rand(2,4)
x =
   0.96   0.06   0.46   0.97
   0.12   0.94   0.42   0.02
>> y = rand(3,3)
y =
   0.77   0.81   0.74
   0.83   0.36   0.89
   0.08   0.04   0.62
>> z = rand(2)
z =
   0.73   0.29
   0.98   0.30
>> w = 10*rand(4,4)
w =
   8.27   7.25   3.44   2.87
   7.39   0.89   4.13   5.23
   7.31   5.62   8.97   7.70
   7.74   4.31   0.22   3.04
>>
```

randi

Funkcija **randi** u Octaveu definira vektor, matricu ili, općenito, višedimenzionalno polje čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [a,b], gdje su $a \in Z$ i $b \in Z$. Funkcija **randi** ima nekoliko različitih sintaksi:

randi(max)

randi(max,n)

randi(max,m,n)

randi([min max])

randi([min max],n)

randi([min max],m,n)

Napomena: naredba **randi(max)** stvara slučajne cijele brojeve iz jednolike razdiobe u rasponu **[1, max]**.

Primjer 5.25: Različite sintakse funkcije **randi**. Matrica **x** dimenzija 2*4 sadrži slučajne cijele brojeve iz jednolike razdiobe u rasponu [-1,1], matrica **y** dimenzija 4*4 sadrži slučajne cijele brojeve iz jednolike razdiobe u rasponu [-5,5].

```
>> x = randi([-1 1],2,4)
x =
   0  -1   1   0
   1  -1  -1   0
>> y = randi([-5 5],4,4)
y =
  -2   5   5   2
   1   0   0   0
  -1   1   1   4
   1  -4  -5  -1
>>
```

randn

Funkcija **randn** u Octaveu definira vektor, matricu ili, općenito, višedimenzionalno polje čiji su elementi slučajni realni brojevi iz normalne (Gaussove) razdiobe sa srednjom vrijednošću 0 i standardnom devijacijom 1. Funkcija **randn** ima nekoliko različitih sintaksi:

randn(m) ili **randn(m,m)**

randn(m,n) ili **randn([m n])**

randn(d1,d2,...,dn) ili **randn([d1 d2 ... dn])**

Napomena: Oblik naredbe **randn(d1,d2,...,dn)** i **randn([d1 d2 ... dn])** daje identičan rezultat samo je sintaksa različita.

Primjer 5.26: Različite sintakse funkcije **randn**. Matrica **x** je dimenzija 2*4, matrica **y** je dimenzija 3*3, matrica **z** je dimenzija 2*2, a matrica **w** je dimenzija 4*4. U ovom je primjeru matrica **w** dimenzija 4*4 pomnožena sa skalarom (u ovom slučaju brojem 10) tako da su stvoreni slučajni realni brojevi iz normalne razdiobe još pomnoženi sa skalarom 10.

```
>> x = randn(2,4)
x =
   1.00  -0.09   2.02  -0.13
  -0.75   0.42  -0.84  -0.20
>> y = randn(3,3)
y =
  -0.16  -0.59  -0.10
  -0.52  -0.38   0.04
   0.63  -0.20   0.77
>> z = randn(2)
z =
   0.09  -0.74
   0.66  -0.66
>> w = 10*randn(4,4)
w =
    8.34   18.99   13.32    3.65
    1.12   -0.48    8.53  -24.96
    0.42   -1.37   12.26    1.63
    5.54   -2.45  -11.17  -16.11
>>
```

Pitanja za provjeru znanja:

1. Koja se funkcija koristi za određivanje duljine vektora (broja elemenata)?
2. Koja se funkcija koristi za određivanje veličine matrice?
3. Koja se funkcija koristi za određivanja broja elemenata matrice?

4. Što rade funkcije **`zeros`**, **`ones`** i **`eye`**, i koje su razlike među njima?
5. Što rade funkcije **`rand`**, **`randi`** i **`randn`**, i koje su razlike među njima?

6. OPERATORI I OPERACIJE

U matematici je operator funkcija koja djeluje na (ili mijenja) drugu funkciju ili operande.

6.1 Aritmetički operatori

Aritmetički su operatori funkcije koje djeluju na brojeve (matrice, vektore). Aritmetički operatori služe za obavljanje aritmetičkih operacija nad matricama, vektorima i skalarima. Popis operatora prema prioritetu izvođenja prikazan je u tablici 6.1.

Tablica 6.1 Prioritet izvođenja aritmetičkih operacija

Prioritet	Operator	Opis
1.	( )	Zagrade grupiraju izraz i pridjeljuju mu najveći prioritet
2.	'	Konjugiranje i transponiranje vektora ili matrice
	.'	Transponiranje vektora ili matrice
3.	^	Potenciranje vektora ili matrice
	.^	Potenciranje među elementima vektora ili matrice
4.	*	Množenje skalara, vektora ili matrica
	.*	Množenje među elementima vektora ili matrica
	/	Desno dijeljenje vektora ili matrica
	\	Lijevo dijeljenje vektora ili matrica
	./	Desno dijeljenje među elementima vektora ili matrica
	.\	Lijevo dijeljenje među elementima vektora ili matrica
5.	+	Zbrajanje skalara, vektora ili matrica
	-	Oduzimanje skalara, vektora ili matrica

Operacije unutar izraza izvršavaju se počevši od najvišeg prioriteta prema najnižem, a izrazi istog prioriteta izvršavaju se od lijeva prema desno. Radi čitljivosti i lakšeg razumijevanja programa preporučuje se što više rabiti zagrade.

6.2 Operator zbrajanja

Operator zbrajanja služi za zbrajanje matrica, vektora, matrica i skalara te vektora i skalara. Za zbrajanje se u Octaveu koristi operator **+**. Zbrajati se mogu međusobno redni vektori ili stupčani vektori. Neka postoje dva redna vektora $\vec{u}$ i $\vec{v}$ dimenzija 1*n:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \text{ i } \vec{v} = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

Zbrajanje tih dvaju vektora daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w} = \vec{u} + \vec{v} = \begin{bmatrix} u_1 + v_1 & u_2 + v_2 & \dots & u_n + v_n \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.1: Zbrajanje rednog vektora **x** dimenzija 1*3 s rednim vektorom **y** dimenzija 1*3 daje redni vektor **z** dimenzija 1*3.

```
>> x = [7 5 8]
x =
     7     5     8
>> y = [4 3 9]
y =
     4     3     9
>> z = x + y
z =
    11     8    17
>>
```

Primjer 6.2: Zbrajanje stupčanog vektora **x** dimenzija 3*1 sa stupčanim vektorom **y** dimenzija 3*1 daje stupčani vektor **z** dimenzija 3*1.

```
>> x = [6; 3; 9]
x =
     6
     3
     9
>> y = [7; 5; 1]
y =
     7
     5
     1
>> z = x + y
z =
    13
     8
    10
>>
```

Zbrajati se mogu i redni vektor sa skalarom i stupčani vektor sa skalarom. Neka je redni vektor $\vec{u}$ dimenzija 1*n:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}$$

Zbrajanje rednog vektora $\vec{u}$ dimenzija 1*n i skalara a daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w} = \vec{u} + a = \begin{bmatrix} u_1 + a & u_2 + a & \dots & u_n + a \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.3: Zbrajanje rednog vektora **x** dimenzija 1*3 sa skalarom **y** daje redni vektor **z** dimenzija 1*3.

```
>> x = [5 -3 6]
x =
     5    -3     6
>> y = 2
y =
     2
>> z = x + y
z =
     7    -1     8
>>
```

Primjer 6.4: Zbrajanje stupčanog vektora **x** dimenzija 4*1 sa skalarom **y** daje stupčani vektor **z** dimenzija 4*1.

```
>> x = [3; 4; 8; -5]
x =
      3
      4
      8
     -5
>> y = 3
y =  3
>> z = x + y
z =
      6
      7
     11
     -2
>>
```

Kako bi se vektori mogli zbrajati, moraju biti jednakih dimenzija (moraju imati jednak broj elemenata). Neka postoje redni vektor $\vec{u}$ dimenzija 1*m i redni vektor $\vec{v}$ dimenzija 1*n:

$\vec{u} = [u_1 \quad u_2 \quad \dots \quad u_m]$ i $\vec{v} = [v_1 \quad v_2 \quad \dots \quad v_n]$

Ako je m≠n, ta se dva vektora ne mogu zbrojiti.

Primjer 6.5: Zbrajanje rednog vektora **x** dimenzija 1*2 s rednim vektorom **y** dimenzija 1*4 nije moguće.

```
>> x = [3 7]
x =
      3      7
>> y = [6 3 8 5]
y =
      6      3      8      5
>> z = x + y
error: operator +: nonconformant arguments (op1 is 1x2, op2 is 1x4)
>>
```

Neka postoje dvije matrice $\underline{A}$ i $\underline{B}$ dimenzija m*n:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } \underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Zbrajanje tih dviju matrica daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{A} + \underline{B} = \begin{bmatrix} A_{11}+B_{11} & A_{12}+B_{12} & \dots & A_{1n}+B_{1n} \\ A_{21}+B_{21} & A_{22}+B_{22} & \dots & A_{2n}+B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}+B_{m1} & A_{m2}+B_{m2} & \dots & A_{mn}+B_{mn} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.6: Zbrajanje matrice **x** dimenzija 2*3 s matricom **y** dimenzija 2*3 daje matricu **z** dimenzija 2*3.

```
>> x = [3 5 2; 6 5 8]
x =
      3      5      2
      6      5      8
>> y = [2 8 9; 8 5 3]
```

```
y =
     2     8     9
     8     5     3
>> z = x + y
z =
     5    13    11
    14    10    11
>>
```

Primjer 6.7: Zbrajanje matrice **x** dimenzija 3*2 s matricom **y** dimenzija 3*2 daje matricu **z** dimenzija 3*2.

```
>> x = [7 5; 4 6; 2 9]
x =
     7     5
     4     6
     2     9
>> y = [8 7; 3 4; 2 5]
y =
     8     7
     3     4
     2     5
>> z = x + y
z =
    15    12
     7    10
     4    14
>>
```

U Octaveu se matrice mogu zbrajati i sa skalarom. Zbraja se svaki element matrice sa skalarom. Neka je matrica $\underline{B}$ dimenzija m*n:

$$\underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Zbrajanje matrice $\underline{B}$ dimenzija m*n i skalara a daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{B} + a = \begin{bmatrix} B_{11}+a & B_{12}+a & \dots & B_{1n}+a \\ B_{21}+a & B_{22}+a & \dots & B_{2n}+a \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1}+a & B_{m2}+a & \dots & B_{mn}+a \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.8: Zbrajanje matrice **x** dimenzija 2*3 sa skalarom **y** daje matricu **z** dimenzija 2*3.

```
>> x = [7 5 3; 4 6 1]
x =
     7     5     3
     4     6     1
>> y = 10
y =  10
>> z = x + y
z =
    17    15    13
    14    16    11
>>
```

Da bi se matrice mogle zbrajati, moraju biti jednakih dimenzija (moraju imati jednak broj redaka i jednak broj stupaca). Neka postoje matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija p*q:

$$\underline{A}=\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } \underline{B}=\begin{bmatrix} B_{11} & B_{12} & \dots & B_{1q} \\ B_{21} & B_{22} & \dots & B_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ B_{p1} & B_{p2} & \dots & B_{pq} \end{bmatrix}$$

Ako je m≠p ili n≠q, te se dvije matrice ne mogu zbrojiti.

Primjer 6.9: Zbrajanje matrice **x** dimenzija 2*3 s matricom **y** dimenzija 3*2 nije moguće.

```
>> x = [3 5 2; 6 5 8]
x =
     3     5     2
     6     5     8
>> y = [8 7; 3 4; 2 5]
y =
     8     7
     3     4
     2     5
>> z = x + y
error: operator +: nonconformant arguments (op1 is 2x3, op2 is 3x2)
>>
```

Za zbrajanje vektora i matrica vrijede zakoni komutacije i asocijacije:

$$\vec{u}+\vec{v}=\vec{v}+\vec{u}$$

$$\underline{A}+\underline{B}=\underline{B}+\underline{A}$$

$$\vec{u}+(\vec{v}+\vec{w})=(\vec{u}+\vec{v})+\vec{w}$$

$$\underline{A}+(\underline{B}+\underline{C})=(\underline{A}+\underline{B})+\underline{C}$$

6.3 Operator oduzimanja

Operator oduzimanja služi za oduzimanje matrica, vektora, matrica i skalara te vektora i skalara. Za oduzimanje se u Octaveu koristi operator **-**. Oduzimati se mogu redni vektori i stupčani vektori. Neka postoje dva redna vektora $\vec{u}$ i $\vec{v}$ dimenzija 1*n:

$$\vec{u}=\begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \text{ i } \vec{v}=\begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

Oduzimanje tih dvaju vektora daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w}=\vec{u}-\vec{v}=\begin{bmatrix} u_1-v_1 & u_2-v_2 & \dots & u_n-v_n \end{bmatrix}=\begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.10: Oduzimanje rednog vektora **x** dimenzija 1*3 od rednog vektora **y** dimenzija 1*3 daje redni vektor **z** dimenzija 1*3.

```
>> x = [4 3 9]
x =
     4     3     9
>> y = [7 5 8]
y =
     7     5     8
>> z = y - x
z =
     3     2    -1
```

```
>>
```

Primjer 6.11: Oduzimanje stupčanog vektora **x** dimenzija 3*1 od stupčanog vektora **y** dimenzija 3*1 daje stupčani vektor **z** dimenzija 3*1.

```
>> x = [7; 5; 1]
x =
     7
     5
     1
>> y = [6; 3; 9]
y =
     6
     3
     9
>> z = y - x
z =
    -1
    -2
     8
>>
```

Oduzimati se mogu i redni vektor sa skalarom i stupčani vektor sa skalarom. Neka je redni vektor $\vec{u}$ dimenzija 1*n:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}$$

Oduzimanje skalara a od rednog vektora $\vec{u}$ dimenzija 1*n daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w} = \vec{u} - a = \begin{bmatrix} u_1 - a & u_2 - a & \dots & u_n - a \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.12: Oduzimanje skalara **x** od rednog vektora **y** dimenzija 1*3 daje redni vektor **z** dimenzija 1*3.

```
>> x = 2
x =  2
>> y = [5 -3 6]
y =
     5    -3     6
>> z = y - x
z =
     3    -5     4
>>
```

Primjer 6.13: Oduzimanje skalara **x** od stupčanog vektora **y** dimenzija 4*1 daje stupčani vektor **z** dimenzija 4*1.

```
>> x = 3
x =  3
>> y = [3; 4; 8; -5]
y =
     3
     4
     8
    -5
>> z = y - x
z =
     0
     1
     5
```

```
     -8
>>
```

Da bi se vektori mogli oduzimati moraju biti jednakih dimenzija, tj. moraju imati jednak broj elemenata. Neka postoje redni vektor $\vec{u}$ dimenzija 1*m i redni vektor $\vec{v}$ dimenzija 1*n:

$\vec{u} = [u_1 \quad u_2 \quad \dots \quad u_m]$ i $\vec{v} = [v_1 \quad v_2 \quad \dots \quad v_n]$

Ako je m≠n, ta se dva vektora ne mogu oduzimati.

Primjer 6.14: Oduzimanje rednog vektora **y** dimenzija 1*4 od rednog vektora **x** dimenzija 1*2 nije moguće.

```
>> x = [3 7]
x =
     3     7
>> y = [6 3 8 5]
y =
     6     3     8     5
>> z = x - y
error: operator -: nonconformant arguments (op1 is 1x2, op2 is 1x4)
>>
```

Neka postoje dvije matrice $\underline{A}$ i $\underline{B}$ dimenzija m*n:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } \underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Oduzimanje matrice $\underline{B}$ od matrice $\underline{A}$ daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{A} - \underline{B} = \begin{bmatrix} A_{11} - B_{11} & A_{12} - B_{12} & \dots & A_{1n} - B_{1n} \\ A_{21} - B_{21} & A_{22} - B_{22} & \dots & A_{2n} - B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} - B_{m1} & A_{m2} - B_{m2} & \dots & A_{mn} - B_{mn} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.15: Oduzimanje matrice **y** dimenzija 2*3 od matrice **x** dimenzija 2*3 daje matricu **z** dimenzija 2*3.

```
>> x = [3 5 2; 6 5 8]
x =
     3     5     2
     6     5     8
>> y = [2 8 9; 8 5 3]
y =
     2     8     9
     8     5     3
>> z = x - y
z =
     1    -3    -7
    -2     0     5
>>
```

Primjer 6.16: Oduzimanje matrice **y** dimenzija 3*2 od matrice **x** dimenzija 3*2 daje matricu **z** dimenzija 3*2.

```
>> x = [7 5; 4 6; 2 9]
x =
     7     5
     4     6
     2     9
>> y = [8 7; 3 4; 2 5]
y =
     8     7
     3     4
     2     5
>> z = x - y
z =
    -1    -2
     1     2
     0     4
>>
```

Matrice se mogu oduzimati i sa skalarom. Oduzima se tako da se skalar oduzima od svakog elementa matrice. Neka je matrica $\underline{B}$ dimenzija m*n:

$$\underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Oduzimanje skalara a od matrice $\underline{B}$ dimenzija m*n daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{B} - a = \begin{bmatrix} B_{11}-a & B_{12}-a & \dots & B_{1n}-a \\ B_{21}-a & B_{22}-a & \dots & B_{2n}-a \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1}-a & B_{m2}-a & \dots & B_{mn}-a \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.17: Oduzimanje skalara **x** od matrice **y** dimenzija 2*3 daje matricu **z** dimenzija 2*3.

```
>> x = 5
x =  5
>> y = [7 5 3; 4 6 1]
y =
     7     5     3
     4     6     1
>> z = y - x
z =
     2     0    -2
    -1     1    -4
>>
```

Da bi se matrice mogle oduzimati, moraju biti jednakih dimenzija, tj. moraju imati jednak broj redaka i jednak broj stupaca. Neka postoje matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija p*q:

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } B = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1q} \\ B_{21} & B_{22} & \dots & B_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ B_{p1} & B_{p2} & \dots & B_{pq} \end{bmatrix}$$

Ako je m≠p ili n≠q, te se dvije matrice ne mogu oduzimati jedna od druge.

Primjer 6.18: Oduzimanje matrice **y** dimenzija 2*3 od matrice **x** dimenzija 3*2 nije moguće.

```
>> x = [3 5 2; 6 5 8]
x =
     3     5     2
     6     5     8
>> y = [8 7; 3 4; 2 5]
y =
     8     7
     3     4
     2     5
>> z = x - y
error: operator -: nonconformant arguments (op1 is 2x3, op2 is 3x2)
>>
```

6.4 Operator množenja

Operator množenja služi za množenje matrica, vektora, matrica i skalara te vektora i skalara. Množiti se mogu redni vektori i stupčani vektori. Neka postoje stupčani vektor $\vec{u}$ dimenzija m*1 i redni vektor $\vec{v}$ dimenzija 1*m:

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \text{ i } \vec{v} = \begin{bmatrix} v_1 & v_2 & \dots & v_m \end{bmatrix}$$

Množenje stupčanog vektora $\vec{u}$ dimenzija m*1 i rednog vektora $\vec{v}$ dimenzija 1*m daje matricu $\underline{W}$ dimenzija m*m:

$$\underline{W} = \vec{u} \cdot \vec{v} = \begin{bmatrix} u_1 \cdot v_1 & u_1 \cdot v_2 & \dots & u_1 \cdot v_m \\ u_2 \cdot v_1 & u_2 \cdot v_2 & \dots & u_2 \cdot v_m \\ \vdots & \vdots & \vdots & \vdots \\ u_m \cdot v_1 & u_m \cdot v_2 & \dots & u_m \cdot v_m \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1m} \\ W_{21} & W_{22} & \dots & W_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ W_{m1} & W_{m2} & & W_{mm} \end{bmatrix}$$

Primjer 6.19: Množenje stupčanog vektora **x** dimenzija 3*1 s rednim vektorom **y** dimenzija 1*3 daje matricu **z** dimenzija 3*3.

```
>> x = [4; 3; 9]
x =
     4
     3
     9
>> y = [7 5 8]
y =
     7     5     8
>> z = x * y
z =
    28    20    32
    21    15    24
    63    45    72
>>
```

Množenje rednog vektora $\vec{v}$ dimenzija 1*m i stupčanog vektora $\vec{u}$ dimenzija m*1 daje skalar a (matricu dimenzija 1*1) što se naziva skalarni produkt vektora:

$$a = \vec{v} \cdot \vec{u} = \begin{bmatrix} v_1 \cdot u_1 + v_2 \cdot u_2 + \dots v_m \cdot u_m \end{bmatrix}$$

Primjer 6.20: Množenje rednog vektora **x** dimenzija 1*3 sa stupčanim vektorom **y** dimenzija 3*1 daje skalar **z** (matricu dimenzija 1*1).

```
>> x = [7 5 8]
x =
     7     5     8
>> y = [4; 3; 9]
y =
     4
     3
     9
>> z = x * y
z =
   115
>>
```

Moguće je množenje rednog vektora ili stupčanog vektora sa skalarom. Neka je redni vektor $\vec{u}$ dimenzija 1*m:

$\vec{u} = [u_1 \quad u_2 \quad \dots \quad u_m]$

Množenje rednog vektora $\vec{u}$ dimenzija 1*m i skalara a daje redni vektor $\vec{w}$ dimenzija 1*m:

$\vec{w} = a \cdot \vec{u} = [a \cdot u_1 \quad a \cdot u_2 \quad \dots \quad a \cdot u_m] = [w_1 \quad w_2 \quad \dots \quad w_m]$

Primjer 6.21: Množenje skalara **x** i rednog vektora **y** dimenzija 1*3 daje redni vektor **z** dimenzija 1*3.

```
>> x = 2
x =  2
>> y = [2 8 5]
y =
     2     8     5
>> z = x * y
z =
     4    16    10
>>
```

Množenje rednog vektora $\vec{u}$ dimenzija 1*n i rednog vektora $\vec{v}$ dimenzija 1*n ili stupčanog vektora $\vec{u}$ dimenzija m*1 i stupčanog vektora $\vec{v}$ dimenzija m*1 nije moguće.

Primjer 6.22: Množenje rednog vektora **x** dimenzija 1*3 i rednog vektora **y** dimenzija 1*3 nije moguće.

```
>> x = [7 5 8]
x =
     7     5     8
>> y = [4 3 9]
y =
     4     3     9
>> z = x * y
error: operator *: nonconformant arguments (op1 is 1x3, op2 is 1x3)
>>
```

U Octaveu je moguće množiti redni vektor $\vec{u}$ dimenzija 1*n i redni vektor $\vec{v}$ dimenzija 1*n ili stupčani vektor $\vec{u}$ dimenzija m*1 i stupčani vektor $\vec{v}$ dimenzija m*1 element po element. Za takvu se operaciju u Octaveu koristi operator **.***. Neka postoji redni vektor $\vec{u}$ dimenzija 1*n i redni vektor $\vec{v}$ dimenzija 1*n:

$\vec{u} = [u_1 \quad u_2 \quad \dots \quad u_n]$ i $\vec{v} = [v_1 \quad v_2 \quad \dots \quad v_n]$

Ta se dva vektora množe element po element i daju redni vektor $\vec{w}$ dimenzija 1*n na ovaj način:

$$\vec{w} = \vec{u} \cdot \vec{v} = [u_1 \cdot v_1 \quad u_2 \cdot v_2 \quad \dots \quad u_n \cdot v_n] = [w_1 \quad w_2 \quad \dots \quad w_n]$$

Primjer 6.23: Množenje rednog vektora **x** dimenzija 1*3 i rednog vektora **y** dimenzija 1*3 element po element daje redni vektor **z** dimenzija 1*3.

```
>> x = [7 5 8]
x =
     7     5     8
>> y = [4 3 9]
y =
     4     3     9
>> z = x .* y
z =
    28    15    72
>>
```

Da bi se dvije matrice $\underline{A}$ i $\underline{B}$ mogle množiti, broj stupaca matrice $\underline{A}$ treba biti jednak broju redaka matrice $\underline{B}$. Tada je umnožak matrice $\underline{A}$ i matrice $\underline{B}$ matrica $\underline{C}$ koja ima broj redaka jednak broju redaka matrice $\underline{A}$, a broj stupaca jednak broju stupaca matrice $\underline{B}$. Neka postoje matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija n*p:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } \underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1p} \\ B_{21} & B_{22} & \dots & B_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ B_{n1} & B_{n2} & \dots & B_{np} \end{bmatrix}$$

Umnožak matrice $\underline{A}$ dimenzija m*n i matrice $\underline{B}$ dimenzija n*p je matrica $\underline{C}$ dimenzija m*p:

$$\underline{C} = \underline{A} \cdot \underline{B} = \begin{bmatrix} A_{11} \cdot B_{11} + \dots + A_{1n} \cdot B_{n1} & \dots & \dots & A_{11} \cdot B_{1p} + \dots + A_{1n} \cdot B_{np} \\ A_{21} \cdot B_{11} + \dots + A_{2n} \cdot B_{n1} & & & A_{21} \cdot B_{1p} + \dots + A_{2n} \cdot B_{np} \\ \vdots & & \vdots \quad \vdots & \vdots \\ A_{m1} \cdot B_{11} + \dots + A_{mn} \cdot B_{n1} & \dots & \dots & A_{m1} \cdot B_{1p} + \dots + A_{mn} \cdot B_{np} \end{bmatrix}$$

$$\underline{C} = \underline{A} \cdot \underline{B} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1p} \\ C_{21} & C_{22} & \dots & C_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mp} \end{bmatrix}$$

Primjer 6.24: Množenje matrice **x** dimenzija 2*3 i matrice **y** dimenzija 3*2 daje matricu **z** dimenzija 2*2.

```
>> x = [3 5 2; 6 5 8]
x =
     3     5     2
     6     5     8
>> y = [8 7; 3 4; 2 5]
y =
     8     7
     3     4
     2     5
>> z = x * y
z =
    43    51
    79   102
>>
```

Primjer 6.25: Množenje matrice **x** dimenzija 3*2 i matrice **y** dimenzija 2*3 daje matricu **z** dimenzija 3*3.

```
>> x = [7 5; 4 6; 2 9]
x =
     7     5
     4     6
     2     9
>> y = [2 8 9; 8 5 3]
y =
     2     8     9
     8     5     3
>> z = x * y
z =
    54    81    78
    56    62    54
    76    61    45
>>
```

Množiti se može i matrica sa skalarom. Neka postoji matrica $\underline{B}$ dimenzija m*n:

$$\underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Množenje matrice $\underline{B}$ dimenzija m*n sa skalarom a daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = a \cdot \underline{B} = \begin{bmatrix} a \cdot B_{11} & a \cdot B_{12} & \dots & a \cdot B_{1n} \\ a \cdot B_{21} & a \cdot B_{22} & \dots & a \cdot B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a \cdot B_{m1} & a \cdot B_{m2} & \dots & a \cdot B_{mn} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.26: Množenje matrice **x** dimenzija 2*3 sa skalarom **y** daje matricu **z** dimenzija 2*3.

```
>> x = [4 7 6; 3 5 1]
x =
     4     7     6
     3     5     1
>> y = 5
y =  5
>> z = y * x
z =
    20    35    30
    15    25     5
>>
```

Ako broj stupaca matrice $\underline{A}$ nije jednak broju redaka matrice $\underline{B}$, nije moguće množiti te dvije matrice.

Primjer 6.27: Množenje matrice **x** dimenzija 2*3 s matricom **y** dimenzija 4*3 nije moguće.

```
>> x = [3 5 2; 6 5 8]
x =
```

```
      3      5      2
      6      5      8
>> y = [8 7 4; 3 4 7; 2 5 1; 9 6 5]
y =
      8      7      4
      3      4      7
      2      5      1
      9      6      5
>> z = x * y
error: operator *: nonconformant arguments (op1 is 2x3, op2 is 4x3)
>>
```

Matrice u Octaveu moguće je množiti i element po element. Za takvu se operaciju koristi operator **.***. Matrice moraju biti jednakih dimenzija kako bi se mogla izvršiti operacija množenja element po element. Neka postoje matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija m*n:

$$\underline{A}=\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } \underline{B}=\begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Množenje matrice $\underline{A}$ dimenzija m*n s matricom $\underline{B}$ dimenzija m*n element po element daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C}=\underline{A}\cdot\underline{B}=\begin{bmatrix} A_{11}\cdot B_{11} & A_{12}\cdot B_{12} & \dots & A_{1n}\cdot B_{1n} \\ A_{21}\cdot B_{21} & A_{22}\cdot B_{22} & \dots & A_{2n}\cdot B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}\cdot B_{m1} & A_{m2}\cdot B_{m2} & \dots & A_{mn}\cdot B_{mn} \end{bmatrix}=\begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.28: Množenje matrice **x** dimenzija 2*3 s matricom **y** dimenzija 2*3 element po element daje matricu **z** dimenzija 2*3.

```
>> x = [2 8 7; 6 5 3]
x =
      2      8      7
      6      5      3
>> y = [3 4 5; 8 3 7]
y =
      3      4      5
      8      3      7
>> z = x .* y
z =
      6     32     35
     48     15     21
>>
```

Za množenje matrica i skalara vrijedi sljedeće:

$$(a+b)\cdot\underline{A}=a\cdot\underline{A}+b\cdot\underline{A}$$

$$a\cdot(\underline{A}+\underline{B})=a\cdot\underline{A}+a\cdot\underline{B}$$

$$a\cdot(b\cdot\underline{A})=a\cdot b\cdot\underline{A}$$

Umnožak dviju matrica općenito nije komutativan, tj. ovisi o poretku članova, pri čemu mogu biti dva slučaja:

a) $\underline{A}\cdot\underline{B}\neq\underline{B}\cdot\underline{A}$ ili

b) $\underline{A} \cdot \underline{B}$ postoji, a $\underline{B} \cdot \underline{A}$ ne postoji.

6.5 Operator dijeljenja

Operator dijeljenja služi za dijeljenje matrica, vektora, matrica i skalara te vektora i skalara. Vektor i matrica mogu se dijeliti sa skalarom. Za takvu se operaciju u Octaveu koristi operator */*. Neka postoji redni vektor $\vec{u}$ dimenzija 1*n:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}$$

Dijeljenje rednog vektora $\vec{u}$ dimenzija 1*n sa skalarom a daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w} = \vec{u} / a = \begin{bmatrix} u_1/a & u_2/a & \dots & u_n/a \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.29: Dijeljenje rednog vektora **x** dimenzija 1*3 sa skalarom **y** daje redni vektor **z** dimenzija 1*3.

```
>> x = [2 8 5]
x =
     2.00     8.00     5.00
>> y = 2
y =  2.00
>> z = x / y
z =
    1.00    4.00    2.50
>>
```

Neka postoji matrica $\underline{A}$ dimenzija m*n:

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix}$$

Dijeljenje matrice $\underline{A}$ dimenzija m*n sa skalarom b daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{A} / b = \begin{bmatrix} A_{11}/b & A_{12}/b & \dots & A_{1n}/b \\ A_{21}/b & A_{22}/b & \dots & A_{2n}/b \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}/b & A_{m2}/b & \dots & A_{mn}/b \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.30: Dijeljenje matrice **x** dimenzija 2*3 sa skalarom **y** daje matricu **z** dimenzija 2*3.

```
>> x = [4 7 6; 3 5 1]
x =
     4.00     7.00     6.00
     3.00     5.00     1.00
>> y = 5
y =  5.00
>> z = x / y
z =
    0.80    1.40    1.20
    0.60    1.00    0.20
>>
```

Moguće je dijeliti vektor s vektorom element po element i matricu s matricom element po element. U Octaveu se takvo dijeljenje izvodi operatorom **./**. Vektori i matrice koje se dijele moraju biti jednakih dimenzija. Neka postoje redni vektor $\vec{u}$ dimenzija 1*n i redni vektor $\vec{v}$ dimenzija 1*n:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \text{ i } \vec{v} = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

Dijeljenje rednog vektora $\vec{u}$ dimenzija 1*n s rednim vektorom $\vec{v}$ dimenzija 1*n daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w} = \vec{u} / \vec{v} = \begin{bmatrix} u_1/v_1 & u_2/v_2 & \dots & u_n/v_n \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.31: Dijeljenje rednog vektora **x** dimenzija 1*3 s rednim vektorom **y** dimenzija 1*3 element po element daje redni vektor **z** dimenzija 1*3.

```
>> x = [7 5 8]
x =
     7.00     5.00     8.00
>> y = [4 3 9]
y =
     4.00     3.00     9.00
>> z = x ./ y
z =
    1.75    1.67    0.89
>>
```

Neka postoje matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija m*n:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } \underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Dijeljenje matrice $\underline{A}$ dimenzija m*n matricom $\underline{B}$ dimenzija m*n daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{A} / \underline{B} = \begin{bmatrix} A_{11}/B_{11} & A_{12}/B_{12} & \dots & A_{1n}/B_{1n} \\ A_{21}/B_{21} & A_{22}/B_{22} & \dots & A_{2n}/B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}/B_{m1} & A_{m2}/B_{m2} & \dots & A_{mn}/B_{mn} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.32: Dijeljenje matrice **x** dimenzija 2*3 s matricom **y** dimenzija 2*3 element po element daje matricu **z** dimenzija 2*3.

```
>> x = [2 8 7; 6 5 3]
x =
     2.00     8.00     7.00
     6.00     5.00     3.00
>> y = [3 4 5; 8 3 7]
y =
     3.00     4.00     5.00
     8.00     3.00     7.00
>> z = x ./ y
z =
    0.67    2.00    1.40
    0.75    1.67    0.43
>>
```

6.6 Operator potenciranja

Operator potenciranja služi za potenciranje matrica, vektora, matrica i skalara te vektora i skalara. Vektor i matrica mogu se potencirati sa skalarom. U Octaveu se za takvu operaciju koristi operator **.^**.Neka postoji redni vektor $\vec{u}$ dimenzija 1*n:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix}$$

Potenciranje rednog vektora $\vec{u}$ dimenzija 1*n sa skalarom a daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w} = \vec{u}^{\,a} = \begin{bmatrix} u_1^a & u_2^a & \dots & u_n^a \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.33: Potenciranje rednog vektora **x** dimenzija 1*3 sa skalarom **y** daje redni vektor **z** dimenzija 1*3.

```
>> x = [2 8 5]
x =
     2     8     5
>> y = 3
y =  3
>> z = x .^ y
z =
     8   512   125
>>
```

Neka postoji matrica $\underline{A}$ dimenzija m*n:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix}$$

Potenciranje matrice $\underline{A}$ dimenzija m*n sa skalarom b daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{A}^b = \begin{bmatrix} A_{11}^b & A_{12}^b & \dots & A_{1n}^b \\ A_{21}^b & A_{22}^b & \dots & A_{2n}^b \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}^b & A_{m2}^b & \dots & A_{mn}^b \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.34: Potenciranje matrice **x** dimenzija 2*3 sa skalarom **y** element po element daje matricu **z** dimenzija 2*3.

```
>> x = [4 7 6; 3 5 1]
x =
     4     7     6
     3     5     1
>> y = 5
y =  5
>> z = x .^ y
z =
        1024       16807        7776
         243        3125           1
>>
```

Moguće je potencirati vektor s vektorom element po element i matricu s matricom element po element. U Octaveu se takvo potenciranje izvodi operatorom **.^**. Vektori i matrice koje se potenciraju moraju biti jednakih dimenzija. Neka postoje redni vektor $\vec{u}$ dimenzija 1*n i redni vektor $\vec{v}$ dimenzija 1*n:

$$\vec{u} = \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \text{ i } \vec{v} = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$$

Potenciranje rednog vektora $\vec{u}$ dimenzija 1*n rednim vektorom $\vec{v}$ dimenzija 1*n element po element daje redni vektor $\vec{w}$ dimenzija 1*n:

$$\vec{w} = \vec{u}^{\vec{v}} = \begin{bmatrix} u_1^{v_1} & u_2^{v_2} & \dots & u_n^{v_n} \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

Primjer 6.35: Potenciranje rednog vektora **x** dimenzija 1*3 rednim vektorom **y** dimenzija 1*3 element po element daje redni vektor **z** dimenzija 1*3.

```
>> x = [7 5 8]
x =
     7     5     8
>> y = [4 3 9]
y =
     4     3     9
>> z = x .^ y
z =
        2401         125   134217728
>>
```

Neka postoje matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija m*n:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix} \text{ i } \underline{B} = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1n} \\ B_{21} & B_{22} & \dots & B_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mn} \end{bmatrix}$$

Potenciranje matrice $\underline{A}$ dimenzija m*n matricom $\underline{B}$ dimenzija m*n element po element daje matricu $\underline{C}$ dimenzija m*n:

$$\underline{C} = \underline{A}^{\underline{B}} = \begin{bmatrix} A_{11}^{B_{11}} & A_{12}^{B_{12}} & \dots & A_{1n}^{B_{1n}} \\ A_{21}^{B_{21}} & A_{22}^{B_{22}} & \dots & A_{2n}^{B_{2n}} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1}^{B_{m1}} & A_{m2}^{B_{m2}} & \dots & A_{mn}^{B_{mn}} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ C_{m1} & C_{m2} & \dots & C_{mn} \end{bmatrix}$$

Primjer 6.36: Potenciranje matrice **x** dimenzija 2*3 matricom **y** dimenzija 2*3 element po element daje matricu **z** dimenzija 2*3.

```
>> x = [2 8 7; 6 5 3]
x =
     2     8     7
     6     5     3
>> y = [3 4 5; 8 3 7]
y =
     3     4     5
     8     3     7
>> z = x .^ y
z =
           8        4096       16807
     1679616         125        2187
>>
```

U Octaveu se mogu potencirati matrice sa skalarom pomoću operatora **^**. Da bi se matrice mogle potencirati pomoću tog operatora, trebaju biti kvadratne, odnosno ako postoji matrica $\underline{A}$:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mn} \end{bmatrix}$$

tada mora biti zadovoljeno da je m=n. Ako nije ispunjen navedeni uvjet, matrica se ne može potencirati. Potenciranje se može prikazati na sljedeći način:

$$\underline{C} = \underline{A}^2 = \underline{A} \cdot \underline{A}$$

$$\underline{C} = \underline{A}^3 = \underline{A} \cdot \underline{A} \cdot \underline{A}$$

$$\underline{C} = \underline{A}^n = \prod_{i=1}^{n} \underline{A}$$

Primjer 6.37: Potenciranje matrice **x** dimenzija 3*3 sa skalarom **y** daje matricu **z** dimenzija 3*3.

```
>> x = [1 4 5; 6 5 4; 8 7 2]
x =
     1     4     5
     6     5     4
     8     7     2
>> y = 3
y =  3
>> z = x ^ y
z =
         667         772         623
         994        1063         764
        1128        1173         798
>>
```

6.7 Relacijski operatori

Relacijski operatori služe za usporedbu vrijednosti između skalara, vektora ili matrica, a kao rezultat daju logičke vrijednosti. Logička vrijednost može imati dvije moguće vrijednosti: 0 ako uvjet nije zadovoljen i 1 ako je uvjet zadovoljen. Kod primjene relacijskih operatora između dvaju vektora ili dviju matrica dimenzije vektora ili matrica moraju biti jednake. Operator između matrice i skalara ili vektora i skalara kao rezultat daje matricu ili vektor jednakih dimenzija kao ulazni vektor ili matrica, a dobije se primjenom operatora između skalara i svakog elementa vektora ili matrice. Octave podržava relacijske operatore prikazane u tablici 6.2.

Tablica 6.2 Relacijski operatori

Operator	Opis
==	Jednako
~=	Različito (nije jednako)
>	Veće od
>=	Veće od ili jednako
<	Manje od
<=	Manje od ili jednako

Relacijski operator jednako (==) između matrice $\underline{A}$ dimenzija m*n i skalara x može se prikazati kao:

$$\begin{bmatrix} A_{11} == x & A_{12} == x & \dots & A_{1n} == x \\ A_{21} == x & A_{22} == x & \dots & A_{2n} == x \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} == x & A_{m2} == x & \dots & A_{mn} == x \end{bmatrix}$$

Relacijski operator različito (nije jednako) (~=) između matrice $\underline{A}$ dimenzija m*n i skalara x može se prikazati kao:

$$\begin{bmatrix} A_{11} \neq x & A_{12} \neq x & \dots & A_{1n} \neq x \\ A_{21} \neq x & A_{22} \neq x & \dots & A_{2n} \neq x \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} \neq x & A_{m2} \neq x & \dots & A_{mn} \neq x \end{bmatrix}$$

Relacijski operator veće od (>) između matrice $\underline{A}$ dimenzija m*n i skalara x može se prikazati kao:

$$\begin{bmatrix} A_{11} > x & A_{12} > x & \dots & A_{1n} > x \\ A_{21} > x & A_{22} > x & \dots & A_{2n} > x \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} > x & A_{m2} > x & \dots & A_{mn} > x \end{bmatrix}$$

Relacijski operator veće od ili jednako (>=) između matrice $\underline{A}$ dimenzija m*n i skalara x može se prikazati kao:

$$\begin{bmatrix} A_{11} \geq x & A_{12} \geq x & \dots & A_{1n} \geq x \\ A_{21} \geq x & A_{22} \geq x & \dots & A_{2n} \geq x \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} \geq x & A_{m2} \geq x & \dots & A_{mn} \geq x \end{bmatrix}$$

Relacijski operator manje od (<) između matrice $\underline{A}$ dimenzija m*n i skalara x može se prikazati kao:

$$\begin{bmatrix} A_{11} < x & A_{12} < x & \dots & A_{1n} < x \\ A_{21} < x & A_{22} < x & \dots & A_{2n} < x \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} < x & A_{m2} < x & \dots & A_{mn} < x \end{bmatrix}$$

Relacijski operator manje od ili jednako (<=) između matrice $\underline{A}$ dimenzija m*n i skalara x može se prikazati kao:

$$\begin{bmatrix} A_{11} \leq x & A_{12} \leq x & \dots & A_{1n} \leq x \\ A_{21} \leq x & A_{22} \leq x & \dots & A_{2n} \leq x \\ \vdots & \vdots & \vdots & \vdots \\ A_{m1} \leq x & A_{m2} \leq x & \dots & A_{mn} \leq x \end{bmatrix}$$

Primjer 6.38: Relacijski operator jednako (**==**) između matrice **x** dimenzija 2*3 i skalara **y** daje matricu **z** dimenzija 2*3. Za slučaj da je vrijednost elementa matrice **x** jednaka vrijednosti skalara **y**, u matrici **z** element matrice ima logičku vrijednost 1, a gdje nije jednak, logičku vrijednost 0.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = 4
y =  4
>> z = x == y
z =
     0     0     0
     1     0     0
```

```
>>
```

Primjer 6.39: Relacijski operator različito (nije jednako, **~=**) između matrice **x** dimenzija 2*3 i skalara **y** daje matricu **z** dimenzija 2*3. Za slučaj da vrijednost elementa matrice **x** nije jednaka vrijednosti skalara **y**, u matrici **z** element matrice ima logičku vrijednost 1, a gdje je jednak, logičku vrijednost 0.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = 4
y =  4
>> z = x ~= y
z =
     1     1     1
     0     1     1
>>
```

Primjer 6.40: Relacijski operator veće od (**>**) između matrice **x** dimenzija 2*3 i skalara **y** daje matricu **z** dimenzija 2*3. Za slučaj da je vrijednost elementa matrice **x** veća od vrijednosti skalara **y**, u matrici **z** element matrice ima logičku vrijednost 1, a gdje nije veća od ili je jednaka, logičku vrijednost 0.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = 4
y =  4
>> z = x > y
z =
     0     1     0
     0     1     1
>>
```

Primjer 6.41: Relacijski operator veće od ili jednako (**>=**) između matrice **x** dimenzija 2*3 i skalara **y** daje matricu **z** dimenzija 2*3. Za slučaj da je vrijednost elementa matrice **x** veća od ili jednaka vrijednosti skalara **y**, u matrici **z** element matrice ima logičku vrijednost 1, a gdje nije veća od, logičku vrijednost 0.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = 4
y =  4
>> z = x >= y
z =
     0     1     0
     1     1     1
>>
```

Primjer 6.42: Relacijski operator manje od (**<**) između matrice **x** dimenzija 2*3 i skalara **y** daje matricu **z** dimenzija 2*3. Za slučaj da je vrijednost elementa matrice **x** manja od vrijednosti skalara **y**, u matrici **z** element matrice ima logičku vrijednost 1, a gdje nije manja od ili je jednaka, logičku vrijednost 0.

```
>> x = [2 5 3; 4 8 6]
```

```
x =
     2     5     3
     4     8     6
>> y = 4
y =  4
>> z = x < y
z =
     1     0     1
     0     0     0
>>
```

Primjer 6.43: Relacijski operator manje od ili jednako (**<=**) između matrice **x** dimenzija 2*3 i skalara **y** daje matricu **z** dimenzija 2*3. Za slučaj da je vrijednost elementa matrice **x** manja od ili jednaka vrijednosti skalara **y**, u matrici **z** element matrice ima logičku vrijednost 1, a gdje nije manja od, logičku vrijednost 0.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = 4
y =  4
>> z = x <= y
z =
     1     0     1
     1     0     0
>>
```

Primjer 6.44: Relacijski operator veće od (**>**) između rednog vektora **x** dimenzija 1*4 i rednog vektora **y** dimenzija 1*4 daje redni vektor **z** dimenzija 1*4. Za slučaj da je vrijednost odgovarajućeg elementa rednog vektora **x** veća od vrijednosti odgovarajućeg elementa rednog vektora **y**, u rednom vektoru **z** element vektora ima logičku vrijednost 1, a gdje nije veća od ili je jednaka, logičku vrijednost 0.

```
>> x = [2 5 8 7]
x =
     2     5     8     7
>> y = [1 6 3 9]
y =
     1     6     3     9
>> z = x > y
z =
     1     0     1     0
>>
```

Primjer 6.45: Relacijski operator manje od (**<**) između matrice **x** dimenzija 2*3 i matrice **y** dimenzija 2*3 daje matricu **z** dimenzija 2*3. Za slučaj da je vrijednost odgovarajućeg elementa matrice **x** manja od vrijednosti odgovarajućeg elementa matrice **y**, u matrici **z** element matrice ima logičku vrijednost 1, a gdje nije manja od, logičku vrijednost 0.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = [3 8 5; 2 5 7]
y =
     3     8     5
     2     5     7
>> z = x < y
z =
```

```
     1     1     1
     0     0     1
>>
```

6.8 Logički operatori

Logički operatori primjenjuju se između logičkih vrijednosti i kao rezultat daju logičke vrijednosti. Logički operatori mogu se primijeniti i između realnih vrijednosti, pri čemu se realne vrijednosti interno pretvaraju u logičke prije samih operacija, i to na način da realna vrijednost 0 prelazi u logičku vrijednost 0, a svi ostali realni brojevi prelaze u logičku jedinicu. Ako se logički operatori primjenjuju između matrice i matrice ili vektora i vektora, moraju biti jednakih dimenzija. Ako se logički operatori primjenjuju između matrice i skalara ili između vektora i skalara, određuje se logička vrijednost između svakog elementa matrice ili vektora i skalara za odgovarajući logički operator. Logički operatori najčešće se koriste u kombinaciji s relacijskim operatorima i/ili naredbama odluka i ponavljanja. Octave podržava logičke operatore prikazane u tablici 6.3.

Tablica 6.3 Logički operatori

Operator	Opis
&, **and**	Logički I
|, **or**	Logički ILI
~, **not**	Logički komplement (NE)
xor	Logički ekskluzivni ILI

Neka su x, y i z logičke vrijednosti. Tablica 6.4 prikazuje tablicu stanja za logičku operaciju I (engl. *AND*).

Tablica 6.4 Logička operacija I (engl. *AND*)

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

U Octaveu se tablica stanja za logičku operaciju I (engl. *AND*) može poopćiti jer vrijednosti mogu biti i realne (tablica 6.5).

Tablica 6.5 Poopćena logička operacija I (engl. *AND*) u Octaveu

x	y	z
0	0	0
0	Različit od 0	0
Različit od 0	0	0
Različit od 0	Različit od 0	1

Tablica 6.6 prikazuje tablicu stanja za logičku operaciju ILI (engl. *OR*).

Tablica 6.6 Logička operacija ILI (engl. *OR*)

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

U Octaveu se tablica stanja za logičku operaciju ILI (engl. *OR*) može poopćiti jer vrijednosti mogu biti i realne (tablica 6.7).

Tablica 6.7 Poopćena logička operacija ILI (engl. *OR*) u Octaveu

x	y	z
0	0	0
0	Različit od 0	1
Različit od 0	0	1
Različit od 0	Različit od 0	1

Tablica 6.8 prikazuje tablicu stanja za logički komplement NE (engl. *NOT*).

Tablica 6.8 Logički komplement NE (engl. *NOT*)

x	z
0	1
1	0

U Octaveu se tablica stanja za logički komplement NE (engl. *NOT*) može poopćiti jer vrijednosti mogu biti i realne (tablica 6.9).

Tablica 6.9 Poopćeni logički komplement NE (engl. *NOT*) u Octaveu

x	z
0	1
Različit od 0	0

Tablica 6.10 prikazuje tablicu stanja za logičku operaciju ekskluzivni ILI (engl. *exclusive OR, XOR*).

Tablica 6.10 Logička operacija ekskluzivni ILI (engl. *exclusive OR, XOR*)

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

U Octaveu se tablica stanja za logički ekskluzivni ILI (engl. *exclusive OR, XOR*) može poopćiti jer vrijednosti mogu biti i realne (tablica 6.11).

Tablica 6.11 Poopćena logička operacija ekskluzivni ILI (engl. *exclusive OR, XOR*) u Octaveu

x	y	z
0	0	0
0	Različit od 0	1
Različit od 0	0	1
Različit od 0	Različit od 0	0

Primjer 6.46: Logička operacija I između matrice **x** dimenzija 2*3 i matrice **y** dimenzija 2*3 daje matricu **z** dimenzija 2*3. Na matricu **x** je prije logičke operacije I primijenjen relacijski operator manje od ili jednako (**<=**), a na matricu **y** relacijski operator jednako (**==**).

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = [3 8 5; 2 5 7]
y =
     3     8     5
     2     5     7
>> x <= 2
ans =
     1     0     0
     0     0     0
>> y == 3
ans =
     1     0     0
     0     0     0
```

```
>> z = (x<=2) & (y==3)
z =
     1     0     0
     0     0     0
>>
```

Primjer 6.47: Na matricu **x** dimenzija 2*3 prije logičkog komplementa (NE) primijenjen je relacijski operator manje od ili jednako (**<=**), i to daje matricu **z** dimenzija 2*3, a na matricu **y** dimenzija 2*3 prije logičkog komplementa primijenjen je relacijski operator jednako (**==**), i to daje matricu **w** dimenzija 2*3.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = [3 8 5; 2 5 7]
y =
     3     8     5
     2     5     7
>> z = ~(x<=2)
z =
     0     1     1
     1     1     1
>> w = ~(y==5)
w =
     1     1     0
     1     0     1
>>
```

Primjer 6.48: Logička operacija ILI između matrice **x** dimenzija 2*3 i matrice **y** dimenzija 2*3 daje matricu **z** dimenzija 2*3. Na matricu **x** dimenzija 2*3 prije logičke operacije ILI primijenjen je relacijski operator manje od ili jednako (**<=**), a na matricu **y** dimenzija 2*3 prije logičke operacije ILI primijenjen je relacijski operator jednako (**==**).

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = [3 8 5; 2 5 7]
y =
     3     8     5
     2     5     7
>> x <= 2
ans =
     1     0     0
     0     0     0
>> y == 5
ans =
     0     0     1
     0     1     0
>> z = (x<=2) | (y==5)
z =
     1     0     1
     0     1     0
>>
```

Primjer 6.49: Logička operacija ekskluzivni ILI između matrice **x** dimenzija 2*3 i matrice **y** dimenzija 2*3 daje matricu **z** dimenzija 2*3. Na matricu **x** dimenzija 2*3 prije logičke operacije ekskluzivni ILI primijenjen je relacijski operator manje od ili jednako (**<=**), a na matricu **y** dimenzija 2*3 prije logičke operacije ekskluzivni ILI primijenjen je relacijski operator jednako (**==**).

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = [3 8 5; 2 5 7]
y =
     3     8     5
     2     5     7
>> x <= 2
ans =
     1     0     0
     0     0     0
>> y == 3
ans =
     1     0     0
     0     0     0
>> z = xor(x<=2,y==3)
z =
     0     0     0
     0     0     0
>>
```

Postoje još i logičke funkcije **all** i **any**. Funkcija **all** određuje jesu li svi elementi nekog polja (matrice, vektora itd.) po nekoj dimenziji različiti od nule. Funkcija **any** određuje je li bilo koji od elemenata nekog polja (matrice, vektora itd.) po nekoj dimenziji različit od nule. I ova dva logička operatora najčešće se koriste u kombinaciji s relacijskim operatorima i/ili naredbama odluka i ponavljanja. Funkcija **any** ima oblik **any(x)** ili **any(x,dim)**. Funkcija **all** ima oblik **all(x)** ili **all(x,dim)**. U funkcijama **all(x,dim)** i **any(x,dim)**, **x** predstavlja matricu ili vektor, a **dim** je dimenzija. Dimenzija **dim** je skalar i za matricu može biti redak ili stupac. Ako se radi o retku, onda je **dim=1**, a ako se radi o stupcu, onda je **dim=2**.

Primjer 6.50: Logička funkcija **any** između matrice **x** dimenzija 2*3 i matrice **y** dimenzija 2*3. Na matricu **x** dimenzija 2*3 prije logičke funkcije **any** primijenjen je relacijski operator manje od ili jednako (**<=**), a na matricu **y** dimenzija 2*3 relacijski operator jednako (**==**). Kod matrice **z** logička funkcija **any** primijenjena je po redcima, a kod matrice **w** logička funkcija **any** primijenjena je po stupcima.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = [3 8 5; 2 5 7]
y =
     3     8     5
     2     5     7
>> x <= 2
ans =
     1     0     0
     0     0     0
>> y == 5
ans =
     0     0     1
     0     1     0
>> (x<=2) | (y==5)
ans =
     1     0     1
     0     1     0
>> z = any((x<=2) | (y==5),1)
z =
     1     1     1
>> w = any((x<=2) | (y==5),2)
w =
     1
     1
>>
```

Primjer 6.51: Logička funkcija **`all`** između matrice **`x`** dimenzija 2*3 i matrice **`y`** dimenzija 2*3. Na matrice **`x`** i **`y`** dimenzija 2*3 prije logičke funkcije **`all`** primijenjen je relacijski operator jednako (**==**). Kod matrice **`z`** logička funkcija **`all`** primijenjena je po redcima, a kod matrice **`w`** logička funkcija **`all`** primijenjena je po stupcima.

```
>> x = [2 5 3; 4 8 6]
x =
     2     5     3
     4     8     6
>> y = [3 8 5; 2 5 7]
y =
     3     8     5
     2     5     7
>> x == 5
ans =
     0     1     0
     0     0     0
>> y == 5
ans =
     0     0     1
     0     1     0
>> (x==5) | (y==5)
ans =
     0     1     1
     0     1     0
>> z = all((x==5) | (y==5),1)
z =
     0     1     0
>> w = all((x==5) | (y==5),2)
w =
     0
     0
>>
```

Pitanja za provjeru znanja:

1. Mogu li se u Octaveu zbrajati ili oduzimati vektori različitih dimenzija?
2. Mogu li se u Octaveu zbrajati ili oduzimati vektori sa skalarima?
3. Mogu li se u Octaveu zbrajati ili oduzimati dvije matrice različitih dimenzija?
4. Mogu li se u Octaveu zbrojiti ili oduzimati matrice sa skalarima?
5. Što je rezultat množenja vektora dimenzija m*1 i vektora dimenzija 1*m?
6. Što je rezultat množenja vektora dimenzija 1*m i vektora dimenzija m*1?
7. Može li se u Octaveu pomnožiti vektor ili matrica sa skalarom?
8. Za što se u Octaveu koristi operator `.*`?
9. Mogu li se u Octaveu dijeliti vektor ili matrica sa skalarom?
10. Za što se u Octaveu koristi operator `./`?
11. Za što se u Octaveu-u koristi operator `.^`?
12. Za što se u Octaveu koriste relacijski operatori?
13. Koji relacijski operatori postoje u Octaveu?
14. Za što se u Octaveu koriste logički operatori?
15. Koji logički operatori postoje u Octaveu?

7. NAREDBE PONAVLJANJA I ODLUKE

U Octaveu postoji nekoliko naredbi koje služe za ponavljanje određenih dijelova programa (petlja) i za odlučivanje o tijeku programa (grananje).

7.1 Naredbe ponavljanja

U naredbe ponavljanja ubrajaju se **`for`** petlja i **`while`** petlja. Kod **`for`** petlje najčešće se dio programa ponavlja unaprijed određeni broj puta dok se kod **`while`** petlje najčešće dio programa ponavlja dok se ne zadovolji uvjet. Broj ponavljanja dijela programa unutar **`while`** petlje nije unaprijed poznat kao što je to kod **`for`** petlje.

for

Petlja **`for`** ima sljedeću sintaksu:

```
for varijabla = izraz
      naredba ili funkcija
      naredba ili funkcija
      ...
end
```

Mogu se rabiti tzv. ugniježđene **`for`** petlje čija je sintaksa sljedeća:

```
for varijabla = izraz
      naredbe ili funkcije
            for varijabla = izraz
                  naredbe ili funkcije
                        for varijabla = izraz
                              naredbe ili funkcije
                        end
                  ...
            end
      ...
end
```

Dio **`for`** petlje **varijabla = izraz** naziva se brojač petlje. Taj dio određuje koliko će se puta dio programa između **`for`** i **`end`** dijela izvršavati. Brojač **varijabla = izraz** u dijelu **`for`** petlje najčešće ima oblik **brojac = pocetak:korak:kraj**. Petlja mijenja vrijednost brojača od vrijednosti **pocetak** do vrijednosti **kraj** uz povećavanje vrijednosti određene vrijednošću **korak**.

Primjer 7.1: **`for`** petlja ponavlja se 50 puta i pri svakom prolasku kroz petlju varijabli **`x`** dodaje se vlastita vrijednost iz prijašnjeg koraka i vrijednost brojača petlje **`j`**. Taj program izračunava zbroj brojeva od 1 do 50. Na početku se varijabli **`x`** dodjeljuje vrijednost 0 kako bi zbroj na početku programa bio jednak nuli.

```
x = 0;
for j = 1:50
    x = x + j;
```

```
end
x
```

Ovo je rezultat izvršavanja gornjeg programa.

```
x =  1275
>>
```

Primjer 7.2: **`for`** petlja ponavlja se 10 puta i pri svakom prolasku kroz petlju elementima (određenim brojačem petlje) rednog vektora **`y`** dimenzija 1*10 upisuje se vrijednost **`1/j`**, gdje je **`j`** vrijednost brojača petlje.

```
for j = 1:10
    y(1,j) = 1/j;
end
y
```

Ovo je rezultat izvršavanja gornjeg programa:

```
y =
    1.00    0.50    0.33    0.25    0.20    0.17    0.14    0.12    0.11
0.10
>>
```

Primjer 7.3: Ugniježđena **`for`** petlja koja se sastoji od dvije **`for`** petlje. Dio programa unutar prve (vanjske) **`for`** petlje ponavlja se 10 puta, a isto tako i dio programa unutar druge (unutarnje) **`for`** petlje. Dio programa unutar druge (unutarnje) **`for`** petlje zbog prve se (vanjske) petlje ponavlja 100 puta (10x10). Pri svakom prolasku kroz drugu **`for`** petlju u odgovarajući se element (određen brojačima petlje **`m`** i **`n`**) matrice **`x`** upisuje umnožak prvog i drugog brojača petlje **`m*n`**. Taj program u matricu **`x`** dimenzija 10*10 upisuje vrijednosti tablice množenja brojeva od 1 do 10.

```
for m = 1:10
    for n = 1:10
        x(m,n) = m * n;
    end
end
x
```

Matrica **`x`** dimenzija 10*10 koja je rezultat izvršavanja gore navedenog programa.

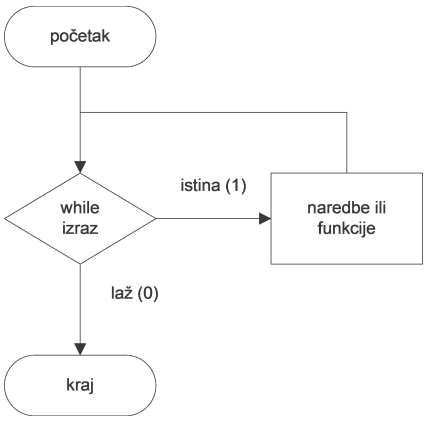
```
x =
     1     2     3     4     5     6     7     8     9    10
     2     4     6     8    10    12    14    16    18    20
     3     6     9    12    15    18    21    24    27    30
     4     8    12    16    20    24    28    32    36    40
     5    10    15    20    25    30    35    40    45    50
     6    12    18    24    30    36    42    48    54    60
     7    14    21    28    35    42    49    56    63    70
     8    16    24    32    40    48    56    64    72    80
     9    18    27    36    45    54    63    72    81    90
    10    20    30    40    50    60    70    80    90   100
>>
```

while

Sintaksa **while** petlje ima sljedeći oblik:

```
while uvjet
      naredba ili funkcija
      naredba ili funkcija
      ...
end
```

Dio programa između **while** i **end** izvršavat će se sve dok je ispunjen uvjet, odnosno dok uvjet ima logičku vrijednost 1. Kada uvjet više nije ispunjen, odnosno vrijednost uvjeta poprima logičku vrijednost 0, izvršavanje petlje prekida se. Na slici 7.1 prikazan je dijagram toka za **while** petlju.



Slika 7.1 Dijagram toka za **while** petlju

Primjer 7.4: Varijabli **x** se dodjeljuje vrijednost 100. U **while** petlji ispituje se je li vrijednost varijable **x** veća od 1. Ako je ispunjen taj uvjet, izvršava se dio programa između **while** i **end**, odnosno u ovom se slučaju vrijednost varijable **x** dijeli sa 2. Dok je uvjet ispunjen, varijabla **x** će se dijeliti sa 2, kada varijabla **x** bude manja od 1, izvršavanje naredbi u petlji prekida se.

```
x = 100
while x > 1
    x = x / 2
end
```

Program unutar **while** petlje izvršio se 7 puta, odnosno vrijednost varijable **x** se 7 puta dijelila sa 2, a potom je njegovo izvršavanje prekinuto jer je vrijednost varijable **x** postala 0,78125 što je manje od 1. Ovo je prikaz izvođenja gore navedenog programa.

```
x =  100
x =  50
x =  25
x =  12.500
x =  6.2500
x =  3.1250
x =  1.5625
x =  0.78125
>>
```

Primjer 7.5: Varijabli **x** dodjeljuje se vrijednost 100, a varijabli **y** vrijednost 1,5. U **while** petlji ispituje se je li vrijednost varijable **x** veća od 1 ili vrijednost varijable **y** manja od 100. Dio programa unutar **while** petlje izvršavat će se dok taj uvjet bude ispunjen, odnosno dok logička vrijednost ispitivanja tog uvjeta bude 1. Kada logička vrijednost tog uvjeta bude 0, prekida se izvršavanje programa. Ako je ispunjen taj uvjet, izvršava se dio programa između **while** i **end**, odnosno u ovom se slučaju vrijednost varijable **x** dijeli sa 2, a vrijednost varijable **y** množi sa 3.

```
x = 100
y = 1.5
while (x>1) || (y<100)
    x = x / 2
    y = y * 3
end
```

Program unutar **while** petlje izvršio se 7 puta, odnosno vrijednost varijable **x** 7 se puta dijelila sa 2, a vrijednost varijable **y** 7 se puta množila sa 3. Prilikom petog prolaska kroz **while** petlju varijabla **y** nije bila manja od 100, ali je varijabla **x** bila veća od 1. Tek kada nijedan uvjet nije bio ispunjen, izvršavanje programa se prekida. Ovo je prikaz izvođenja gore navedenog programa.

```
x =  100
y =  1.5000
x =  50
y =  4.5000
x =  25
y =  13.500
x =  12.500
y =  40.500
x =  6.2500
y =  121.50
x =  3.1250
y =  364.50
x =  1.5625
y =  1093.5
x =  0.78125
y =  3280.5
>>
```

7.2 Naredbe odluke

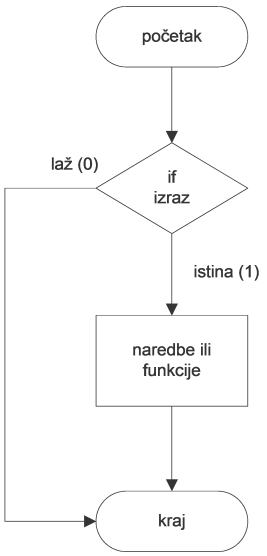
U naredbe odluke ubrajaju se **if-elseif-else-end** i **switch-case-otherwise-end**.

if

Prvi oblik:

```
if izraz
      naredbe ili funkcije
end
```

Izraz u **if** naredbi odluke može biti sastavljen od relacijskih i/ili logičkih operatora. Ako je taj uvjet ispunjen (poprima logičku vrijednost 1), izvršavaju se naredbe ili funkcije između **if** i **end**. Ako uvjet nije ispunjen, izlazi se iz tog dijela programa. Na slici 7.2 prikazan je dijagram toka za **if-end** naredbu odluke.

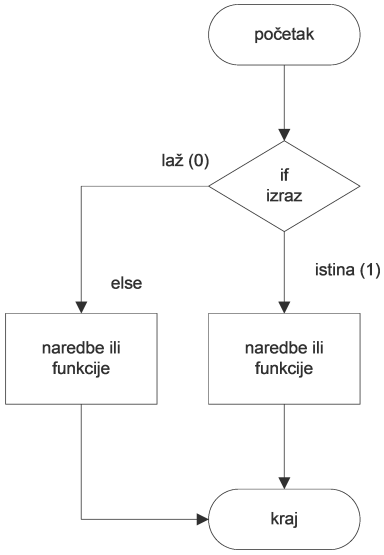


Slika 7.2 Dijagram toka za **`if-end`** naredbu odluke

Drugi oblik:

```
if izraz
      naredbe ili funkcije
else
      naredbe ili funkcije
end
```

Izraz u **`if`** naredbi odluke može biti sastavljen od relacijskih i/ili logičkih operatora. Ako je taj uvjet ispunjen (poprima logičku vrijednost 1), izvršavaju se naredbe ili funkcije između **`if`** i **`else`**. Ako uvjet nije ispunjen (poprima logičku vrijednost 0), izvršavaju se naredbe ili funkcije između **`else`** i **`end`**. Na slici 7.3 prikazan je dijagram toka za **`if-else-end`** naredbu odluke.



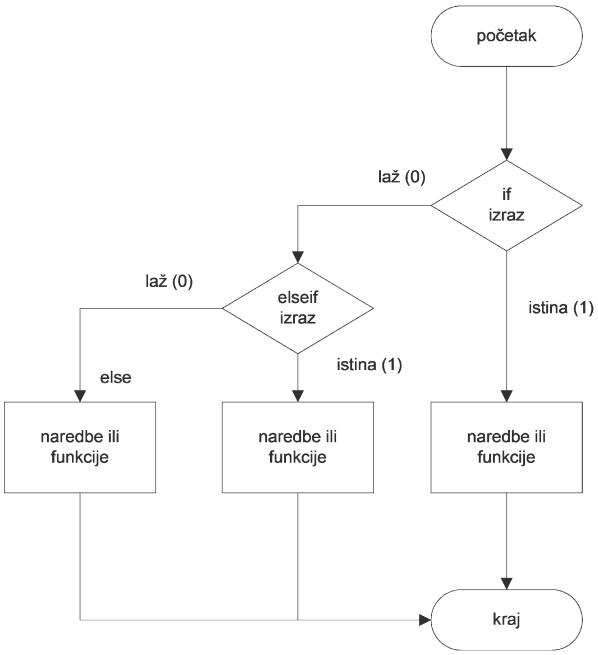
Slika 7.3 Dijagram toka za **`if-else-end`** naredbu odluke

Treći oblik:

```
if izraz
      naredbe ili funkcije
elseif izraz
      naredbe ili funkcije
elseif izraz
```

```
      naredbe ili funkcije
...
elseif izraz
      naredbe ili funkcije
else
      naredbe ili funkcije
end
```

Izraz u **`if`** naredbi odluke može biti sastavljen od relacijskih i/ili logičkih operatora. Ako je taj uvjet ispunjen (poprima logičku vrijednost 1), izvršavaju se naredbe ili funkcije između **`if`** i **`elseif`**. Ako uvjet nije ispunjen (poprima logičku vrijednost 0), ispituje se uvjet u **`elseif`** izrazu koji slijedi. Ako je taj uvjet ispunjen, izvršavaju se naredbe ili funkcije između tog **`elseif`** i sljedećeg **`elseif`**, a ako nije ispunjen, ispituje se uvjet u **`elseif`** izrazu koji slijedi, itd. do kraja. U slučaju da nijedan uvjet nije ispunjen, izvršavaju se naredbe ili funkcije između **`else`** i **`end`**. Na slici 7.4 prikazan je dijagram toka za **`if-elseif-else-end`** naredbu odluke.



Slika 7.4 Dijagram toka za **`if-elseif-else-end`** naredbu odluke

Primjer 7.6: Varijablama **`x`** i **`y`** dodjeljuje se slučajna realna vrijednost iz jednolike razdiobe u rasponu [0,100]. Zatim se ispituje pomoću **`if`** naredbe odluke je li vrijednost varijable **`x`** veća od **`y`**. Ako je ispunjen uvjet (logička vrijednost 1), ispisuje se "Broj x je veći od broja y.", a ako nije, izlazi se iz **`if`** naredbe odluke i ne ispisuje se ništa.

```
x = 100*rand
y = 100*rand
if x > y
    disp('Broj x je veci od broja y.')
end
```

Ovdje su prikazani rezultati izvršavanja programa za oba slučaja.

```
x =  96.944
y =  61.894
Broj x je veći od broja y.
>>
```

```
x =  26.206
y =  79.164
>>
```

Primjer 7.7: Varijabli **x** slučajno se dodjeljuje cjelobrojna vrijednost iz jednolike razdiobe u rasponu [0,100000]. Zatim se ispituje pomoću **if-else-end** naredbe odluke je li vrijednost varijable **x** parna ili neparna (to se radi pomoću funkcije **mod**). Ako je parna, ispisuje se "Broj x je paran.", a ako je neparna, ispisuje se "Broj x je neparan.".

```
x = randi([0 100000],1,1)
if mod(x,2) == 0
    disp('Broj x je paran.')
else
    disp('Broj x je neparan.')
end
```

Ovdje su prikazani rezultati izvršavanja programa za oba slučaja.

```
x =  48056
Broj x je paran.
>>
```

```
x =  60815
Broj x je neparan.
>>
```

Primjer 7.8: Varijabli **x** slučajno se dodjeljuje realna vrijednost iz jednolike razdiobe u rasponu [0,1]. Zatim se ispituje pomoću **if-else-end** naredbe odluke je li vrijednost varijable **x** manja od 0,5 ili je vrijednost varijable **x** veća od ili jednaka 0,5. Ako je vrijednost varijable **x** manja od 0,5, ispisuje se "pismo", a ako je veća od ili jednaka 0,5, ispisuje se "glava".

```
x = rand
if x < 0.5
    disp('pismo')
else
    disp('glava')
end
```

Ovdje su prikazani rezultati izvršavanja programa za oba slučaja.

```
x =  0.35459
pismo
>>
```

```
x =  0.62639
glava
>>
```

Primjer 7.9: Varijabli **x** slučajno se dodjeljuje realna vrijednosti iz jednolike razdiobe u rasponu [-1,1]. Zatim se pomoću **if-elseif-else-end** naredbe odluke ispituje je li broj **x** veći od 0,01 te ako je, ispisuje se "Broj x je pozitivan.". Ako nije ispunjen taj uvjet, ispituje se je li broj **x** manji od 0,01 i veći od -0,01 te ako je, ispisuje se "Broj x je približno jednak nuli.". Ako nije ispunjen ni prvi ni drugi uvjet, ispisuje se "Broj x je negativan.". To se događa u slučaju kad je broj **x** manji od -0,01.

```
x = 2*rand-1
if x > 0.1
    disp('Broj x je pozitivan.')
```

```
elseif (x>=-0.1) && (x<=0.1)
    disp('Broj x je približno jednak nuli.')
else
    disp('Broj x je negativan.')
end
```

Ovdje su prikazani rezultati izvršavanja programa za sva tri slučaja.

```
x =  0.14924
Broj x je pozitivan.
>>
```

```
x =  0.0013776
Broj x je približno jednak nuli.
>>
```

```
x =  -0.64727
Broj x je negativan.
>>
```

Primjer 7.10: Varijabli **x** slučajno se dodjeljuje cjelobrojna vrijednost iz jednolike razdiobe u rasponu [0,100] koja predstavlja bodove testa. Ispituje se je li vrijednost varijable **x** manja od 50 te ako je, varijabli **y** se dodjeljuje vrijednost 1. Varijabla **y** predstavlja ocjenu testa. Ako je vrijednost varijable **x** veća od ili jednaka 50, a manja od 60, varijabli **y** dodjeljuje se vrijednost 2. Ako je vrijednost varijable **x** veća od ili jednaka 60, a manja od 70, varijabli **y** dodjeljuje se vrijednost 3. Ako je vrijednost varijable **x** veća od ili jednaka 70, a manja od 80, varijabli **y** dodjeljuje se vrijednost 4. Ako je vrijednost varijable **x** veća od ili jednaka 80, varijabli **y** dodjeljuje se vrijednost 5.

```
x = randi([0 100],1,1)
if x < 50
    y = 1
elseif x <= 60
    y = 2
elseif x <= 70
    y = 3
elseif x <= 80
    y = 4
else
    y = 5
end
```

Ovdje su prikazani rezultati izvršavanja programa za svih pet slučajeva.

```
x =  13
y =  1
>>
```

```
x =  57
y =  2
>>
```

```
x =  68
y =  3
>>
```

```
x =  74
y =  4
>>
```

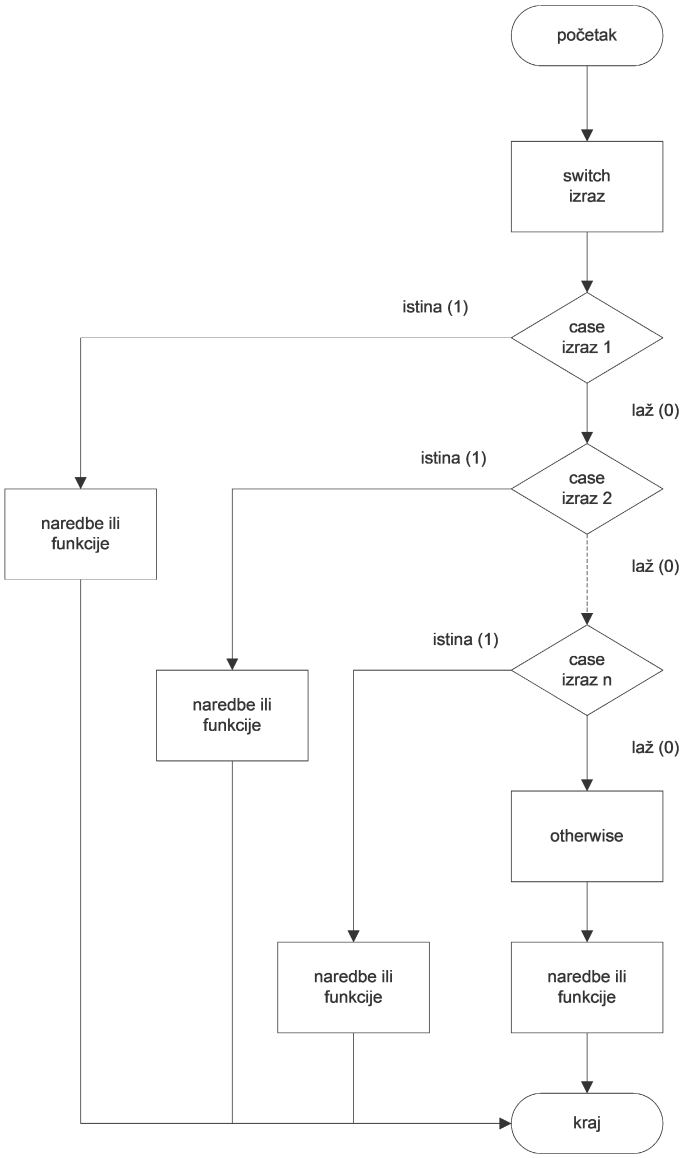
```
x =  86
y =  5
>>
```

switch

Naredba odluke **`switch-case-otherwise-end`** ima sljedeću sintaksu:

```
switch izraz
       case izraz 1
              naredbe ili funkcije
       case izraz 2
              naredbe ili funkcije
       ...
       otherwise
              naredbe ili funkcije
end
```

Na temelju vrijednosti izraza u dijelu **`switch`**, dijelovi **`case`** ispituju odgovara li vrijednost njihovim izrazima te ako da, izvršavaju se naredbe ili funkcije neposredno iza **`case`**. U izrazima iza **`case`** naredbe može biti i više vrijednosti koje su navedene u vitičastim zagradama, npr. **`{1, 5, 8}`**. U izrazima kod **`switch`** i **`case`** naredbi mogu se koristiti i znakovni nizovi (eng. *string*). Na slici 7.5 prikazan je dijagram toka za **`switch-case-otherwise-end`** naredbu odluke.



Slika 7.5 Dijagram toka za **switch-case-otherwise-end** naredbu odluke

Primjer 7.11: Varijabli **y** slučajno se dodjeljuje cjelobrojna vrijednost iz jednolike razdiobe u rasponu [1,5]. Zatim se u dijelu **switch** postavlja varijabla **y** kao uvjet ispitivanja. Slijede **case** dijelovi koji ispituju je li vrijednost varijable **y** 1 ili 2 ili 3 ili 4 ili 5, te se ispisuju odgovarajuće poruke.

```
y = randint(1,1,[1 5])
switch y
    case 1
        disp('Ocjena je nedovoljan.')
    case 2
        disp('Ocjena je dovoljan.')
    case 3
        disp('Ocjena je dobar.')
    case 4
        disp('Ocjena je vrlo dobar.')
    otherwise
        disp('Ocjena je izvrstan.')
end
```

Ovdje su prikazani rezultati izvršavanja programa za svih pet slučajeva.

```
y =  1
Ocjena je nedovoljan.
>>
```

```
y =  2
Ocjena je dovoljan.
>>
```

```
y =  3
Ocjena je dobar.
>>
```

```
y =  4
Ocjena je vrlo dobar.
>>
```

```
y =  5
Ocjena je izvrstan.
>>
```

Primjer 7.12: Varijabli **x** dodjeljuje se cjelobrojna vrijednost iz jednolike razdiobe u rasponu [0,10]. Zatim se u dijelu **switch** postavlja varijabla **x** kao uvjet ispitivanja. Slijede **case** dijelovi koji ispituju je li varijabla **x** jednaka 1 ili 2 ili 3, zatim **case** dio koji ispituje je li varijabla **x** jednaka 4 ili 5 ili 6, zatim **case** dio koji ispituje je li varijabla **x** jednaka 7 ili 8, zatim **case** dio koji ispituje je li varijabla **x** jednaka 9 ili 10, te ako ništa od gore navedenog nije ispunjeno, slijedi **otherwise** dio. Ovdje je potrebno uočiti da se u svim **case** dijelovima ispituje više uvjeta koji su navedeni u vitičastim zagradama i odvojeni zarezima.

```
x = randint(1,1,[0 10])
switch x
    case {1,2,3}
        disp('Broj x je 1 ili 2 ili 3.')
        y = 2*x
    case {4,5,6}
        disp('Broj x je 4 ili 5 ili 6.')
        y = 3*x
    case {7,8}
        disp('Broj x je 7 ili 8.')
        y = 4*x
```

```
    case {9,10}
        disp('Broj x je 9 ili 10.')
        y = 5*x
    otherwise
        disp('Broj x je 0.')
end
```

Ovdje su prikazani rezultati izvršavanja programa za četiri slučaja.

```
x =  1
Broj x je 1 ili 2 ili 3.
>>

x =  5
Broj x je 4 ili 5 ili 6.
>>

x =  8
Broj x je 7 ili 8.
>>

x =  9
Broj x je 9 ili 10.
>>
```

Primjer 7.13: Varijabli **x** dodjeljuje se znakovni niz **'breskva'**. Zatim se u dijelu **switch** postavlja znakovna varijabla **x** kao uvjet ispitivanja. Slijede **case** dijelovi koji ispituju je li znakovna varijabla **x** jednaka **'jabuka'** ili **'kruska'** ili **'sljiva'** ili ništa od toga, te se ispisuju odgovarajuće poruke.

```
x = 'breskva'
switch x
    case 'jabuka'
        disp('Izabrano voce je jabuka.')
    case 'kruska'
        disp('Izabrano voce je kruska.')
    case 'sljiva'
        disp('Izabrano voce je sljiva.')
    otherwise
        disp('Ne znam koje je voce izabrano.')
end
```

Ovdje je prikazan rezultat izvršavanja programa.

```
x = breskva
Ne znam koje je voce izabrano.
>>
```

break

Naredba **break** prekida izvođenje programa, odnosno izlazi iz petlje. Ako postoji više ugniježđenih petlji, ona predaje nadzor prvoj vanjskoj (višoj) petlji.

Primjer 7.14: Postoje tri **for** petlje. Jedna s brojačem **p** koji ide od 7 do 8 i korakom 1, druga s brojačem **q** koji ide od 3 do 5 i korakom 1, te treća s brojačem **r** koji ide od 1 do 2 i korakom 1. U drugoj petlji nalazi se **if** naredba odluke koja ispituje je li brojač **q** jednak 4. Ako je, izlazi iz petlje i predaje kontrolu petlji s brojačem **p**.

```
% Break primjer
for p = 7:8
    for q = 3:5
        if q == 4
            break
        end
        for r = 1:2
            fprintf('  %3.0f,  %3.0f,  %3.0f\n',p,q,r)
        end
    end
end
```

Podrobnije objašnjenje prikaza formata broja može se vidjeti naredbom **`help fprintf`**.

U nastavku je prikazan rezultat izvršavanja gornjeg programa.

```
    7,    3,    1
    7,    3,    2
    8,    3,    1
    8,    3,    2
>>
```

Primjer 7.15: Traži se nultočka polinoma trećeg stupnja pomoću **`while`** naredbe ponavljanja metodom bisekcije. Program će završiti ako je razlika između dvije točke na krivulji manja od 0,001, ili će izaći iz **`while`** naredbe ponavljanja ako je vrijednost funkcije (polinoma) između -0,01 i 0,01. Taj zadnji uvjet postiže se pomoću naredbe **`break`**.

```
a = 0;
fa = a^3-2*a-5;
b = 3;
fb = b^3-2*b-5;
while (b-a) > 0.001
    x = (a+b)/2
    fx = x^3-2*x-5
    if (fx>-0.01) && (fx<0.01)
        disp('prekid while-end petlje')
        break
    elseif sign(fx) == sign(fa)
        a = x;
        fa = fx;
    else
        b = x;
        fb = fx;
    end
end
```

```
x =  1.5000
fx = -4.6250
x =  2.2500
fx =  1.8906
x =  1.8750
fx = -2.1582
x =  2.0625
fx = -0.35132
x =  2.1562
fx =  0.71280
x =  2.1094
fx =  0.16684
x =  2.0859
fx = -0.095679
x =  2.0977
fx =  0.034714
x =  2.0918
```

```
fx = -0.030698
x =  2.0947
fx =  0.0019543
prekid while-end petlje
>>
```

continue

Naredba **`continue`** prelazi na sljedeću iteraciju petlje.

Primjer 7.16: Postoje tri **`for`** petlje. Jedna s brojačem **`p`** koji ide od 7 do 8 i korakom 1, druga s brojačem **`q`** koji ide od 3 do 5 i korakom 1, te treća s brojačem **`r`** koji ide od 1 do 2 i korakom 1. To je isti primjer koji se koristio kod **`break`** naredbe, samo se umjesto te naredbe koristi naredba **`continue`**. U drugoj se petlji nalazi **`if`** naredba odluke koja ispituje je li brojač **`q`** jednak 4. Ako je brojač **`q`** jednak 4, preskače se brojač **`q`** koji je jednak 4 i ide se na sljedeću vrijednost brojača, u ovom slučaju 5.

```
% Continue primjer
for p = 7:8
    for q = 3:5
        if q == 4
            continue
        end
        for r = 1:2
            fprintf('  %3.0f,  %3.0f,  %3.0f\n',p,q,r)
        end
    end
end
```

U nastavku je prikazan rezultat izvršavanja gornjeg programa.

```
    7,    3,    1
    7,    3,    2
    7,    5,    1
    7,    5,    2
    8,    3,    1
    8,    3,    2
    8,    5,    1
    8,    5,    2
>>
```

7.3 Vektorizacija programa

U Octaveu je ponekad moguće izbjeći korištenje naredbi odluke i ponavljanja pomoću tzv. vektorizacije programa. Vektorizacija programa znači da se matrice i vektori koriste umjesto **`for`** petlji ili **`if`** naredbi odluka zato što se na taj način programi brže izvršavaju. Koristi se mogućnost indeksiranja elemenata matrice ili vektora pomoću drugih matrica ili vektora koji se nazivaju maske.

Primjer 7.17: Neka postoje dva vektora **`x`** i **`y`**. Vrijednosti elemenata vektora **`y`** koristit će se za indeksiranje elemenata vektora **`x`** kako bi se dobio vektor **`z`**. Vrijednosti elemenata vektora **`y`** su 9, 4 i 6, što znači da su odabrani 9. element vektora **`x`** koji iznosi 26, 4. element vektora **`x`** koji iznosi 11 te 6. element vektora **`x`** koji iznosi 17.

```
>> x = [2 5 8 11 14 17 20 23 26 29]
x =
     2     5     8    11    14    17    20    23    26    29
>> y = [9 4 6]
y =
     9     4     6
>> z = x(y)
```

```
z =
    26    11    17
>>
```

Primjer 7.18: Definiran je vektor **x**, a zatim je definiran vektor **y** pomoću relacijskog operatora veće od **>**. Vektor **y** za sve elemente u vektoru **x** koji su veći od 6 poprima logičku vrijednost 1, a za ostale logičku vrijednost 0. Zatim se vektor **y** koristi kao maska za dobivanje svih elemenata vektora **x** koji su veći od 6. Potrebno je obratiti pozornost na to da je vektor **y** logička varijabla.

```
>> x = 1:10
x =
     1     2     3     4     5     6     7     8     9    10
>> y = x > 6
y =
     0     0     0     0     0     0     1     1     1     1
>> x(y)
ans =
     7     8     9    10
>>
```

Primjer 7.19: Neka se želi u nekim podatcima, koji su predstavljeni matricom **x**, sve podatke koji su veći od nule postaviti u vrijednost 0. To se može napraviti pomoću **for** naredbe ponavljanja i **if** naredbe odluke na sljedeći način. Prvo se stvori matrica x dimenzija 10*10 sa slučajnim cjelobrojnim vrijednostima iz jednolike razdiobe u rasponu [-5,5], a zatim se pomoću **for** i **if** naredbi ispituje koji su elementi matrice x veći od 0 i ti se elementi izjednače sa 0.

```
x = randi([-5 5],10,10)
for i = 1:10
    for j = 1:10
        if x(i,j) > 0
            x(i,j) = 0;
        end
    end
end
x
```

Rezultat izvršavanja gornjeg programa prikazan je u nastavku.

```
x =
    2    0    2    3   -2    5   -4   -5    5   -3
   -5    2   -1    3   -5   -4   -2   -3    2   -4
    1    4    3    1   -2    1    0    5    2   -5
   -5    4   -5    1   -1    3   -2   -2    3    2
   -2   -1    5    2   -4    0    5    3    1   -2
   -2   -1    0    1   -4    3   -3    3    4    5
    3    5    5   -5    4   -3   -2    2   -2   -5
    0   -1    1    2    4   -5    1   -5    5    1
   -3    1   -5   -1   -2    1    5   -5   -4   -1
   -4   -1    0    5    5   -2    0    0    2    0
x =
    0    0    0    0   -2    0   -4   -5    0   -3
   -5    0   -1    0   -5   -4   -2   -3    0   -4
    0    0    0    0   -2    0    0    0    0   -5
   -5    0   -5    0   -1    0   -2   -2    0    0
   -2   -1    0    0   -4    0    0    0    0   -2
   -2   -1    0    0   -4    0   -3    0    0    0
    0    0    0   -5    0   -3   -2    0   -2   -5
    0   -1    0    0    0   -5    0   -5    0    0
   -3    0   -5   -1   -2    0    0   -5   -4   -1
   -4   -1    0    0    0   -2    0    0    0    0
>>
```

To se može jednostavnije i brže izvesti ovako:

```
>> x = randi([-5 5],10,10)
x =
    0    4    5   -5   -4    2   -2   -4   -5    2
   -2   -5    0    5    0    1    0   -2   -1    4
    0   -1   -5   -2   -5   -1   -1    3   -4    4
    0    1   -5    5    3    5    0   -3    0   -5
   -5   -3    5    2   -3    3   -2   -2   -1    2
   -4    2   -2    2   -1    2   -2    0    0    5
   -5   -3    1   -1   -2    4   -4   -2   -4   -3
    0    1   -2    1    0    5   -1   -2    2    3
   -5    4    5   -1   -3    0   -5   -1    0    0
    2   -2    1    2   -4   -4    0   -5    5   -1
>> x(x>0) = 0
x =
    0    0    0   -5   -4    0   -2   -4   -5    0
   -2   -5    0    0    0    0    0   -2   -1    0
    0   -1   -5   -2   -5   -1   -1    0   -4    0
    0    0   -5    0    0    0    0   -3    0   -5
   -5   -3    0    0   -3    0   -2   -2   -1    0
   -4    0   -2    0   -1    0   -2    0    0    0
   -5   -3    0   -1   -2    0   -4   -2   -4   -3
    0    0   -2    0    0    0   -1   -2    0    0
   -5    0    0   -1   -3    0   -5   -1    0    0
    0   -2    0    0   -4   -4    0   -5    0   -1
>>
```

Primjer 7.20: Neka se želi izračunati korijen iz zbroja kvadrata brojeva koji predstavljaju indekse elemenata matrice dimenzija 10*10.To se može učiniti pomoću **`for`** naredbe ponavljanja.

```
for i = 1:10
    for j = 1:10
        r(i,j) = sqrt(i^2 + j^2);
    end
end
r
```

Rezultat izvršavanja gornjeg programa prikazan je u nastavku.

```
r =
    1.41    2.24    3.16    4.12    5.10    6.08    7.07    8.06    9.06
10.05
    2.24    2.83    3.61    4.47    5.39    6.32    7.28    8.25    9.22
10.20
    3.16    3.61    4.24    5.00    5.83    6.71    7.62    8.54    9.49
10.44
    4.12    4.47    5.00    5.66    6.40    7.21    8.06    8.94    9.85
10.77
    5.10    5.39    5.83    6.40    7.07    7.81    8.60    9.43   10.30
11.18
    6.08    6.32    6.71    7.21    7.81    8.49    9.22   10.00   10.82
11.66
    7.07    7.28    7.62    8.06    8.60    9.22    9.90   10.63   11.40
12.21
    8.06    8.25    8.54    8.94    9.43   10.00   10.63   11.31   12.04
12.81
    9.06    9.22    9.49    9.85   10.30   10.82   11.40   12.04   12.73
13.45
   10.05   10.20   10.44   10.77   11.18   11.66   12.21   12.81   13.45
14.14
>>
```

To se može jednostavnije i brže izvesti ovako:

```
>> [i,j] = meshgrid(1:10,1:10)
i =
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
     1.00      2.00      3.00      4.00      5.00      6.00      7.00      8.00      9.00
10.00
j =
     1.00      1.00      1.00      1.00      1.00      1.00      1.00      1.00      1.00
1.00
     2.00      2.00      2.00      2.00      2.00      2.00      2.00      2.00      2.00
2.00
     3.00      3.00      3.00      3.00      3.00      3.00      3.00      3.00      3.00
3.00
     4.00      4.00      4.00      4.00      4.00      4.00      4.00      4.00      4.00
4.00
     5.00      5.00      5.00      5.00      5.00      5.00      5.00      5.00      5.00
5.00
     6.00      6.00      6.00      6.00      6.00      6.00      6.00      6.00      6.00
6.00
     7.00      7.00      7.00      7.00      7.00      7.00      7.00      7.00      7.00
7.00
     8.00      8.00      8.00      8.00      8.00      8.00      8.00      8.00      8.00
8.00
     9.00      9.00      9.00      9.00      9.00      9.00      9.00      9.00      9.00
9.00
    10.00     10.00     10.00     10.00     10.00     10.00     10.00     10.00     10.00
10.00
>> r = sqrt(i.^2 + j.^2)
r =
     1.41      2.24      3.16      4.12      5.10      6.08      7.07      8.06      9.06
10.05
     2.24      2.83      3.61      4.47      5.39      6.32      7.28      8.25      9.22
10.20
     3.16      3.61      4.24      5.00      5.83      6.71      7.62      8.54      9.49
10.44
     4.12      4.47      5.00      5.66      6.40      7.21      8.06      8.94      9.85
10.77
     5.10      5.39      5.83      6.40      7.07      7.81      8.60      9.43     10.30
11.18
     6.08      6.32      6.71      7.21      7.81      8.49      9.22     10.00     10.82
11.66
     7.07      7.28      7.62      8.06      8.60      9.22      9.90     10.63     11.40
12.21
     8.06      8.25      8.54      8.94      9.43     10.00     10.63     11.31     12.04
12.81
     9.06      9.22      9.49      9.85     10.30     10.82     11.40     12.04     12.73
13.45
    10.05     10.20     10.44     10.77     11.18     11.66     12.21     12.81     13.45
14.14
>>
```

Primjer 7.21: Neka postoji redni vektor **x** dimenzija 1*10 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [0,10]. Potrebno je za sve elemente rednog vektora **x**, čija je vrijednost veća od 3, oduzeti broj 3. To je moguće postići pomoću **for** i **if** naredbi ovako:

```
x = randi([0 10],1,10)
for i = 1:10
    if x(i) > 3
        x(i) = x(i)-3;
    end
end
x
```

Rezultat izvršavanja gornjeg programa prikazan je u nastavku.

```
x =
   1   0   5   1   3   8   8   1   1   6
x =
   1   0   2   1   3   5   5   1   1   3
>>
```

To se može jednostavnije i brže izvesti ovako:

```
>> x = randi([0 10],1,10)
x =
    4    5   10    4   10   10    7    1    9    1
>> x(x>3) = x(x>3)-3
x =
   1   2   7   1   7   7   4   1   6   1
>>
```

Primjer 7.22: Neka postoji matrica **x** dimenzija 10*10 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [-20,20]. Neka je potrebno svim elementima matrice **x** koji su veći od 10 postaviti vrijednost na 10 i svim elementima matrice **x** koji su manji od -10 postaviti vrijednost na -10. To je moguće postići pomoću **for** i **if** naredbi ovako:

```
x = randi([-20 20],10,10)
for i = 1:10
    for j = 1:10
        if x(i,j) > 10
            x(i,j) = 10;
        elseif x(i,j) < -10
            x(i,j) = -10;
        end
    end
end
x
```

Rezultat izvršavanja gornjeg programa prikazan je u nastavku.

```
x =
   20  -13   16   -4   -8    9    8  -12   18   -2
  -16   20   20  -10   18  -13   14    2    7   13
    5  -11  -18  -12  -12    3   12   -4    2    1
   19    3  -10    3  -19    0  -10   10   13    4
  -15  -15   14  -15   20    3   15   10   19   -7
  -18  -20   11  -20    2    6   12  -19   17   -2
   17   12   -5    7   19    5   -4   -9   -8    3
   -1  -17    8    8   12    9   14   18  -16    0
```

```
    13    15    -1    16   -18    17     4    17    10   -17
    -1    10   -13    20     6    -6   -12   -10     8   -16
x =
    10   -10    10    -4    -8     9     8   -10    10    -2
   -10    10    10   -10    10   -10    10     2     7    10
     5   -10   -10   -10   -10     3    10    -4     2     1
    10     3   -10     3   -10     0   -10    10    10     4
   -10   -10    10   -10    10     3    10    10    10    -7
   -10   -10    10   -10     2     6    10   -10    10    -2
    10    10    -5     7    10     5    -4    -9    -8     3
    -1   -10     8     8    10     9    10    10   -10     0
    10    10    -1    10   -10    10     4    10    10   -10
    -1    10   -10    10     6    -6   -10   -10     8   -10
>>
```

To se može jednostavnije i brže izvesti ovako:

```
>> x = randi([-20 20],10,10)
x =
     8    12    16     3     3    19    -2     6     8    12
     4    20    14     5     0    10    19    -7    18    19
     5   -13    13    -8   -12     8     1     9   -19    -5
   -18    -7    15    10    -1    10    12    10    -9     2
    12   -11    18   -12     0   -12    -7    10     8     3
     9    -5    17     5     5    -2   -19    16    10    19
    -5    13    -2   -11   -10     8     3   -16     5    -7
    -8    15     6    -4   -12    -3    15    14    14    17
   -10   -11    -7    11   -12    10    -1   -19   -18    -3
     4    19     8   -18   -15    -9     4     6    -4    -3
>> x = min(max(x,-10),10)
x =
     8    10    10     3     3    10    -2     6     8    10
     4    10    10     5     0    10    10    -7    10    10
     5   -10    10    -8   -10     8     1     9   -10    -5
   -10    -7    10    10    -1    10    10    10    -9     2
    10   -10    10   -10     0   -10    -7    10     8     3
     9    -5    10     5     5    -2   -10    10    10    10
    -5    10    -2   -10   -10     8     3   -10     5    -7
    -8    10     6    -4   -10    -3    10    10    10    10
   -10   -10    -7    10   -10    10    -1   -10   -10    -3
     4    10     8   -10   -10    -9     4     6    -4    -3
>>
```

Pitanja za provjeru znanja:

1. Koje su naredbe ponavljanja u Octaveu?
2. Je li broj ponavljanja dijela programa unutar **`while`** petlje unaprijed poznat?
3. Je li broj ponavljanja dijela programa unutar **`for`** petlje unaprijed poznat?
4. Kada se prekida izvršavanje **`while`** petlje?
5. Koje su naredbe odluka u Octaveu?
6. Kada se prekida izvršavanje naredbi ili funkcija između **`if`** i **`end`**?
7. Čemu služi naredba odluka **`if-else-end`**, odnosno **`if-elseif-else-end`**?
8. Čemu služi naredba **`break`**?
9. Čemu služi naredba **`continue`**?
10. Čemu služi vektorizacija programa u Octaveu i koje su njene prednosti?

8. VRSTE FUNKCIJA

Funkcije su važan dio svakog programskog jezika. Bitno olakšavaju izradu programa, povećavaju njegovu pouzdanost i olakšavaju održavanje. Funkcije omogućuju da se program podijeli u manje neovisne cjeline koje se mogu pisati i provjeravati neovisno jedna od druge. Manje je cjeline puno lakše napisati, provjeriti i mijenjati od velikih i složenih programa jer na umu treba imati samo relativno jednostavan zadatak koji ta cjelina treba obaviti. Takve se cjeline onda mogu rabiti u različitim programima, ako je to potrebno, ili višekratno u istom programu. Složeni se programi tako mogu sastavljati u velikoj mjeri od gotovih dijelova (funkcija) koje je napisao i provjerio netko drugi, slično kao što se složeni strojevi sastavljaju od gotovih i provjerenih dijelova (sklopova).

U Octaveu se funkcije mogu podijeliti u tri skupine: lokalne funkcije, primarne korisničke funkcije i ugrađene funkcije.

Glavna je namjena lokalnih funkcija stvaranje funkcija bez pisanja i pohrane funkcijske M-datoteke. To je praktično pri izradi jednostavnijih programa izravno iz naredbenog prozora ili pri pisanju skripti. Lokalne su funkcije ograničene primjene s obzirom na činjenicu da su vidljive samo unutar skripte u kojoj se nalaze ili unutar trenutačno aktivnog radnog prostora. Glavna je namjena lokalnih funkcija izrada jednostavnih priručnih funkcija, bez potrebe stvaranja funkcijskih M-datoteka.

Primarne korisničke funkcije piše korisnik prema svojim potrebama. Takve funkcije korisnik sprema u funkcijske M-datoteke i može ih rabiti u bilo kojem programu. Mogu ih rabiti i drugi kojima ih korisnik stavi na raspolaganje. Primarne korisničke funkcije temeljni su gradbeni blokovi svakog složenijeg programa.

Ugrađene se funkcije isporučuju uz Octave program. To su funkcije koje su napisali i provjerili programeri Octavea i na raspolaganju su korisnicima. Ugrađenim je funkcijama moguće riješiti zadatke koji se javljaju u mnogim programima pa se može reći da se ugrađenim funkcijama proširuju osnovne mogućnosti Octavea.

8.1 Lokalne funkcije

Pretpostavimo da unutar skripte treba na mnogo mjesta izračunavati istu vrijednost (računati isti izraz ili isti odsječak programa). To se može učiniti tako da se pri svakom izračunu u skripti ponovi cjelokupan izraz ili odsječak programa. Postupak se može pojednostavniti uporabom lokanih funkcija (engl. *inline function*).

Za lokalnu funkciju vrijedi:

- Matematički izraz može imati jedan ili više argumenata (ulaznih varijabli).
- Matematički izraz može sadržavati sve ugrađene Octaveove funkcije i funkcije koje je definirao korisnik (primarne korisničke funkcije).
- Izraz mora biti napisan u skladu s dimenzijama argumenta (izvršavanje nad pojedinim elementima ili po pravilima linearne algebre).
- Lokalna funkcija poziva se upisivanjem njezina imena i vrijednosti njezinih argumenata.
- Lokalna funkcija može se rabiti kao argument drugih funkcija.
- Lokalna funkcija postoji samo u radnom prostoru (engl. *base workspace*) te se izlaskom iz Octave programa gubi.

inline

Lokalne funkcije definirane pomoću naredbe **`inline`** imaju sljedeću sintaksu:

```
function_name = inline('math expresion')
```

ili

```
function_name = inline('math expresion ','var1','var2','var3',…,'varn')
```

Primjer 8.1: Definirana je lokalna funkcija **`y1`** naredbom **`inline`**. Prikazano je nekoliko načina pozivanja lokalne funkcije za različite argumente. Najprije se poziva lokalna funkcija **`y1`** samo za jedan argument kao **`y1(2)`**. Izračunava se vrijednost lokalne funkcije za vrijednost 2. Zatim se poziva lokalna funkcija **`y1`** za tri argumenta kao **`y1([-2 0 1])`**, a rezultat je vrijednost lokalne funkcije za vrijednosti -2, 0 i 1. Zatim se definira varijabla **`x`** od -1 do 1 s korakom 0,2. Lokalna funkcija poziva se kao **`y1(x)`** i izračunava za sve vrijednosti varijable **`x`**.

```
>> y1 = inline('2*x.^2+3*x-5')
y1 = f(x) = 2*x.^2+3*x-5
>> y1(2)
ans =  9
>> y1([-2 0 1])
ans =
    -3    -5     0

>> x = -1:0.2:1
x =
   -1.00    -0.80    -0.60    -0.40    -0.20     0.00     0.20     0.40     0.60
0.80     1.00

>> y1(x)
ans =
   -6.00    -6.12    -6.08    -5.88    -5.52    -5.00    -4.32    -3.48    -2.48
-1.32    0.00
```

Primjer 8.2: Definirana je lokalna funkcija **`z`** naredbom **`inline`**. U prvom pozivanju rezultatu lokalne funkcije **`z`** dodjeljuje se varijabla **`z1`** za argumente **`x=2`** i **`y=3`**. U drugom pozivanju rezultatu lokalne funkcije **`z`** dodjeljuje se varijabla **`z2`** za argumente **`x=3`** i **`y=2`**.

```
>> z = inline('x.^2-y.^2')
z = f(x, y) = x.^2-y.^2
>> z1 = z(2,3)
z1 = -5.00
>> z2 = z(3,2)
z2 =  5.00
```

U sljedeća dva primjera bit će prikazan oblik definiranja lokalne funkcije kod kojeg je važan poredak unosa argumenata kod pozivanja lokalne funkcije.

Primjer 8.3: Definirana je lokalna funkcija **`z`** sa dva ulazna argumenta **`x`** i **`y`** naredbom **`inline`**. U naredbi **`inline`** definirano je da prvi argument prilikom pozivanja lokalne funkcije mora biti **`x`**, a drugi **`y`**. Prikazana su i dva poziva lokalne funkcije **`z`**, gdje je u prvom slučaju lokalnoj funkciji **`z`** dodijeljena varijabla **`z1`** za argumente **`x=0.1`** i **`y=0.5`**, a u drugom je slučaju lokalnoj funkciji **`z`** dodijeljena varijabla **`z2`** za argumente **`x=0.2`** i **`y=0.3`**.

```
>> z = inline('sin(x)+cos(y)','x','y')
z = f(x, y) = sin(x)+cos(y)
>> z1 = z(0.1,0.5)
z1 = 0.98
>> z2 = z(0.2,0.3)
z2 =  1.15
```

Primjer 8.4: Definirana je lokalna funkcija **z** sa dva ulazna argumenta **x** i **y** naredbom **inline**. U naredbi **inline** definirano je da prvi argument prilikom pozivanja mora biti **y**, a drugi **x** (obratno nego u prethodnom primjeru, a lokalna funkcija je ista). Prikazana su i dva poziva lokalne funkcije **z**, gdje je u prvom vrijednosti lokalne funkcije **z** dodijeljena varijabla **z1** za argumente **y=0.1** i **x=0.5**, a u drugom je vrijednosti lokalne funkcije **z** dodijeljena varijabla **z2** za argumente **y=0.2** i **x=0.3**. Iako je na prvi pogled u ova dva primjera sve isto, rezultati su različiti jer lokalna funkcija drukčije dodjeljuje ulazne argumente.

```
>> z = inline('sin(x)+cos(y)','y','x')
z = f(y, x) = sin(x)+cos(y)
>> z1 = z(0.1,0.5)
z1 =  1.47
>> z2 = z(0.2,0.3)
z2 =  1.28
```

Kod novijih se inačica Octave programa preporučuje rabiti anonimne funkcije umjesto **inline** funkcija.

argnames

Argumenti **inline** funkcije mogu se prikazati pomoću naredbe **argnames**, npr.:

```
>> z = inline('sin(x)+cos(y)','x','y')
z = f(x, y) = sin(x)+cos(y)
>> argnames(z)
ans =
{
  [1,1] = x
  [2,1] = y
}
>>
```

formula

Izraz **inline** funkcije može se prikazati pomoću naredbe **formula**, npr.:

```
>> formula(z)
ans = sin(x)+cos(y)
>>
```

8.2 Anonimne funkcije

Lokalne funkcije definirane pomoću anonimnih funkcija (engl. *anonymous functions*) imaju sljedeću sintaksu:

```
fhandle = @(arglist) expr
```

Anonimna se zove jer s desne strane nema imena, već je definirana znakom **@** i svojim argumentima (u gornjem primjeru samo jednim argumentom **x**). Tijelo funkcije dio je koji se nastavlja iza desne okrugle zagrade i praznine. S lijeve strane znaka jednakosti pokazivač je funkcije (engl. *function handle*). Pokazivač funkcije omogućuje indirektno pozivanje funkcije, kako je opisano u nastavku. Primjer anonimne funkcije:

```
>> sqrequ = @(x) 5*x^2 + 2*x - 4
sqrequ =
@(x) 5 * x ^ 2 + 2 * x - 4
>>
```

Nakon definiranja anonimne funkcije (u gornjem primjeru **sqrequ**), u radnom prostoru postoji funkcija koja može rabiti varijable radnog prostora ili lokalne varijable funkcije koja je poziva, npr.:

```
>> x = 2
x =  2.00
>> sqrequ(x)
ans =  20.00
>>
```

Najprije je u radnom prostoru definirana varijabla **x** i pripisana joj vrijednost 2, a zatim je pozvana anonimna funkcija **sqrequ** te joj kao argument predana varijabla **x**.

Anonimne funkcije mogu ponekad zamijeniti globalne varijable, tj. varijable koje trebaju biti dostupne (vidljive) u radnom prostoru i funkciji.

Ako anonimna funkcija nema argumenata unutar zagrada, iza znaka `@` se ostavlja prazno. Obvezno se moraju pisati zagrade!

```
>> t = @() datestr(now);
```

Pri pozivu anonimne funkcije bez argumenata također treba napisati zagrade:

```
>> t()
ans = 16-Oct-2019 12:04:54
>>
```

Važno svojstvo anonimnih funkcija je mogućnost da se one rabe kao argumenti drugih funkcija. Npr. anonimna funkcija:

```
>> sqr = @(x) x.^2;
```

može se rabiti, npr. kao argument funkcije **quad**, koja računa integral funkcije **sqr**:

```
>> quad(sqr,0,1)
ans = 0.33
>>
```

func2str

Anonimna funkcija može se prikazati naredbom **func2str**, npr.:

```
>> func2str(sqrequ)
ans = @(x) 5 * x ^ 2 + 2 * x - 4
>>
```

Osim anonimnih funkcija, naredba **func2str** prikazat će bilo koju funkciju čiji se pokazivač navede kao argument naredbe **func2str**, uključujući i ugrađene funkcije, npr.:

```
>> x = @sin
x = @sin
>> func2str(x)
ans = sin
>>
```

functions

Podrobniji prikaz anonimne funkcije može se dobiti naredbom **`functions`**, npr.:

```
>> functions(sqrequ)
ans =
  scalar structure containing the fields:
    function = @(x) 5 * x ^ 2 + 2 * x - 4
    type = anonymous
    file =
>>
```

Osim anonimnih funkcija, naredba **`functions`** prikazat će podroban prikaz bilo koje funkcije čiji se pokazivač navede kao argument naredbe **`functions`**, uključujući i ugrađene funkcije, npr.:

```
>> x = @sin
x = @sin
>> functions(x)
ans =
  scalar structure containing the fields:
    function = sin
    type = simple
    file =
>>
```

Lokalne funkcije mogu se pisati i unutar korisničke funkcije. U tom slučaju dostupne (vidljive) su joj sve lokalne varijable funkcije unutar koje se anonimna funkcija nalazi.

Primjer **`inline`** lokalne funkcije unutar primarne korisničke funkcije:

```
function y = cinline(r)
sq = inline('r^2','r');
y = sq(r)*pi;
```

Primjer anonimne lokalne funkcije unutar primarne korisničke funkcije:

```
function y = canon(r)
sq = @(r) r^2;
y = sq(r)*pi;
```

Umjesto lokalnih funkcija unutar primarnih korisničkih funkcija mogu se rabiti podfunkcije.

8.3 Funkcije

Funkcije su M-datoteke koje počinju definicijom funkcije (engl. *function*). Takve se funkcije nazivaju i primarne funkcije. Funkcija koju definira korisnik (engl. *user-defined function*), ili primarna korisnička funkcija, Octaveov je program koji je korisnik napisao i pohranio u M-datoteku, a naziva se i funkcijska datoteka (engl. *function file*). U nastavku će se korisničke funkcije nazivati jednostavno funkcije.

Za razliku od skripti, funkcije ne rabe varijable radnog prostora. Funkcije komuniciraju s okolinom preko argumenata (ulaznih varijabli) i izlaznih varijabli (rezultata funkcije). Varijable unutar funkcije nisu vidljive

ni dostupne u radnom prostoru. Za razliku od skripti, funkcija je odvojena od radnog prostora i ne može izravno mijenjati varijable radnog prostora. Funkcija ima oblik:

```
function [output1,output2,…,outputN]=function_name(inp1,inp2,…,inpM)
% Comments
commands or functions (function body)
```

Prvi red svake funkcijske datoteke MORA BITI red s njezinom definicijom. Ako nema tog reda, ta datoteka nije funkcija nego skripta.

Red s definicijom funkcije definira:

- M-datoteku kao funkcijsku datoteku (funkciju)
- ime funkcije i
- broj i redoslijed ulaznih argumenata i izlaznih varijabli.

Red s definicijom funkcije ima sljedeći oblik:

```
function [output variables] = function_name(arguments)
```

odnosno

```
function [output1,output2, …,outputN] = function_name(inp1,inp2,…,inpM)
```

Funkcija navedena u prvom redu M-datoteke naziva se još i primarna funkcija (engl. *primary function*), dok se ostale funkcije koje se nalaze unutar primarne funkcije nazivaju podfunkcije (engl. *subfunctions*). Podfunkcije su opisane u nastavku.

Unutar funkcije na raspolaganju su samo argumenti (ulazne varijable) i varijable definirane unutar funkcije (lokalne varijable). Varijable definirane unutar funkcije čuvaju se u posebnom memorijskom prostoru (engl. *function workspace*) i nisu izravno dostupne u radnom prostoru (engl. *base workspace*). U radnom prostoru (engl. *base workspace*) dostupne su samo izlazne varijable (rezultat funkcije).

Funkciju predstavlja zatvoren sustav koji s radnim prostorom i drugim programima može komunicirati samo posredstvom argumenata (ulaznih varijabli) i izlaznih varijabli (rezultata funkcije). Memorije za pohranu varijabli radnog prostora (engl. *base workspace*) i memorije za pohranu varijabli funkcija (engl. *function workspace*) odvojene su.

8.4 Ulazne varijable (argumenti) funkcije

Ulazne varijable (argumenti) navode se unutar oblih zagrada iza imena funkcije. Najčešće postoji barem jedan ulazni argument, iako je moguće definirati i funkciju bez ulaznih argumenata. Argumenti su vidljivi i dostupni unutar tijela funkcije. Više ulaznih argumenata razdvaja se zarezima. Matematički izrazi u funkcijskoj datoteci moraju se pisati u skladu s dimenzijama argumenata zato što argumenti mogu biti skalari, vektori i matrice. Argumenti mogu biti brojevi, izraz čiji se rezultat može izračunati, ili varijabla kojoj je dodijeljena vrijednost. Argumentima se vrijednost dodjeljuje prema redoslijedu kojim su navedeni u popisu argumenata u definiciji funkcije.

Nazivi argumenata navedeni u definiciji funkcije su tzv. "slijepe" varijable (engl. *dummy variables*) i služe za posredan prijenos vrijednosti argumenata iz radnog okoliša funkciji. Nazivi tih varijabli važeći su samo unutar funkcije. Pri pozivu funkcije argumenti navedeni u okruglim zagradama iza naziva funkcije preslikavaju se u "slijepe" varijable. Npr. neka postoji funkcija:

```
function w = carea(x, y, z)
```

Funkcija se iz radnog okoliša (ili skripte ili druge funkcije poziva ovako):

```
>> carea(3, 4, 5)
```

ili:

```
>> p = 3
p =  3.00
>> r = 4
r =  4.00
>> q = 5
q =  5.00
>> carea(p, r, q)
```

Unutar funkcije varijabli **x** pripisat će se vrijednost varijable **p** iz radnog prostora, varijabli **y** vrijednost varijable **r** iz radnog prostora, a varijabli **z** vrijednost varijable **q** iz radnog prostora. Vrijednosti varijabli **p**, **q** i **r** odvojene su i neovisne o vrijednostima varijabli **x**, **y**, **z**. Pri pozivu funkcije važan je samo njihov redoslijed unutar zagrada funkcije gdje su navedeni argumenti.

Moguće je (ali ne i nužno) u radnom prostoru varijablama dati iste nazive koji su navedeni u definiciji funkcije, npr.:

```
>> x = 3
x =  3.00
>> y = 4
y =  4.00
>> z = 5
z =  5.00
>> carea(x, y, z)
```

Treba imati na umu da su varijable **x**, **y** i **z** definirane u radnom prostoru različite i neovisne o varijablama **x**, **y** i **z** unutar funkcije, iako se isto zovu.

Funkcija se može pozvati i s manje argumenata nego što je to navedeno u definiciji funkcije, npr.

```
>> carea(3, 4)
```

ali se pritom unutar funkcije (u ovom primjeru funkcije **carea**) treba pobrinuti što učiniti ako je broj argumenata manji od predviđenog.

Funkcija se ne može pozvati s više argumenata nego što je navedeno u definiciji funkcije, npr.:

```
>> carea(3, 4, 5, 6, 7)
Too many arguments!
```

8.5 Izlazne varijable (rezultati) funkcije

Izlazne varijable (rezultati) funkcije navode se unutar uglatih zagrada na lijevoj strani reda s definicijom funkcije iza riječi **function**. U definiciji funkcije više izlaznih varijabli odvajaju se zarezima, npr.:

```
function [x, y] = circumarea(r)
```

Pri pozivu funkcije s više izlaznih varijabli, varijable radnog prostora kojima se pripisuje rezultat funkcije mogu biti odvojene zarezima ili bjelinama, npr.:

```
>> [m, r] = circumarea(5)
```

ili

```
>> [m r] = circumarea(5)
```

U oba primjera varijabli **m** radnog prostora pripisat će se vrijednost izlazne varijable **x** funkcije **circumarea**, a varijabli **r** radnog prostora vrijednost izlazne varijable **y** funkcije **circumarea**.

Ako postoji samo jedna izlazna varijabla, ona se ne mora pisati unutar uglatih zagrada, npr.:

```
function y = sqsum(x)
```

i poziva se:

```
>> q = sqsum(5)
```

Funkcije mogu biti bez izlaznih varijabli, npr.:

```
function plotparab(x, y)
```

i poziva se:

```
>> plotparab(x, y)
```

Izlazne varijable funkcije neovisne su o varijablama radnog prostora, čak i ako imaju iste nazive. Npr. neka postoji funkcija:

```
function [x, y] = test
x = 10;
y = 20;
```

Ako se u radnom prostoru varijablama **x** i **y** pridijele vrijednosti, npr.:

```
>> x = 3
x =  3.00xx
>> y = 4
y =  4.00
```

i zatim pozove funkcija **test**, dobit će se ovaj rezultat:

```
>> [w, q] = test
w =  10.00
q =  20.00
```

Vrijednosti varijabli **x = 3** i **y = 4** u radnom prostoru različite su od vrijednosti varijabli **x = 10** i **y = 20** unutar funkcije, što se može provjeriti ispisom vrijednosti varijabli radnog prostora:

```
>> x
x =  3.00
>> y
y =  4.00
```

8.6 Komentar funkcije

Preporučuje se iza reda s definicijom funkcije pisati komentar u kojem su navedene temeljne značajke funkcije. Redova s komentarima može biti i više, a služe za upoznavanje korisnika s namjenom i načinom rada funkcije. Ti redovi moraju počinjati znakom %. Posebno je važan prvi red komentara koji slijedi neposredno iza definicije funkcije. Taj se red naziva H1. Funkcija **lookfor** pretražuje sve H1 redove svih funkcija, pa odgovarajući sadržaj reda H1 može bitno olakšati pronalaženje željene funkcije.

Kada korisnik u naredbenom prozoru napiše **help function_file_name**, Octave će prikazati komentare koji slijede neposredno iza retka s definicijom funkcije. Prikazat će se svi redovi komentara, od prvog komentara iza definicije funkcije pa do prvog reda u kojem nije komentar. To se naziva zaglavlje funkcije. Preporučljivo je u tom dijelu kratko opisati sve važne značajke funkcije.

8.7 Tijelo funkcije

Tijelo funkcije sadrži program napisan u M jeziku. U tijelu funkcije mogu se koristiti sve Octaveove ugrađene funkcije, funkcije koje je definirao korisnik, naredbe odluke i ponavljanja, sve vrste operatora i operacija i dr.

Funkcija se izvršava tako da se navede ime M-datoteke u koju je funkcija pohranjena, npr.:

```
>> rctngl(3, 4);
```

Funkcija se uvijek poziva tako da se navede naziv funkcijske M-datoteke. Ako se funkcijska M-datoteka nazove različito od naziva primarne funkcije, naziv primarne funkcije naveden u definiciji funkcije zanemaruje se. Preporučljivo je da se M-datoteci u koju je pohranjena funkcija dodjeljuje isto ime kao u retku s njezinom definicijom. U suprotnom može biti zbunjujuće što funkcija ima jedno ime, a pri pozivu funkcije treba navesti drugo ime (ime M-datoteke).

Funkcija koju je definirao korisnik rabi se na isti način kao i ugrađene Octaveove funkcije. Funkcije se mogu pozivati iz naredbenog prozora, skript datoteka i drugih funkcija. Za razliku od skripte funkcija se NE MOŽE pokrenuti naredbom **run** u programu za pisanje skripti i funkcija (**M-file editor**).

Funkcija se može rabiti dodjeljivanjem njezina rezultata varijabli ili varijablama, kao dio matematičkog izraza, kao argument druge funkcije ili navođenjem njezina imena u naredbenom prozoru ili skripti. U svim tim slučajevima korisnik mora točno znati kakve su vrste argumenti (ulazne varijable) i izlazne varijable (rezultat).

Pri pozivu funkcije pretpostavlja se da se funkcija nalazi u tekućem imeniku. Ako to nije tako, pri pozivu funkcije treba navesti cjelokupni put (engl. *path*) do datoteke.

Treba uočiti sljedeće značajke funkcija:

- Vrijednosti svih podataka koji se rabe u funkciji moraju biti određene argumentima (ulaznim varijablama) ili moraju biti određene unutar funkcije.
- Rezultat funkcije izlazna je varijabla čija se vrijednost može prenijeti u radni prostor.
- Funkcije se mogu prevesti u drugi programski jezik ili u izvršni oblik (kompajlirati, engl. *compile*).
- Varijable definirane unutar funkcije odvojene su od varijabli definiranih u radnom prostoru, čak i ako imaju iste nazive (osim ako nisu definirane kao globalne). To znači da promjena vrijednosti varijabli definiranih unutar funkcije ne utječe na varijable definirane u radnom prostoru i obrnuto.

Primjer 8.5: Funkcija **`circumference`** ima jedan argument (ulaznu varijablu) i jednu izlaznu varijablu (rezultat). Naziv je funkcije **`circumference`**, a naziv funkcijske datoteke **`circumference.m`**. Funkcija računa opseg kruga čiji je polumjer jednak argumentu funkcije (zbog kratkoće izbačen je komentar funkcije).

```
function [y] = circumference(r)
y = 2*r*pi;
```

Funkcija se može pozvati tako da se u naredbenom prozoru napiše, npr.:

```
>> circumference(2)
ans =  12.57
>>
```

Rezultat pozivanja funkcijske datoteke **`circumference.m`** za **`circumference(2)`**, odnosno za vrijednost argumenta **`r=2`**, je 12,57.

Budući da s lijeve strane nema varijable kojoj se dodjeljuje vrijednost, Octave rabi varijablu **`ans`**.

Primjer 8.6: Pozivanje funkcijske datoteke **`circlearea.m`** za **`circlearea(5)`**, odnosno vrijednost argumenta **`r=5`** i prikaz rezultata:

```
function [y] = circlearea(r)
y = (r^2)*pi;
```

```
>> circlearea(5)
ans =  78.54
>>
```

Budući da s lijeve strane nema varijable kojoj se dodjeljuje vrijednost, Octave rabi varijablu **`ans`**.

U ovom je primjeru argument **`r`** skalar. Ako bi trebalo izračunati površinu kruga za više različitih polumjera, treba pozivati funkciju više puta:

```
>> circlearea(1)
ans =  3.14
>> circlearea(3.5)
ans =  38.48
>> circlearea(2.7)
ans =  22.90
>> circlearea(6.3)
ans =  124.69
>>
```

Ako je potrebno izračunati površinu kruga za više vrijednosti argumenta **`r`**, ovaj je postupak zamoran i nepotreban. Bilo bi korisno da se mogu upisati sve vrijednosti argumenata **`r`** za koje se želi izračunati površina kruga te proslijediti funkciji. To je moguće, ali se mora paziti da je funkcija napisana na ogovarajući način. U upravo navedenom primjeru to nije bio slučaj.

Ako se funkcijskoj datoteci **`circlearea.m`** proslijedi više vrijednosti ulazne varijable, program Octave javlja pogrešku. Izračun površine kruga nije moguć za više vrijednosti jer se ulazna varijabla prosljeđuje kao vektor, a u funkcijskoj se datoteci s njom računa kao sa skalarom.

```
>> circlearea([1 3.5 2.7 6.3])
error: for x^A, A must be a square matrix.  Use .^ for elementwise power.
```

```
error: called from
    circlearea at line 2 column 3
>>
```

Funkciju **circlearea** treba preurediti kako bi prihvaćala vrijednosti argumenta **r** kao vektor. Funkcija **circlearea** koja prihvaća ulaznu varijablu **r** kao vektor izgleda ovako:

```
function [y] = circlearea(r)
y = (r.^2)*pi;
```

Razlika je u operatoru **^** koji je sada **.^** (uočite točku ispred operatora **^**). To je operator potenciranja matrice element po element. Taj se postupak često naziva i vektorizacija (vektorizacija je opisana u poglavlju 7.3).

Ako se tako prerađenoj funkciji proslijedi argument **r** kao vektor **r=[1 3.5 2.7 6.3]**, rezultat će biti:

```
>> r = [1 3.5 2.7 6.3]
r =
    1.00    3.50    2.70    6.30
>> circlearea(r)
ans =
    3.14   38.48   22.90  124.69
>>
```

Vidljivo je da je sada rezultat funkcije vrijednost površine kruga za sve vrijednosti ulazne varijable **r**. Treba skrenuti pozornost da postoje posebni slučajevi koji mogu dovesti do zabune. Npr. primijeni li se naredba potenciranja bez točke na kvadratnu matricu, izvršit će se program bez pogreške, ali rezultat vjerojatno neće biti ono što je korisnik imao na umu. U ovom je primjeru korisnik želio potencirati svaki element matrice, ali ovako napisanom naredbom rezultat je kvadrirana matrica. Npr.:

```
>> x = [1 2 3; 4 5 6; 7 8 9]
x =
   1.00   2.00   3.00
   4.00   5.00   6.00
   7.00   8.00   9.00
>> y = x^2
y =
    30.00    36.00    42.00
    66.00    81.00    96.00
   102.00   126.00   150.00
>> y = x.^2
y =
     1.00     4.00     9.00
    16.00    25.00    36.00
    49.00    64.00    81.00
>>
```

Treba stoga uvijek imati na umu rezultat operatora.

Do sada su bile prikazane funkcije koje imaju jedan argument i jednu izlaznu varijablu. Riječ je bila o jednom argumentu i jednoj izlaznoj varijabli bez obzira što su one u zadnjem primjeru bile vektori, a ne skalari. U nastavku će biti prikazana funkcija koja ima jedan argument i dvije izlazne varijable. U ovom će se primjeru spojiti dvije prethodne funkcije u jednu, odnosno napisat će se funkcija koja računa i opseg i površinu kruga. Argument je **r** (polumjer kruga), a izlazne varijable su **y** (opseg kruga) i **w** (površina kruga).

```
function [y, w] = circumarea(r)
y = 2*r*pi;
w = (r.^2)*pi;
```

Ako se u naredbeni prozor napiše **`circumarea(4.6)`**, rezultat je:

```
>> circumarea(4.6)
ans =  28.90
>>
```

Prikazan je samo rezultat za opseg kruga. To je zato što je poziv funkcijske datoteke bio bez varijable na lijevoj strani. Kako funkcija ima dvije izlazne varijable, prikazana je samo prva izlazna varijabla.

Kod funkcija koje imaju više izlaznih varijabli mora se prije poziva funkcije na lijevoj strani jednakosti napisati onoliko varijabli unutar uglatih zagrada koliko funkcija ima izlaznih varijabli. U ovom je primjeru potrebno napisati sljedeće:

```
>> [q, z] = circumarea(4.6)
q =  28.90
z =  66.48
>>
```

Treba uočiti da varijable u uglatim zagradama imaju nazive **`q`** i **`z`**, što je različito od **`y`** i **`w`** navedenih unutar funkcije. Budući da su varijable u radnom prostoru (engl. *base workspace*) odvojene od varijabli navedenih unutar funkcije (engl. *function workspace*), svejedno je koji će se nazivi s lijeve strane navesti pri pozivu funkcije. Važan je samo redoslijed (prva varijabla će poprimiti vrijednost varijable **`y`** iz funkcije, a druga vrijednost varijable **`w`**). Mogu se navesti i ista imena varijabli kako su navedena unutar funkcije. Pritom treba znati da su unatoč istog imena, varijable u radnom prostoru i unutar funkcije odvojene i neovisne (osim ako je riječ o globalnim varijablama).

Funkcija može imati i više argumenata i više izlaznih varijabli. Npr. funkcija **`rect`** ima dvije ulazne varijable i dvije izlazne varijable, pohranjena je u datoteku **`rect.m`** i izgleda ovako:

```
function [w, z] = rect(x,y)
w = 2*(x+y);
z = x.*y;
```

Argumenti su **`x`** i **`y`** (duljine stranica pravokutnika), a izlazne varijable su **`w`** i **`z`** (opseg i površina pravokutnika).

Ovdje je prikazan rezultat pozivanja funkcije **`rect`** gdje su ulazne varijable **`x`** i **`y`** vektori **`x = [1 2 3 4 5]`** i **`y = [1 2 3 4 5]`**, a izlazne varijable **`w`** i **`z`** su također vektori **`w = [4 8 12 16 20]`** i **`z = [1 4 9 16 25]`**, odnosno u radni prostor preslikane varijable **`p`** i **`q`**.

```
>> [p, q] = rect(1:5,1:5)
p =
     4.00     8.00    12.00    16.00    20.00
q =
     1.00     4.00     9.00    16.00    25.00
>>
```

Primjer 8.7: Tri primjera funkcije s jednom ulaznom varijablom **`x`** i jednom izlaznom varijablom **`y`**.

```
function [y] = fun1(x)
y = x.^2 + 3*x - 5;
```

```
function [y] = fun2(x)
y = exp(-x).*sin(10*x);
```

```
function [y] = fun3(x)
y = -x.^3 + 3*x.^2 + x - 1;
```

Primjer 8.8: Dva primjera funkcija s dvije ulazne varijable **x** i **y** te jednom izlaznom varijablom **z**.

```
function [z] = fun4(x,y)
z = x.^2 - y.^2;
```

```
function [z] = fun5(x,y)
z = x.*exp(-x.^2 - y.^2);
```

8.8 Funkcijska M-datoteka

M-datoteka u koju je pohranjena funkcija naziva se funkcijska M-datoteka ili kraće funkcijska datoteka. Preporučuje se da naziv funkcijske datoteke bude isti kao i naziv primarne funkcije. Funkcija se uvijek poziva tako da se navede naziv funkcijske M-datoteke. Ako se funkcijska M-datoteka nazove različito od naziva primarne funkcije, naziv primarne funkcije naveden u definiciji funkcije zanemaruje se. Npr. neka postoji funkcija:

```
function [y] = fun1(x)
```

i neka je ona pohranjena u funkcijsku M-datoteku s nazivom **test1.m**. Funkcija se poziva ovako:

```
>>test1(3)
```

Pokuša li se funkcija pozvati ovako:

```
>>fun1(3)
```

Octave će javiti pogrešku jer ne može pronaći M-datoteku s nazivom **fun1**.

echo

Funkcije se obično pišu tako da se pri izvršenju funkcije ne prikazuju njezine naredbe ni međurezultati (osim ako to autor nije izričito omogućio). Naredbom **echo** mogu se prikazati i naredbe funkcije pri njezinu izvršenju, ali to usporava njezino izvršavanje pa naredbu **echo** kod funkcija treba rabiti samo kad se program provjerava ili traži pogreška. Naredba **echo** kod funkcije ima oblike prikazane u tablici 8.1.

Tablica 8.1 Oblici **echo** naredbe za funkcije

echo on	Uključuje ispis naredbi.
echo off	Isključuje ispis naredbi.
echo on all	Uključuje ispis naredbi za sve funkcije.
echo off all	Isključuje ispis naredbi za sve funkcije.

return

Ponekad se u funkcijama rabi naredba **return**. Naredba **return** prekida izvršavanje funkcije i vraća nadzor pozivnoj funkciji ili skripti iz koje je bila pozvana funkcija.

Primjer 8.9: Za računanje opsega kruga koristi se funkcija **`circumfer`** i unutar nje naredba **`return`**. U slučaju da se funkciji **`circumfer`** proslijedi argument koji je manji od 0 (negativna vrijednost), funkcija će ispisati "Polumjer negativan!" i vratiti nadzor pozivnoj funkciji ili skripti.

```
function [y] = circumfer(r)
if r<0
    disp('Radius negative!')
    return
end
y = 2*r*pi;
```

Rezultat funkcije ako je argument funkcije pozitivan:

```
>> circumfer(2)
ans =  12.57
```

Rezultat funkcije ako je argument funkcije negativan:

```
>> circumfer(-5)
Radius negative!
>>
```

nargin

Rezultat naredbe **`nargin`** broj je argumenata (ulaznih varijabli) proslijeđenih funkciji. Naime, pri pozivu funkcije može se dogoditi da se funkcija poziva s brojem argumenata različitim od onog navedenog u definiciji funkcije. Rezultat će u tom slučaju biti pogrešan ili će doći do prekida programa. Kako bi se to preduhitrilo, u tijelo se funkcije na početku navede naredba **`nargin`** i provjeri je li broj argumenata ispravan. Ako nije, program se može prekinuti i korisniku ispisati poruka o tome da je unio pogrešan broj argumenta. Npr. pozove li se funkcija:

```
function f = sumpoly(x,y,z)
f = x^3 + y^2 + z
```

tako da se navede samo jedan argument, Octave će javiti pogrešku:

```
>> sumpoly(3)
error: 'y' undefined near line 2 column 11
error: called from
    sumpoly at line 2 column 3
>>
```

Uporabom naredbe **`nargin`** program može javiti razumljiviju poruku korisniku:

```
function f = sumpoly(x,y,z)

if nargin < 3
    error('Potrebne su tri ulazne varijable')
end

f = x^3 + y^2 + z
```

```
>> sumpoly(3)
error: Potrebne su tri ulazne varijable
```

```
error: called from
    sumpoly at line 4 column 3
>>
```

Korisnik može prije izvršenja funkcije provjeriti potreban broj argumenata naredbom **`nargin(@function_name)`**, npr.:

```
>> nargin(@sumpoly)
ans =  3
>>
```

nargchk

Rezultat naredbe **`nargchk`** tekstualna je poruka (znakovni niz) koja govori o tome je li broj argumenata koji se prosljeđuje funkciji ispravan. Oblik naredbe je:

`nargchk(minargs, maxargs, numargs)` – gdje je **`minargs`** najmanji dopušteni broj argumenata, **`maxargs`** najveći dopušteni broj argumenata, a **`numargs`** stvarni broj argumenata koji se prosljeđuju funkciji. Naredba **`nargchk`** rabi se unutar funkcije redovito zajedno s naredbom **`nargin`** i **`error`**. Npr. ako se funkcija:

```
function f = foo(x,y,z)

nargchk(2,3,nargin)
```

pozove s neodgovarajućim brojem argumenata, ispisat će se pogreška čiji je sadržaj znakovni niz koji je rezultat naredbe **`nargchk`**.

```
>> foo(1)
ans = not enough input arguments
>> foo(1,2)
ans =
>> foo(5,10,15)
ans =
>> foo(5,6,7,8)
ans = too many input arguments
>>
```

nargout

Rezultat naredbe **`nargout`** broj je izlaznih varijabli definiranih pri pozivu funkcije. Ako se pri pozivu funkcije navede manje od predviđenog broja izlaznih varijabli, prikazat će se samo onaj broj koji je naveden pri pozivu. Uporabom naredbe **`nargout`** rezultat se može promijeniti ako je naveden manji broj izlaznih varijabli od potrebnog ili se može korisniku porukom dati do znanja da nisu prikazane sve vrijednosti izlaznih varijabli. Npr. ako se funkcija:

```
function [y, w] = circumarea(r)
y = 2*r*pi;
w = (r.^2)*pi;
```

pozove ovako:

```
>> circumarea(3)
ans =  18.85
>>
```

prikazat će se samo jedna od predviđene dvije izlazne varijable. Prepravimo li funkciju ovako:

```
function [y, w] = circumarea(r)
if nargout <2
    error('Postoje dvije izlazne varijable!')
end
y = 2*r*pi;
w = (r.^2)*pi;
```

i pozove ovako:

```
>> circumarea(3)
error: Postoje dvije izlazne varijable!
error: called from
    circumarea at line 3 column 3
>>
```

prikazuje se poruka o broju izlaznih varijabli funkcije.

Korisnik može prije izvršenja funkcije provjeriti broj izlaznih varijabli naredbom **`nargout(@function_name)`**, npr.:

```
>> nargout(@circumarea)
ans =  2
>>
```

nargoutchk

Rezultat naredbe **`nargoutchk`** tekstualna je poruka (znakovni niz) koja govori o tome je li broj izlaznih varijabli koji se očekuje od funkcije ispravan. Oblik naredbe je **`nargoutchk(minargs, maxargs, numargs)`**, gdje je **`minargs`** najmanji dopušteni broj izlaznih varijabli, **`maxargs`** najveći dopušteni broj izlaznih varijabli, a **`numargs`** stvarni broj izlaznih varijabli koji se očekuje od funkcije. Naredba **`nargoutchk`** rabi se unutar funkcije redovito zajedno s naredbom **`nargout`** i **`error`**. Npr. ako se funkcija:

```
function [x, y] = foo(x, y, z)
error(nargoutchk(2,2, nargout))
```

pozove s neodgovarajućim brojem argumenata ispisat će se pogreška čiji je sadržaj znakovni niz koji je rezultat naredbe **`nargoutchk`**.

```
>> x = foo(1, 2, 3)
??? Error using ==> foo
Not enough output arguments.
```

varargin

Ponekad je u trenutku pisanja funkcije nepoznat broj argumenata koji će se navesti pri pozivu funkcije. Naredba **`varargin`** omogućuje da se funkcija pozove s promjenjivim brojem argumenata. Npr. funkcija:

```
function f = foo(x,y)
```

očekuje točno dva argumenta pri pozivu, pa se ispravno poziva ovako:

```
>> foo(1,2)
```

Ako želimo funkciju **foo** pozivati s različitim brojem argumenata, rabit ćemo oblik funkcije s naredbom **varargin**, npr.:

```
function y = foo(varargin{:})
```

Ako se funkcija **foo** pozove ovako:

```
>> foo(1,'red','uppercase',[1 2 3])
```

funkciji **foo** će se proslijediti argumenti redom **vargin{1}=1**, **vargin{2}='red'**, **vargin{3}='uppercase'** i **vargin{4}=[1 2 3]**.

Navođenje naredbe **varargin** u funkciji **foo** omogućilo je pozivanje funkcije **foo** s promjenjivim brojem argumenata. Ključno je uočiti da se funkcija **foo** mogla pozivati s po volji odabranim brojem argumenata odvojenim zarezima.

Primjer 8.10: Primjer funkcije **carea** koja prihvaća različit broj argumenata. Ovisno o broju argumenata računa se površina kvadrata, pravokutnika ili trokuta. U slučaju da pri pozivu nema argumenata ili ih je više od tri, ispisuje se poruka upozorenja.

```
function w = carea(varargin)
% carea(varargin)
%
% Synopsis: carea(x)        square area
%           carea(x, y)     rectangle area
%           carea(x, y, z)  triangle area
%
% Input:  x - square side
%         x, y - rectangle sides
%         x, y, z - triangle sides
% Output: square area or rectangle area or
%         triangle area
%
% Calculate area of square, rectangle or triangle,
% depending on number of arguments.

switch nargin
     case 0    % No arguments - warning message
         disp('At least one arguments requested!');
     case 1    % One argument - square area
         w = varargin{1}^2;
         disp(['Square area = ', num2str(w)]);
     case 2    % Two arguments - rectangle area
         w = varargin{1} * varargin{2};
         disp(['Rectangle area = ', num2str(w)]);
     case 3    % Three arguments - triangle area
         % Heron's formula for triangle area
         x = varargin{1};
         y = varargin{2};
         z = varargin{3};
         s = (x + y + z)/2;
         w =  sqrt(s*(s - x)*(s - y)*(s - z));
         disp(['Triangle area = ', num2str(w)]);
    otherwise   % More then three arguments
```

```
        disp('Too many arguments!');
end
```

Prikazani su rezultati poziva funkcije **carea** s različitim brojem argumenata.

```
>> carea();
At least one arguments requested!
>> carea(3);
Sqare area = 9
>> carea(3,4);
Rectangle area = 12
>> carea(3,4,5);
Tiangle area = 6
>> carea(3,4,5,6);
Too many arguments!
>>
```

varargout

Uz uporabu naredbe **varargout** moguće je kao rezultat funkcije dobiti različit broj izlaznih varijabli. Npr. neka postoji funkcija:

```
function [varargout] = test20(x)
for i = 1:x
     varargout(i) = {i};
end
```

Pozovemo li funkciju **test20** s vrijednošću argumenta **x=1**, funkcija će dobiti jednu vrijednost:

```
>> [x] = test20(1)
x =  1.00
>>
```

Pozovemo li funkciju **test20** s vrijednošću argumenta **x=3**, funkcija će dobiti tri vrijednosti:

```
>> [x,y,z] = test20(3)
x =  1.00
y =  2.00
z =  3.00
>>
```

8.9 Funkcije funkcija

Argumenti do sada opisanih funkcija numeričke su vrijednosti. Te numeričke vrijednosti mogu biti unesene izravno pri pozivu funkcije ili neizravno posredstvom varijabli radnog prostora. Npr.:

```
>> w = functx(3,4)
```

ili

```
>> x = 3
x =  3.00
>> y = 4
y =  4.00
>> w = functx(x,y)
```

Octave omogućuje da argument funkcije bude druga funkcija. Da bi to bilo moguće, argument mora biti pokazivač funkcije (engl. *function handle*). Pokazivač funkcije tvori se pomoću operatora `@`, npr.:

```
>> functxhandle = @functx
```

U gornjem primjeru varijabla **`functxhandle`** pokazivač je funkcije **`functx`**. Pokazivač funkcije omogućuje da funkcija bude argument druge funkcije. Funkcije koje mogu prihvatiti pokazivač druge funkcije kao argument nazivaju se funkcije funkcija (engl. *function functions*). Takve su, npr. ugrađene Octaveove funkcije **`feval`** i **`fplot`** te sve korisničke funkcije koje su pisane tako da mogu prihvatiti pokazivač funkcije kao argument.

Pokazivač funkcije može se prikazati naredbama **`functions(funhandle)`** i **`func2str(funhandle)`**.

feval

Naredba **`feval`** izvršava funkciju koja je navedena kao argument funkcije **`feval`**. Sintaksa funkcije **`feval`** je:

`feval('functionname', arg1, arg2, …, argn)` – gdje je **`functionname`** ime funkcije koja će se izvršiti, a **`arg1, arg2, …, argn`** jedan ili više argumenata funkcije **`functionname`**. Npr.

```
>> feval('circumfer',3)
```

isto je kao i:

```
>> circumfer(3)
```

Argument **`functionname`** mora biti ugrađena Octaveova funkcija ili korisnički definirana funkcija.

Umjesto naziva funkcije navedenog pod apostrofima **`'functionname'`**, kao argument moguće je proslijediti pokazivač funkcije, npr.:

```
>> fncirc = @circumfer
>> feval(fncirc,3)
```

isto je kao:

```
>> circumf(3)
```

Npr. u primjeru funkcije **`bisect`** izvršit će se funkcija **`fn`** koja je argument funkcije **`bisect`**. To omogućuje da se unutar funkcije **`bisect`** izvršavaju različite funkcije **`fn`** ovisno o argumentu.

```
function c = bisect(fn, a, b, tol)
```

```
fa = feval(fn, x);
```

U drugom će retku funkcija **fa** postati **fn(x)** zbog primjene naredbe **feval**.

Korisnička funkcija **bisect** pronalazi rješenje funkcije **fn=0** metodom bisekcije. Pomoću funcije **feval** moguće je funkciju **bisect** pozivati za različite funkcije **fn**, npr.:

```
>> bisect('sin', x, b, tol)
>> bisect('cos', x, b, tol)
```

U prvom slučaju izračunat će se točka u kojoj je **sin** jednak nuli, a u drugom točka u kojoj je **cos** jednak nuli. U oba se primjera pozivala ista funkcija **bisect**, ali s različitim argumentima. U načelu, prvi argument funkcije **feval** pokazivač je funkcije koja se želi izvršiti (ili njezin naziv naveden pod apostrofima). U navedenom primjeru to su funkcije **sin** i **cos**.

8.10 Podfunkcije

Podfunkcije (engl. *subfunctions*) su funkcije definirane unutar primarne funkcije (engl. *primary function*). One se mogu rabiti (pozivati) samo unutar primarne funkcije unutar koje su definirane te nisu vidljive (dostupne) izvan nje. Npr.:

```
function y = sumpoly2(x)
y = sum(polthr(x));
function z = polthr(x)
z(1) = x^3;
z(2) = x^2;
z(3) = x;
```

U ovom je primjeru funkcija **polthr(x)** vidljiva samo unutar primarne funkcije **sumpoly2(x)** i nije vidljiva izvan nje.

Varijable definirane unutar podfunkcija spremaju se u poseban memorijski prostor i vidljive su samo podfunkciji te nisu dostupne primarnoj funkciji ni radnom prostoru.

Podfunkcije ima smisla rabiti ako se unutar primarne funkcije treba više puta izvršavati isti odsječak programa ili se pojedina zadaća želi odvojiti u posebnu cjelinu kako bi se program razdijelio na module, tj. dijelove koje je lakše provjeravati. Primjer podfunkcija može se naći u mnogim ugrađenim funkcijama programa Octave (npr. pomoću naredbe **type ezplot** mogu se vidjeti podfunkcije ugrađene funkcije **ezplot**).

8.11 Ugrađene funkcije

Octave program ima mnoge gotove funkcije koje su provjerene i optimirane glede brzine izvođena. Stoga, kad god je to moguće, treba rabiti gotove Octaveove funkcije te prije pisanja vlastitog odsječka programa za rješenje nekog zadatka valja provjeriti postoji li gotova Octaveova funkcija koja može riješiti dio zadatka ili cijeli zadatak. Takve se gotove funkcije nazivaju ugrađene funkcije (engl. *built-in functions*). Njima se rukuje isto kao i s korisničkim funkcijama. Pomoću naredbe **help** može se vidjeti cjelokupan popis raspoloživih gotovih Octaveovih funkcija.

Dio je Octaveovih funkcija preveden pa nije moguće vidjeti izvorni program. Pokuša li se vidjeti sadržaj prevedene funkcije naredbom **type**, ispisat će se da je to ugrađena funkcija i neće se vidjeti izvorni program. Takve su, primjerice, funkcije sin, cos, sqrt i dr. Npr.:

```
>> type sin
sin is built-in function
>>
```

Ostale su Octaveove funkcije M-datoteke pa se njihov sadržaj može vidjeti kao i sadržaj korisničkih M-datoteka. Npr. naredba **`type isprime`** prikazat će sadržaj M-datoteke **`isprime.m`**. O programiranju u Octaveu puno se može naučiti proučavajući ugrađene Octaveove funkcije. Pomoću naredbe **`type nazivfunkcije`** može se provjeriti koja je ugrađena funkcija prevedena, a koja je u obliku M-datoteke.

8.12 Naredbe i funkcije

U prvim inačicama Octave programa razlikovale su se naredbe i funkcije. Naredbe koje su imale argumente pisale su se na ovaj način:

```
>> save test x y z
```

dok bi se funkcije koje su imale argumente pisale ovako:

```
>> w = funcx(x,y,z)
```

Postoje naredbe koje mijenjaju okoliš (npr. pohrana datoteke, crtanje grafikona i dr.), ali ne vraćaju nikakav rezultat:

```
>> y = clc
error: value on right hand side of assignment is undefined
```

Nakon inačice Octavea 4 naredbe i funkcije postale su dvojne u smislu da se naredbe smatraju funkcijama koje prihvaćaju argumente tipa niz (engl. *string*). Tako je npr. naredba **`axis off`** isto što i naredba **`axis('off')`**.

To omogućuje da se naredbama proslijede nizovi (engl. *strings*) koji se mogu oblikovati (mijenjati) unutar programa. Npr. naredba **`axis`** može se pozvati ovako:

```
>> axis(q)
```

gdje je **`q`** varijabla tipa niz (engl. *string*) kojoj se u programu može pridjeljivati vrijednost ovisno o tijeku odvijanja programa.

Druga je prednost dvojnosti naredbi i funkcija da korisnik može napisati nove naredbe pomoću funkcija (funkcijskih M-datoteka).

8.13 Razlike između skripti i funkcija

Ponekad je teško uočiti razliku između skripti i funkcija, pa će se u nastavku navesti njihove sličnosti i razlike:

- I skripte i funkcije spremaju se u M-datoteke s produžetkom m.
- Naziv funkcije može biti različit od naziva M-datoteke u koju je funkcija pohranjena, ali se takvo imenovanje ne preporučuje. Naziv skripte ujedno je i naziv M-datoteke u koju je skripta pohranjena te ne može biti različit od nje.
- Prvi red funkcije mora biti definicija funkcije.
- Varijable definirane unutar funkcije lokalne su (osim ako ih izričito ne definiramo kao globalne) i nisu vidljive u radnom prostoru.
- Istoimene varijable definirane unutar skripti i radnog prostora iste su. Skripte razmjenjuju podatke s radnim prostorom izravno jer rabe iste varijable (varijable se nalaze u istom memorijskom prostoru).
- Funkcije prihvaćaju podatke pomoću ulaznih varijabli (argumenata), a vraćaju ih preko izlaznih varijabli (rezultata). Funkcije razmjenjuju podatke s radnim prostorom samo pomoću ulaznih i

izlaznih varijabli. Istoimene varijable definirane unutar funkcije i radnog prostora odvojene su i različite (nalaze se u odvojenim memorijskim prostorima).

9. TRANSCEDENTNE FUNKCIJE

Realne funkcije dijele se na algebarske i transcedentne funkcije. Funkcija je algebarska ako se pri računanju zavisne varijable koriste samo algebarske operacije: zbrajanje, oduzimanje, množenje, dijeljenje i potenciranje. Algebarske funkcije dijele se na cijele racionalne funkcije (polinomi), razlomljene racionalne funkcije i iracionalne funkcije. Sve funkcije koje nisu algebarske su transcedentne. Najvažnije transcedentne funkcije su eksponencijalna funkcija, logaritamska funkcija, trigonometrijske funkcije i ciklometrijske funkcije.

9.1 Trigonometrijske funkcije

U trigonometrijske funkcije ubrajaju se funkcije sinus, kosinus, tangens i kotangens. U tablici 9.1 prikazane su domene i kodomene trigonometrijskih funkcija.

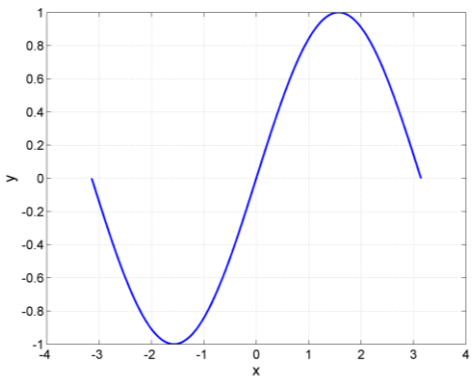
Tablica 9.1 Domene i kodomene trigonometrijskih funkcija

Funkcija	Domena	Kodomena
$y=\sin(x)$	$-\pi \le x \le \pi$	$-1 \le y \le 1$
$y=\cos(x)$	$-\pi \le x \le \pi$	$-1 \le y \le 1$
$y=tg(x)$	$-\pi/2 < x < \pi/2$	$-\infty < y < \infty$
$y=ctg(x)$	$0 < x < \pi$	$-\infty < y < \infty$

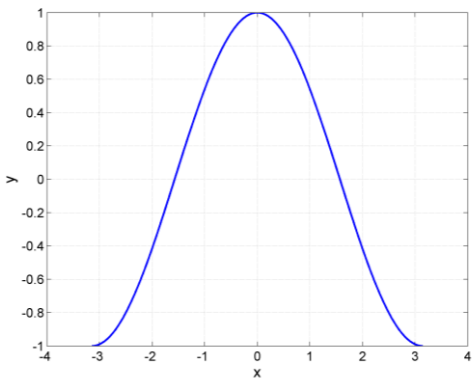
Na slici 9.1 prikazani su grafovi trigonometrijskih funkcija.

Za trigonometrijske funkcije vrijedi:

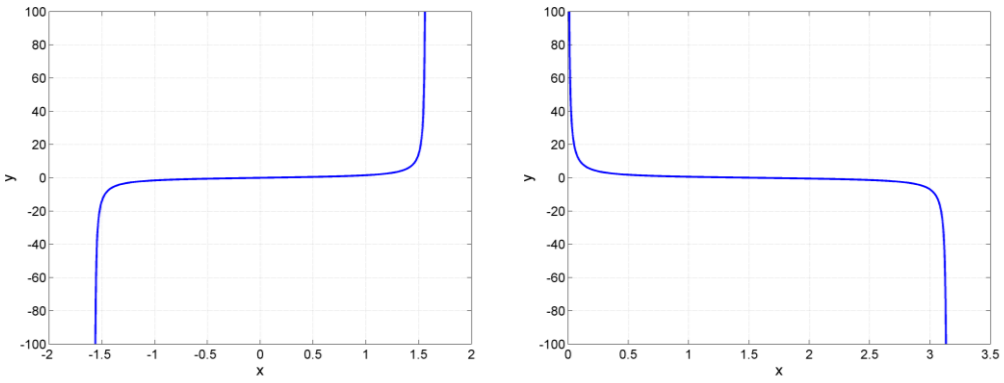
$$y=\cos(x)=\sin(x+\pi/2) \qquad y=tg(x)=\frac{\sin(x)}{\cos(x)} \qquad y=ctg(x)=\frac{1}{tg(x)}=\frac{\cos(x)}{\sin(x)}$$



a. $y=\sin(x)$ za $-\pi \le x \le \pi$



b. $y=\cos(x)$ za $-\pi \le x \le \pi$



c. $y = tg(x)$ za $-\pi/2 < x < \pi/2$ d. $y = ctg(x)$ za $0 < x < \pi$

Slika 9.1 Grafovi trigonometrijskih funkcija

sin

U Octaveu se za izračunavanje funkcije sinus koristi funkcija **`sin(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija sinus računa se element po element za vektore i matrice. Argument funkcije **`sin`** mora biti u radijanima.

Primjer 9.1: Funkcija **`sin`** primijenjena na matricu **`x`** dimenzija 2*3 daje matricu **`y`** dimenzija 2*3.

```
>> x = [0.5 0.8 0.7; 1.2 3.1 2.4]
x =
    0.50    0.80    0.70
    1.20    3.10    2.40
>> y = sin(x)
y =
    0.48    0.72    0.64
    0.93    0.04    0.68
>>
```

sind

Ako je argument funkcije sinus u stupnjevima, koristi se funkcija **`sind(x)`**, gdje **`x`** može biti skalar, vektor ili matrica.

Primjer 9.2: Funkcija **`sind`** primijenjena na matricu **`x`** dimenzija 2*2 daje matricu **`y`** dimenzija 2*2.

```
>> x = [30 45; 60 90]
x =
    30.00    45.00
    60.00    90.00
>> y = sind(x)
y =
    0.50    0.71
    0.87    1.00
>>
```

cos

U Octaveu se za izračunavanje funkcije kosinus koristi funkcija **cos(x)**, gdje **x** može biti skalar, vektor ili matrica. Funkcija kosinus računa se element po element za vektore i matrice. Argument funkcije **cos** mora biti u radijanima.

Primjer 9.3: Funkcija **cos** primijenjena na matricu **x** dimenzija 2*3 daje matricu **y** dimenzija 2*3.

```
>> x = [0.5 0.8 0.7; 1.2 3.1 2.4]
x =
    0.50    0.80    0.70
    1.20    3.10    2.40
>> y = cos(x)
y =
    0.88    0.70    0.76
    0.36   -1.00   -0.74
>>
```

cosd

Ako je argument funkcije kosinus u stupnjevima, koristi se funkcija **cosd(x)**, gdje **x** može biti skalar, vektor ili matrica.

Primjer 9.4: Funkcija **cosd** primijenjena na matricu **x** dimenzija 2*2 daje matricu **y** dimenzija 2*2.

```
>> x = [30 45; 60 90]
x =
    30.00    45.00
    60.00    90.00
>> y = cosd(x)
y =
    0.87    0.71
    0.50    0.00
>>
```

tan

U Octaveu se za izračunavanje funkcije tangens koristi funkcija **tan(x)**, gdje **x** može biti skalar, vektor ili matrica. Funkcija tangens računa se element po element za vektore i matrice. Argument funkcije **tan** mora biti u radijanima.

Primjer 9.5: Funkcija **tan** primijenjena na redni vektor **x** dimenzija 1*4 daje redni vektor **y** dimenzija 1*4.

```
>> x = [0.1 1.1 1.5 0.7]
x =
    0.10    1.10    1.50    0.70
>> y = tan(x)
y =
    0.10    1.96   14.10    0.84
>>
```

tand

Ako je argument funkcije tangens u stupnjevima, koristi se funkcija **tand(x)**, gdje **x** može biti skalar, vektor ili matrica.

Primjer 9.6: Funkcija **tand** primijenjena na redni vektor **x** dimenzija 1*4 daje redni vektor **y** dimenzija 1*4.

```
>> x = [10 30 70 60]
x =
    10.00    30.00    70.00    60.00
>> y = tand(x)
y =
    0.18    0.58    2.75    1.73
>>
```

cot

U Octaveu se za izračunavanje funkcije kotangens koristi funkcija **cot(x)**, gdje **x** može biti skalar, vektor ili matrica. Funkcija kotangens računa se element po element za vektore i matrice. Argument funkcije **cot** mora biti u radijanima.

Primjer 9.7: Funkcija **cot** primijenjena na matricu **x** dimenzija 2*3 daje matricu **y** dimenzija 2*3.

```
>> x = [0.1 1.1 1.5; 0.5 0.9 0.3]
x =
    0.10    1.10    1.50
    0.50    0.90    0.30
>> y = cot(x)
y =
    9.97    0.51    0.07
    1.83    0.79    3.23
>>
```

cotd

Ako je argument funkcije kotangens u stupnjevima, koristi se funkcija **cotd(x)**, gdje **x** može biti skalar, vektor ili matrica.

Primjer 9.8: Funkcija **cotd** primijenjena na matricu **x** dimenzija 2*3 daje matricu **y** dimenzija 2*3.

```
>> x = [10 80 50; 60 15 70]
x =
    10.00    80.00    50.00
    60.00    15.00    70.00
>> y = cotd(x)
y =
    5.67    0.18    0.84
    0.58    3.73    0.36
>>
```

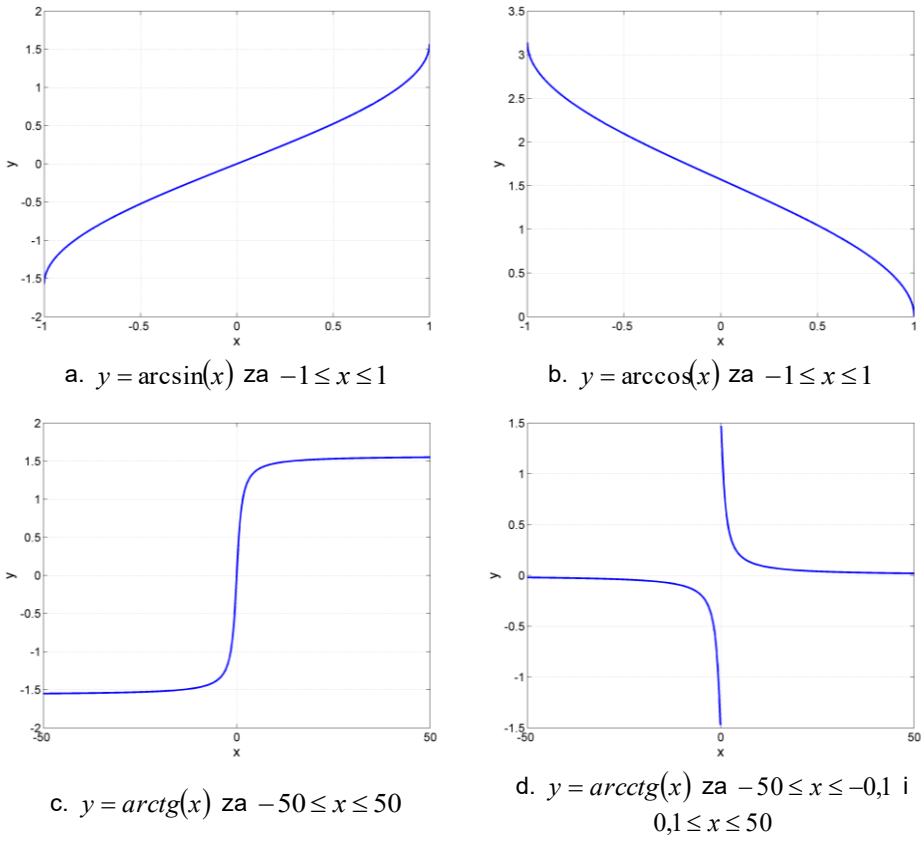
9.2 Ciklometrijske funkcije

U ciklometrijske funkcije ubrajaju se funkcije arkus sinus, arkus kosinus, arkus tangens i arkus kotanges. To su inverzne funkcije trigonometrijskih funkcija. U tablici 9.2 prikazane su domene i kodomene ciklometrijskih funkcija.

Tablica 9.2 Domene i kodomene ciklometrijskih funkcija

Funkcija	Domena funkcije	Kodomena funkcije
$y = \arcsin(x)$	$-1 \le x \le 1$	$-\pi/2 \le y \le \pi/2$
$y = \arccos(x)$	$-1 \le x \le 1$	$0 \le y \le \pi$
$y = arctg(x)$	$-\infty < x < \infty$	$-\pi/2 < y < \pi/2$
$y = arcctg(x)$	$-\infty < x < \infty$	$0 < y < \pi$

Na slici 9.2 prikazani su grafovi ciklometrijskih funkcija.



a. $y = \arcsin(x)$ za $-1 \le x \le 1$

b. $y = \arccos(x)$ za $-1 \le x \le 1$

c. $y = arctg(x)$ za $-50 \le x \le 50$

d. $y = arcctg(x)$ za $-50 \le x \le -0{,}1$ i $0{,}1 \le x \le 50$

Slika 9.2 Grafovi ciklometrijskih funkcija

asin

U Octaveu se za izračunavanje funkcije arkus sinus koristi funkcija **`asin(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija arkus sinus računa se element po element za vektore i matrice. Funkcija **`asin`** daje rezultat u radijanima.

Primjer 9.9: Funkcija **`asin`** primijenjena na matricu **`x`** dimenzija 2*3 daje matricu **`y`** dimenzija 2*3.

```
>> x = [0.5 0.8 0.7; -0.4 -0.2 -0.6]
x =
    0.50    0.80    0.70
   -0.40   -0.20   -0.60
>> y = asin(x)
y =
    0.52    0.93    0.78
   -0.41   -0.20   -0.64
```

```
>>
```

asind

Funkcija **asind(x)** daje rezultat u stupnjevima.

Primjer 9.10: Funkcija **asind** primijenjena na matricu **x** dimenzija 2*2 daje matricu **y** dimenzija 2*2.

```
>> x = [0.5 0.8; -0.4 -0.2]
x =
    0.50    0.80
   -0.40   -0.20
>> y = asind(x)
y =
   30.00   53.13
  -23.58  -11.54
>>
```

acos

U Octaveu se za izračunavanje funkcije arkus kosinus koristi funkcija **acos(x)**, gdje **x** može biti skalar, vektor ili matrica. Funkcija arkus kosinus računa se element po element za vektore i matrice. Funkcija **acos** daje rezultat u radijanima.

Primjer 9.11: Funkcija **acos** primijenjena na matricu **x** dimenzija 2*3 daje matricu **y** dimenzija 2*3.

```
>> x = [0.5 0.8 0.7; -0.4 -0.2 -0.6]
x =
    0.50    0.80    0.70
   -0.40   -0.20   -0.60
>> y = acos(x)
y =
    1.05    0.64    0.80
    1.98    1.77    2.21
>>
```

acosd

Funkcija **acosd(x)** daje rezultat u stupnjevima.

Primjer 9.12: Funkcija **acosd** primijenjena na matricu **x** dimenzija 2*2 daje matricu **y** dimenzija 2*2.

```
>> x = [0.5 0.8; -0.4 -0.2]
x =
    0.50    0.80
   -0.40   -0.20
>> y = acosd(x)
y =
   60.00   36.87
  113.58  101.54
>>
```

atan

U Octaveu se za izračunavanje funkcije arkus tangens koristi funkcija **`atan(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija arkus tangens računa se element po element za vektore i matrice. Funkcija **`atan`** daje rezultat u radijanima.

Primjer 9.13: Funkcija **`atan`** primijenjena na redni vektor **`x`** dimenzija 1*4 daje redni vektor **`y`** dimenzija 1*4.

```
>> x = [-50 -1 2 20]
x =
   -50.00    -1.00     2.00    20.00
>> y = atan(x)
y =
   -1.55   -0.79    1.11    1.52
>>
```

atand

Funkcija **`atand(x)`** daje rezultat u stupnjevima.

Primjer 9.14: Funkcija **`atand`** primijenjena na redni vektor **`x`** dimenzija 1*4 daje redni vektor **`y`** dimenzija 1*4.

```
>> x = [-50 -1 2 20]
x =
   -50.00    -1.00     2.00    20.00
>> y = atand(x)
y =
  -88.85  -45.00   63.43   87.14
>>
```

acot

U Octaveu se za izračunavanje funkcije arkus kotangens koristi funkcija **`acot(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija arkus kotangens računa se element po element za vektore i matrice. Funkcija **`acot`** daje rezultat u radijanima.

Primjer 9.15: Funkcija **`acot`** primijenjena na matricu **`x`** dimenzija 2*3 daje matricu **`y`** dimenzija 2*3.

```
>> x = [-8 -7 -4; 1 2 3]
x =
    -8.00    -7.00    -4.00
     1.00     2.00     3.00
>> y = acot(x)
y =
   -0.12   -0.14   -0.24
    0.79    0.46    0.32
>>
```

acotd

Funkcija **`acotd(x)`** daje rezultat u stupnjevima.

Primjer 9.16: Funkcija **`acotd`** primijenjena na matricu **`x`** dimenzija 2*3 daje matricu **`y`** dimenzija 2*3.

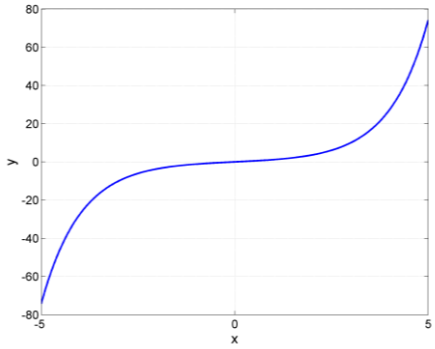
```
>> x = [-8 -7 -4; 1 2 3]
x =
    -8.00    -7.00    -4.00
     1.00     2.00     3.00
>> y = acotd(x)
y =
   -7.13   -8.13  -14.04
   45.00   26.57   18.43
>>
```

9.3 Hiperbolne funkcije

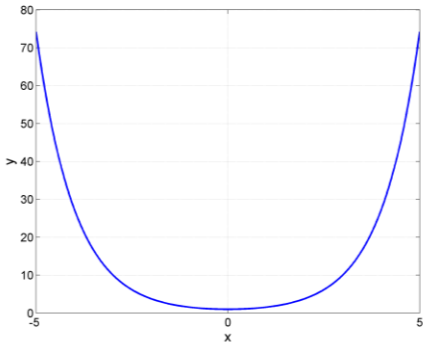
U hiperbolne funkcije ubrajaju se funkcije sinus hiperbolni, kosinus hiperbolni, tangens hiperbolni i kotangens hiperbolni. U tablici 9.3 prikazane su domene i kodomene hiperbolnih funkcija. Na slici 9.3 prikazani su grafovi hiperbolnih funkcija.

Tablica 9.3 Domene i kodomene hiperbolnih funkcija

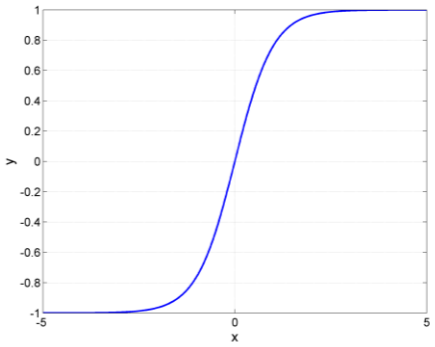
Funkcija	Domena	Kodomena
$y = sh(x)$	$-\infty < x < \infty$	$-\infty < y < \infty$
$y = ch(x)$	$-\infty < x < \infty$	$1 < y < \infty$
$y = th(x)$	$-\infty < x < \infty$	$-1 < y < 1$
$y = cth(x)$	$-\infty < x < \infty$	$-\infty < y < \infty$



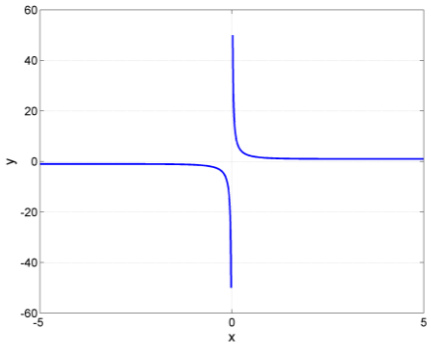
a. $y = sh(x)$ za $-5 \le x \le 5$



b. $y = ch(x)$ za $-5 \le x \le 5$



c. $y = th(x)$ za $-5 \le x \le 5$



d. $y = cth(x)$ za $-5 \le x \le -0{,}02$ i $0{,}02 \le x \le 5$

Slika 9.3 Grafovi hiperbolnih funkcija

Za hiperbolne funkcije vrijedi:

$$sh(x)=\frac{e^x-e^{-x}}{2} \qquad ch(x)=\frac{e^x+e^{-x}}{2} \qquad th(x)=\frac{sh(x)}{ch(x)}=\frac{e^x-e^{-x}}{e^x+e^{-x}} \qquad cth(x)=\frac{ch(x)}{sh(x)}=\frac{e^x+e^{-x}}{e^x-e^{-x}}$$

sinh

U Octaveu se za izračunavanje funkcije sinus hiperbolni koristi funkcija **`sinh(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija sinus hiperbolni računa se element po element za vektore i matrice.

Primjer 9.17: Funkcija **`sinh`** primijenjena na matricu **`x`** dimenzija 2*3 daje matricu **`y`** dimenzija 2*3.

```
>> x = [2 1 3; 5 -1 -2]
x =
     2.00     1.00     3.00
     5.00    -1.00    -2.00
>> y = sinh(x)
y =
    3.63    1.18   10.02
   74.20   -1.18   -3.63
>>
```

cosh

U Octaveu se za izračunavanje funkcije kosinus hiperbolni koristi funkcija **`cosh(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija kosinus hiperbolni računa se element po element za vektore i matrice.

Primjer 9.18: Funkcija **`cosh`** primijenjena na matricu **`x`** dimenzija 2*2 daje matricu **`y`** dimenzija 2*2.

```
>> x = [-1 -2; 2 3]
x =
    -1.00    -2.00
     2.00     3.00
>> y = cosh(x)
y =
    1.54    3.76
    3.76   10.07
>>
```

tanh

U Octaveu se za izračunavanje funkcije tangens hiperbolni koristi funkcija **`tanh(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija tangens hiperbolni računa se element po element za vektore i matrice.

Primjer 9.19: Funkcija **`tanh`** primijenjena na redni vektor **`x`** dimenzija 1*4 daje redni vektor **`y`** dimenzija 1*4.

```
>> x = [-5 -1 2 3]
x =
   -5.00   -1.00    2.00    3.00
>> y = tanh(x)
y =
   -1.00   -0.76    0.96    1.00
>>
```

coth

U Octaveu se za izračunavanje funkcije kotangens hiperbolni koristi funkcija **coth(x)**, gdje **x** može biti skalar, vektor ili matrica. Funkcija kotangens hiperbolni računa se element po element za vektore i matrice.

Primjer 9.20: Funkcija **coth** primijenjena na matricu **x** dimenzija 2*3 daje matricu **y** dimenzija 2*3.

```
>> x = [-1 -2 -3; 1 2 3]
x =
    -1.00    -2.00    -3.00
     1.00     2.00     3.00
>> y = coth(x)
y =
   -1.31   -1.04   -1.00
    1.31    1.04    1.00
>>
```

9.4 Logaritamske funkcije

Logaritam broja $x \in \Re^+$ baze $a \in N(a \neq 1)$ eksponent je kojim treba potencirati bazu a da se dobije broj x. Logaritamska funkcija inverzna je funkcija eksponencijalne funkcije. Svojstva logaritamske funkcije su:

$$\log_a(x \cdot y) = \log_a x + \log_a y \qquad \log_a \frac{x}{y} = \log_a x - \log_a y$$

$$\log_a x^k = k \cdot \log_a x \qquad \log_a \sqrt[n]{x} = \log_a x^{1/n} = \frac{1}{n} \cdot \log_a x$$

Veze različitih logaritamskih funkcija su:

$$\log_b x = \frac{\log_a x}{\log_a b} \qquad \log_a b = \frac{1}{\log_b a}$$

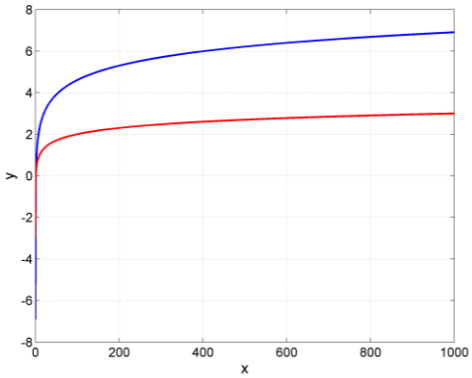
$$\log_{a^k} x = \frac{1}{k} \cdot \log_a x \qquad \log_{\frac{1}{a}} x = -\log_a x$$

Logaritmi brojeva baze 10 zovu se dekadski ili Briggsovi logaritmi. Označuju se sa $\log_{10} x$, odnosno $\log x$. Logaritmi brojeva baze e zovu se prirodni ili Napierovi logaritmi. Označuju se sa $\log_e x$, odnosno $\ln x$. U tablici 9.4 prikazane su domene i kodomene za prirodne logaritme, dekadske logaritme i eksponencijalnu funkciju.

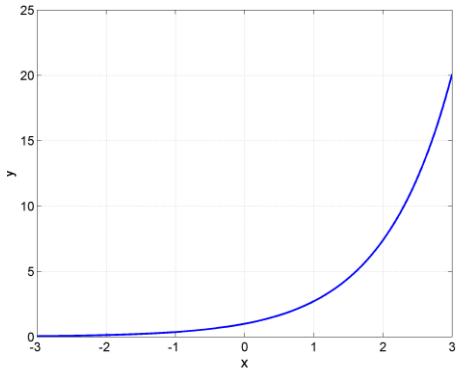
Tablica 9.4 Domene i kodomene za prirodne logaritme, dekadske logaritme i eksponencijalnu funkciju

Funkcija	Domena	Kodomena
$y = \ln(x)$	$0 < x < \infty$	$-\infty < y < \infty$
$y = \log(x)$	$0 < x < \infty$	$-\infty < y < \infty$
$y = e^x$	$-\infty < x < \infty$	$0 < y < \infty$

Na slici 9.4 prikazani su grafovi logaritamskih funkcija i eksponencijalne funkcije.



a. Funkcija prirodnog logaritma (plavo) $y = \ln(x)$ za $-0{,}001 \le x \le 1000$ i dekadskog logaritma (crveno) $y = \log(x)$ za $-0{,}001 \le x \le 1000$



b. Eksponencijalna funkcija $y = e^x$ za $-3 \le x \le 3$

Slika 9.4 Grafovi logaritamskih funkcija i eksponencijalne funkcije

log

U Octaveu se za izračunavanje funkcije prirodnog logaritma koristi funkcija **`log(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija prirodnog logaritma računa se element po element za vektore i matrice.

Primjer 9.21: Funkcija **`log`** primijenjena na matricu **`x`** dimenzija 2*3 daje matricu **`y`** dimenzija 2*3.

```
>> x = [0.5 0.8 0.7; 1.2 8.1 15.4]
x =
    0.50    0.80    0.70
    1.20    8.10   15.40
>> y = log(x)
y =
   -0.69   -0.22   -0.36
    0.18    2.09    2.73
>>
```

log10

U Octaveu se za izračunavanje funkcije dekadskog logaritma koristi funkcija **`log10(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Funkcija dekadskog logaritma računa se element po element za vektore i matrice.

Primjer 9.22: Funkcija **`log10`** primijenjena na matricu **`x`** dimenzija 2*3 daje matricu **`y`** dimenzija 2*3.

```
>> x = [0.2 0.6 0.8; 1.2 18.1 35.4]
x =
    0.20    0.60    0.80
    1.20   18.10   35.40
>> y = log10(x)
y =
   -0.70   -0.22   -0.10
    0.08    1.26    1.55
>>
```

9.5 Eksponencijalna funkcija

exp

U Octaveu se za izračunavanje eksponencijalne funkcije koristi funkcija **`exp(x)`**, gdje **`x`** može biti skalar, vektor ili matrica. Eksponencijalna funkcija računa se element po element za vektore i matrice.

Primjer 9.23: Funkcija **`exp`** primijenjena na redni vektor **`x`** dimenzija 1*4 daje redni vektor retka **`y`** dimenzija 1*4.

```
>> x = [0.1 2.1 3.5 0.7]
x =
    0.10    2.10    3.50    0.70
>> y = exp(x)
y =
    1.11    8.17   33.12    2.01
>>
```

Pitanja za provjeru znanja:

1. Koje trigonometrijske funkcije postoje u Octaveu?
2. Koja je razlika između funkcija **`sin`** i **`sind`**?
3. Koje ciklometrijske funkcije postoje u Octaveu?
4. Koja je razlika između funkcija **`acos`** i **`acosd`**?
5. Koje hiperbolne funkcije postoje u Octaveu?
6. Koje logaritamske funkcije postoje u Octaveu?
7. Može li argument funkcije **`sin`** biti matrica i ako da, što je rezultat?
8. Može li argument funkcije **`acos`** bit vektor i ako da, što je rezultat?
9. Može li argument funkcije **`exp`** biti negativan skalar?

10. STATISTIČKE FUNKCIJE

U svakodnevnom životu prikuplja se mnoštvo različitih podataka koje je potrebno na određeni način prikazati i usporediti. Statistika je znanost koja se bavi prikupljanjem, analizom, interpretacijom ili objašnjenjem i prezentiranjem podataka, te omogućuje donošenje zaključaka na osnovi informacija sadržanih u tim podatcima. Statističke metode koje se koriste za opisivanje prikupljenih podataka nazivaju se opisnom statistikom (engl. *descriptive statistics*). U parametre opisne statistike ubrajaju se raspon (najmanja i najveća vrijednost), suma, aritmetička sredina, standardna devijacija, varijanca, kovarijanca i koeficijent korelacije, koji su opisani u nastavku poglavlja.

min

Funkcija **`min`** pronalazi element vektora ili matrice najmanje vrijednosti. Oblik funkcije je:

`min(x)` – gdje je **`x`** redni ili stupčani vektor.

Primjer 10.1: Definiran je redni vektor **`x`** dimenzija 1*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U skalar **`y`** pohranjuje se najmanji element vektora **`x`**.

```
>> x = randi([1 20],1,5)
x =
    12.00    16.00    20.00    15.00     6.00
>> y = min(x)
y =
     6.00
>>
```

Ako se želi dobiti podatak najmanje vrijednosti elementa vektora te koji je indeks najmanjeg elementa vektora (odnosno njegov položaj), koristi se oblik funkcije:

`[y,i] = min(x)` – gdje je **`x`** vektor, **`y`** najmanji element vektora, a **`i`** indeks (položaj) najmanjeg elementa vektora.

Primjer 10.2: Definiran je redni vektor **`x`** dimenzija 1*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U skalar **`y`** pohranjuje se najmanji element vektora **`x`**, a u skalar **`i`** položaj najmanjeg elementa vektora **`x`**.

```
>> x = randi([1 20],1,5)
x =
    15.00    20.00     7.00    17.00    11.00
>> [y,i] = min(x)
y =
     7.00
i =
     3.00
>>
```

Ako je **x** matrica, tada je oblik funkcije:

min(x,[],1) – gdje je **x** matrica, a **1** označuje da se želi pronaći najmanji element po redcima za svaki stupac

min(x,[],2) – gdje je **x** matrica, a **2** označuje da se želi pronaći najmanji element po stupcima za svaki redak.

Primjer 10.3: Definirana je matrica **x** dimenzija 3*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **y** pohranjuju najmanji elementi matrice **x**, po redcima za svaki stupac. U drugom se slučaju u vektor **y** pohranjuju najmanji elementi matrice **x**, po stupcima za svaki redak.

```
>> x = randi([1 20],3,5)
x =
    8.00   16.00   14.00   10.00   15.00
   14.00   11.00   15.00    7.00   11.00
    9.00   17.00   17.00   19.00    8.00
>> y = min(x,[],1)
y =
    8.00   11.00   14.00    7.00    8.00
>> y = min(x,[],2)
y =
   8.00
   7.00
   8.00
>>
```

Ako se želi dobiti podatak najmanje vrijednosti elementa matrice te koji je indeks retka, odnosno stupca najmanjeg elementa matrice u retku ili stupcu (odnosno njegov položaj), koristi se oblik funkcije:

[y,i] = min(x,[],1) – gdje je **x** matrica, **1** označuje da se želi pronaći najmanji element po redcima za svaki stupac, **y** je najmanji element po redcima matrice za svaki stupac, a **i** je indeks (položaj) elementa po redcima

[y,i] = min(x,[],2) – gdje je **x** matrica, **2** označuje da se želi pronaći najmanji element po stupcima za svaki redak, **y** je najmanji element po stupcima matrice za svaki redak, a **i** je indeks (položaj) elementa po stupcima.

Primjer 10.4: Definirana je matrica **x** dimenzija 3*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **y** pohranjuju najmanji elementi matrice **x**, po redcima za svaki stupac, a u vektor **i** njihov položaj. U drugom se slučaju u vektor **y** pohranjuju najmanji elementi matrice **x**, po stupcima za svaki redak, a u vektor **i** njihov položaj.

```
>> x = randi([1 20],3,5)
x =
    2.00    9.00   10.00    2.00    4.00
    6.00    5.00    6.00    2.00    1.00
    4.00    8.00   15.00    3.00   15.00
>> [y,i] = min(x,[],1)
y =
   2.00   5.00   6.00   2.00   1.00
i =
   1.00   2.00   2.00   1.00   2.00
>> [y,i] = min(x,[],2)
y =
   2.00
   1.00
   3.00
i =
   1.00
   5.00
   4.00
>>
```

U slučaju da ima više istih najmanjih vrijednosti, Octave daje podatak o prvoj pronađenoj najmanjoj vrijednosti.

max

Funkcija **max** pronalazi element vektora ili matrice najveće vrijednosti. Oblik funkcije je:

max(x) – gdje je **x** redni ili stupčani vektor.

Primjer 10.5: Definiran je redni vektor **x** dimenzija 1*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U skalar **y** pohranjuje se najveći element vektora **x**.

```
>> x = randi([1 20],1,5)
x =
     7.00     19.00      6.00     11.00      6.00
>> y = max(x)
y =
    19.00
>>
```

Ako se želi dobiti podatak najveće vrijednosti elementa vektora te koji je indeks najvećeg elementa vektora (odnosno njegov položaj), koristi se oblik funkcije:

[y,i] = max(x) – gdje je **x** vektor, **y** najveći element vektora, a **i** indeks (položaj) najvećeg elementa vektora.

Primjer 10.6: Definiran je redni vektor **x** dimenzija 1*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U skalar **y** pohranjuje se najveći element vektora **x**, a u skalar **i** položaj najvećeg elementa vektora **x**.

```
>> x = randi([1 20],1,5)
x =
    20.00     11.00     12.00     17.00     19.00
>> [y,i] = max(x)
y =
    20.00
i =
     1.00
>>
```

Ako je **x** matrica, tada je oblik funkcije:

max(x,[],1) – gdje je **x** matrica, a **1** označuje da se želi pronaći najveći element po redcima za svaki stupac

max(x,[],2) – gdje je **x** matrica, a **2** označuje da se želi pronaći najveći element po stupcima za svaki redak.

Primjer 10.7: Definirana je matrica **x** dimenzija 3*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **y** pohranjuju najveći elementi matrice **x**, po redcima za svaki stupac. U drugom se slučaju u vektor **y** pohranjuju najveći elementi matrice **x**, po stupcima za svaki redak.

```
>> x = randi([1 20],3,5)
x =
   12.00    6.00    1.00    2.00   13.00
    5.00    3.00   15.00   10.00   14.00
```

```
    8.00   17.00    5.00   13.00   11.00
>> y = max(x,[],1)
y =
   12.00   17.00   15.00   13.00   14.00
>> y = max(x,[],2)
y =
   13.00
   15.00
   17.00
>>
```

Ako se želi dobiti podatak najveće vrijednosti elementa matrice te koji je indeks retka, odnosno stupca najvećeg elementa matrice (odnosno njegov položaj), koristi se oblik funkcije:

`[y,i] = max(x,[],1)` – gdje je **`x`** matrica, **`1`** označuje da se želi pronaći najveći element po redcima za svaki stupac, **`y`** je najveći element po redcima matrice za svaki stupac, a **`i`** je indeks (položaj) elementa po redcima

`[y,i] = max(x,[],2)` – gdje je **`x`** matrica, **`2`** označuje da se želi pronaći najveći element po stupcima za svaki redak, **`y`** je najveći element po stupcima matrice za svaki redak, a **`i`** je indeks (položaj) elementa po stupcima.

Primjer 10.8: Definirana je matrica **`x`** dimenzija 3*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **`y`** pohranjuju najveći elementi matrice **`x`**, po redcima za svaki stupac, a u vektor **`i`** njihov položaj. U drugom se slučaju u vektor **`y`** pohranjuju najveći elementi matrice **`x`**, po stupcima za svaki redak, a u vektor **`i`** njihov položaj.

```
>> x = randi([1 20],3,5)
x =
   19.00   17.00   13.00   20.00   16.00
    9.00    6.00   13.00   16.00    5.00
    3.00   14.00    8.00   11.00    6.00
>> [y,i] = max(x,[],1)
y =
   19.00   17.00   13.00   20.00   16.00
i =
   1.00   1.00   1.00   1.00   1.00
>> [y,i] = max(x,[],2)
y =
   20.00
   16.00
   14.00
i =
   4.00
   4.00
   2.00
>>
```

U slučaju da ima više istih najvećih vrijednosti, Octave daje podatak o prvoj pronađenoj najvećoj vrijednosti.

sum

Funkcija **`sum`** izračunava zbroj elemenata vektora ili matrica. Oblik funkcije je:

`sum(x)` – gdje je **`x`** vektor.

Primjer 10.9: Definiran je redni vektor **`x`** dimenzija 1*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U skalar **`y`** pohranjuje se zbroj elementa vektora **`x`**.

```
>> x = randi([1 20],1,5)
x =
```

```
     2.00    19.00     6.00    11.00    15.00
>> y = sum(x)
y =
    53.00
>>
```

Ako je **x** matrica tada je oblik funkcije:

sum(x,1) – rezultat je zbroj elemenata po redcima za svaki stupac matrice **x**

sum(x,2) – rezultat je zbroj elemenata po stupcima za svaki redak matrice **x**.

Primjer 10.10: Definirana je matrica **x** dimenzija 3*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **y** pohranjuju zbroj elementa matrice **x**, po redcima za svaki stupac. U drugom se slučaju u vektor **y** pohranjuju zbroj elementa matrice **x**, po stupcima za svaki redak.

```
>> x = randi([1 20],3,5)
x =
    8.00   17.00   17.00    3.00   15.00
    4.00    6.00   15.00    7.00   15.00
    1.00    2.00    3.00    6.00   10.00
>> y = sum(x,1)
y =
   13.00   25.00   35.00   16.00   40.00
>> y = sum(x,2)
y =
   60.00
   47.00
   22.00
>>
```

Ako se želi izračunati zbroj svih elemenata matrice **x**, to se može postići pomoću funkcije **sum(sum(x))**.

Primjer 10.11: Definirana je matrica **x** dimenzija 3*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U skalar **y** pohranjuje se zbroj svih elementa matrice **x**.

```
>> x = randi([1 20],3,5)
x =
   18.00   19.00   12.00   19.00   11.00
    1.00    7.00    7.00   12.00    2.00
    6.00   15.00   14.00   10.00   14.00
>> y = sum(sum(x))
y =  167.00
>>
```

mean

Pri mjerenju neke varijabilne veličine X (statističko obilježje), dobit će se n izmjerenih vrijednosti (statistički niz ili uzorak je $x_1, x_2, \ldots, x_{n-1}, x_n$). Aritmetička sredina $\bar{x}$ statističkog niza ili uzorka je:

$$\bar{x} = \frac{x_1 + x_2 + \ldots + x_{n-1} + x_n}{n} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Funkcija **mean** izračunava aritmetičku sredinu elemenata vektora ili matrice. Oblik funkcije je:

mean(x) – gdje je **x** vektor.

Primjer 10.12: Definiran je redni vektor **x** dimenzija 1*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U skalar **y** pohranjuje se aritmetička sredina elemenata vektora **x**.

```
>> x = randi([1 20],1,5)
x =
    6.00    5.00    9.00   20.00   18.00
>> y = mean(x)
y =  11.60
>>
```

Ako je **x** matrica, oblik funkcije je:

mean(x,1) – rezultat je aritmetička sredina elemenata po redcima za svaki stupac matrice **x**

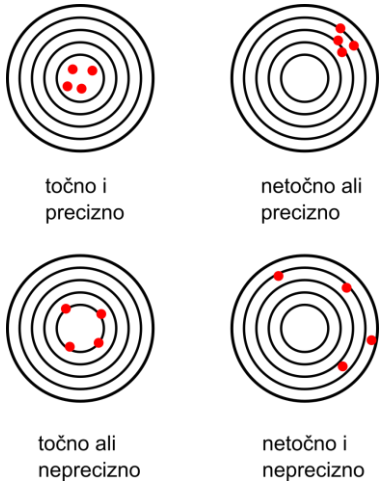
mean(x,2) – rezultat je aritmetička sredina elemenata po stupcima za svaki redak matrice **x**.

Primjer 10.13: Definirana je matrica **x** dimenzija 3*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **y1** pohranjuju aritmetičke sredine elemenata matrice **x**, po redcima za svaki stupac. U drugom se slučaju u vektor **y2** pohranjuju aritmetičke sredine elemenata matrice **x**, po stupcima za svaki redak.

```
>> x = randi([1 20],3,5)
x =
   10.00    5.00    9.00    1.00    7.00
   17.00    6.00    6.00    8.00    6.00
    6.00    1.00    9.00   15.00   20.00
>> y1 = mean(x,1)
y1 =
   11.00    4.00    8.00    8.00   11.00
>> y2 = mean(x,2)
y2 =
    6.40
    8.60
   10.20
>>
```

std

Pojedinačni rezultati mjerenja međusobno se manje razlikuju što je mjerni postupak precizniji. Preciznost je mjera pouzdanosti mjernog uređaja ili bilo čega drugoga. Ako se npr. duljina procjenjuje preko vizualnog dojma, preciznost je mala. Ako se upotrijebi ravnalo preciznost je veća. Preciznost se često pogrešno upotrebljava umjesto pojma točnost. Za razliku od točnosti, preciznost se ne može definirati za jedno mjerenje, preciznost je sposobnost mjernog uređaja da se ponovnim mjerenjem izmjerena veličina znatno ne mijenja, dok točnost opisuje odstupanje izmjerene veličine od njene stvarne vrijednosti. Razlika između preciznosti i točnosti prikazana je na slici 10.1.



Slika 10.1 Preciznost i točnost

Brojčanu procjenu preciznosti mjernog postupka daje standardno odstupanje (standardna devijacija):

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}$$

Funkcija **std** izračunava standardno odstupanje elemenata vektora ili matrice. Oblik funkcije je:

std(x,0) – gdje je **x** vektor, a **0** označuje da će se računati standardno odstupanje za n-1 uzoraka

std(x,1) – gdje je **x** vektor, a **1** označuje da će se računati standardno odstupanje za n uzoraka.

Primjer 10.14: Definiran je redni vektor **x** dimenzija 1*10 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u skalar **y1** pohranjuje standardno odstupanje elemenata vektora **x**, za n-1 uzoraka. U drugom se slučaju u skalar **y2** pohranjuje standardno odstupanje elemenata vektora **x**, za n uzoraka.

```
>> x = randi([1 20],1,10)
x =
    4.00   20.00    2.00    8.00    7.00    9.00    4.00   18.00    2.00
10.00
>> y1 = std(x,0)
y1 =  6.26
>> y2 = std(x,1)
y2 =  5.94
>>
```

Ako je **x** matrica, oblik funkcije je:

std(x,0,1) – rezultat je standardno odstupanje elemenata po redcima za svaki stupac matrice **x**

std(x,0,2) – rezultat je standardno odstupanje elemenata po stupcima za svaki redak matrice **x**, gdje u oba oblika funkcije **0** iza **x** označuje da će se računati standardno odstupanje za n-1 uzoraka

std(x,1,1) – rezultat je standardno odstupanje elemenata po redcima za svaki stupac matrice **x**

std(x,1,2) – rezultat je standardno odstupanje elemenata po stupcima za svaki redak matrice **x**, gdje u oba oblika funkcije **1** iza **x** označuje da će se računati standardno odstupanje za n uzoraka.

Primjer 10.15: Definirana je matrica **x** dimenzija 5*10 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **y1** pohranjuje standardno odstupanje elemenata matrice **x**, po redcima za svaki stupac matrice **x**, za n-1 uzoraka. U drugom se slučaju u vektor **y2** pohranjuje standardno odstupanje elemenata matrice **x**, po stupcima za svaki redak matrice **x**, za n-1 uzoraka. U trećem se slučaju u vektor **y3** pohranjuje standardno odstupanje elemenata matrice **x**, po redcima za svaki stupac matrice **x**, za n uzoraka. U četvrtom se slučaju u vektor **y4** pohranjuje standardno odstupanje elemenata matrice **x**, po stupcima za svaki redak matrice **x**, za n uzoraka.

```
>> x = randi([1 20],5,10)
x =
   19.00   14.00   11.00    7.00    7.00   19.00    9.00   18.00   16.00
7.00
   19.00   20.00   16.00   19.00   11.00    4.00    2.00    6.00   11.00
18.00
   12.00    1.00   19.00   14.00   18.00   13.00    9.00   15.00   13.00
4.00
   10.00    1.00   20.00   15.00    2.00   15.00   16.00    1.00   11.00
9.00
    2.00    1.00    6.00    9.00    9.00    4.00    9.00    6.00    3.00
13.00
>> y1 = std(x,0,1)
y1 =
   7.09   9.02   5.86   4.82   5.86   6.75   4.95   7.05   4.82   5.45
>> y2 = std(x,0,2)
y2 =
```

```
    5.10
    6.77
    5.71
    6.78
    3.79
>> y3 = std(x,1,1)
y3 =
    6.34    8.06    5.24    4.31    5.24    6.03    4.43    6.31    4.31    4.87
>> y4 = std(x,1,2)
y4 =
    4.84
    6.42
    5.42
    6.43
    3.60
>>
```

var

Uz standardno odstupanje kao procjena preciznosti mjernih rezultata koristi se i varijanca:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2$$

Funkcija **var** izračunava varijancu elemenata vektora ili matrice. Oblik funkcije je:

var(x,0) – gdje je **x** vektor, a **0** označuje da će se računati varijanca za n-1 uzoraka

var(x,1) – gdje je **x** vektor, a **1** označuje da će se računati varijanca za n uzoraka.

Primjer 10.16: Definiran je redni vektor **x** dimenzija 1*10 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u skalar **y1** pohranjuje varijanca elemenata vektora **x**, za n-1 uzoraka. U drugom se slučaju u skalar **y2** pohranjuje varijanca elemenata vektora **x**, za n uzoraka.

```
>> x = randi([1 20],1,10)
x =
     3.00    18.00    18.00    19.00    18.00    10.00     5.00    13.00     5.00
10.00
>> y1 = var(x,0)
y1 =  38.32
>> y = var(x,1)
y2 =  34.49
>>
```

Ako je **x** matrica tada je oblik funkcije:

var(x,0,1) – rezultat je varijanca elemenata po redcima za svaki stupac matrice **x**

var(x,0,2) – rezultat je varijanca elemenata po stupcima za svaki redak matrice **x**, gdje u oba oblika funkcije **0** iza **x** označuje da će se računati varijanca za n-1 uzoraka

var(x,1,1) – rezultat je varijanca elemenata po redcima za svaki stupac matrice **x**

var(x,1,2) – rezultat je varijanca elemenata po stupcima za svaki redak matrice **x**, gdje u oba oblika funkcije **1** iza **x** označuje da će se računati varijanca za n uzoraka.

Primjer 10.17: Definirana je matrica **x** dimenzija 5*10 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,20]. U prvom se slučaju u vektor **y1** pohranjuje varijanca elemenata matrice **x**, po redcima za svaki stupac matrice **x**, za n-1 uzoraka. U drugom se slučaju u vektor **y2** pohranjuje varijanca elemenata matrice **x**, po stupcima za svaki redak matrice **x**, za n-1 uzoraka. U trećem se slučaju u vektor **y3** pohranjuje varijanca elemenata matrice **x**, po redcima za svaki stupac matrice **x**,

za n uzoraka. U četvrtom se slučaju u vektor **y4** pohranjuje varijanca elemenata matrice **x**, po stupcima za svaki redak matrice **x**, za n uzoraka.

```
>> x = randi([1 20],5,10)
x =
    1.00     6.00    18.00     4.00     2.00    16.00     6.00     4.00    12.00
15.00
    3.00    13.00    14.00    10.00    10.00     5.00     9.00     9.00    14.00
20.00
    9.00     9.00     1.00     8.00    11.00     5.00    12.00     2.00     7.00
13.00
   20.00    13.00    10.00    13.00    11.00    18.00    16.00     3.00    13.00
12.00
   13.00     3.00     1.00    12.00    19.00    15.00    20.00     7.00     6.00
16.00
>> y1 = var(x,0,1)
y1 =
   59.20    19.20    58.70    12.80    36.30    39.70    30.80     8.50    13.30
9.70
>> y2 = var(x,0,2)
y2 =
   39.16
   23.57
   16.23
   21.88
   43.96
>> y3 = var(x,1,1)
y3 =
   47.36    15.36    46.96    10.24    29.04    31.76    24.64     6.80    10.64
7.76
>> y4 = var(x,1,2)
y4 =
   35.24
   21.21
   14.61
   19.69
   39.56
>>
```

cov

Kovarijanca pokazuje koliko se dvije varijable mijenjaju zajedno. To je različito od varijance koja pokazuje koliko se jedna varijabla mijenja. Kovarijanca postaje pozitivnija za svaki par vrijednosti koji se razlikuje od njihovih srednjih vrijednosti u istom smjeru, te postaje negativnija za svaki par vrijednosti koji se razlikuje od njihovih srednjih vrijednosti u suprotnim smjerovima. Neka su X, Y i Z tri statistička obilježja (slučajne varijable) i postoje tri niza vrijednosti $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ i $(z_1, z_2, \ldots, z_n)$, tada su njihove kovarijance:

$$s_{xy}^2 = s_{yx}^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\cdot(y_i - \bar{y})}{n-1};\ s_{xz}^2 = s_{zx}^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\cdot(z_i - \bar{z})}{n-1};\ s_{yz}^2 = s_{zy}^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})\cdot(z_i - \bar{z})}{n-1}$$

ili u matričnom obliku:

$$C = \begin{bmatrix} s_{xx}^2 & s_{xy}^2 & s_{xz}^2 \\ s_{yx}^2 & s_{yy}^2 & s_{yz}^2 \\ s_{zx}^2 & s_{zy}^2 & s_{zz}^2 \end{bmatrix}$$

Važno je uočiti da vrijedi $s_{xy}^2 = s_{yx}^2$, $s_{xz}^2 = s_{zx}^2$ i $s_{yz}^2 = s_{zy}^2$. Općenito, ako postoji m nizova od n podataka, kovarijanca se može napisati kao matrica m*m (matrica kovarijanci je simetrična):

$$C = \begin{bmatrix} s_{11}^2 & s_{12}^2 & \cdots & s_{1m}^2 \\ s_{21}^2 & s_{22}^2 & \cdots & s_{2m}^2 \\ \vdots & \vdots & \vdots & \vdots \\ s_{m1}^2 & s_{m2}^2 & \cdots & s_{mm}^2 \end{bmatrix}$$

Funkcija **cov** izračunava kovarijancu elemenata matrice po redcima za svaki stupac (za vektor je kovarijanca jednaka varijanci). Oblik funkcije je:

cov(x,0) – gdje je **x** matrica, a **0** označuje da će se računati kovarijanca za n-1 uzoraka

cov(x,1) – gdje je **x** matrica, a **1** označuje da će se računati kovarijanca za n uzoraka.

Primjer 10.18: Definirana je matrica **x** dimenzija 5*5 čiji su elementi slučajni cijeli brojevi iz jednolike razdiobe u rasponu [1,10]. U prvom se slučaju u matricu **y1** pohranjuju kovarijance matrice **x**, za n-1 uzoraka. U drugom se slučaju u matricu **y2** pohranjuju kovarijance matrice **x**, za n uzoraka. Važno je uočiti da je matrica kovarijanci simetrična s obzirom na glavnu dijagonalu.

```
>> x = randi([1 10],5,5)
x =
     3.00     6.00     4.00    10.00     5.00
     1.00     6.00     7.00     9.00     1.00
     4.00     7.00     2.00     4.00     9.00
     9.00     5.00     1.00     3.00     5.00
     1.00     1.00    10.00     9.00     2.00
>> y1 = cov(x,0)
y1 =
    10.80     2.00   -10.10    -8.75     4.95
     2.00     5.50    -6.25    -2.25     4.00
   -10.10    -6.25    13.70     9.00    -8.90
    -8.75    -2.25     9.00    10.50    -6.50
     4.95     4.00    -8.90    -6.50     9.80
>> y2 = cov(x,1)
y2 =
     8.64     1.60    -8.08    -7.00     3.96
     1.60     4.40    -5.00    -1.80     3.20
    -8.08    -5.00    10.96     7.20    -7.12
    -7.00    -1.80     7.20     8.40    -5.20
     3.96     3.20    -7.12    -5.20     7.84
>>
```

corrcoef

Koeficijent korelacije između dvije slučajne varijable x i y s aritmetičkim sredinama $\bar{x}$ i $\bar{y}$ te standardnim odstupanjima s_x i s_y definiran je izrazom:

$$r_{xy} = \frac{s_{xy}^2}{\sqrt{s_{xx}^2 s_{yy}^2}} = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\cdot\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2 \cdot \sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}}$$

Napomena: $s_{xx}^2 = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)\cdot\left(x_i - \bar{x}\right)}{n-1} = \frac{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}{n-1} = s_x^2$

Koeficijent korelacije pokazuje stupanj linearne ovisnosti između varijabli. Što je koeficijent korelacije bliže 1 ili -1, veća je korelacija između varijabli. Ako su varijable neovisne, koeficijent korelacije je 0, ali obratno ne vrijedi (ako su dvije varijable ovisne, njihov koeficijent korelacije može biti 0) zbog toga što koeficijent korelacije ustanovljuje jedino linearnu ovisnost između varijabli. Pretpostavke kod računanja koeficijenta korelacije su:

- linearni odnos između dviju varijabli x i y

- kontinuirane slučajne varijable
- obje varijable moraju imati normalnu razdiobu
- varijable x i y moraju biti neovisne jedna o drugoj.

Ako postoje tri niza vrijednosti $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ i $(z_1, z_2, \ldots, z_n)$, tada su njihovi koeficijenti korelacije:

$$r_{xy} = r_{yx} = \frac{s_{xy}^2}{\sqrt{s_{xx}^2 s_{yy}^2}};\ r_{xz} = r_{zx} = \frac{s_{xz}^2}{\sqrt{s_{xx}^2 s_{zz}^2}};\ r_{yz} = r_{zy} = \frac{s_{yz}^2}{\sqrt{s_{yy}^2 s_{zz}^2}}$$

ili u matričnom obliku:

$$r = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix}$$

Važno je uočiti da vrijedi $r_{xy} = r_{yx}$, $r_{xz} = r_{zx}$ i $r_{yz} = r_{zy}$. Općenito, ako postoji m nizova od n mjerenja, koeficijent korelacije može se napisati kao matrica m*m (matrica koeficijenta korelacije je simetrična):

$$r = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ r_{m1} & r_{m2} & \cdots & r_{mm} \end{bmatrix}$$

Funkcija **corrcoef** izračunava korelaciju između dva vektora ili za matricu po redcima za svaki stupac. Oblik funkcije je:

[r,p] = corrcoef(x,y) – gdje je **x** vektor koji sadrži vrijednosti varijable x, **y** vektor koji sadrži vrijednosti varijable y, matrica **r** sadrži izračunate vrijednosti korelacije za te dvije varijable, a matrica **p** sadrži izračunate koeficijente značajnosti

[r,p] = corrcoef(x) – gdje je **x** matrica koja sadrži vrijednosti varijabli za koje se želi izračunati korelacija, matrica **r** sadrži izračunate vrijednosti korelacije za te varijable, a matrica **p** sadrži izračunate koeficijente značajnosti.

Primjer 10.19: Stvoren je stupčani vektor **x** dimenzija 10*1 s rastućim vrijednostima u rasponu [-2,2]. Vektor **y** se izračunava kao zbroj elementa vektora **x** i vrijednosti koje su nastale iz normalne (Gaussove) razdiobe. Matrica **r** sadrži izračunate vrijednosti korelacije između vektora **x** i **y**, a matrica **p** sadrži izračunate koeficijente značajnosti.

```
>> x = linspace(-2,2,10)'
x =
  -2.00
  -1.56
  -1.11
  -0.67
  -0.22
   0.22
   0.67
   1.11
   1.56
   2.00
>> y = x + 0.5*randn(10,1)
y =
  -2.57
  -1.33
  -0.57
  -0.62
```

```
  -0.69
  -0.59
   0.51
   1.01
   1.12
   0.82
>> r = corr(x,y)
r = 0.93
>>
```

Primjer 10.20: Stvoren je stupčani vektor **x** dimenzija 10*1 s rastućim vrijednostima u rasponu [-2,2]. Zatim se stvara matrica **x**. Drugi stupac matrice **x** izračunava se kao zbroj elementa prvog stupca matrice **x** i vrijednosti koje su nastale iz normalne (Gaussove) razdiobe. Treći stupac matrice **x** izračunava se kao zbroj elementa prvog stupca matrice **x** i vrijednosti koje su nastale iz normalne (Gaussove) razdiobe. Četvrti stupac matrice **x** izračunava se kao zbroj elementa prvog stupca matrice **x** i vrijednosti koje su nastale iz normalne (Gaussove) razdiobe. Matrica **r** sadrži izračunate vrijednosti korelacije između svih stupca matrice **x**, a matrica **p** sadrži izračunate koeficijente značajnosti.

```
>> x = linspace(-2,2,10)';
>> x(:,2) = x(:,1) + 0.2*randn(10,1);
>> x(:,3) = x(:,1) + 0.5*randn(10,1);
>> x(:,4) = x(:,1) + 1.0*randn(10,1)
x =
  -2.00  -1.72  -3.46  -1.19
  -1.56  -1.72  -1.79  -1.58
  -1.11  -1.07  -1.08  -1.55
  -0.67  -0.35  -0.84  -0.17
  -0.22  -0.03   0.51  -1.73
   0.22   0.45  -0.44   0.69
   0.67   0.66   0.79   0.87
   1.11   1.36   1.09   1.92
   1.56   1.69   0.93   3.32
   2.00   1.62   2.26   2.43
>> r = corr(x)
r =
   1.00   0.99   0.94   0.90
   0.99   1.00   0.91   0.90
   0.94   0.91   1.00   0.71
   0.90   0.90   0.71   1.00
>>
```

Pitanja za provjeru znanja:

1. Koja se funkcija u Octaveu koristi za pronalaženje najmanje vrijednosti elemenata vektora ili matrice?
2. Čemu služi funkcija **max**?
3. Koja funkcija u Octaveu izračunava zbroj elementa vektora ili matrice?
4. Čemu služi funkcija **mean**?
5. Koja funkcija u Octaveu izračunava standardno odstupanje vektora ili matrice?
6. Čemu služi funkcija **var**?
7. Čemu služi funkcija **cov**?
8. Koja funkcija u Octaveu izračunava koeficijent korelacije?

11. FUNKCIJE ZA RAD S POLINOMIMA

Polinomi su funkcije oblika:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

gdje su: $a_i \in \Re$ za $i = 0,1,2,\ldots,n$ koeficijenti polinoma, a $n \in N$ predstavlja stupanj polinoma.

U Octaveu se polinomi prikazuju vektorom čiji su elementi koeficijenti polinoma. Prvi element vektora je koeficijent polinoma najvišeg stupnja, pa zatim niži itd., odnosno u vektoru su koeficijenti polinoma poredani od najvišeg stupnja do najnižeg. U tablici 11.1 prikazani su primjeri polinoma i njihovi prikazi u Octaveu.

Tablica 11.1 Prikaz polinoma u Octaveu

Polinom	Prikaz u Octaveu
$f(x) = 2x + 3$	`p = [2 3]`
$f(x) = x^2 - 2x + 5$	`p = [1 -2 5]`
$f(x) = 3x^3 + x - 10$	`p = [3 0 1 -10]`

U zadnjem primjeru polinoma u tablici 11.1 koeficijent polinoma uz x^2 je 0. Uobičajeno je da se u matematici koeficijenti polinoma jednaki 0 i potencije varijable x uz te koeficijente ne prikazuju. U Octaveu je potrebno navesti sve vrijednosti koeficijenta polinoma bez obzira jesu li neke od njih jednake 0.

polyval

Funkcija **`polyval`** izračunava vrijednost polinoma u određenim točkama. Oblik naredbe je:

`polyval(p,x)` – gdje je **`p`** vektor koji sadrži koeficijente polinoma, a **`x`** je skalar, vektor ili matrica koji sadrži elemente što predstavljaju vrijednosti točaka u kojima se želi izračunati vrijednost polinoma.

Primjer 11.1: Za polinom $f(x) = 2x + 3$ izračunava se vrijednost polinoma u točki $f(5) = 13$.

```
>> p = [2 3]
p =
     2.00    3.00
>> x = 5
x =
     5.00
>> y = polyval(p,x)
y =  13.00
>>
```

Primjer 11.2: Za polinom $f(x)=x^2-2x+5$ izračunava se vrijednost polinoma u točkama $f(0)=5$, $f(1)=4$, $f(2)=5$, $f(3)=8$, $f(4)=13$ i $f(5)=20$.

```
>> p = [1 -2 5]
p =
     1.00    -2.00     5.00
>> x = 0:1:5
x =
     0.00     1.00     2.00     3.00     4.00     5.00
>> y = polyval(p,x)
y =
     5.00     4.00     5.00     8.00    13.00    20.00
>>
```

Primjer 11.3: Za polinom $f(x)=3x^3+x-10$ izračunava se vrijednost polinoma u točkama $f(-2)=-36$; $f(-1{,}5)=-21{,}625$; $f(-1)=-14$; $f(-0{,}5)=-10{,}875$; $f(0)=-10$; $f(0{,}5)=-9{,}125$; $f(1)=-6$; $f(1{,}5)=1{,}625$ i $f(2)=16$.

```
>> p = [3 0 1 -10]
p =
     3.00     0.00     1.00   -10.00
>> x = -2:0.5:2
x =
   -2.00   -1.50   -1.00   -0.50    0.00    0.50    1.00    1.50    2.00
>> y = polyval(p,x)
y =
  -36.00  -21.62  -14.00  -10.88  -10.00   -9.12   -6.00    1.62   16.00
>>
```

roots

Funkcija **`roots`** izračunava nultočke polinoma, odnosno vrijednosti varijable x za koje je vrijednost polinoma jednaka nuli. Oblik naredbe je:

`roots(p)` – gdje je **`p`** vektor čiji su elementi koeficijenti polinoma.

Rješenja polinoma vrijednosti su varijable x za koje je vrijednost polinoma jednaka nuli, pa se često zovu i nultočke polinoma ili korijeni polinoma. Za polinom oblika $f(x)=a_nx^n+a_{n-1}x^{n-1}+\ldots+a_1x+a_0$ potrebno je pronaći takve vrijednosti varijable x da je vrijednost polinoma u tim točkama jednaka nuli $a_nx^n+a_{n-1}x^{n-1}+\ldots+a_1x+a_0=0$.

Primjer 11.4: Za polinom $f(x)=x+3$ izračunava se nultočaka polinoma koja se nalazi u $f(-3)=0$.

```
>> p = [1 3]
p =
     1.00    3.00
>> r = roots(p)
r = -3.00
>>
```

Primjer 11.5: Za polinom $f(x)=x^2-2x-3$ izračunavaju se nultočke polinoma koje se nalaze u $f(3)=0$ i $f(-1)=0$.

```
>> p = [1 -2 -3]
p =
    1.00   -2.00   -3.00
>> r = roots(p)
r =
    3.00
   -1.00
>>
```

Primjer 11.6: Za polinom $f(x)=x^5-12{,}1x^4+40{,}59x^3-17{,}015x^2-71{,}95x+35{,}88$ izračunavaju se nultočke polinoma koje se nalaze u $f(6{,}5)=0$; $f(4)=0$, $f(2{,}3)=0$; $f(-1{,}2)=0$ i $f(0{,}5)=0$.

```
>> p = [1 -12.1 40.59 -17.015 -71.95 35.88]
p =
    1.00  -12.10   40.59  -17.02  -71.95   35.88
>> r = roots(p)
r =
    6.50
    4.00
    2.30
   -1.20
    0.50
>>
```

poly

Funkcija **`poly`** izračunava koeficijente polinoma ako su poznate njegove nultočke, odnosno korijeni polinoma. Oblik naredbe:

`poly(r)` – gdje je **`r`** vektor koji sadrži nultočke, odnosno korijene polinoma.

Primjer 11.7: Za nultočke polinoma koje su elementi vektora **`r`** izračunavaju se koeficijenti polinoma i pohranjuju u vektor **`p`**.

```
>> r = [6.5 4.0 2.3 -1.2 0.5]
r =
    6.50    4.00    2.30   -1.20    0.50
>> p = poly(r)
p =
    1.00  -12.10   40.59  -17.02  -71.95   35.88
>>
```

Dva se polinoma zbrajaju ili oduzimaju tako da se zbroje ili oduzmu odgovarajući koeficijenti polinoma. Neka postoje dva polinoma:

$$f_1(x)=a_nx^n+a_{n-1}x^{n-1}+\ldots+a_1x+a_0$$

$$f_2(x)=b_nx^n+b_{n-1}x^{n-1}+\ldots+b_1x+b_0$$

Zbroj tih dvaju polinoma je:

$$f_1(x)+f_2(x)=(a_n+b_n)x^n+(a_{n-1}+b_{n-1})x^{n-1}+\ldots+(a_1+b_1)x+a_0+b_0$$

Razlika tih dvaju polinoma je:

$$f_1(x)-f_2(x)=(a_n-b_n)x^n+(a_{n-1}-b_{n-1})x^{n-1}+\ldots+(a_1-b_1)x+a_0-b_0$$

U Octaveu se dva polinoma zbrajaju ili oduzimaju tako da se zbroje ili oduzmu vektori čiji su elementi koeficijenti polinoma. Ako polinomi nisu istog stupnja (što znači da vektori čiji su elementi koeficijenti

polinoma nisu jednake duljine), kraći se vektor mora dopuniti nulama kako bi imao isti broj elemenata kao dulji vektor.

Primjer 11.8: Prvo se zbrajaju, a zatim oduzimaju dva polinoma. Prvi polinom je $f_1(x)=3x^6+15x^5-10x^3-3x^2+15x-40$, a drugi polinom je $f_2(x)=3x^3-2x-6$. Zbroj tih dvaju polinoma je $f_1(x)+f_2(x)=3x^6+15x^5-7x^3-3x^2+13x-46$, a njihova razlika je $f_1(x)-f_2(x)=3x^6+15x^5-13x^3-3x^2+17x-34$.

```
>> p1 = [3 15 0 -10 -3 15 -40]
p1 =
     3.00    15.00     0.00   -10.00    -3.00    15.00   -40.00
>> p2 = [3 0 -2 -6]
p2 =
     3.00     0.00    -2.00    -6.00
>> p = p1 + [0 0 0 p2]
p =
     3.00    15.00     0.00    -7.00    -3.00    13.00   -46.00
>> p = p1 - [0 0 0 p2]
p =
     3.00    15.00     0.00   -13.00    -3.00    17.00   -34.00
>>
```

conv

Funkcija **conv** množi dva polinoma: Oblik naredbe je:

conv(p1,p2) – gdje su **p1** i **p2** vektori čiji su elementi koeficijenti polinoma.

Primjer 11.9: Množe se polinomi $f_1(x)=x+3$ i $f_2(x)=x^2-2x-3$. Rezultat umnoška tih dvaju polinoma je polinom $f(x)=x^3+x^2-9x-9$.

```
>> p1 = [1 3]
p1 =
     1.00     3.00
>> p2 = [1 -2 -3]
p2 =
     1.00    -2.00    -3.00
>> p = conv(p1,p2)
p =
     1.00     1.00    -9.00    -9.00
>>
```

deconv

Funkcija **deconv** dijeli dva polinoma: Oblik naredbe je:

[q,r] = deconv(p1,p2) – gdje su **p1** vektor čiji su elementi koeficijenti polinoma u brojniku, **p2** vektor čiji su elementi koeficijenti polinoma u nazivniku, **q** vektor čiji su elementi koeficijenti polinoma koji je količnik dijeljenja tih dvaju polinoma, a **r** vektor čiji su elementi koeficijenti polinoma koji je ostatak dijeljenja tih dvaju polinoma.

Primjer 11.10: Dijele se polinomi $f_1(x)=2x^3+9x^2+7x-6$ i $f_2(x)=x+3$. Rezultat dijeljenja tih dvaju polinoma je polinom $f(x)=2x^2+3x-2$.

```
>> p1 = [2 9 7 -6]
p1 =
```

```
     2.00     9.00     7.00    -6.00
>> p2 = [1 3]
p2 =
     1.00     3.00
>> [q,r] = deconv(p1,p2)
q =
     2.00     3.00    -2.00
r =
     0.00     0.00     0.00     0.00
>>
```

Primjer 11.11: Dijele se polinomi $f_1(x) = 2x^6 - 13x^5 + 75x^3 + 2x^2 - 60$ i $f_2(x) = x^2 - 5$. Rezultat dijeljenja tih dvaju polinoma je polinom $f(x) = 2x^4 - 13x^3 + 10x^2 + 10x + 52 + (50x + 200)/(x^2 - 5)$.

```
>> p1 = [2 -13 0 75 2 0 -60]
p1 =
     2.00   -13.00     0.00    75.00     2.00     0.00   -60.00
>> p2 = [1 0 -5]
p2 =
     1.00     0.00    -5.00
>> [q,r] = deconv(p1,p2)
q =
     2.00   -13.00    10.00    10.00    52.00
r =
     0.00     0.00     0.00     0.00     0.00    50.00   200.00
>>
```

polyder

Funkcija **`polyder`** izračunava derivaciju polinoma, derivaciju umnoška dvaju polinoma ili derivaciju kvocijenta dvaju polinoma. Oblik naredbe je:

`polyder(p)` – gdje je **`p`** vektor čiji su elementi koeficijenti polinoma.

`polyder(p1,p2)` – gdje su **`p1`** i **`p2`** vektori čiji su elementi koeficijenti polinoma koji se množe. Funkcija izračunava derivaciju umnoška tih dvaju polinoma.

`[q,d] = polyder(p1,p2)` – gdje su **`p1`** i **`p2`** vektori čiji su elementi koeficijenti polinoma koji se dijele. Funkcija izračunava derivaciju kvocijenta tih dvaju polinoma i pohranjuje rezultat u vektore **`q`** i **`d`**.

Primjer 11.12: Izračunava se derivacija polinoma $f(x) = 2x^2 + 5x + 2$ koja je jednaka $f'(x) = 4x + 5$.

```
>> p = [2 5 2]
p =
     2.00     5.00     2.00
>> pd = polyder(p)
pd =
     4.00     5.00
>>
```

Primjer 11.13: Izračunava se derivacija polinoma $f(x) = -3x^3 + 4x^2 - 8x - 2$ koja je jednaka $f'(x) = -9x^2 + 8x - 8$.

```
>> p = [-3 4 -8 -2]
```

```
p =
    -3.00     4.00    -8.00    -2.00
>> pd = polyder(p)
pd =
    -9.00     8.00    -8.00
>>
```

Primjer 11.14: Izračunava se derivacija umnoška polinoma $f_1(x) = x + 3$ i polinoma $f_2(x) = x^2 - 3x$. Njihov umnožak jednak je $f(x) = f_1(x) \cdot f_2(x) = x^3 - 9x$, a derivacija umnoška jednaka je $f'(x) = 3x^2 - 9$.

```
>> p1 = [1 3]
p1 =
     1.00     3.00
>> p2 = [1 -3 0]
p2 =
     1.00    -3.00     0.00
>> pd = polyder(p1,p2)
pd =
     3.00     0.00    -9.00
>>
```

Primjer 11.15: Izračunava se derivacija kvocijenta polinoma $f_1(x) = 2x^3 + 9x^2 + 7x - 6$ i polinoma $f_2(x) = x + 3$. Njihov kvocijent jednak je $f(x) = 2x^2 + 3x - 2$, a derivacija kvocijenta jednaka je $f'(x) = \left(4x^3 + 27x^2 + 54x + 27\right) / \left(x^2 + 6x + 9\right) = 4x + 3$.

```
>> p1 = [2 9 7 -6]
p1 =
     2.00     9.00     7.00    -6.00
>> p2 = [1 3]
p2 =
     1.00     3.00
>> [q,d] = polyder(p1,p2)
q =
     4.00     3.00
d =
     1.00
>>
```

Pitanja za provjeru znanja:

1. Čemu služi funkcija **`polyval`**?
2. Koja funkcija u Octaveu izračunava nultočke polinoma?
3. Čemu služi funkcija **`poly`**?
4. Koja funkcija u Octaveu množi dva polinoma?
5. Koja funkcija u Octaveu dijeli dva polinoma?
6. Čemu služi funkcija **`polyder`**?

12. OSTALE FUNKCIJE

U ovom su poglavlju opisane funkcije koje su korisne i često se rabe, ali ne pripadaju u neku određenu kategoriju funkcija.

abs

Pomoću funkcije **abs** izračunava se apsolutna vrijednost broja. Ulazna varijabla funkcije **abs(x)** može biti skalar, vektor ili matrica. Apsolutna vrijednost broja može se prikazati na sljedeći način:

$$|x| = \begin{cases} x & za \quad x > 0 \\ 0 & za \quad x = 0 \\ -x & za \quad x < 0 \end{cases}$$

Primjer 12.1: Definirana je matrica **x** dimenzija 3*2. Matrica **y** dimenzija 3*2 apsolutna je vrijednost matrice **x**.

```
>> x = [-5.2 2.6; -3.4 -8.5; 7.2 6.5]
x =
   -5.20    2.60
   -3.40   -8.50
    7.20    6.50
>> y = abs(x)
y =
    5.20    2.60
    3.40    8.50
    7.20    6.50
>>
```

sign

Pomoću funkcije **sign** određuje se predznak broja. Ulazna varijabla funkcije **sign(x)** može biti skalar, vektor ili matrica. Predznak broja može se prikazati na sljedeći način:

$$\mathrm{sgn}(x) = \begin{cases} 1 & za \quad x > 0 \\ 0 & za \quad x = 0 \\ -1 & za \quad x < 0 \end{cases}$$

Primjer 12.2: Definirana je matrica **x** dimenzija 3*3. Matrica **y** dimenzija 3*3 predznak je matrice **x**.

```
>> x = [0 -4.5 -6.3; -8.3 7.5 -3.8; 9.5 0 2.3]
x =
    0.00   -4.50   -6.30
   -8.30    7.50   -3.80
    9.50    0.00    2.30
```

```
>> y = sign(x)
y =
     0.00    -1.00    -1.00
    -1.00     1.00    -1.00
     1.00     0.00     1.00
>>
```

sqrt

Pomoću funkcije **sqrt** izračunava se drugi korijen. Ulazna varijabla funkcije **sqrt(x)** može biti skalar, vektor ili matrica. Izlazna varijabla funkcije **sqrt** može biti i kompleksna ako je ulazna varijabla negativna.

Primjer 12.3: Definirana je matrica **x** dimenzija 3*3. Matrica **y** dimenzija 3*3 drugi je korijen matrice **x**. Budući da su neki elementi matrice **x** negativni, za njih je rezultat u matrici **y** kompleksni broj. To je vidljivo ako se napiše naredba **whos** koja prikazuje detalje o varijablama.

```
>> x = [1.5 -4.5 -6.3; -8.3 7.5 -3.8; 9.5 5.2 2.3]
x =
    1.5000   -4.5000   -6.3000
   -8.3000    7.5000   -3.8000
    9.5000    5.2000    2.3000
>> y = sqrt(x)
y =
   1.22474 + 0.00000i   0.00000 + 2.12132i   0.00000 + 2.50998i
   0.00000 + 2.88097i   2.73861 + 0.00000i   0.00000 + 1.94936i
   3.08221 + 0.00000i   2.28035 + 0.00000i   1.51658 + 0.00000i
>>
```

deg2rad

Pomoću funkcije **deg2rad** pretvaraju se stupnjevi u radijane. Ulazna varijabla funkcije **deg2rad(x)** može biti skalar, vektor ili matrica.

Primjer 12.4: Definirana je matrica **x** dimenzija 2*2. Pomoću funkcije **deg2rad(x)** pretvaraju se vrijednosti iz stupnjeva u radijane i rezultat je matrica **y** dimenzija 2*2.

```
>> x = [15 60; 45 30]
x =
    15.00    60.00
    45.00    30.00
>> y = deg2rad(x)
y =
    0.26    1.05
    0.79    0.52
>>
```

rad2deg

Pomoću funkcije **rad2deg** pretvaraju se radijani u stupnjeve. Ulazna varijabla funkcije **rad2deg(x)** može biti skalar, vektor ili matrica.

Primjer 12.5: Definirana je matrica **x** dimenzija 2*3. Pomoću funkcije **rad2deg(x)** pretvaraju se vrijednosti iz radijana u stupnjeve i rezultat je matrica **y** dimenzija 2*3.

```
>> x = [pi/2 pi/4 pi/8; pi/3 pi 2*pi]
x =
    1.57    0.79    0.39
```

```
    1.05    3.14    6.28
>> y = rad2deg(x)
y =
   90.00   45.00   22.50
   60.00  180.00  360.00
>>
```

ceil

Pomoću funkcije **ceil** zaokružuju se vrijednosti prema najbližem većem cijelom broju. Ulazna varijabla funkcije **ceil(x)** može biti skalar, vektor ili matrica.

Primjer 12.6: Definirana je matrica **x** dimenzija 3*3. Pomoću funkcije **ceil(x)** zaokružuju se vrijednosti prema najbližem većem cijelom broju i rezultat je matrica **y** dimenzija 3*3.

```
>> x = [2.3 -3.8 8.9; -4.5 1.7 -6.2; -5.2 7.6 9.3]
x =
    2.30   -3.80    8.90
   -4.50    1.70   -6.20
   -5.20    7.60    9.30
>> y = ceil(x)
y =
     3.00    -3.00     9.00
    -4.00     2.00    -6.00
    -5.00     8.00    10.00
>>
```

floor

Pomoću funkcije **floor** zaokružuju se vrijednosti prema najbližem manjem cijelom broju. Ulazna varijabla funkcije **floor(x)** može biti skalar, vektor ili matrica.

Primjer 12.7: Definirana je matrica **x** dimenzija 3*3. Pomoću funkcije **floor(x)** zaokružuju se vrijednosti prema najbližem manjem cijelom broju i rezultat je matrica **y** dimenzija 3*3.

```
>> x = [2.3 -3.8 8.9; -4.5 1.7 -6.2; -5.2 7.6 9.3]
x =
    2.30   -3.80    8.90
   -4.50    1.70   -6.20
   -5.20    7.60    9.30
>> y = floor(x)
y =
    2.00   -4.00    8.00
   -5.00    1.00   -7.00
   -6.00    7.00    9.00
>>
```

fix

Pomoću funkcije **fix** zaokružuju se vrijednosti prema najbližem cijelom broju u smjeru 0. Ulazna varijabla funkcije **fix(x)** može biti skalar, vektor ili matrica.

Primjer 12.8: Definirana je matrica **x** dimenzija 3*3. Pomoću funkcije **fix(x)** zaokružuju se vrijednosti prema najbližem cijelom broju u smjeru 0 i dobiva se matrica **y** dimenzija 3*3.

```
>> x = [2.3 -3.8 8.9; -4.5 1.7 -6.2; -5.2 7.6 9.3]
x =
```

```
    2.30   -3.80    8.90
   -4.50    1.70   -6.20
   -5.20    7.60    9.30
>> y = fix(x)
y =
    2.00   -3.00    8.00
   -4.00    1.00   -6.00
   -5.00    7.00    9.00
>>
```

round

Pomoću funkcije **round** zaokružuju se vrijednosti prema najbližem cijelom broju. Ulazna varijabla funkcije **round(x)** može biti skalar, vektor ili matrica.

Primjer 12.9: Definirana je matrica **x** dimenzija 3*3. Pomoću funkcije **round(x)** zaokružuju se vrijednosti prema najbližem cijelom broju i rezultat je matrica **y** dimenzija 3*3.

```
>> x = [2.3 -3.8 8.9; -4.5 1.7 -6.2; -5.2 7.6 9.3]
x =
    2.30   -3.80    8.90
   -4.50    1.70   -6.20
   -5.20    7.60    9.30
>> y = round(x)
y =
    2.00   -4.00    9.00
   -5.00    2.00   -6.00
   -5.00    8.00    9.00
>>
```

mod

Funkcija **mod(x,y)** daje ostatak rezultata dijeljenja ulaznih argumenata **x** i **y**. Ulazne varijable funkcije **mod(x,y)** mogu biti skalari, vektori ili matrice, s time da vektori ili matrice **x** i **y** moraju biti istih dimenzija. Funkcija **mod(x,y)** daje ostatak rezultata dijeljenja u smislu **x-floor(x./y).*y**.

Primjer 12.10: Definirana je matrica **x** dimenzija 3*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-20,20]. Definirana je matrica **y** dimenzija 3*4 pomoću funkcije **randint** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [2,4]. Pomoću funkcije **mod(x,y)** nastaje matrica **z** dimenzija 3*4.

```
>> x = randi([-20 20],3,4)
x =
   -18.00    20.00    19.00   -18.00
    -7.00   -11.00   -20.00   -15.00
   -11.00    12.00     9.00    18.00
>> y = randi([2 4],3,4)
y =
     4.00     3.00     4.00     4.00
     3.00     3.00     3.00     2.00
     4.00     3.00     3.00     3.00
>> z = mod(x,y)
z =
     2.00     2.00     3.00     2.00
     2.00     1.00     1.00     1.00
     1.00     0.00     0.00     0.00
>>
```

rem

Funkcija **rem(x,y)** daje ostatak rezultata dijeljenja ulaznih argumenata **x** i **y**. Ulazne varijable funkcije **rem(x,y)** mogu biti skalari, vektori ili matrice s time da vektori ili matrice **x** i **y** moraju biti istih dimenzija. Funkcija **rem(x,y)** daje ostatak rezultata dijeljenja u smislu **x-fix(x./y).*y**.

Primjer 12.11: Definirana je matrica **x** dimenzija 3*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-20,20]. Definirana je matrica **y** dimenzija 3*4 pomoću funkcije **randint** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [2,4]. Pomoću funkcije **rem(x,y)** nastaje matrica **z** dimenzija 3*4.

```
>> x = randi([-20 20],3,4)
x =
    -8.00     4.00    19.00   -10.00
    11.00     8.00   -10.00    15.00
    -5.00   -13.00   -16.00   -17.00
>> y = randi([2 4],3,4)
y =
     3.00     4.00     4.00     2.00
     4.00     4.00     4.00     4.00
     4.00     3.00     2.00     3.00
>> z = rem(x,y)
z =
    -2.00     0.00     3.00    -0.00
     3.00     0.00    -2.00     3.00
    -1.00    -1.00    -0.00    -2.00
>>
```

cart2pol

Pomoću funkcije **cart2pol** pretvaraju se Kartezijeve koordinate u polarne koordinate. Oblik funkcije je: **[th,r] = cart2pol(x,y)**.

Ulazne varijable **x** i **y** mogu biti skalari, vektori ili matrice i oni su koordinate (x,y) u Kartezijevu koordinatnom sustavu. Izlazne varijable **th** i **r** su skalari, vektori ili matrice istih dimenzija kao ulazne varijable **x** i **y**, i oni su polarne koordinate. Varijabla **th** je u radijanima.

Polarne koordinate mogu se iz Kartezijevih koordinata izračunati na sljedeći način:

$$r = \sqrt{x^2 + y^2}$$

$$\varphi = arctg(y/x)$$

Primjer 12.12: Definirane su matrice **x** i **y** dimenzija 2*2 koje sadrže (x,y) koordinate u Kartezijevu koordinatnom sustavu. Pomoću funkcije **cart2pol** pretvaraju se u polarne koordinate i rezultat su matrice **th** i **r** dimenzija 2*2. Matrica **r** sadrži amplitudu, a matrica **th** kut polarnih koordinata u radijanima.

```
>> x = [1.2 2.3; 5.4 6.8]
x =
    1.20    2.30
    5.40    6.80
>> y = [2.2 8.8; 1.5 3.4]
y =
    2.20    8.80
    1.50    3.40
>> [th,r] = cart2pol(x,y)
th =
    1.07    1.32
    0.27    0.46
r =
    2.51    9.10
    5.60    7.60
```

```
>>
```

pol2cart

Pomoću funkcije **pol2cart** pretvaraju se polarne koordinate u Kartezijeve koordinate. Oblik funkcije je: **[x,y] = pol2cart(th,r)**.

Ulazne varijable **th** i **r** mogu biti skalari, vektori ili matrice i oni su polarne koordinate. Izlazne varijable **x** i **y** su skalari, vektori ili matrice istih dimenzija kao ulazne varijable **th** i **r**, i oni su Kartezijeve koordinate. Varijabla **th** je u radijanima.

Kartezijeve koordinate mogu se iz polarnih koordinata izračunati na sljedeći način:

$$x = r \cdot \cos\varphi$$

$$y = r \cdot \sin\varphi$$

Primjer 12.13: Definirane su matrice **th** i **r** dimenzija 2*3 koje su polarne koordinate. Pomoću funkcije **pol2cart** pretvaraju se u koordinate u Kartezijevu koordinatnom sustavu i rezultat su matrice **x** i **y** dimenzija 2*3.

```
>> th = [pi/2 pi/4 pi/3; pi 3/2*pi pi/8]
th =
    1.57    0.79    1.05
    3.14    4.71    0.39
>> r = [1.7 8.3 4.3; 5.7 7.2 2.5]
r =
    1.70    8.30    4.30
    5.70    7.20    2.50
>> [x,y] = pol2cart(th,r)
x =
    0.00    5.87    2.15
   -5.70   -0.00    2.31
y =
    1.70    5.87    3.72
    0.00   -7.20    0.96
>>
```

diag

Pomoću funkcije **diag** mogu se odrediti elementi glavne dijagonale matrice ili stvoriti dijagonalna matrica s određenim elementima na njoj. Oblik funkcije **diag(x)**, ako je **x** matrica dimenzija m*m, daje stupčani vektor dimenzija m*1 koji se sastoji od elemenata glavne dijagonale matrice **x**. Ako matrica **x** nije kvadratna nego ima dimenzije m*n, tada funkcija **diag(x)** daje stupčani vektor dimenzija m*1 koji se sastoji od pseudoelemenata glavne dijagonale matrice **x**. Ako je **x** redni vektor dimenzije 1*n, tada funkcija **diag(x)** stvara dijagonalnu matricu dimenzije n*n s elementima na glavnoj dijagonali koji odgovaraju elementima vektora **x**.

Primjer 12.14: Definirana je matrica **x** dimenzija 4*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-10,10]. Stupčani vektor **y** dimenzija 1*4 nastaje pomoću funkcije **diag(x)** i sastoji se od elemenata glavne dijagonale matrice **x**.

```
>> x = randi([-10 10],4,4)
x =
    -7.00     6.00     1.00     8.00
    -9.00     1.00    -5.00     4.00
     0.00    -3.00    -8.00    -1.00
    -1.00     9.00     2.00    -4.00
>> y = diag(x)
y =
    -7.00
```

```
     1.00
    -8.00
    -4.00
>>
```

Primjer 12.15: Definirana je matrica **x** dimenzija 3*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [0,10]. Stupčani vektor **y** dimenzija 1*3 nastaje pomoću funkcije **diag(x)** i sastoji se od pseudoelemenata glavne dijagonale matrice **x**.

```
>> x = randi([0 10],3,4)
x =
     5.00     0.00     4.00     1.00
     7.00    10.00     5.00     3.00
     4.00     6.00     9.00     7.00
>> y = diag(x)
y =
     5.00
    10.00
     9.00
>>
```

Primjer 12.16: Definiran je redni vektor **x** dimenzija 1*4. Pomoću funkcije **diag(x)** nastaje dijagonalna matrica **y** dimenzija 4*4 koja na glavnoj dijagonali sadrži elemente vektora **x**.

```
>> x = [5 2 7 3]
x =
     5.00     2.00     7.00     3.00
>> y = diag(x)
y =
     5.00        0        0        0
        0     2.00        0        0
        0        0     7.00        0
        0        0        0     3.00
>>
```

trace

Funkcija **trace(x)** daje zbroj elemenata glavne dijagonale matrice **x**. Ako matrica nije kvadratna, funkcija **trace** daje zbroj elemenata pseudo glavne dijagonale matrice.

Primjer 12.17: Definirana je matrica **x** dimenzija 4*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [0,10]. Pomoću funkcije **trace(x)** dobiva se zbroj elemenata glavne dijagonale matrice **x**. To su elementi 1, 6, 0 i 6, i njihov zbroj iznosi 13.

```
>> x = randi([0 10],4,4)
x =
     9.00    2.00    3.00    9.00
     3.00    7.00    4.00    9.00
     1.00    8.00    5.00    4.00
     2.00    3.00    3.00    4.00
>> y = trace(x)
y =  25
>>
```

Primjer 12.18: Definirana je matrica **x** dimenzija 3*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [0,10]. Pomoću funkcije **trace(x)** dobiva se zbroj elemenata pseudo glavne dijagonale matrice **x**. To su elementi 4, 7 i 1, i njihov zbroj iznosi 12.

```
>> x = randi([0 10],3,4)
x =
     9.00     4.00     2.00     5.00
     6.00     5.00     7.00     4.00
     4.00    10.00     0.00     6.00
>> y = trace(x)
y =  14
>>
```

fliplr

Pomoću funkcije **fliplr** zrcali se matrica u smjeru lijevo/desno. Ulazna varijabla funkcije **fliplr(x)** je matrica. Matrica može biti kvadratna ili pravokutna. Mijenja se poredak stupaca u smjeru lijevo/desno. Neka postoji matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija m*n koja je zrcalna matrici $\underline{A}$ u smjeru lijevo/desno:

$$\underline{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \dots & \vdots \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{bmatrix}$$

$$\underline{B} = \begin{bmatrix} A_{1,n} & A_{1,n-1} & \dots & A_{1,1} \\ A_{2,n} & A_{2,n-1} & \dots & A_{2,1} \\ \vdots & \vdots & \dots & \vdots \\ A_{m,n} & A_{m,n-1} & \dots & A_{m,1} \end{bmatrix}$$

Primjer 12.19: Definirana je matrica **x** dimenzija 3*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-10,10]. Pomoću funkcije **fliplr** nastaje matrica **y** dimenzija 3*4 koja je zrcalna matrici **x** u smjeru lijevo/desno.

```
>> x = randi([-10 10],3,4)
x =
    -2.00     9.00     7.00    -2.00
     3.00    -2.00    -7.00     7.00
     0.00     3.00     9.00     4.00
>> y = fliplr(x)
y =
    -2.00     7.00     9.00    -2.00
     7.00    -7.00    -2.00     3.00
     4.00     9.00     3.00     0.00
>>
```

flipud

Pomoću funkcije **flipud** zrcali se matrica u smjeru gore/dolje. Ulazna varijabla funkcije **flipud(x)** je matrica. Matrica može biti kvadratna ili pravokutna. Mijenja se poredak redaka u smjeru gore/dolje. Neka postoji matrica $\underline{A}$ dimenzija m*n i matrica $\underline{B}$ dimenzija m*n koja je zrcalna matrici $\underline{A}$ u smjeru gore/dolje:

$$\underline{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \dots & \vdots \\ A_{m,1} & A_{m,2} & \dots & A_{m,n} \end{bmatrix}$$

$$\underline{B} = \begin{bmatrix} A_{m,1} & A_{m,2} & \dots & A_{m,n} \\ A_{m-1,1} & A_{m-1,2} & \dots & A_{m-1,n} \\ \vdots & \vdots & \dots & \vdots \\ A_{1,1} & A_{1,2} & \dots & A_{1,n} \end{bmatrix}$$

Primjer 12.20: Definirana je matrica **x** dimenzija 4*3 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-10,10]. Pomoću funkcije **flipud** nastaje matrica **y** dimenzija 4*3 koja je zrcalna matrici **x** u smjeru gore/dolje.

```
>> x = randi([-10 10],4,3)
x =
     3.00    -7.00     1.00
    -3.00    -7.00    -4.00
     5.00     5.00     5.00
     9.00     1.00    -1.00
>> y = flipud(x)
y =
     9.00     1.00    -1.00
     5.00     5.00     5.00
    -3.00    -7.00    -4.00
     3.00    -7.00     1.00
>>
```

triu

Pomoću funkcije **triu** dobiva se gornja trokutasta matrica. Ulazna varijabla funkcije **triu(x)** je matrica. Matrica može biti kvadratna ili pravokutna. Ako je matrica pravokutna, dobiva se pseudo gornja trokutasta matrica. Neka postoji matrica $\underline{A}$ dimenzija m*m i matrica $\underline{B}$ dimenzija m*m koja je gornja trokutasta matrica matrice $\underline{A}$:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ A_{21} & A_{22} & \dots & A_{2m} \\ \vdots & \vdots & \dots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mm} \end{bmatrix}$$

$$\underline{B} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ 0 & A_{22} & \dots & A_{2m} \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & A_{mm} \end{bmatrix}$$

Primjer 12.21: Definirana je matrica **x** dimenzija 4*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-5,5]. Pomoću funkcije **triu(x)** nastaje gornja trokutasta matrica **y** dimenzija 4*4.

```
>> x = randi([-5 5],4,4)
x =
    -2.00     5.00    -1.00     2.00
    -1.00     0.00    -3.00    -1.00
     1.00    -4.00    -1.00     0.00
```

```
    -2.00    -1.00     5.00    -2.00
>> y = triu(x)
y =
    -2.00     5.00    -1.00     2.00
     0.00     0.00    -3.00    -1.00
     0.00     0.00    -1.00     0.00
     0.00     0.00     0.00    -2.00
>>
```

Primjer 12.22: Definirana je matrica **x** dimenzija 2*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [0,10]. Pomoću funkcije **triu(x)** nastaje pseudo gornja trokutasta matrica **y** dimenzija 2*4.

```
>> x = randi([0 10],2,4)
x =
     9.00     2.00    10.00     3.00
     9.00    10.00     8.00     5.00
>> y = triu(x)
y =
     9.00     2.00    10.00     3.00
     0.00    10.00     8.00     5.00
>>
```

tril

Pomoću funkcije **tril** dobiva se donja trokutasta matrica. Ulazna varijabla funkcije **tril(x)** je matrica. Matrica može biti kvadratna ili pravokutna. Ako je matrica pravokutna, dobiva se pseudo donja trokutasta matrica. Neka postoji matrica $\underline{A}$ dimenzija m*m i matrica $\underline{B}$ dimenzija m*m koja je donja trokutasta matrica matrice $\underline{A}$:

$$\underline{A} = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ A_{21} & A_{22} & \dots & A_{2m} \\ \vdots & \vdots & \dots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mm} \end{bmatrix}$$

$$\underline{B} = \begin{bmatrix} A_{11} & 0 & \dots & 0 \\ A_{21} & A_{22} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ A_{m1} & A_{m2} & \dots & A_{mm} \end{bmatrix}$$

Primjer 12.23: Definirana je matrica **x** dimenzija 4*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-5,5]. Pomoću funkcije **tril(x)** nastaje donja trokutasta matrica **y** dimenzija 4*4.

```
>> x = randi([-5 5],4,4)
x =
   -3.00   -1.00    5.00    2.00
    3.00   -5.00   -3.00   -1.00
   -2.00    0.00   -1.00   -3.00
    3.00    4.00    1.00    3.00
>> y = tril(x)
y =
   -3.00    0.00    0.00    0.00
    3.00   -5.00    0.00    0.00
   -2.00    0.00   -1.00    0.00
    3.00    4.00    1.00    3.00
>>
```

Primjer 12.24: Definirana je matrica **x** dimenzija 2*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [0,10]. Pomoću funkcije **tril(x)** nastaje pseudo donja trokutasta matrica **y** dimenzija 2*4.

```
>> x = randi([0 10],2,4)
x =
     1.00   10.00    5.00    3.00
     9.00    2.00    6.00    1.00
>> y = tril(x)
y =
     1.00    0.00    0.00    0.00
     9.00    2.00    0.00    0.00
>>
```

rot90

Pomoću funkcije **rot90(x)** zakreće se matrica **x** za 90 stupnjeva u smjeru suprotnom od smjera kretanja kazaljki na satu. Pomoću funkcije **rot90(x,k)** zakreće se matrica **x** za k·90 stupnjeva, gdje je k= ±1, ±2, ±3,... . Ako je **k** pozitivan cijeli broj, matrica **x** zakreće se u smjeru suprotnom od smjera kretanja kazaljki na satu, a ako je **k** negativan cijeli broj, matrica **x** zakreće se u smjeru kretanja kazaljki na satu.

Primjer 12.25: Definirana je matrica **x** dimenzija 3*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [-10,10]. Pomoću funkcije **rot90(x)** dobiva se matrica **y** koja je za 90 stupnjeva u smjeru suprotnom od smjera kretanja kazaljki na satu rotirana matrica **x**. Pomoću funkcije **rot90(x,2)** dobiva se matrica **z** koja je matrica **x** rotirana za 180 stupnjeva u smjeru suprotnom od smjera kretanja kazaljki na satu. Pomoću funkcije **rot90(x,-3)** dobiva se matrica **w** koja je matrica **x** rotirana za 270 stupnjeva u smjeru kretanja kazaljki na satu.

```
>> x = randi([-10 10],3,4)
x =
   -2.00   -3.00   -8.00   -8.00
   -2.00   -9.00   10.00    0.00
    2.00   -8.00    2.00   -5.00
>> y = rot90(x)
y =
   -8.00    0.00   -5.00
   -8.00   10.00    2.00
   -3.00   -9.00   -8.00
   -2.00   -2.00    2.00
>> z = rot90(x,2)
z =
   -5.00    2.00   -8.00    2.00
    0.00   10.00   -9.00   -2.00
   -8.00   -8.00   -3.00   -2.00
>> w = rot90(x,-3)
w =
   -8.00    0.00   -5.00
   -8.00   10.00    2.00
   -3.00   -9.00   -8.00
   -2.00   -2.00    2.00
>>
```

repmat

Funkcija **repmat** stvara matricu ponavljajući skalar, vektor ili matricu određeni broj puta u smjeru redaka i stupaca. Oblik funkcije je:

repmat(x,[m n]) – gdje je **x** skalar, vektor ili matrica, **m** skalar koji određuje broj ponavljanja po redcima, a **n** skalar koji određuje broj ponavljanja po stupcima.

Primjer 12.26: Definiran je skalar **x**. Pomoću funkcije **repmat** ponavlja se skalar **x**, 5 puta u smjeru redaka i 5 puta u smjeru stupaca.

```
>> x = 8
x =
     8.00
>> y = repmat(x,[5 5])
y =
     8.00     8.00     8.00     8.00     8.00
     8.00     8.00     8.00     8.00     8.00
     8.00     8.00     8.00     8.00     8.00
     8.00     8.00     8.00     8.00     8.00
     8.00     8.00     8.00     8.00     8.00
>>
```

Primjer 12.27: Definiran je redni vektor **x** dimenzija 1*3. Pomoću funkcije **repmat** ponavlja se vektor **x**, 2 puta u smjeru redaka i 3 puta u smjeru stupaca.

```
>> x = [5 3 8]
x =
   5.00   3.00   8.00
>> y = repmat(x,[2 3])
y =
   5.00   3.00   8.00   5.00   3.00   8.00   5.00   3.00   8.00
   5.00   3.00   8.00   5.00   3.00   8.00   5.00   3.00   8.00
>>
```

Primjer 12.28: Definirana je matrica **x** dimenzija 2*2. Pomoću funkcije **repmat** ponavlja se matrica **x**, 3 puta u smjeru redaka i 2 puta u smjeru stupaca te nastaje matrica **y** dimenzija 6*4.

```
>> x = [6 8; 7 5]
x =
   6.00   8.00
   7.00   5.00
>> y = repmat(x,[3 2])
y =
   6.00   8.00   6.00   8.00
   7.00   5.00   7.00   5.00
   6.00   8.00   6.00   8.00
   7.00   5.00   7.00   5.00
   6.00   8.00   6.00   8.00
   7.00   5.00   7.00   5.00
>>
```

reshape

Funkcija **reshape** preoblikuje vektor ili matricu tako da mijenja broj redaka i stupaca matrice, odnosno vektora. Oblik funkcije je:

reshape(x,[m n]) – gdje je **x** vektor ili matrica, **m** skalar koji predstavlja novi broj redaka, a **n** skalar koji predstavlja novi broj stupaca.

Važno je da se vektor ili matrica **x** može pretvoriti u novi vektor ili matricu s novim brojem redaka **m** i novim brojem stupaca **n**, odnosno mora biti zadovoljeno da početni vektor ili matrica te vektor ili matrica, dobivena pomoću funkcije **reshape**, imaju isti broj elemenata. Ako to nije ispunjeno, Octave će javiti pogrešku. Preoblikovanje se radi po stupcima.

Primjer 12.29: Definirana je matrica **x** dimenzija 2*2. Pomoću funkcije **reshape** matrica **x** je preoblikovana u redni vektor **y** dimenzija 1*4.

```
>> x = [6 8; 7 5]
x =
   6.00   8.00
   7.00   5.00
>> y = reshape(x,[1 4])
y =
   6.00   7.00   8.00   5.00
>>
```

Primjer 12.30: Definirana je matrica **x** dimenzija 3*3. Pomoću funkcije **reshape** matrica **x** preoblikovana je u redni vektor **y** dimenzija 1*9.

```
>> x = [1 2 3; 4 5 6; 7 8 9]
x =
   1.00   2.00   3.00
   4.00   5.00   6.00
   7.00   8.00   9.00
>> y = reshape(x,[1 9])
y =
   1.00   4.00   7.00   2.00   5.00   8.00   3.00   6.00   9.00
>>
```

Primjer 12.31: Definirana je matrica **x** dimenzija 3*4 pomoću funkcije **randi** i sadrži slučajne cjelobrojne vrijednosti iz jednolike razdiobe u rasponu [0,10]. Pomoću funkcije **reshape** matrica **x** preoblikovana je u matricu **y** dimenzija 2*6.

```
>> x = randi([0 10],3,4)
x =
    4.00    0.00    2.00    3.00
    3.00   10.00    1.00    8.00
    8.00    0.00    3.00    5.00
>> y = reshape(x,[2 6])
y =
    4.00    8.00   10.00    2.00    3.00    8.00
    3.00    0.00    0.00    1.00    3.00    5.00
>>
```

Primjer 12.32: Definiran je redni vektor **x** dimenzija 1*9. Pomoću funkcije **reshape** redni vektor **x** preoblikovan je u matricu **y** dimenzija 3*3.

```
>> x = [1 2 3 4 5 6 7 8 9]
x =
   1.00   2.00   3.00   4.00   5.00   6.00   7.00   8.00   9.00
>> y = reshape(x,[3 3])
y =
   1.00   4.00   7.00
   2.00   5.00   8.00
   3.00   6.00   9.00
>>
```

Primjer 12.33: Definirana je matrica **x** dimenzija 2*3. Pomoću funkcije **reshape** matrica **x** pokušava se preoblikovati u matricu **y** dimenzija 2*4. Budući da matrica **x** ima 6 elemenata, ne može se preoblikovati u matricu **y** koja bi imala 8 elemenata.

```
>> x = [1 2 3; 4 5 6]
x =
```

```
   1.00   2.00   3.00
   4.00   5.00   6.00
>> y = reshape(x,[2 4])
error: reshape: can't reshape 2x3 array to 2x4 array
>>
```

Pitanja za provjeru znanja:

1. Koja je razlika između funkcija **`abs`** i **`sign`**?
2. Čemu služi funkcija **`sqrt`**?
3. Koja je razlika između funkcija **`deg2rad`** i **`rad2deg`**?
4. Koja je razlika između funkcija **`ceil`**, **`floor`**, **`fix`** i **`round`**?
5. Koja je razlika između funkcija **`mod`** i **`rem`**?
6. Čemu služi funkcija **`cart2pol`**?
7. Čemu služi funkcija **`pol2cart`**?
8. Čemu služi funkcija **`diag`**?
9. Čemu služi funkcija **`trace`**?
10. Koja je razlika između funkcija **`fliplr`** i **`flipud`**?
11. Koja je razlika između funkcija **`triu`** i **`tril`**?
12. Čemu služi funkcija **`rot90`**?
13. Čemu služi funkcija **`repmat`**?
14. Čemu sliži funkcija **`reshape`**?

13. 2D PRIKAZ PODATAKA

Octave ima više naredbi za izradu različitih vrsta grafova. Grafovi mogu biti s jednolikom i logaritamskom podjelom (skalom), te u različitim oblicima. Moguće je izabrati boju, debljinu i vrstu crte, dodati oznake (engl. *marker*) i crte mreže, oznake koordinatnih osi, naslove, legendu, smjestiti više grafova u isti grafički prozor itd.

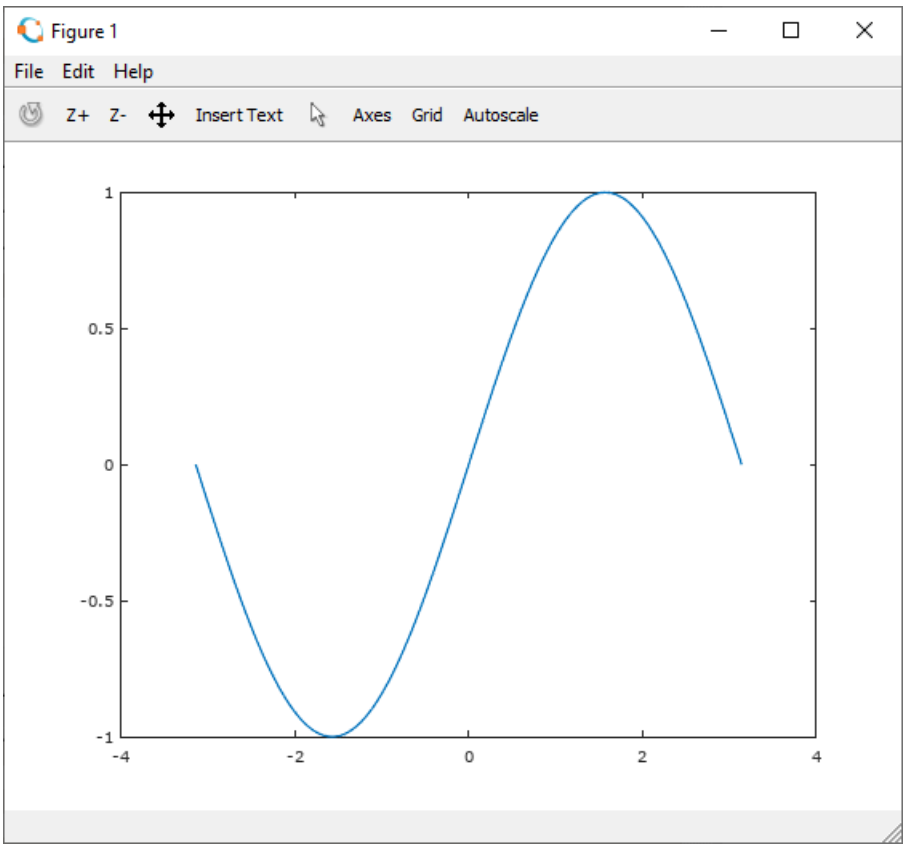
plot

Najčešća je naredba koja se koristi za crtanje 2D grafova **`plot`**. Naredba **`plot`** ima više oblika, ali najčešći je **`plot(x,y)`**. Argumenti **`x`** i **`y`** mogu biti ili redni ili stupčani vektori, ali moraju biti istih dimenzija. Argumenti **`x`** i **`y`** predstavljaju x i y podatke (točke) za crtanje 2D grafa. Izvršavanjem naredbe **`plot`** nastaje slika u grafičkom prozoru (engl. *figure*). Grafički se prozor automatski otvara pri izvršavanju naredbe **`plot`** (ako već nije otvoren). Graf sadrži krivulju sa x vrijednostima na apscisi i y vrijednostima na ordinati. Krivulja se sastoji od pravocrtnih segmenata koji povezuju točke čije se koordinate (x,y) nalaze u vektorima **`x`** i **`y`**. Za apscisu se rabi vektor koji je naveden kao prvi argument, a za ordinatu vektor koji je naveden kao drugi argument u naredbi **`plot(x,y)`**. Stvoreni graf ima osi s jednolikom podjelom i automatski određenim rasponima osi.

Primjer 13.1: Definiran je redni vektor **`x`** dimenzija 1*100 pomoću funkcije **`linspace`** u rasponu [-pi,pi] od 100 elemenata (točaka). Redni vektor **`y`** dimenzija 1*100 je vrijednost funkcije sinus za vrijednosti vektora **`x`**. Naredba **`plot(x,y)`** prikazuje funkciju sinus.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x i y
x = linspace(-pi,pi,100);
y = sin(x);

% Prikaz grafa funkcije y=sin(x)
plot(x,y)
```

Otvaranje grafičkog prozora (engl. *figure*) i prikaz funkcije sinus u rasponu [-pi,pi], prikazano je na slici 13.1.



Slika 13.1 Grafički prozor u kojem je prikazan graf funkcije sinus

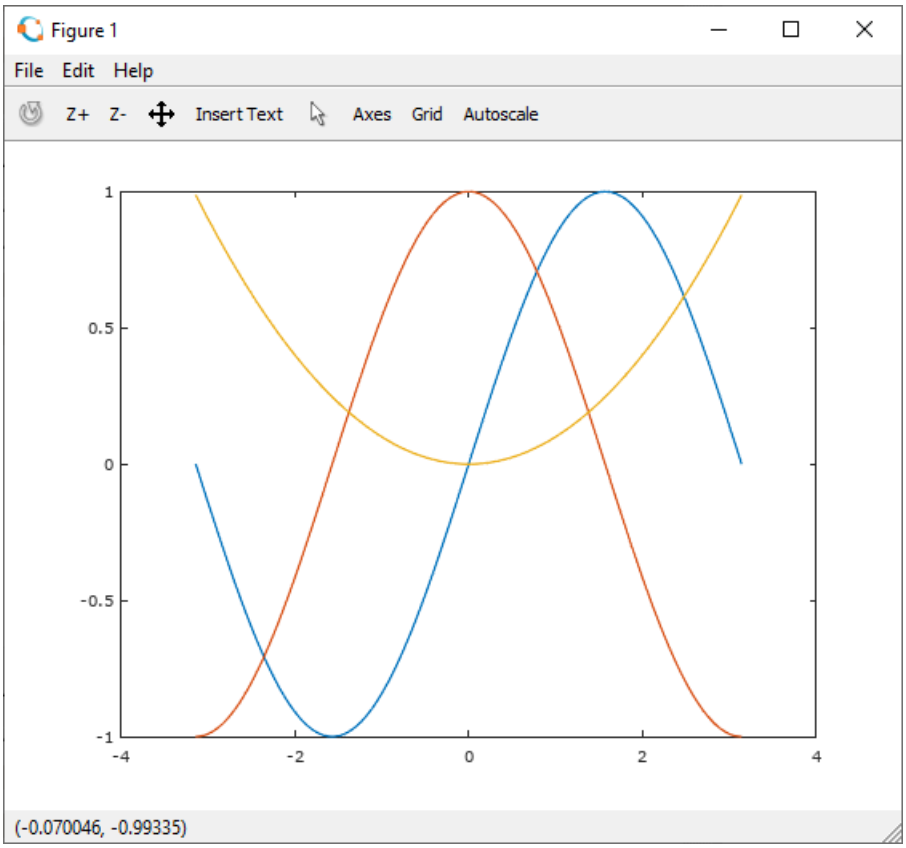
Naredba **`plot`** može se koristiti i za crtanje više krivulja na istom grafu. Potrebno je navesti onoliko parova vektora koliko se želi nacrtati krivulja. Naredba **`plot`** u tom obliku je **`plot(x1,y1,x2,y2,...,xn,yn)`**.

Svaki par vektora predstavlja x i y podatke krivulje i oni moraju imati jednak broj elemenata. Octave crta krivulje različitim bojama.

Primjer 13.2: Definiran je redni vektor **`x`** dimenzija 1*100 pomoću funkcije **`linspace`** u rasponu [-pi,pi] od 100 elemenata (točaka). Redni vektor **`y1`** dimenzija 1*100 vrijednost je funkcije sinus za vrijednosti vektora **`x`**. Redni vektor **`y2`** dimenzija 1*100 vrijednost je funkcije kosinus za vrijednosti vektora **`x`**. Redni vektor **`y3`** dimenzija 1*100 vrijednost je kvadratne funkcije za vrijednost vektora **`x`**. Naredba **`plot(x,y1,x,y2,x,y3)`** prikazuje te tri funkcije (krivulje) na istom grafu (u istom koordinatnom sustavu).

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1, y2 i y3
x = linspace(-pi,pi,100);
y1 = sin(x);
y2 = cos(x);
y3 = (x.^2)/10;

% Prikaz grafova funkcija y1=sin(x), y2=cos(x) i y3=(x.^2)/10
plot(x,y1,x,y2,x,y3)
```

Rezultat je otvaranje grafičkog prozora (engl. *figure*) i prikaz triju funkcija u rasponu [-pi,pi], što je prikazano na slici 13.2.



Slika 13.2 Grafički prozor u kojem su prikazane tri funkcije

Pomoću naredbe **`plot`** moguće je definirati vrstu crte i boju svake krivulje na grafu, kao i vrstu oznake (engl. *marker*). Marker je vizualna oznaka (križić, krug, točka i sl.) na mjestu u grafu koji se odnosi na točku s koordinatama (x,y). Oznaka vrste i boja crte te oznaka vrste markera prikazane su u tablici 13.1.

Tablica 13.1 Oznake vrste i boje crta te oznake vrste markera

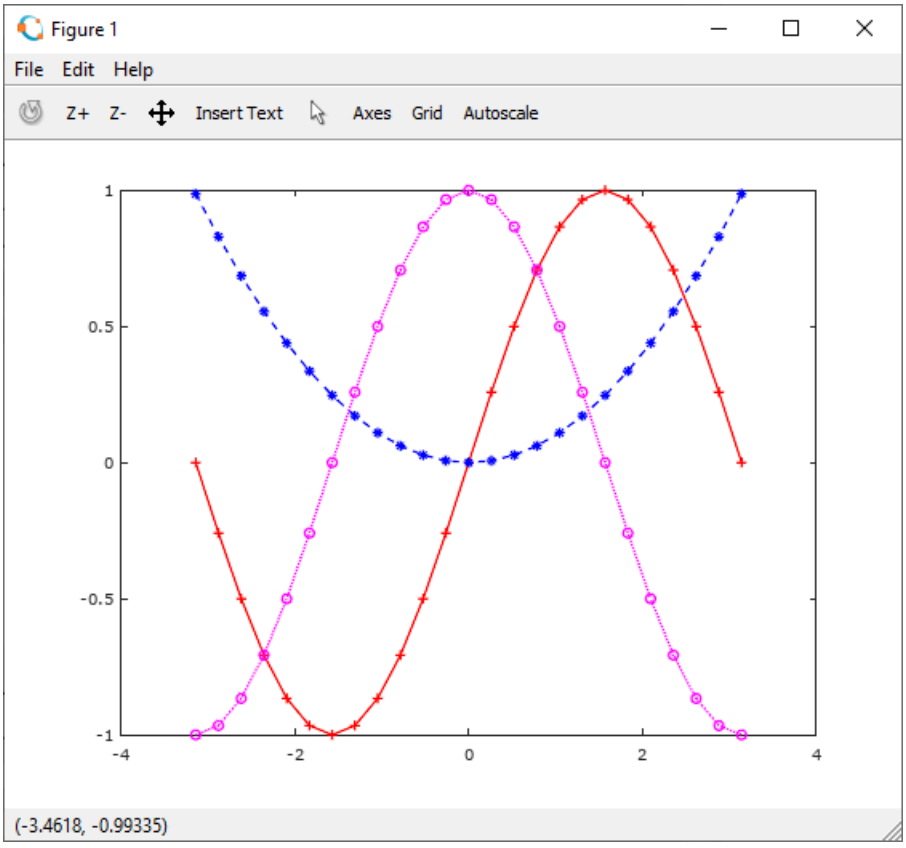
Vrsta crte	Oznaka	Boja crte	Oznaka	Vrsta markera	Oznaka
Puna	`-`	Plava	`b`	Točka	`.`
Točkasta	`:`	Zelena	`g`	Krug	`o`
Crta-točka	`-.`	Crvena	`r`	X-znak	`x`
Isprekidana	`--`	Cijan	`c`	Plus	`+`
		Magenta	`m`	Zvijezda	`*`
		Žuta	`y`	Kvadrat	`s`
		Crna	`k`	Dijamant	`d`
				Trokut (prema dolje)	`v`
				Trokut (prema gore)	`^`
				Trokut (nalijevo)	`<`
				Trokut (nadesno)	`>`
				Zvijezda s pet krakova	`p`
				Zvijezda sa šest krakova	`h`

Primjer 13.3: Definiran je redni vektor **`x`** dimenzija 1*25 pomoću funkcije **`linspace`** u rasponu [-pi,pi] od 25 elemenata (točaka). Redni vektor **`y1`** dimenzija 1*25 vrijednost je funkcije sinus za vrijednosti vektora **`x`**. Redni vektor **`y2`** dimenzija 1*25 vrijednost je funkcije kosinus za vrijednosti vektora **`x`**. Redni vektor **`y3`** dimenzija 1*25 vrijednost je kvadratne funkcije za vrijednost vektora **`x`**. Naredba **`plot`** prikazuje te tri funkcije (krivulje) na istom grafu (i u istom koordinatnom sustavu). Za prvu je krivulju izabrana crvena boja, puna crta i oznaka markera plusom. Za drugu je krivulju izabrana magenta boja,

točkasta crta i oznaka markera krugom. Za treću je krivulju izabrana plava boja, isprekidana crta i oznaka markera zvijezdom.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1, y2 i y3
x = linspace(-pi,pi,25);
y1 = sin(x);
y2 = cos(x);
y3 = (x.^2)/10;

% Prikaz grafova funkcija y1=sin(x), y2=cos(x) i y3=(x.^2)/10
plot(x,y1,'r+-',x,y2,'mo:',x,y3,'b*--')
```

Rezultat je otvaranje grafičkog prozora (engl. *figure*) i prikaz triju funkcija u rasponu [-pi,pi], što je prikazano na slici 13.3.



Slika 13.3 Grafički prozor u kojem su prikazane tri funkcije s različitim vrstama i bojama crta i markera

Krivuljama koje su prikazane pomoću naredbe **`plot`** mogu se mijenjati značajke. U tablici 13.2 opisane su četiri značajke krivulje prikazane pomoću naredbe **`plot`** koje se često koriste.

Tablica 13.2 Značajke crte i markera prikazane pomoću naredbe **`plot`**

Svojstvo	Opis	Moguće vrijednosti svojstva
`LineWidth`	Debljina crte	Broj
`MarkerSize`	Veličina markera	Broj
`MarkerEdgeColor`	Boja obruba markera	Oznaka boje
`MarkerFaceColor`	Boja ispune markera	Oznaka boje

hold

Kada bi se naredbe **plot** navodile jedna za drugom, svaka sljedeća obrisala bi sadržaj grafičkog prozora, odnosno krivulja prikazanih u njemu. Ako se želi prikazati više krivulja na istom grafu koristeći više naredbi **plot**, mora se koristiti naredba **hold on**. Naredba **hold on** zadržava postojeći otvoreni grafički prozor, prethodno nacrtanu krivulju, vrijednosti osi itd. Ako se nakon naredbe **hold on** upotrijebi naredba **plot**, nova će se krivulja nacrtati u već otvorenom grafičkom prozoru. Ako se više ne želi na istom grafu crtati krivulje, koristi se naredba **hold off**.

grid

Naredba **grid on** dodaje crte mreže na graf, a naredba **grid off** uklanja crte mreže s grafa. Gustoća crta mreže određena je naredbom **axis** (opisano u nastavku).

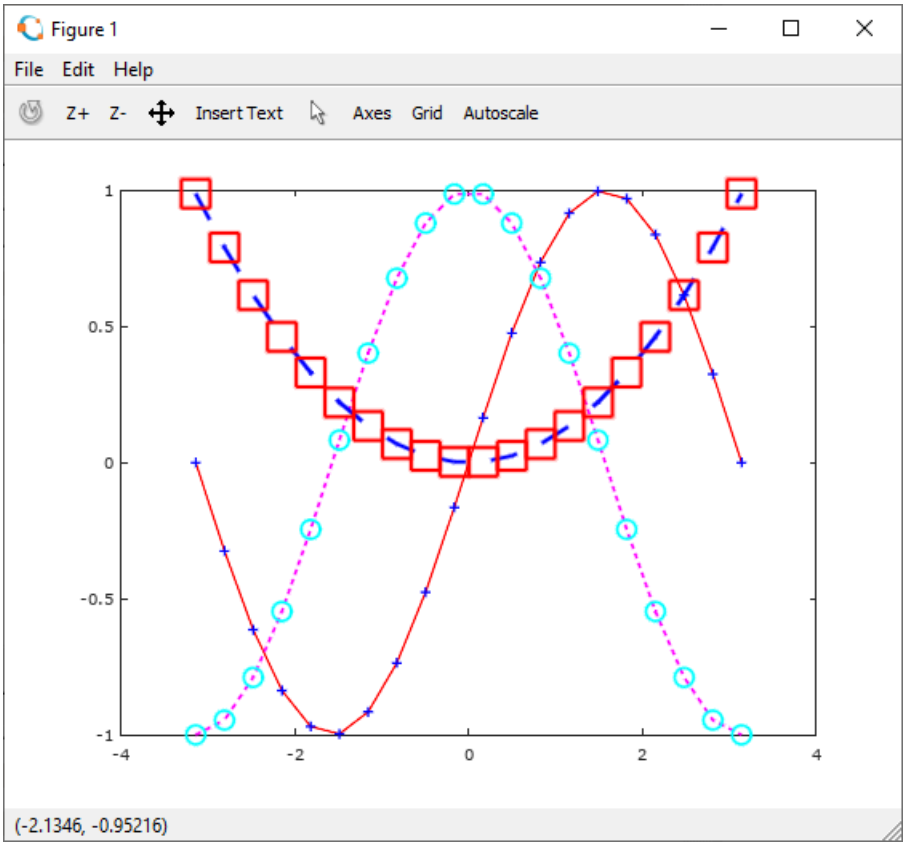
figure

Naredba **figure** otvara novi grafički prozor.

Primjer 13.4: Definiran je redni vektor **x** dimenzija 1*20 pomoću funkcije **linspace** u rasponu [-pi,pi] od 100 elemenata (točaka). Redni vektor **y1** dimenzija 1*20 vrijednost je funkcije sinus za vrijednosti vektora **x**. Redni vektor **y2** dimenzija 1*20 vrijednost je funkcije kosinus za vrijednosti vektora **x**. Redni vektor **y3** dimenzija 1*20 vrijednost je kvadratne funkcije za vrijednost vektora **x**. Naredba **plot** prikazuje te tri funkcije (krivulje) na istom grafu. Za prvu je krivulju izabrana crvena boja, puna crta, oznaka markera plus, debljina crte je 2, veličina markera 6, a boja ruba markera plava. Za drugu je krivulju izabrana magenta boja, točkasta crta, oznaka markera krug, debljina crte je 3, veličina markera 12, a boja ruba markera cijan. Za treću je krivulju izabrana plava boja, isprekidana crta, oznaka markera kvadrat, debljina crte je 4, veličina markera 18, a boja ruba markera crvena.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1, y2 i y3
x = linspace(-pi,pi,20);
y1 = sin(x);
y2 = cos(x);
y3 = (x.^2)/10;

% Prikaz grafova funkcija y1=sin(x), y2=cos(x) i y3=(x.^2)/10
plot(x,y1,'r+-','LineWidth',2,'MarkerSize',6,'MarkerEdgeColor','b')
hold on
plot(x,y2,'mo:','LineWidth',3,'MarkerSize',12,'MarkerEdgeColor','c')
plot(x,y3,'bs--','LineWidth',4,'Markersize',18,'MarkerEdgeColor','r')
```

Rezultat je otvaranje grafičkog prozora (engl. *figure*) i prikaz triju funkcija u rasponu [-pi,pi], što je prikazano na slici 13.4.

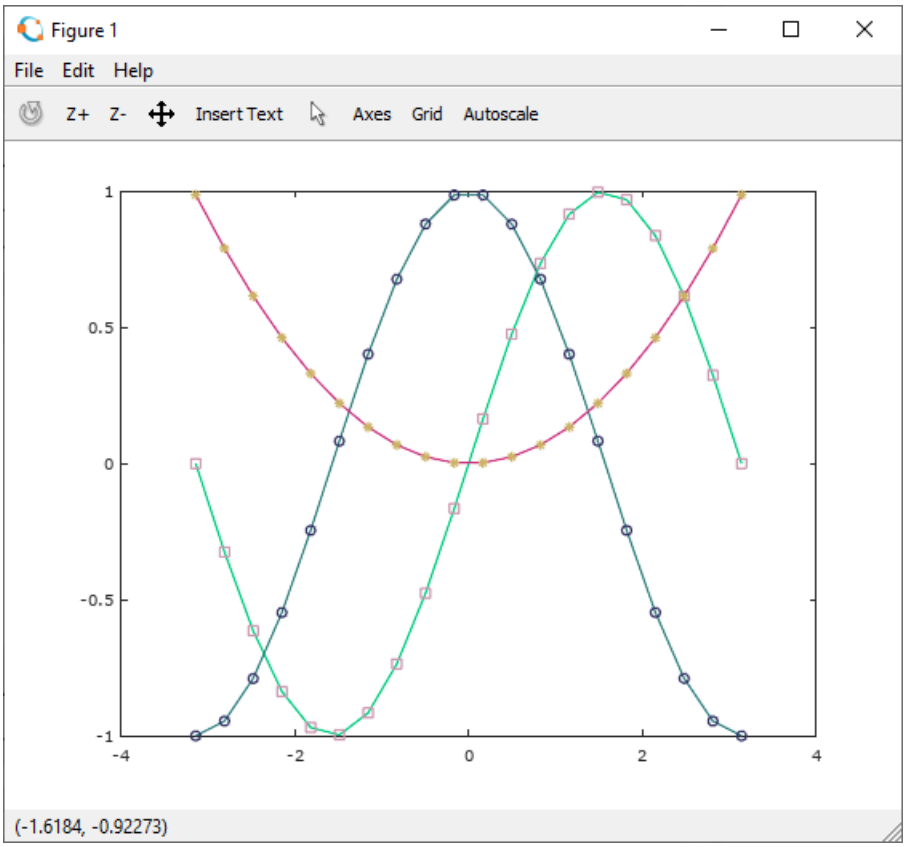


Slika 13.4 Grafički prozor u kojem su prikazane tri funkcije s različitim vrstama, debljinama i bojama crta i markera

Osim osnovnih boja koje se mogu rabiti za sve značajke grafova, a koje su navedene u tablici 13.1, moguće je rabiti svih 16777216 boja koje su na raspolaganju računalu. To se radi tako da se iza svake značajke grafa napisanog kao tekstualni niz navede vektor s tri elementa. Ta tri elementa vektora odgovaraju udjelima crvene (engl. *red*, r), zelene (engl. *green*, g) i plave (engl. *blue*, b) komponente boje. Vrijednosti tih triju elemenata vektora moraju se nalaziti u rasponu [0,1]. Vrijednost 0 znači da nema udjela te komponente u boji, dok vrijednost 1 znači da je udio te komponente u boji potpun.

Primjer 13.5: U grafičkom se prozoru prikazuju tri krivulje na grafu s različitim bojama crta i markera koje su određene pomoću vektora s tri elementa, a koji predstavljaju udjele pojedinih komponenata boja (crvene, zelene i plave), što je prikazano na slici 13.5.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1, y2 i y3
x = linspace(-pi,pi,20);
y1 = sin(x);
y2 = cos(x);
y3 = (x.^2)/10;

% Prikaz grafova funkcija y1=sin(x), y2=cos(x) i y3=(x.^2)/10
plot(x,y1,'Color',[0.0                                                0.8
0.5],'LineWidth',2,'Marker','s','MarkerEdgeColor',[0.8 0.6 0.7])
hold on
plot(x,y2,'Color',[0.2                                                0.5
0.5],'LineWidth',2,'Marker','o','MarkerEdgeColor',[0.2 0.2 0.4])
plot(x,y3,'Color',[0.8                                                0.2
0.5],'LineWidth',2,'Marker','*','MarkerEdgeColor',[0.8 0.7 0.4])
```

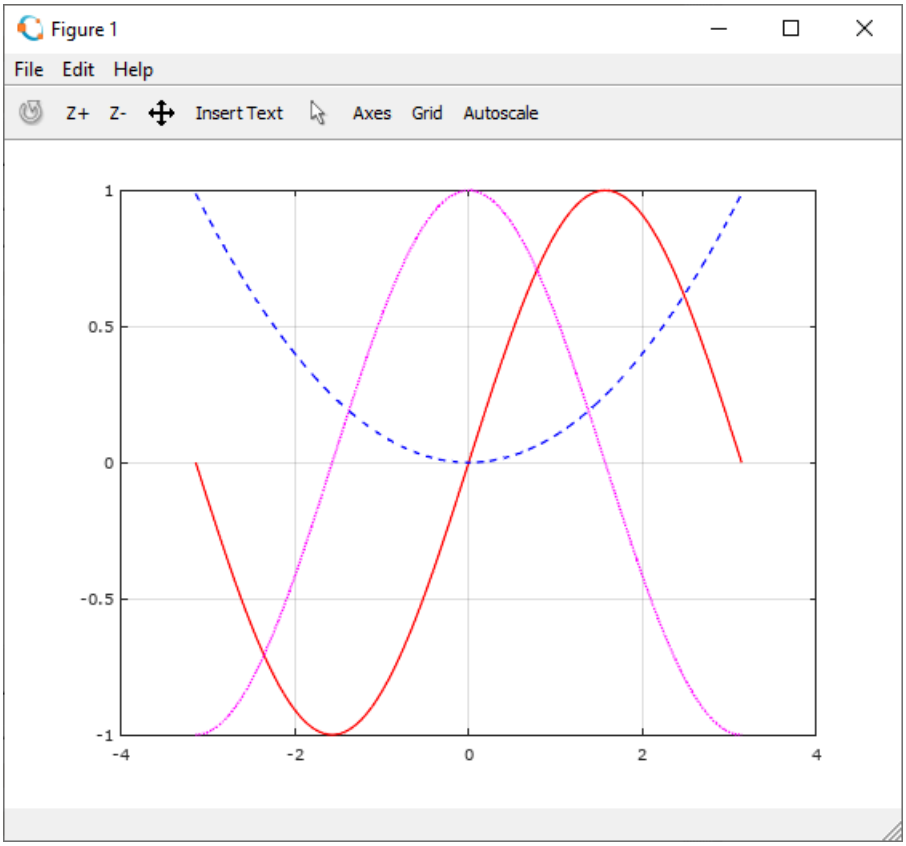


Slika 13.5 Tri krivulje prikazane pomoću naredbe **`plot`** kod koje se za određivanje boje krivulja i markera koristi vektor s tri elementa koji predstavljaju udjele komponenata boja

Primjer 13.6: Definiran je redni vektor **`x`** dimenzija 1*100 pomoću funkcije **`linspace`** u rasponu [-pi,pi] od 100 elemenata (točaka). Redni vektor **`y1`** dimenzija 1*100 vrijednost je funkcije sinus za vrijednosti vektora **`x`**. Redni vektor **`y2`** dimenzija 1*100 vrijednost je funkcije kosinus za vrijednosti vektora **`x`**. Redni vektor **`y3`** dimenzija 1*100 vrijednost je kvadratne funkcije za vrijednost vektora **`x`**. Naredba **`plot`** prikazuje prvu funkciju. Zatim se koristi naredba **`hold on`** za zadržavanje grafičkog prozora. Iza nje slijede dvije naredbe **`plot`** za prikazivanje druge i treće krivulje. Nakon toga korištena je naredba **`grid on`** za dodavanje crta mreže na graf.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1, y2 i y3
x = linspace(-pi,pi,100);
y1 = sin(x);
y2 = cos(x);
y3 = (x.^2)/10;

% Prikaz grafova funkcija y1=sin(x), y2=cos(x) i y3=(x.^2)/10
plot(x,y1,'r-')
hold on
plot(x,y2,'m:')
plot(x,y3,'b--')
grid on
```

Rezultat je otvaranje grafičkog prozora (engl. *figure*) i prikaz triju funkcija u rasponu [-pi,pi], što je prikazano na slici 13.6.



Slika 13.6 Grafički prozor u kojem su prikazane tri krivulje

13.1 Naredbe za opisivanje grafova

Naredbama za opisivanje grafova moguće je opisati osi, dodati naslov i legendu.

xlabel

Naredba `xlabel` služi za natpis ispod apscise. Naredba se koristi u obliku `xlabel('znakovni niz')`, gdje se pod navodnicima navodi tekst ispod apscise (znakovna varijabla).

ylabel

Naredba `ylabel` služi za natpis pored ordinate. Naredba se koristi u obliku `ylabel('znakovni niz')`, gdje se pod navodnicima navodi tekst pored ordinate (znakovna varijabla).

title

Naredba `title` služi za natpis iznad grafa (naslov). Naredba se koristi u obliku `title('znakovni niz')`, gdje se pod navodnicima navodi tekst naslova grafa (znakovna varijabla).

legend

Naredba `legend` služi za legendu grafa. Legenda prikazuje uzorak crte i markera (ako postoji) kojom je nacrtana svaka krivulja na grafu te pored njega natpis koji zadaje korisnik. Oblik naredbe je `legend('znakovni niz 1','znakovni niz 2',…,'znakovni niz n','Location')`. Znakovnih nizova može biti manje ili točno toliko koliko ima krivulja na grafu. Legenda se može smjestiti na razna mjesta (`'Location'` parametar u naredbi `legend`) koja su prikazana u tablici 13.3.

Tablica 13.3 Položaj legende grafa

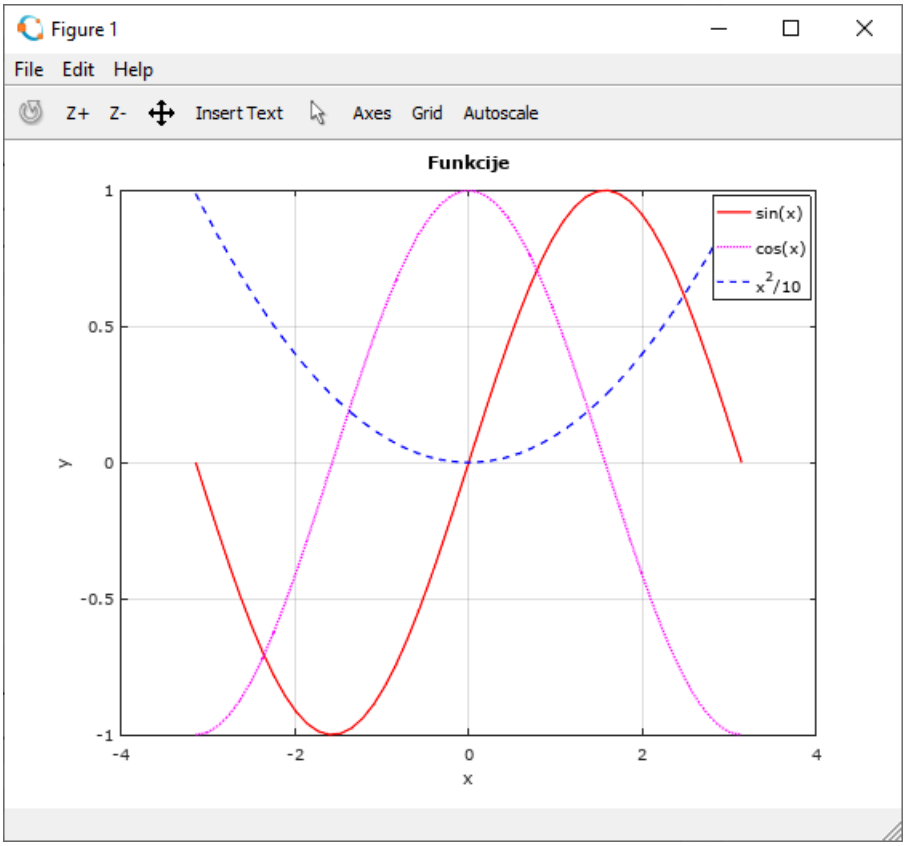
Položaj	Opis položaja
`'North'`	Na vrh grafa

'South'	Na dno grafa
'East'	Na desni dio grafa
'West'	Na lijevi dio grafa
'NorthEast'	Na gornji desni dio grafa
'NorthWest'	Na gornji lijevi dio grafa
'SouthEast'	Na donji desni dio grafa
'SouthWest'	Na donji lijevi dio grafa
'NorthOutside'	Na gornji dio izvan grafa
'SouthOutside'	Na donji dio izvan grafa
'EastOutside'	Na desni dio izvan grafa
'WestOutside'	Na lijevi dio izvan grafa
'NorthEastOutside'	Na gornji desni dio izvan grafa
'NorthWestOutside'	Na gornji lijevi dio izvan grafa
'SouthEastOutside'	Na donji desni dio izvan grafa
'SouthWestOutside'	Na donji lijevi dio izvan grafa
'Best'	Na mjesto gdje najmanje smeta iscrtanim krivuljama na grafu
'BestOutside'	Na mjesto gdje najmanje smeta izvan grafa

Primjer 13.7: Definiran je redni vektor **x** dimenzija 1*50 pomoću funkcije **linspace** u rasponu [-pi,pi] od 50 elemenata (točaka). Redni vektor **y1** dimenzija 1*50 vrijednost je funkcije sinus za vrijednosti vektora **x**. Redni vektor **y2** dimenzija 1*50 vrijednost je funkcije kosinus za vrijednosti vektora **x**. Redni vektor **y3** dimenzija 1*50 vrijednost je kvadratne funkcije za vrijednost vektora **x**. Naredba **plot** prikazuje prvu funkciju. Zatim se koristi naredba **hold on** za zadržavanje grafičkog prozora. Iza nje slijede dvije naredbe **plot** za prikazivanje druge i treće krivulje. Slijede naredbe **xlabel**, **ylabel** i **title** koje postavljaju oznake na apscisu, ordinatu i naslov iznad grafa. Nakon toga naredba **grid on** dodaje crte mreže na graf. Naredba **legend** postavlja legendu na graf.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1, y2 i y3
x = linspace(-pi,pi,50);
y1 = sin(x);
y2 = cos(x);
y3 = (x.^2)/10;

% Prikaz grafova funkcija y1=sin(x), y2=cos(x) i y3=(x.^2)/10
plot(x,y1,'r-')
hold on
plot(x,y2,'m:')
plot(x,y3,'b--')
% Opis grafa
xlabel('x')
ylabel('y')
title('Funkcije')
% Prikaz crta mreze grafa
grid on
% Prikaz legende grafa
legend('sin(x)','cos(x)','x^2/10')
```

Rezultat je otvaranje grafičkog prozora (engl. *figure*) i prikaz triju funkcija u rasponu [-pi,pi], što je prikazano na slici 13.7.



Slika 13.7 Grafički prozor u kojem su prikazane tri krivulje, označene apscisa i ordinata, naslov iznad grafa i legenda

text

Naredba `text(x,y,'znakovni niz')` prikazuje natpis na grafu tako da je prvi znak natpisa na mjestu čije su koordinate (x,y).

13.2 Oblikovanje teksta

Moguće je oblikovati prikaz znakovnog niza uključenog u naredbe `xlabel`, `ylabel`, `title`, `legend` i `text`. Oblikovanjem se zadaje vrsta slova (engl. *font*), veličina, položaj (indeks i eksponent), stil (italic, bold, normal), boja teksta, boja pozadine itd. U tablici 13.4 prikazani su neki od najčešće korištenih svojstava za oblikovanje teksta.

Ako se ispred znaka upiše _ (engl. *underscore*) ili ^ (engl. *caret*), znak će biti prikazan u indeksu, odnosno eksponentu. Da bi se više znakova ispisalo u indeksu, odnosno eksponentu, upisuju se unutar vitičastih zagrada {}, koje slijede neposredno iza znaka _ ili ^, npr. x_{r1}.

Tablica 13.4 Svojstva za oblikovanje teksta

Ime svojstva	Opis	Moguće vrijednosti svojstva
`Rotation`	Orijentacija (kut) znakova	Broj u stupnjevima
`FontAngle`	Normalan ili ukošen stil znakova	Normal, italic
`FontName`	Vrsta slova	Ime fonta dostupno sustavu
`FontSize`	Veličina znakova	Broj
`FontWeight`	Debljina znakova	Light, normal, bold
`Color`	Boja znakova	Oznaka boje
`BackgroundColor`	Boja pozadine znakova	Oznaka boje
`EdgeColor`	Boja ruba okvira oko znakova	Oznaka boje
`LineWidth`	Debljina ruba okvira oko znakova	Broj

Grčko slovo može se uključiti u tekst ako se upiše \englesko ime slova unutar znakovnog niza. Ako se ime slova upiše malim slovima, ispisat će se malo grčko slovo, a ako se ime slova upiše velikim početnim slovom, ispisat će se veliko grčko slovo, npr. \alpha ili \Alpha. U tablici 13.5 prikazane su samo neke oznake grčkih slova, a za detalje može se pogledati Octave help.

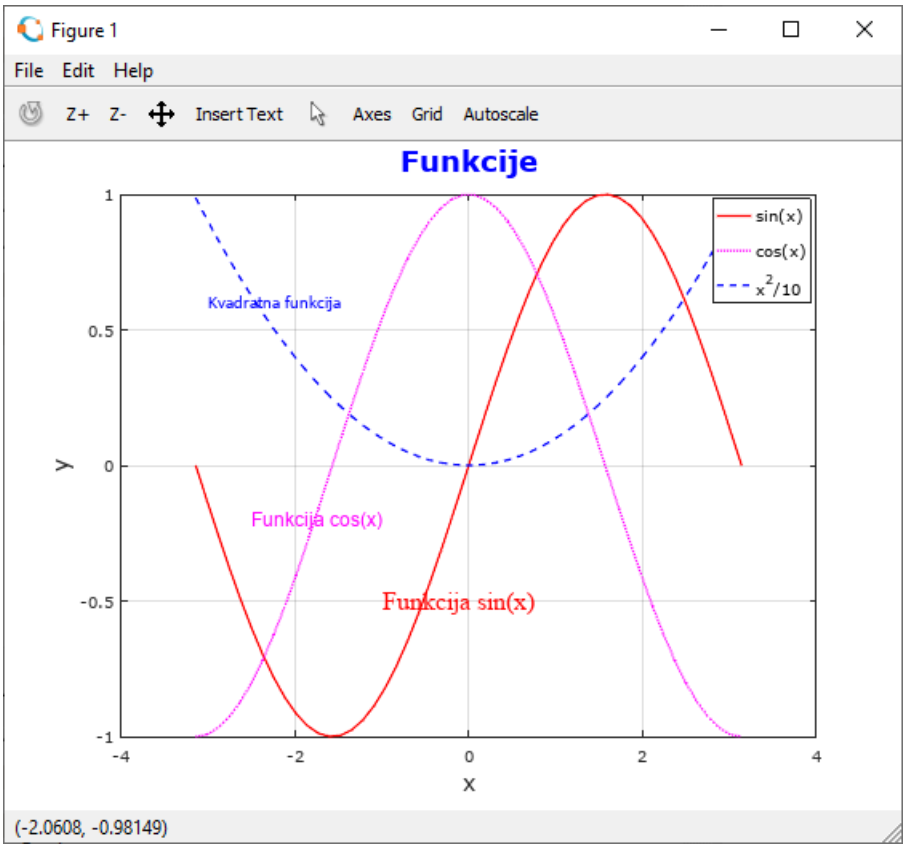
Tablica 13.5 Oznake grčkih slova u znakovnim nizovima

Slova u znakovnom nizu	Grčko slovo
\alpha	α
\beta	β
\gamma	γ
\theta	θ
\pi	π
\sigma	σ
\Phi	Φ
\Delta	Δ
\Gamma	Γ
\Lambda	Λ
\Omega	Ω
\Sigma	Σ

Primjer 13.8: Definiran je redni vektor **x** dimenzija 1*50 pomoću funkcije **linspace** u rasponu [-pi,pi] od 50 elemenata (točaka). Redni vektor **y1** dimenzija 1*50 vrijednost je funkcije sinus za vrijednosti vektora **x**. Redni vektor **y2** dimenzija 1*50 vrijednost je funkcije kosinus za vrijednosti vektora **x**. Redni vektor **y3** dimenzija 1*50 vrijednost je kvadratne funkcije za vrijednost vektora **x**. Naredba **plot** prikazuje prvu funkciju. Zatim se koristi naredba **hold on** za zadržavanje grafičkog prozora. Iza nje slijede dvije naredbe **plot** za prikazivanje druge i treće krivulje. Slijede naredbe **xlabel**, **ylabel** i **title** koje prikazuju oznake na apscisi, ordinati i naslovu iznad grafa. Nakon toga naredba **grid on** dodaje crte mreže na graf. Slijedi naredba **legend** za postavljanje legende na graf. Slijede naredbe **text** koje na samom grafu ispisuju tekst. U naredbama **xlabel**, **ylabel**, **title** i **text** koriste se parametri za oblikovanje teksta.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1, y2 i y3
x = linspace(-pi,pi,50);
y1 = sin(x);
y2 = cos(x);
y3 = (x.^2)/10;

% Prikaz grafova funkcija y1=sin(x), y2=cos(x) i y3=(x.^2)/10
plot(x,y1,'r-')
hold on
plot(x,y2,'m:')
plot(x,y3,'b--')
% Opis grafa
xlabel('x','FontSize',14)
ylabel('y','FontSize',14)
title('Funkcije','FontSize',18,'Color','b')
% Prikaz crta mreze grafa
grid on
% Prikaz legende grafa
legend('sin(x)','cos(x)','x^2/10')
% Ispis teksta na grafu
text(-1,-0.5,'Funkcija            sin(x)','FontName','Times            New
Roman','FontSize',16,'Color','r')
text(-2.5,-0.2,'Funkcija
cos(x)','FontName','Arial','FontSize',12,'Color','m')
text(-3,0.6,'Kvadratna
funkcija','FontName','Tahoma','FontSize',10,'Color','b')
```

Rezultat je otvaranje grafičkog prozora (engl. *figure*) i prikaz triju funkcija u rasponu [-pi,pi], što je prikazano na slici 13.8.



Slika 13.8 Grafički prozor u kojem su prikazane tri krivulje, označene apscisa i ordinata, naslov iznad grafa, legenda i tekst

13.3 Grafovi s logaritamskom podjelom

U mnogim znanstvenim i tehničkim primjenama, podjela jedne ili obje osi na grafu ima logaritamsku podjelu. Logaritamska podjela omogućuje prikazivanje velikog opsega vrijednosti, prepoznavanje karakteristika podataka i mogućih oblika matematičkih odnosa pogodnih za modeliranje podataka.

semilogx

Naredba `semilogx(x,y)` crta funkciju y(x) s logaritamskom podjelom apscise i jednolikom podjelom ordinate.

semilogy

Naredba `semilogy(x,y)` crta funkciju y(x) s jednolikom podjelom apscise i logaritamskom podjelom ordinate.

loglog

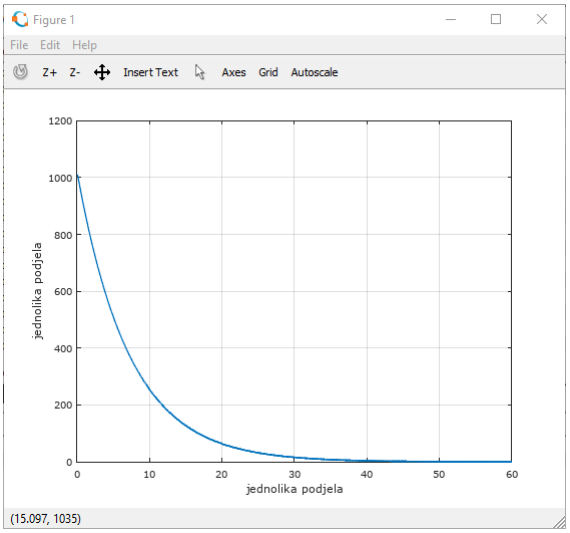
Naredba `loglog(x,y)` crta funkciju y(x) s logaritamskom podjelom apscise i logaritamskom podjelom ordinate.

Primjer 13.9: Definiran je redni vektor `x` dimenzija 1*1000 pomoću funkcije `linspace` u rasponu od 0,1 do 60 s 1000 elemenata (točaka). Redni vektor `y` dimenzija 1*1000 vrijednost je eksponencijalne funkcije za vrijednosti vektora `x`. Otvaraju se 4 grafička prozora. U prvom se grafičkom prozoru pomoću naredbe `plot` prikazuje krivulja s jednolikom podjelom obje osi. U drugom se grafičkom prozoru pomoću naredbe `semilogx` prikazuje krivulja s logaritamskom podjelom apscise i jednolikom podjelom ordinate. U trećem se grafičkom prozoru pomoću naredbe `semilogy` prikazuje krivulja s

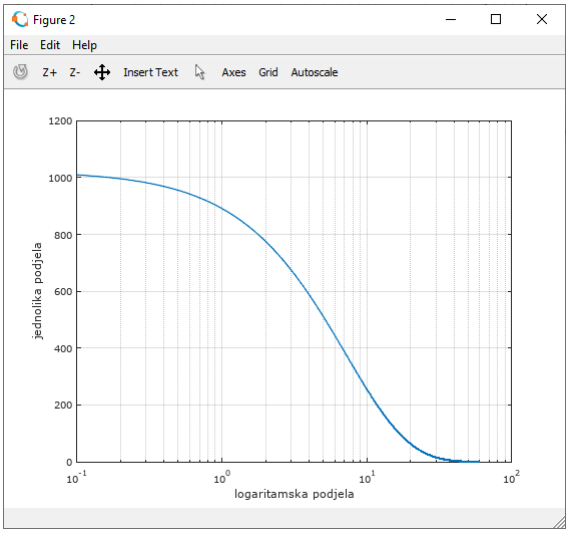
jednolikom podjelom apscise i logaritamskom podjelom ordinate. U četvrtom se grafičkom prozoru pomoću naredbe **`loglog`** prikazuje krivulja s logaritamskom podjelom obje osi.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x i y
x = linspace(0.1,60,1000);
y = 2.^(-0.2*x+10);

% Prikaz grafa funkcije y=f(x)
figure
plot(x,y,'LineWidth',2)
grid on
xlabel('jednolika podjela')
ylabel('jednolika podjela')

% Prikaz grafa funkcije y=f(x) s logaritamskom podjelom x-osi
figure
semilogx(x,y,'LineWidth',2)
grid on
xlabel('logaritamska podjela')
ylabel('jednolika podjela')

% Prikaz grafa funkcije y=f(x) s logaritamskom podjelom y-osi
figure
semilogy(x,y,'LineWidth',2)
grid on
xlabel('jednolika podjela')
ylabel('logaritamska podjela')

% Prikaz grafa funkcije y=f(x) s logaritamskom podjelom x-osi i y-osi
figure
loglog(x,y,'LineWidth',2)
grid on
xlabel('logaritamska podjela')
ylabel('logaritamska podjela')
```

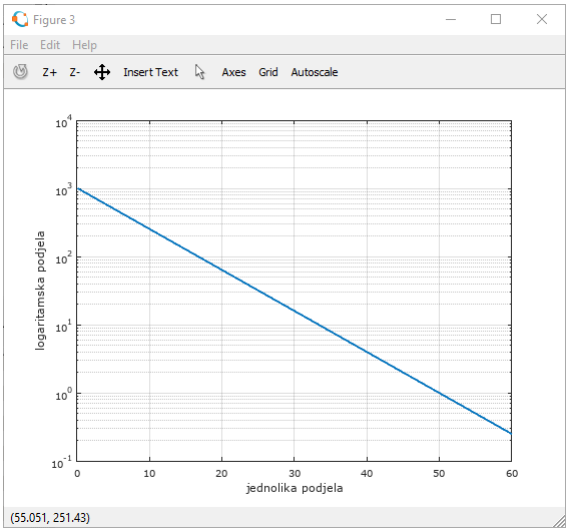
Rezultat je otvaranje četiriju grafičkih prozora (engl. *figure*) i prikaz funkcije u svakom od njih s različitom podjelom osi, što je prikazano na slici 13.9.



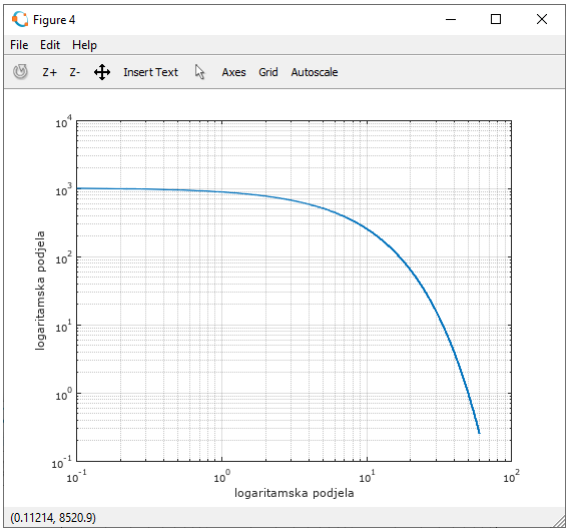
a. jednolika podjela apscise, jednolika podjela ordinate



b. logaritamska podjela apscise, jednolika podjela ordinate



c. jednolika podjela apscise, logaritamska podjela ordinate



d. logaritamska podjela apscise, logaritamska podjela ordinate

Slika 13.9 Grafovi s jednolikom i logaritamskom podjelom

axis

Kada se izvrši naredba **`plot`**, Octave automatski određuje raspon vrijednosti na osima na temelju vrijednosti funkcije koja se prikazuje. Raspon osi može se promijeniti pomoću naredbe **`axis`**. Naredba **`axis`** ima različite oblike, a ovdje će biti navedene one koje se najčešće koriste:

`axis([xmin xmax ymin ymax])` – određuje najmanju i najveću vrijednost osi na temelju vrijednosti **`xmin`**, **`xmax`**, **`ymin`** i **`ymax`**

`axis auto` – automatsko određivanje raspona osi

`axis equal` – odnos između apscise i ordinate je 1:1

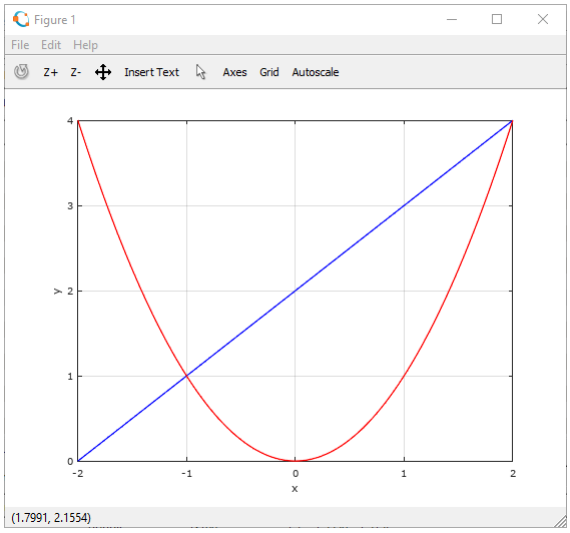
`axis square` – kvadratičan oblik područja osi (raspon apscise i ordinate jednake je duljine).

Primjer 13.10: Definiran je redni vektor **`x`** dimenzija 1*100 pomoću funkcije **`linspace`** u rasponu [-2,2] od 100 elemenata (točaka). Redni vektor **`y1`** dimenzija 1*100 vrijednost je funkcije **`y=x+2`** za vrijednosti vektora **`x`**. Redni vektor **`y2`** dimenzija 1*100 vrijednost je funkcije **`y=x.^2`** za vrijednosti vektora **`x`**. Otvaraju se 4 grafička prozora. U prvom se grafičkom prozoru prikazuju dvije krivulje s automatskom podjelom obje osi. U drugom se grafičkom prozoru pomoću naredbe **`axis([-3 3 -1 5])`** prikazuju dvije krivulje s apscisom u rasponu [-3,3] i ordinatom u rasponu [-1,5]. U trećem se grafičkom prozoru pomoću naredbe **`axis equal`** prikazuju dvije krivulje kod kojih je odnos apscise i ordinate 1:1. U četvrtom se grafičkom prozoru pomoću naredbe **`axis square`** prikazuju dvije krivulje s kvadratičnim oblikom područja osi (raspon apscise i ordinate jednake je duljine).

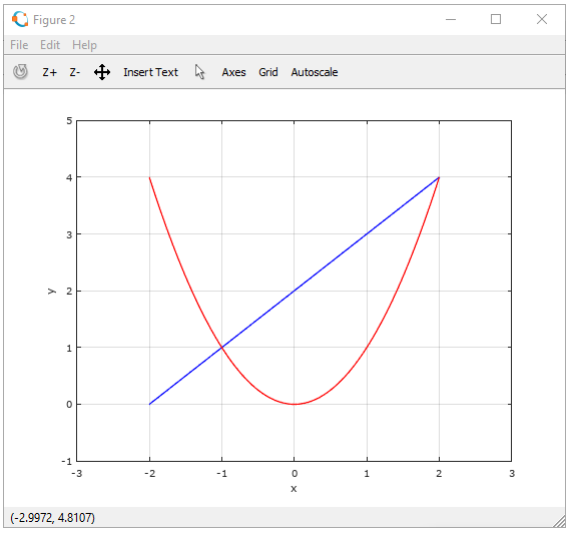
```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1 i y2
x = linspace(-2,2,100);
y1 = x+2;
y2 = x.^2;

% Prikaz grafova funkcija y1 i y2
figure
plot(x,y1,'b','LineWidth',2)
hold on
plot(x,y2,'r','LineWidth',2)
```

```
grid on
xlabel('x')
ylabel('y')

% Prikaz grafova funkcija y1 i y2 s axis([-3 3 -1 5])
figure
plot(x,y1,'b','LineWidth',2)
hold on
plot(x,y2,'r','LineWidth',2)
grid on
xlabel('x')
ylabel('y')
axis([-3 3 -1 5])

% Prikaz grafova funkcija y1 i y2 s axis equal
figure
plot(x,y1,'b','LineWidth',2)
hold on
plot(x,y2,'r','LineWidth',2)
grid on
xlabel('x')
ylabel('y')
axis equal

% Prikaz grafova funkcija y1 i y2 s axis square
figure
plot(x,y1,'b','LineWidth',2)
hold on
plot(x,y2,'r','LineWidth',2)
grid on
xlabel('x')
ylabel('y')
axis square
```

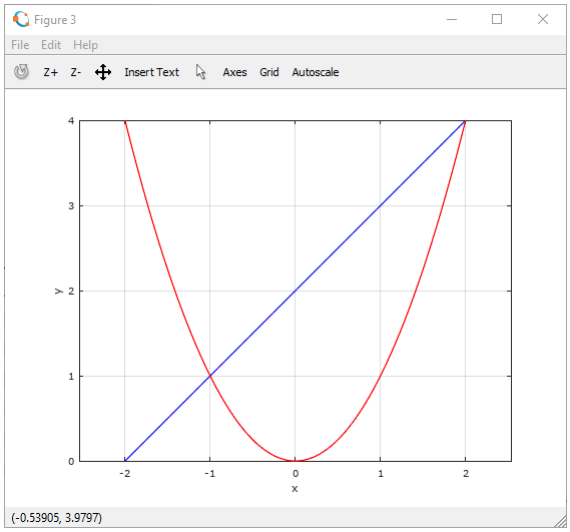
Rezultat je otvaranje četiriju grafičkih prozora (engl. *figure*) i prikaz dviju funkcija u svakom od njih pomoću različitih oblika naredbe **`axis`**, što je prikazano na slici 13.10.



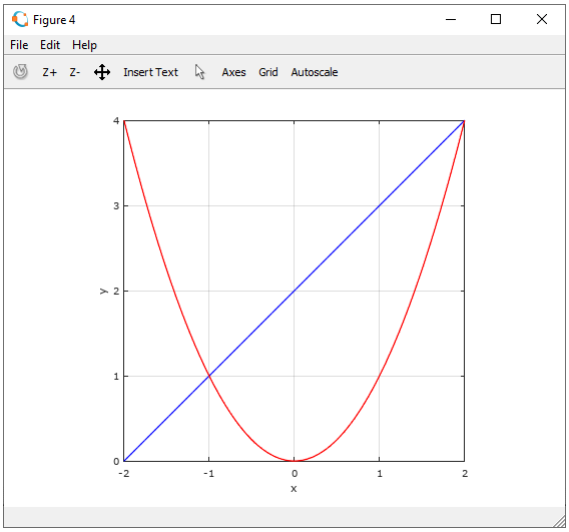
a. automatsko oblikovanje podjela osi



b. oblikovanje podjela osi naredbom **`axis([-3 3 -1 5])`**



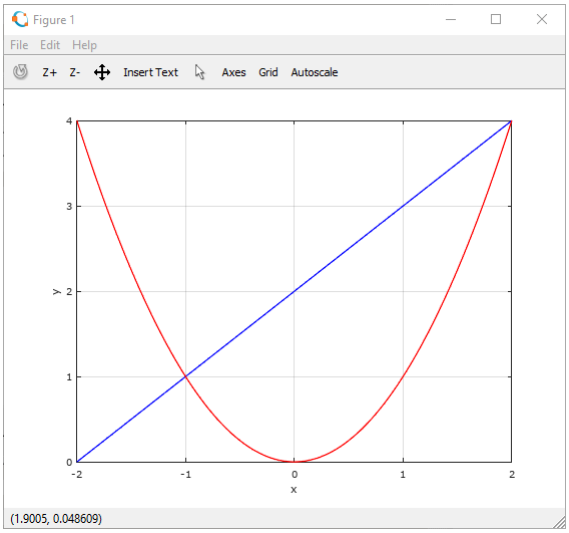
c. oblikovanje podjela osi naredbom **`axis equal`**



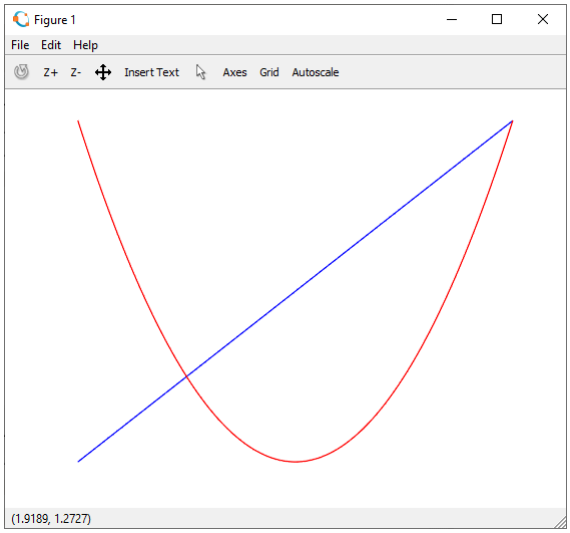
d. oblikovanje podjela osi naredbom **`axis square`**

Slika 13.10 Izgled grafova dobiven pomoću različitih oblika naredbe **`axis`**

Pri crtanju grafova automatski se prikazuju osi grafa. Pomoću naredbe **`axis`** moguće je uključiti ili isključiti osi grafa. Naredba za isključivanje osi je **`axis off`**, a naredba za uključivanje osi **`axis on`**. Na slici 13.11 prikazano je kako izgleda isti graf s uključenim i isključenim osima.



a. graf s prikazom osi



b. graf bez prikaza osi

Slika 13.11 Grafički prozor s uključenim i isključenim prikazom osi

13.4 Posebne vrste grafova

Pomoću naredbe **`plot`** crtaju se grafovi u kojima su točke podataka (vrijednosti funkcije) povezane pravocrtnim segmentima. Grafovi s drukčije povezanim ili prikazanim vrijednostima ponekad bolje predočavaju podatke.

bar

Pomoću naredbe **`bar`** crta se uspravni vrpčasti graf. Oblik naredbe je:

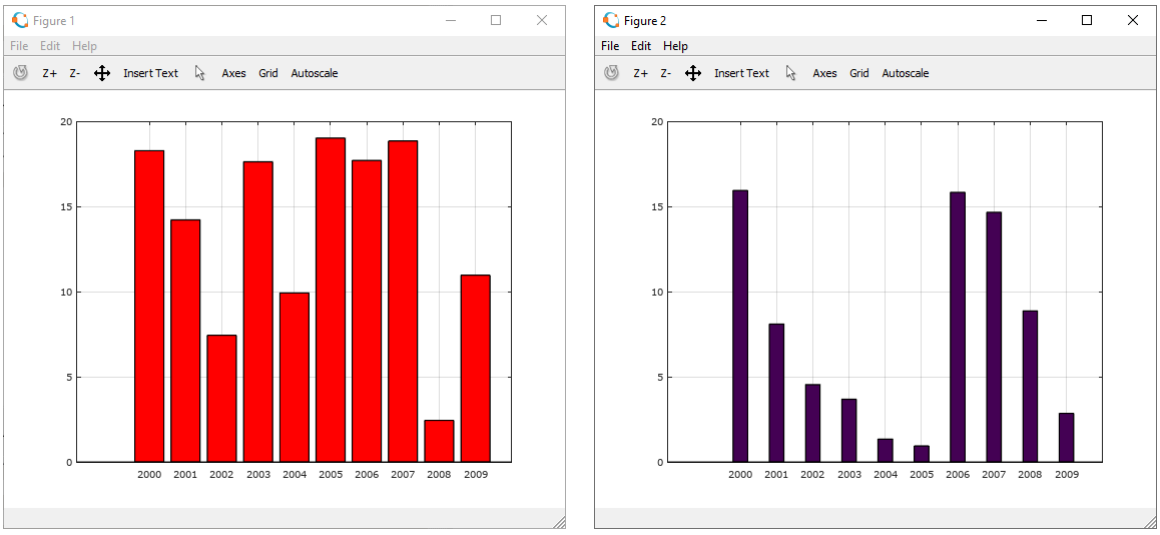
`bar(x,y)` – vektori **`x`** i **`y`** predstavljaju podatke

bar(x,y,w) – vektori **x** i **y** predstavljaju podatke, a skalar **w** predstavlja širinu vrpci u uspravnom vrpčastom grafu.

Primjer 13.11: Definirani su vektori **x** i **y** s podatcima. Vektor **y** sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,20]. Program crta dva grafa, svaki u svom grafičkom prozoru. U drugom grafičkom prozoru uspravnom vrpčastom grafu određuje se širina vrpca od 0,4. U naredbi **bar** moguće je odrediti boju vrpca na isti način kao i kod naredbe **plot**.

```
close all
clear all
clc

format short
format compact

figure
% Definiranje vektora x i y
x = 2000:2009;
y = 20*rand(1,10);
% Prikaz uspravnog vrpcastog grafa
bar(x,y,'r')
grid on

figure
% Definiranje vektora x i y
x = 2000:2009;
y = 20*rand(1,10);
% Prikaz uspravnog vrpcastog grafa
bar(x,y,0.4)
grid on
```

Rezultat je otvaranje dvaju grafičkih prozora (engl. *figure*) i prikaz uspravnog vrpčastog grafa u svakom od njih, što je prikazano na slici 13.12.



a. debljina vrpci 0,8

b. debljina vrpci 0,4

Slika 13.12 Uspravni vrpčasti grafovi

barh

Pomoću naredbe **barh** crta se vodoravni vrpčasti graf. Oblik naredbe je:

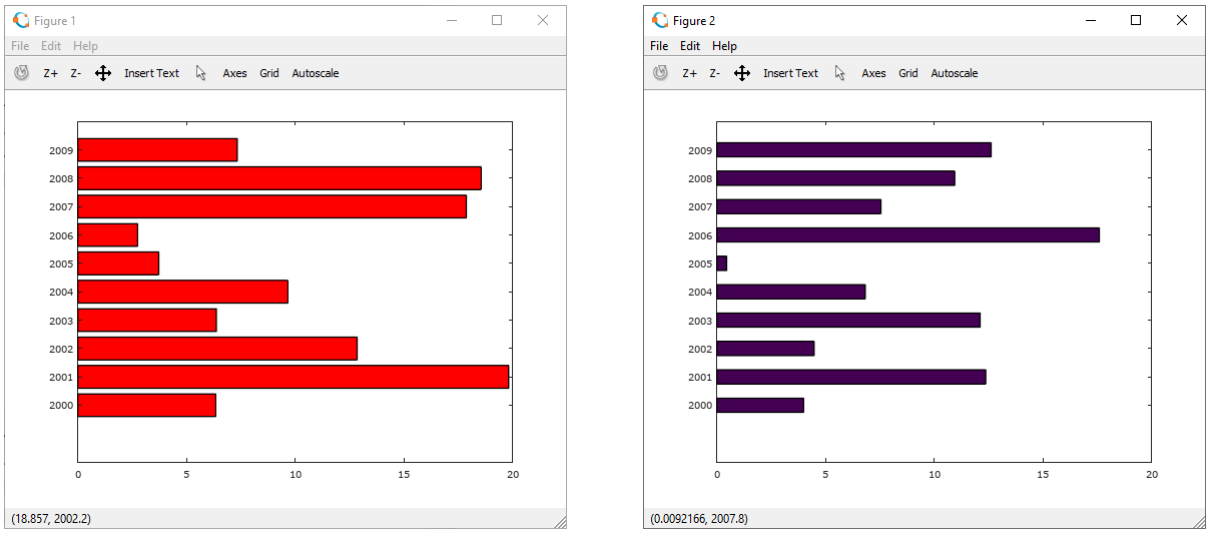
barh(x,y) – vektori **x** i **y** predstavljaju podatke

barh(x,y,w) – vektori **x** i **y** predstavljaju podatke, a skalar **w** predstavlja širinu vrpci u vodoravnom vrpčastom grafu.

Primjer 13.12: Definirani su vektori **x** i **y** s podatcima. Vektor **y** sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,20]. Program crta dva grafa, svaki u svom grafičkom prozoru. U drugom grafičkom prozoru vodoravnom vrpčastom grafu određuje se širina vrpca od 0,5. U naredbi **barh** moguće je odrediti boju vrpca na isti način kao i kod naredbe **plot**.

```
close all
clear all
clc

format short
format compact

figure
% Definiranje vektora x i y
x = 2000:2009;
y = 20*rand(1,10);
% Prikaz vodoravnog vrpcastog grafa
barh(x,y,'r')
axis auto

figure
% Definiranje vektora x i y
x = 2000:2009;
y = 20*rand(1,10);
% Prikaz vodoravnog vrpcastog grafa
barh(x,y,0.5)
axis auto
```

Rezultat je otvaranje dvaju grafičkih prozora (engl. *figure*) i prikaz vodoravnog vrpčastog grafa u svakom od njih, što je prikazano na slici 13.13.



a. debljina vrpci 0,8

b. debljina vrpci 0,5

Slika 13.13 Vodoravni vrpčasti grafovi

Naredbe **bar** i **barh** osim vektora prihvaćaju i matricu kao ulazni podatak za prikaz uspravnih ili vodoravnih vrpčastih grafova. Kada je matrica ulazni podatak za naredbe **bar** i **barh**, podatci svakog retka predstavljaju grupe podataka koji se prikazuju na mjestima koji predstavljaju redne brojeve redaka matrice. Oblik naredbi **bar** i **barh** za takav tip grafova je:

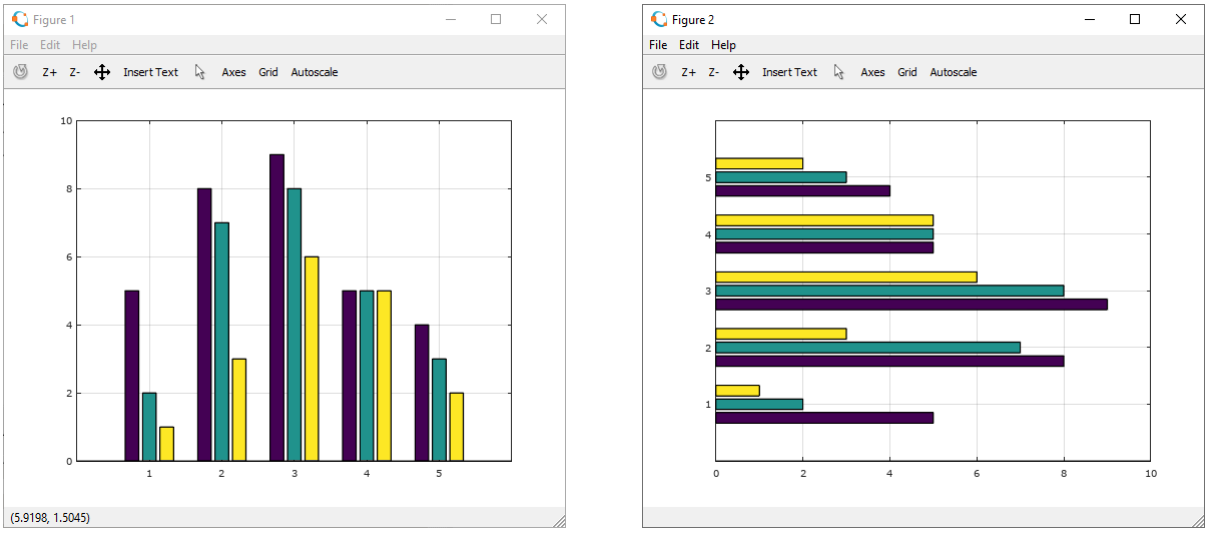
`bar(y,'grouped')` – **`y`** je matrica

`barh(y,'grouped')` – **`y`** je matrica.

Primjer 13.13: Matrica **`y`** dimenzija 5*3, što znači da ima 5 redaka i 3 stupca, ulazni je podatak za naredbe **`bar`** i **`barh`**. Svaki od pet redaka s tri podatka predstavlja jednu grupu podataka. Na temelju matrice **`y`**, naredbe **`bar`** i **`barh`** crtaju 5 grupa s tri podatka čije vrijednosti predstavljaju visinu vrpca.

```
close all
clear all
clc

format short
format compact

% Matrica y
y = [5 2 1; 8 7 3; 9 8 6; 5 5 5; 4 3 2];

% Prikaz uspravnog grupiranog vrpcastog grafa
figure
bar(y,'grouped')
grid on

% Prikaz vodoravnog grupiranog vrpcastog grafa
figure
barh(y,'grouped')
grid on
```

Rezultat je otvaranje dvaju grafičkih prozora (engl. *figure*) i prikaz uspravnog i vodoravnog grupiranog vrpčastog grafa, što je prikazano na slici 13.14.



a. uspravni graf

b. vodoravni graf

Slika 13.14 Grupirani vrpčasti grafovi

Osim ovakvog oblika uspravnog i vodoravnog vrpčastog grafa kada je ulazni podatak matrica, postoji i drugi oblik kada se podatci svakog retka matrice "slažu jedan na drugoga". Kada je matrica ulazni podatak za naredbe **`bar`** i **`barh`**, podatci svakog retka predstavljaju grupe podataka koji se prikazuju jedan iznad drugog na mjestima koji predstavljaju redne brojeve redaka matrice. Taj je tip grafa ponekad

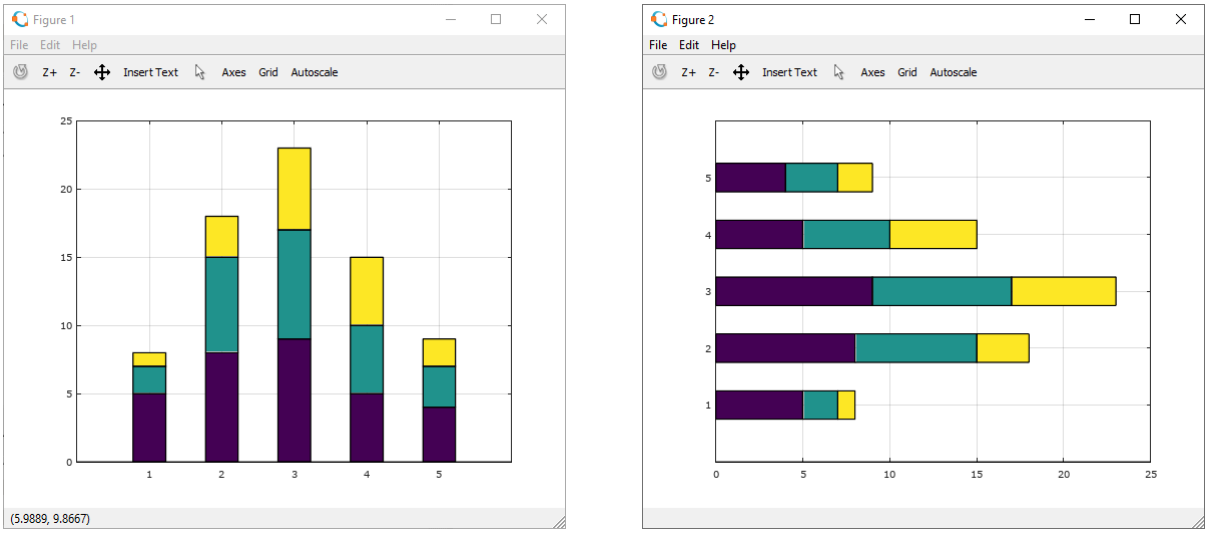
pogodno koristiti jer prikazuje udio pojedinog podatka u grupi podataka. Oblik naredbi **bar** i **barh** za takav tip grafova je:

bar(y,'stacked') – **y** je matrica

barh(y,'stacked') – **y** je matrica.

Primjer 13.14: Matrica **y** dimenzija 5*3, što znači da ima 5 redaka i 3 stupca, ulazni je podatak za naredbe **bar** i **barh**. Svaki od pet redaka s tri podatka predstavlja jednu grupu podataka koji se crtaju jedan iznad drugog. Na temelju matrice **y**, naredbe **bar** i **barh** crtaju pet grupa s tri podatka čiji zbroj predstavlja visinu vrpce. U prvom se grafičkom prozoru prikazuje rezultat naredbe **bar** gdje je širina vrpca 0,45, a u drugom se grafičkom prozoru prikazuje rezultat naredbe **barh** gdje je širina vrpca 0,5 što je prikazano na slici 13.15.

```
close all
clear all
clc

format short
format compact

% Matrica y
y = [5 2 1; 8 7 3; 9 8 6; 5 5 5; 4 3 2];

% Prikaz uspravnog "slaganog jedan na drugi" vrpcastog grafa
figure
bar(y,0.45,'stacked')
grid on

% Prikaz vodoravnog "slaganog jedan na drugi" vrpcastog grafa
figure
barh(y,0.5,'stacked')
grid on
```



a. uspravni graf

b. vodoravni graf

Slika 13.15 Vrpčasti grafovi "slagani jedan na drugi"

area

Pomoću naredbe **area** crtaju se popunjeni grafovi. Oblik naredbe je:

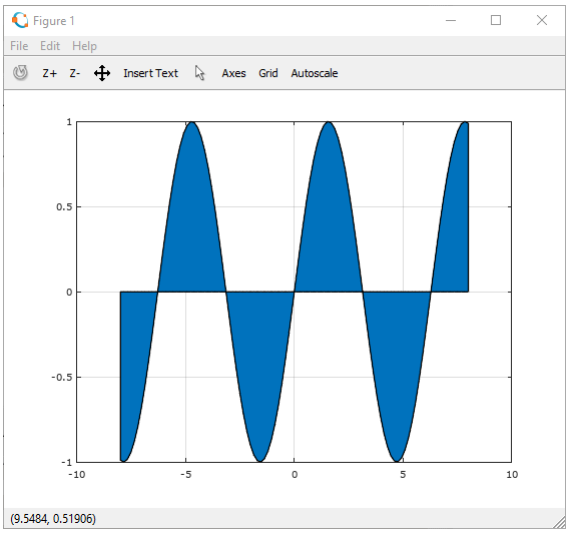
area(x,y) – gdje su **x** i **y** vektori.

Ako su **x** i **y** vektori, naredba **area(x,y)** prikazuje sličan graf koji se dobiva pomoću naredbe **plot(x,y)**, osim što je krivulja ispunjena između vrijednosti 0 i y.

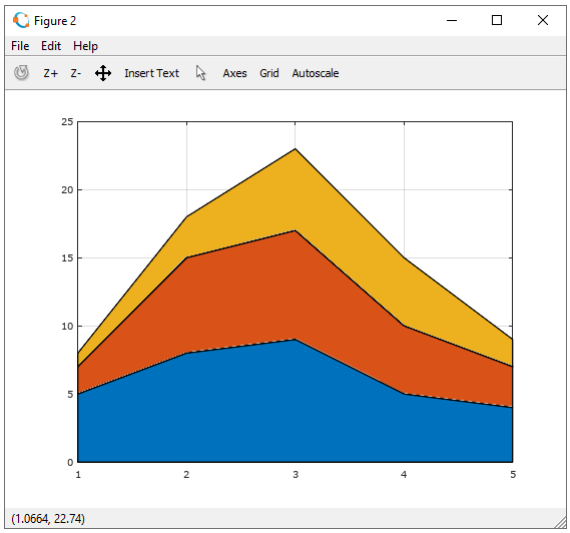
Kao ulazni podatak naredbe **area** može se rabiti i matrica. Tada se svaki redak matrice rabi kao grupa podataka koja se prikazuje na mjestu koje predstavlja redni broj retka matrice. Taj je tip grafa ponekad pogodno koristiti jer prikazuje udio pojedinog podatka u grupi podataka.

Primjer 13.15: Definiran je redni vektor **x** pomoću funkcije **linspace** u rasponu [-8,8], a sastoji se od 50 elemenata. Vektor **y** vrijednost je funkcije sinus za vrijednosti vektora **x**. U prvom se grafičkom prozoru pomoću naredbe **area(x,y)** prikazuje krivulja sinus koja je ispunjena između vrijednosti 0 i vrijednosti funkcije sinus (slika 13.16). U drugom se grafičkom prozoru prikazuje pomoću naredbe **area(y)** popunjeni graf koji koristi matricu **y** dimenzija 5*3 kao ulazni podatak (slika 13.16). Za svaki redni broj retka matrice koriste se podatci u njemu, slažući se jedan na drugoga (slično kao kod naredbe **bar(y,'stacked')**). To znači da postoji pet grupa po tri podatka.

```
close all
clear all
clc

format short
format compact

figure
% Definiranje vektora x i y
x = linspace(-8,8,100);
y = sin(x);
% Prikaz popunjenog grafa
area(x,y)
grid on

figure
% Matrica y
y = [5 2 1; 8 7 3; 9 8 6; 5 5 5; 4 3 2];
% Prikaz popunjenog grafa
area(y)
grid on
```



a. primjer grafa s podatcima zadanim u vektoru



b. primjer grafa s podatcima zadanim u matrici

Slika 13.16 Popunjeni grafovi dobiveni naredbom **area**

stairs

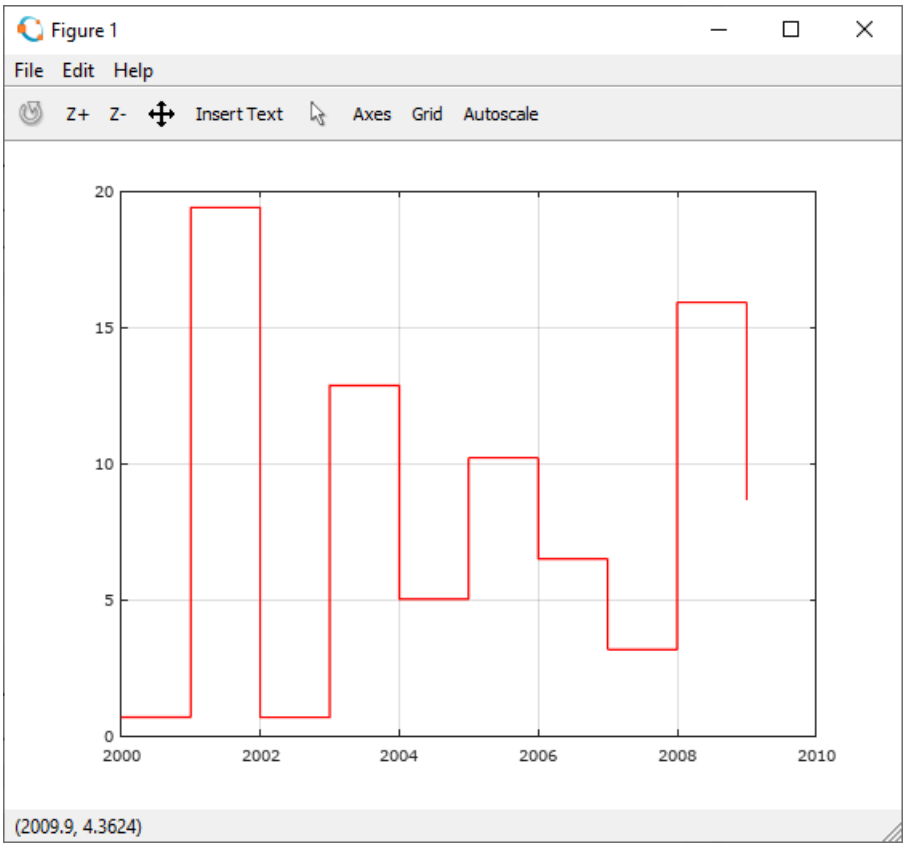
Pomoću naredbe **stairs** crtaju se stepeničasti grafovi. Oblik naredbe je:

stairs(x,y) – vektori **x** i **y** predstavljaju podatke

stairs(x,y,'znakovni niz') – vektori **x** i **y** predstavljaju podatke, a **'znakovni niz'** određuje boju, debljinu i stil crte kao kod naredbe **plot**.

Primjer 13.16: Definirani su vektori **x** i **y**. Vektor **y** sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,20]. Program crta stepeničasti graf na temelju podataka u vektorima **x** i **y**. Stepeničastom grafu definirana je crvena boja crte debljine 2, što je prikazano na slici 13.17.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x i y
x = 2000:2009;
y = 20*rand(1,10);
% Prikaz stepenicastog grafa
stairs(x,y,'r','LineWidth',2)
grid on
```



Slika 13.17 Stepeničasti graf

stem

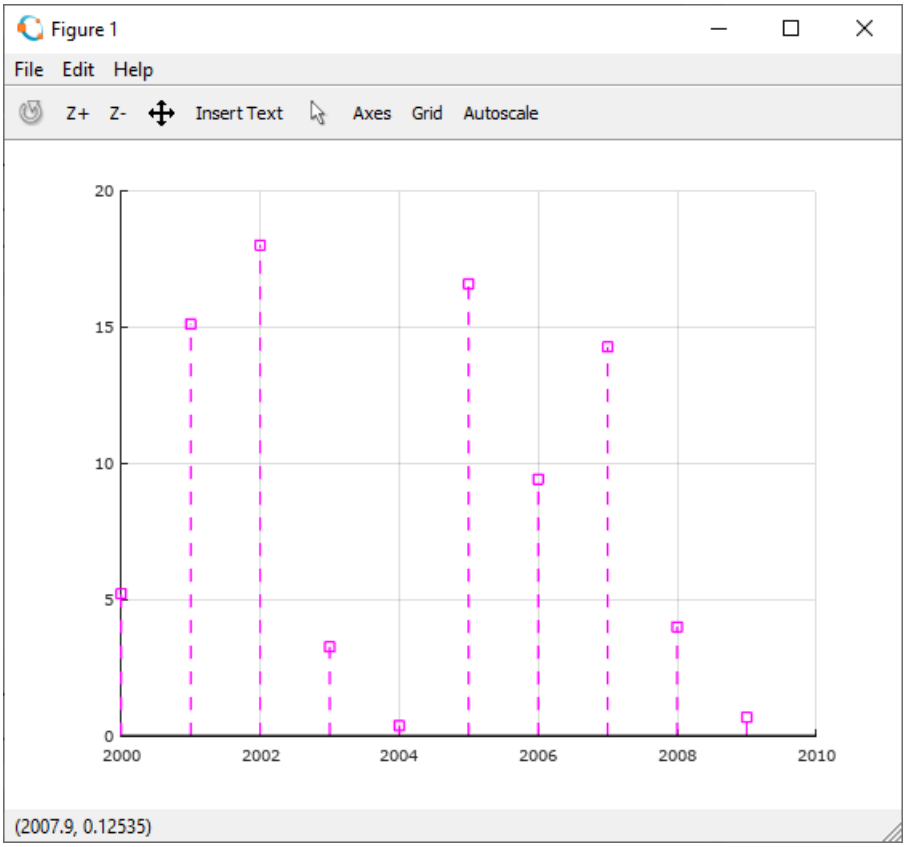
Pomoću naredbe **stem** crta se graf diskretnih podataka. Oblik naredbe je:

stem(x,y) – vektori **x** i **y** predstavljaju podatke

`stem(x,y,'znakovni niz')` – vektori **`x`** i **`y`** predstavljaju podatke, a **`'znakovni niz'`** određuje boju, debljinu i stil crte, kao kod naredbe **`plot`**.

Primjer 13.17: Definirani su vektori **`x`** i **`y`**. Vektor **`y`** sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,20]. Program crta graf diskretnih podataka na temelju podataka u vektorima **`x`** i **`y`**. Grafu diskretnih podataka definirana je magenta boja, crtkana crta, debljine 2 te marker oblika kvadrata (slika 13.18).

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x i y
x = 2000:2009;
y = 20*rand(1,10);
% Prikaz grafa diskretnih podataka
stem(x,y,'ms--','LineWidth',2)
grid on
```



Slika 13.18 Graf diskretnih podataka

pie

Pomoću naredbe **`pie`** crta se tortni graf. Oblici naredbe su:

`pie(x)` – crta tortni graf s podatcima sadržanim u vektoru **`x`**.

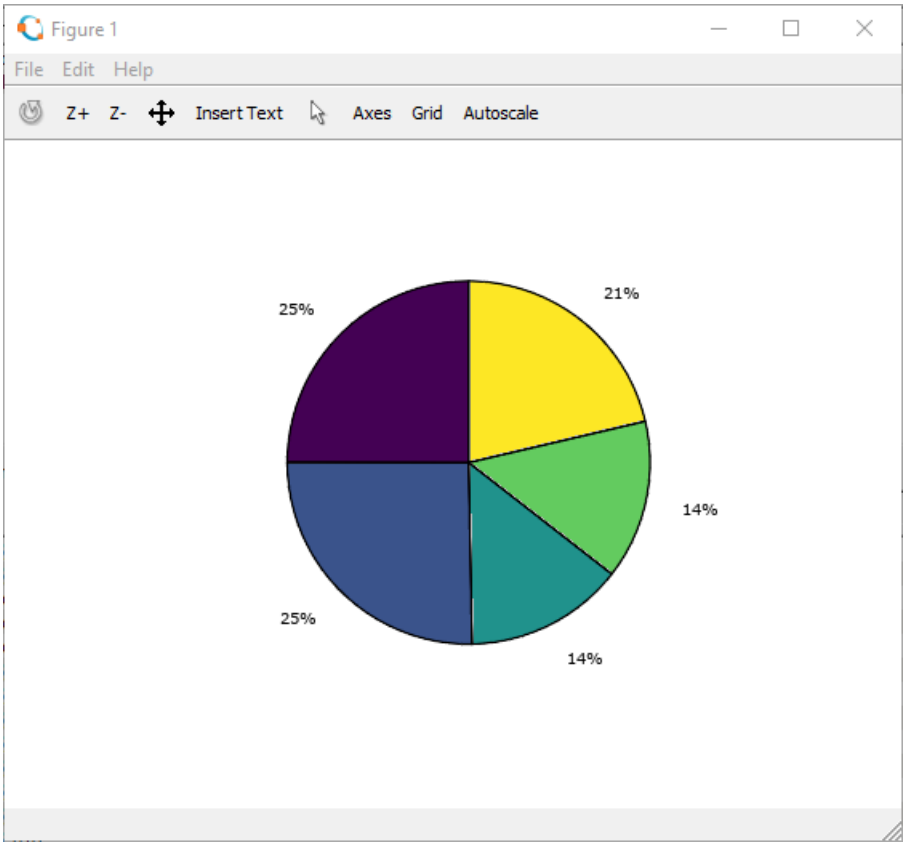
`pie(x,explode)` – crta tortni graf s podatcima sadržanim u vektoru **`x`**, a vektor **`explode`** služi za izdvajanje segmenata tortnog grafa. Vektor **`explode`** mora biti iste dimenzije kao i vektor **`x`**. Ako su

elementi vektora **explode** različiti od 0, odgovarajući elementi vektora **x** bit će izdvojeni segmenti u tortnom grafu. Najčešće vektor **explode** sadrži vrijednosti 0 i 1.

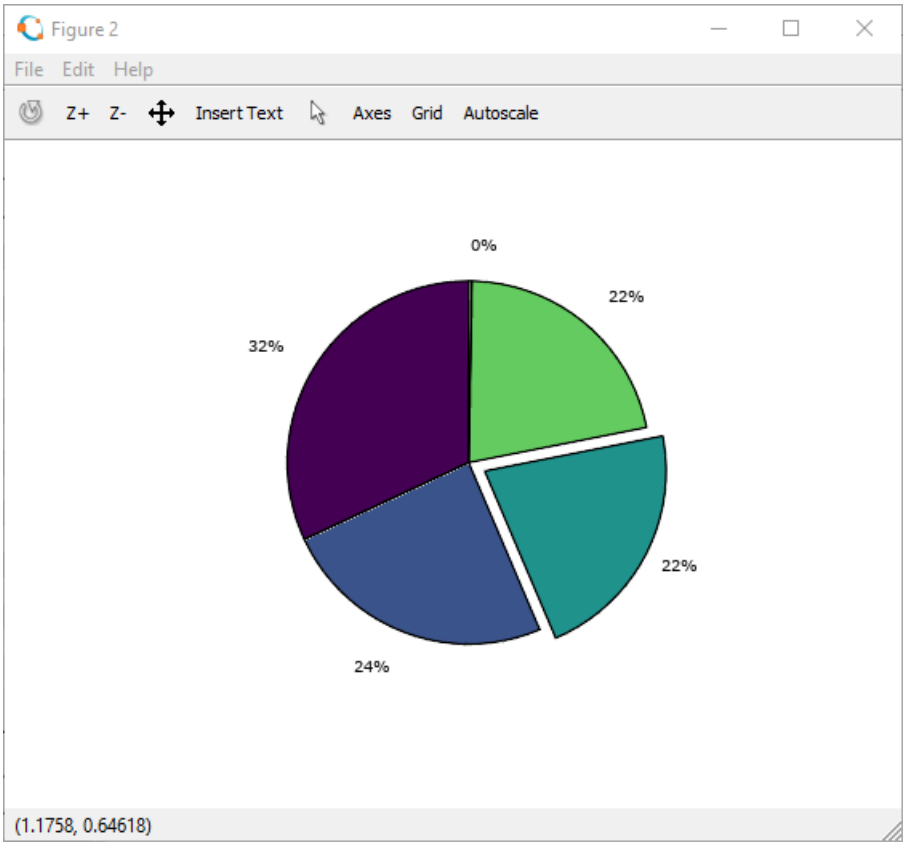
pie(x,'znakovni niz') – crta tortni graf s podatcima sadržanim u vektoru **x**, a **'znakovni niz'**, koji mora biti iste dimenzije kao i vektor **x**, sadrži opise segmenata tortnog grafa.

Primjer 13.18: Vektor **x** sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,20]. Program otvara tri grafička prozora i prikazuje tortne grafove u svakom od njih. U prvom se grafičkom prozoru prikazuje tortni graf sa pet segmenta i oznaka segmenata u postotcima (slika 13.19). U drugom se grafičkom prozoru prikazuje tortni graf sa pet segmenta kojem je treći segment izdvojen (slika 13.20). U trećem se grafičkom prozoru prikazuje tortni graf sa četiri segmenta na kojem su segmenti opisani tekstom (slika 13.21).

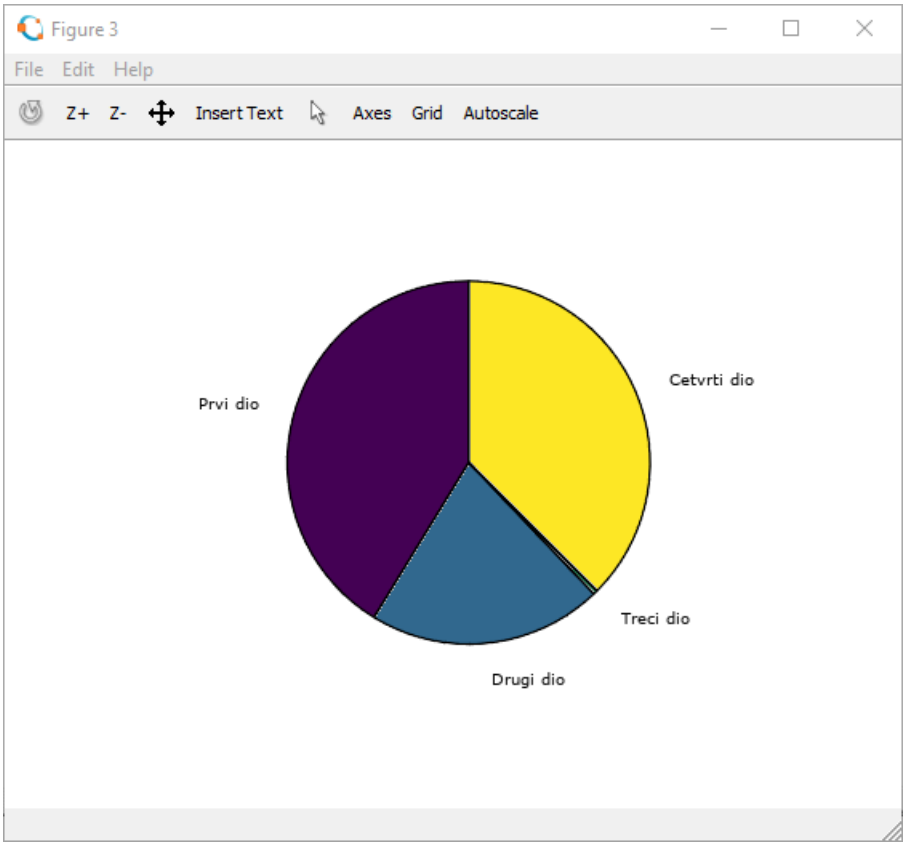
```
close all
clear all
clc

format short
format compact

% Prikaz tortnog grafa
figure
x = 20*rand(1,5);
pie(x)

% Prikaz tortnog grafa
figure
x = 20*rand(1,5);
explode = [0 0 1 0 0];
pie(x,explode)

% Prikaz tortnog grafa
figure
x = 10*rand(1,4);
labels = {'Prvi dio','Drugi dio','Treci dio','Cetvrti dio'};
pie(x,labels)
```



Slika 13.19 Tortni graf



Slika 13.20 Tortni graf s jednim izdvojenim segmentom



Slika 13.21 Tortni graf s opisanim segmentima

scatter

Pomoću naredbe **`scatter`** crtaju se samo markeri na koordinatama (x,y) koje predstavljaju podatke. Naredba ima nekoliko oblika:

`scatter(x,y)` – prikazuje markere na mjestima određenim vrijednostima vektora **`x`** i **`y`**.

`scatter(x,y,w)` – prikazuje markere na mjestima određenim vrijednostima vektora **`x`** i **`y`**, a vektor **`w`** određuje veličinu markera kao površinu (u zaslonskim točkama na kvadrat).

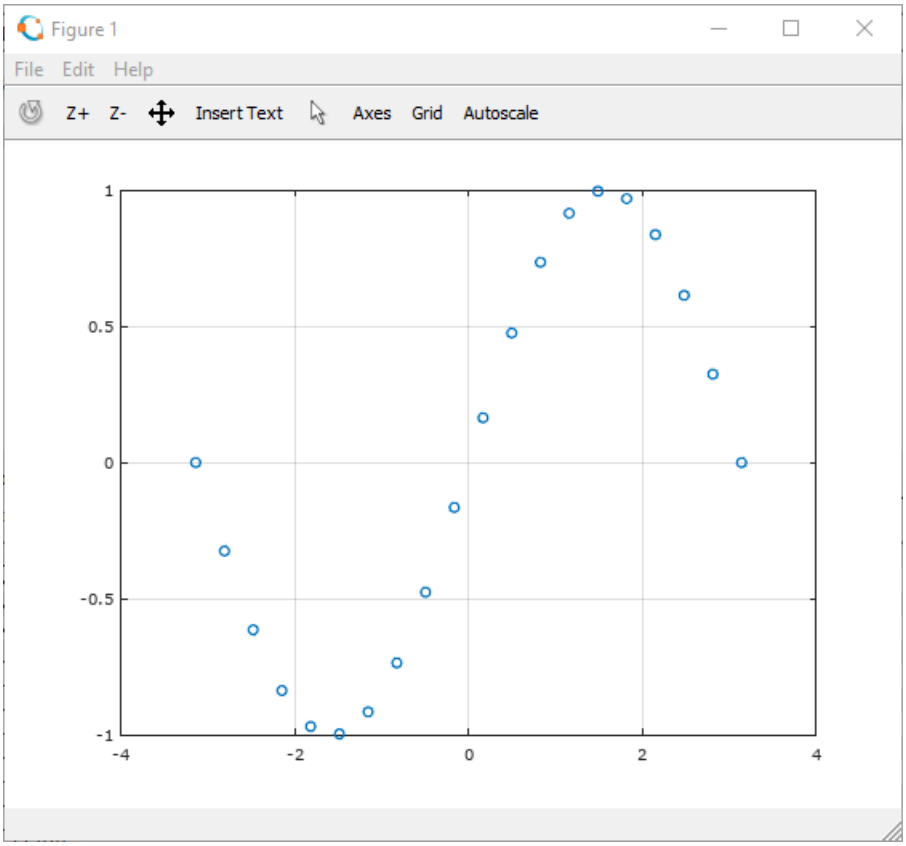
`scatter(x,y,w,'filled')` – prikazuje markere na mjestima određenim vrijednostima vektora **`x`** i **`y`**, a vektor **`w`** određuje veličinu markera kao površinu (u zaslonskim točkama na kvadrat). Parametar **`'filled'`** određuje da će markeri biti ispunjeni.

`scatter(x,y,w,q,'filled')` – prikazuje markere na mjestima određenim vrijednostima vektora **`x`** i **`y`**, a vektor **`w`** određuje veličinu markera kao površinu (u zaslonskim točkama na kvadrat). Ako je **`q`** vektor iste dimenzije kao vektori **`x`** i **`y`**, on određuje redni broj boje u trenutačnoj paleti boja. Ako je **`q`** matrica dimenzija (dimenzija vektora x ili y)*3, svaki redak matrice sa tri elementa određuje udjele komponenata boja (kao kod naredbe **`plot`**). Parametar **`'filled'`** određuje da će markeri biti ispunjeni.

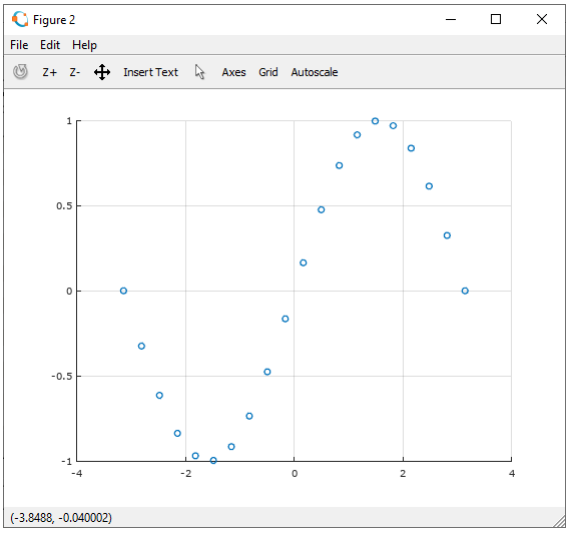
U svom najjednostavnijem obliku naredba **`scatter(x,y)`** daje isti rezultat kao i naredba **`plot(x,y,'oznaka markera')`**.

Primjer 13.19: Definiran je redni vektor **`x`** dimenzija 1*20 pomoću funkcije **`linspace`** u rasponu [-pi,pi], sa 20 elemenata. Vektor **`y`** je vrijednost funkcije sinus za vrijednosti vektora **`x`**. Otvara se pet grafičkih prozora koji su prikazani na slikama 13.22 i 13.23. U prvom se grafičkom prozoru pomoću naredbe **`plot`** prikazuju samo markeri na mjestima određenim vektorima **`x`** i **`y`**. U drugom se grafičkom prozoru prikazuju markeri pomoću naredbe **`scatter`**. Treba uočiti da su prvi i drugi grafički prozor isti, iako su nastali pomoću naredbi **`plot`** i **`scatter`**. U trećem se prozoru prikazuju markeri različite veličine pomoću naredbe **`scatter`**. U četvrtom se prozoru prikazuju popunjeni markeri različite veličine pomoću naredbe **`scatter`**. U petom se prozoru prikazuju popunjeni markeri u različitim bojama i različite veličine pomoću naredbe **`scatter`**.

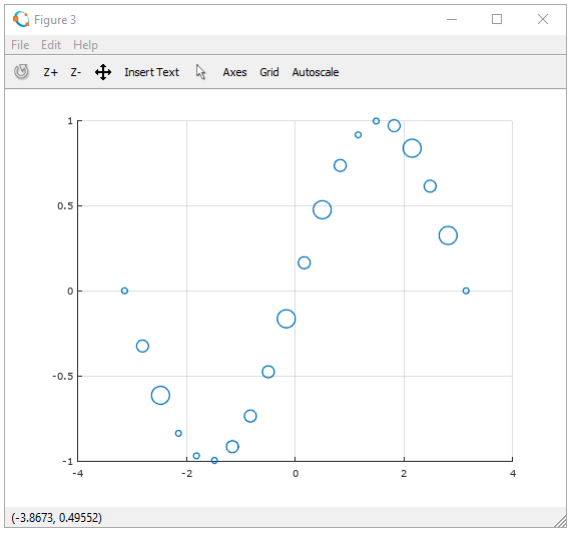
```
close all
clear all
clc

format short
format compact

% Definiranje vektora x i y
x = linspace(-pi,pi,20);
y = sin(x);

% Prikaz grafa funkcije y=sin(x)
figure
plot(x,y,'o')
grid on

% Prikaz grafa funkcije y=sin(x) pomocu naredbe scatter
figure
scatter(x,y)
grid on

% Prikaz grafa funkcije y=sin(x) pomocu naredbe scatter
figure
w = [6 12 18 6 6 6 12 12 12 18 12 18 12 6 6 12 18 12 18 6];
scatter(x,y,w.^2)
grid on

% Prikaz grafa funkcije y=sin(x) pomocu naredbe scatter
figure
w = [6 12 18 6 6 6 12 12 12 18 12 18 12 6 6 12 18 12 18 6];
scatter(x,y,w.^2,'filled')
grid on

% Prikaz grafa funkcije y=sin(x) pomocu naredbe scatter
figure
w = [6 12 18 6 6 6 12 12 12 18 12 18 12 6 6 12 18 12 18 6];
q = rand(20,3);
scatter(x,y,w.^2,q,'filled')
grid on
```



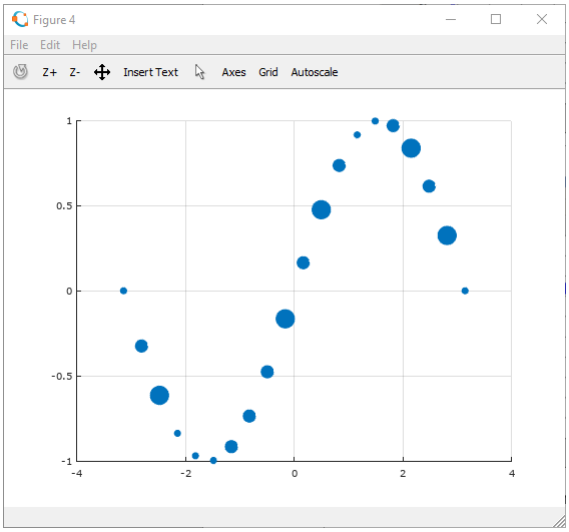
Slika 13.22 Graf prikazan pomoću naredbe `plot` koji se sastoji samo od markera



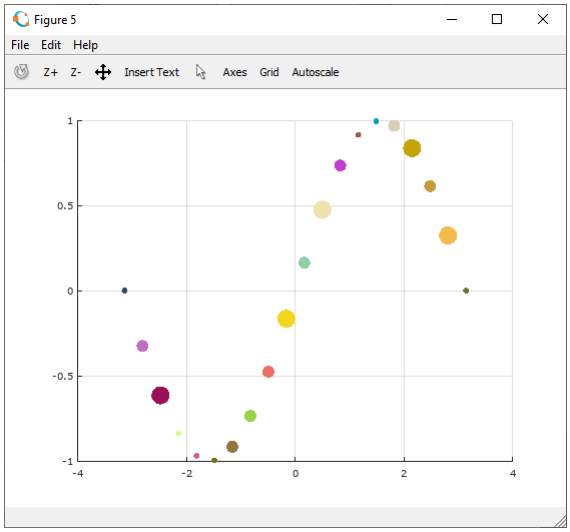
a. jednaka veličina oznaka



b. definirane različite veličine oznaka



c. definirane različite veličine popunjenih oznaka



d. definirane različite veličine popunjenih raznobojnih oznaka

Slika 13.23 Grafovi nastali uporabom različitih oblika naredbe **`scatter`**

polar

Ponekad je potrebno prikazati graf čiji podatci x i y, odnosno koordinate (x,y), nisu u Kartezijevu koordinatnom sustavu nego su podatci u polarnim koordinatama. Polarne koordinate sastoje se od udaljenosti (amplituda) točke u ravnini do ishodišta koordinatnog sustava i kuta koji zatvara ta amplituda s apscisom. Za takav se graf u Octaveu koristi naredba **`polar`** čiji je oblik:

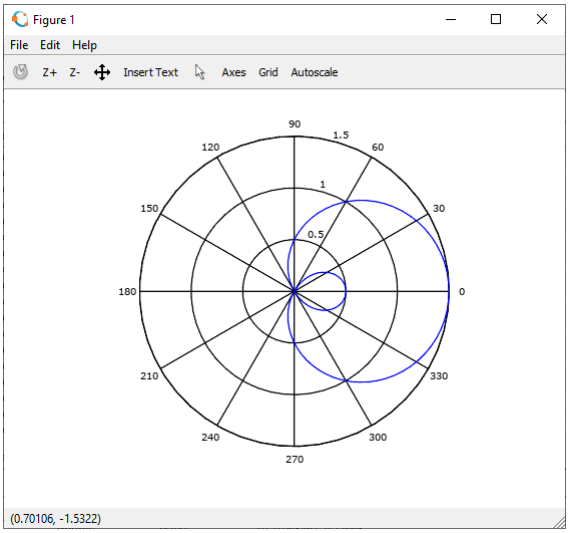
`polar(fi,r)` – gdje vektor **`fi`** sadrži kutove u radijanima, a vektor **`r`** sadrži amplitude.

U prikazu polarnog grafa kut **`fi`** se iz radijana pretvara u stupnjeve.

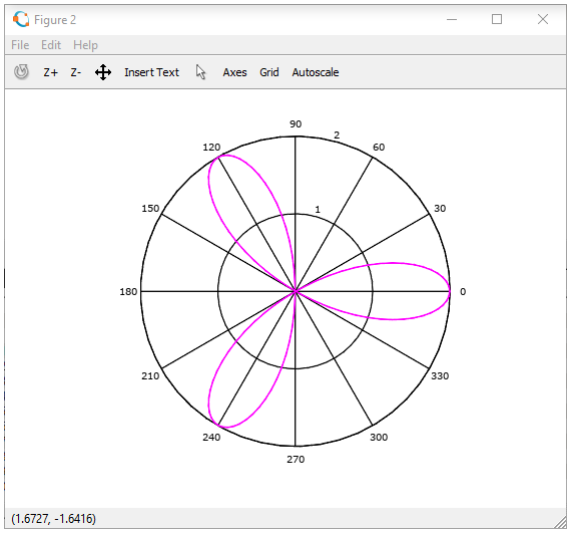
Primjer 13.20: Otvaraju se četiri grafička prozora i u svakom se prikazuje po jedan polarni graf. Za svaki polarni graf definiran je redni vektor **`fi`** dimenzija 1*100 u rasponu [0,2*pi] sa 100 elemenata (točaka). Za svaki polarni graf izračunava se vektor **`r`** koji je funkcija vektora **`fi`**. Sva su četiri grafička prozora s polarnim grafovima prikazana na slici 13.24.

```
close all
clear all
clc

format short
format compact

figure
% Definiranje vektora fi i r te prikaz polarnog grafa
fi = linspace(0,2*pi,100);
r = 0.5 + cos(fi);
polar(fi,r,'b-')

figure
% Definiranje vektora fi i r te prikaz polarnog grafa
fi = linspace(0,2*pi,100);
r = 2*cos(3*fi);
polar(fi,r,'m-')

figure
% Definiranje vektora fi i r te prikaz polarnog grafa
fi = linspace(0,2*pi,100);
r = cos(4*fi);
polar(fi,r,'b-')

figure
```

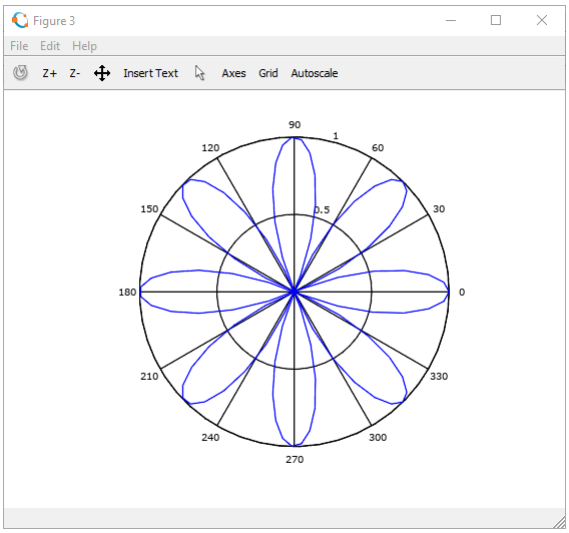
```
% Definiranje vektora fi i r te prikaz polarnog grafa
fi = linspace(0,2*pi,100);
r = sin(3*fi) + cos(4*fi);
polar(fi,r,'r-')
```



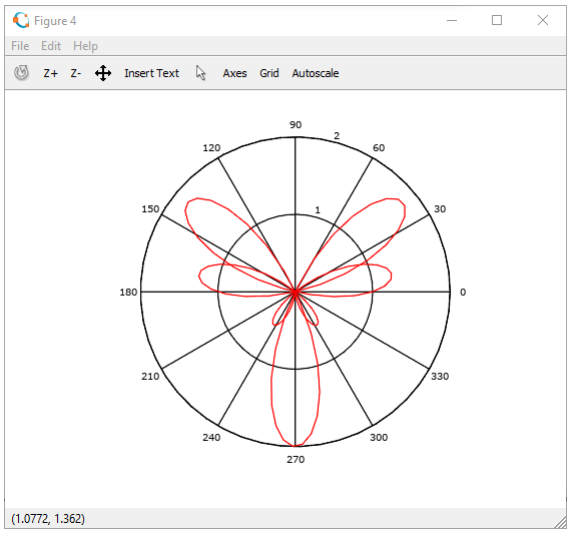
a. funkcija `r = 0.5 + cos(fi)`



b. funkcija `r = 2*cos(3*fi)`



c. funkcija `r = cos(4*fi)`



d. funkcija `r = sin(3*fi) + cos(4*fi)`

Slika 13.24 Polarni grafovi različitih funkcija

13.5 Crtanje više krivulja s različitim uspravnim osima na istom grafu

Ponekad je potrebno prikazati podatke čije se vrijednosti na ordinati jako razlikuju, odnosno imaju vrlo različit raspon vrijednosti. To znači da se podatci jedne krivulje neće vidjeti jer će raspon vrijednosti na ordinati biti preveliki ili premali u odnosu na podatke druge krivulje. U Octaveu postoji mogućnost crtanja dviju krivulja na istom grafu tako da se stvore dvije ordinate s različitim rasponom vrijednosti, i to jedna s lijeve strane i druga s desne strane. Krivulje nacrtane na takav način dobro se vide jer svaka od njih ima ordinatu s različitim rasponom vrijednosti. Za takve se grafove koristi naredba `plotyy` koja ima sljedeći oblik:

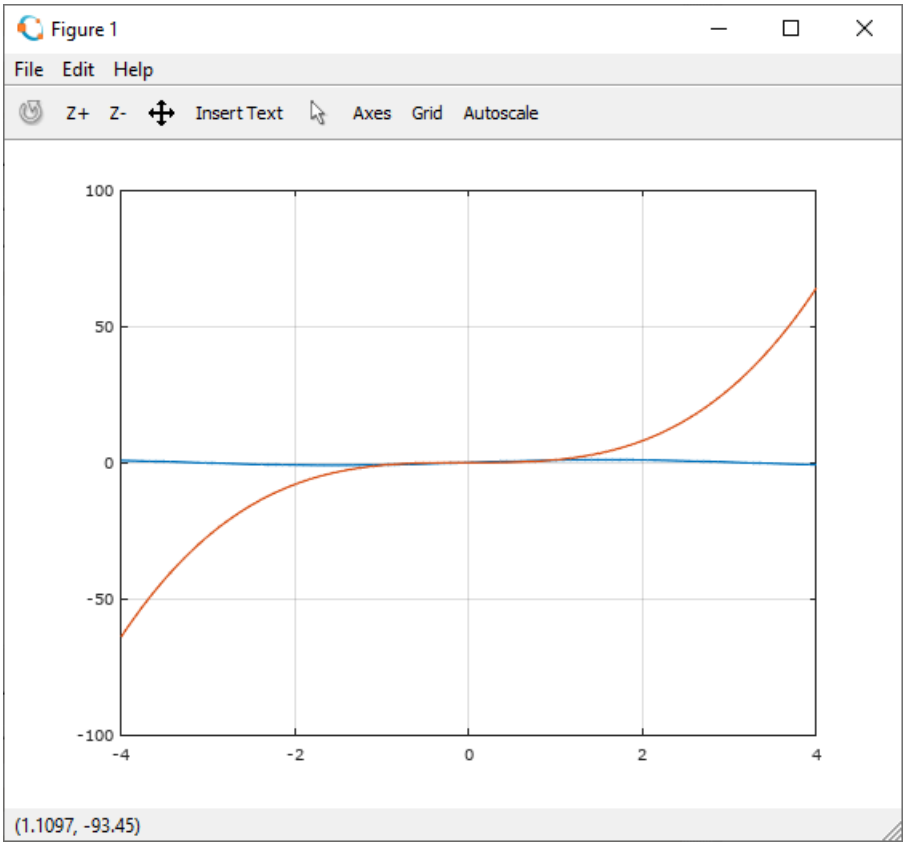
`plotyy(x1,y1,x2,y2)` – gdje su `x1` i `y1` vektori (podatci) za crtanje prve krivulje, a `x2` i `y2` vektori (podatci) za crtanje druge krivulje

`plotyy(x1,y1,x2,y2,'oblik grafa 1','oblik grafa 2')` – gdje su `x1` i `y1` vektori (podatci) za crtanje prve krivulje, a `x2` i `y2` vektori (podatci) za crtanje druge krivulje. Znakovni niz `'oblik grafa 1'`

je parametar za crtanje oblika prve krivulje, a znakovni niz **`'oblik grafa 2'`** je parametar za crtanje oblika druge krivulje.

Primjer 13.21: Pokazat će se kako bi izgledalo kada bi se pomoću naredbe **`plot`** nacrtale dvije krivulje čiji se podatci na ordinati jako razlikuju, koristeći samo jednu ordinatu. Definiran je redni vektor **`x`** dimenzija 1*100 pomoću funkcije **`linspace`** u rasponu [-4,4] koji se sastoji od 100 elementa. Zatim su definirani vektori **`y1`** i **`y2`**, gdje je **`y1`** vrijednost funkcije sinus za vrijednosti vektora **`x`**, a **`y2`** vrijednost je funkcije **`x.^3`** za vrijednosti vektora **`x`**. Sa slike 13.25 vidljivo je da se funkcija sinus jako slabo vidi jer je njezina vrijednost u rasponu [-1,1], dok je vrijednost funkcije **`x.^3`** u rasponu [-64,64]. To znači da je najveća vrijednost elemenata vektora **`y2`** 64 puta veća od vektora **`y1`**.

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x, y1 i y2
x = linspace(-4,4,100);
y1 = sin(x);
y2 = x.^3;

% Prikaz grafova funkcija y1=sin(x) i y2=x.^3
figure
plot(x,y1,x,y2)
grid on
```

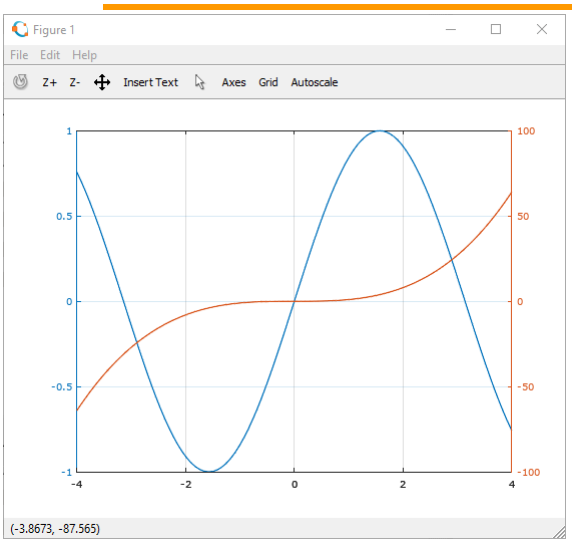


Slika 13.25 Krivulje funkcije **`sin(x)`** i **`x.^3`** na istom grafu

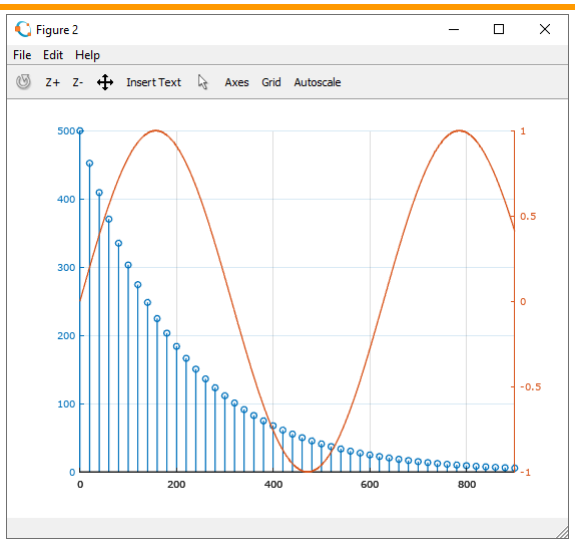
Primjer 13.22: Definirani su redni vektori **`x1`** i **`x2`** dimenzija 1*100 pomoću funkcije **`linspace`**. Vektor **`y1`** sadrži vrijednost funkcije sinus za vrijednosti vektora **`x1`**. Vektor **`y2`** sadrži vrijednosti funkcije **`x.^3`** za vrijednosti vektora **`x2`**. Otvara se prvi grafički prozor te se pomoću naredbe **`plotyy`** prikazuju te

dvije funkcije na istom grafu s različitim rasponima ordinata (slika 13.26a). Lijeva je ordinata za funkciju **sin**, a desna ordinata za funkciju **x.^3**. Vidljivo je da lijeva ordinata ima raspon [-1,1], a desna ordinata raspon [-80,80]. U drugom se grafičkom prozoru pomoću naredbe **plotyy** prikazuju dvije funkcije od kojih je jedna padajuća eksponencijalna, a druga sinus (slika 13.26b). Lijeva je ordinata za padajuću eksponencijalnu funkciju, a desna ordinata za funkciju sinus. Vidljivo je da lijeva ordinata ima raspon [0,500], a desna ordinata raspon [-1,1]. Ovdje je u naredbi **plotyy** definiran i oblik prve i druge krivulje. Za prvu se krivulju koristio graf diskretnih podataka (**stem**), a za drugu krivulju normalni graf (**plot**).

```
close all
clear all
clc

format short
format compact

% Definiranje vektora x1, y1, x2 i y2
x1 = linspace(-4,4,100);
y1 = sin(x1);
x2 = linspace(-4,4,100);
y2 = x2.^3;

% Prikaz grafova funkcija y1 i y2 s različitim ordinatama
figure
plotyy(x1,y1,x2,y2)
grid on

% Definiranje vektora t1, t2, y1 i y2 te skalara A, a i b
t1 = 0:20:900;
t2 = 0:900;
A = 500;
a = 0.005;
b = 0.01;
y1 = A * exp(-a*t1);
y2 = sin(b*t2);

% Prikaz grafova funkcija y1 i y2 s različitim ordinatama
figure
plotyy(t1,y1,t2,y2,'stem','plot')
grid on
```



a. funkcije **sin** i **x.^3**



b. funkcije **sin** i **A * exp(-a*t)**

Slika 13.26 Prikaz crtanja dviju krivulja na istom grafu s različitim rasponima ordinata

13.6 Crtanje više grafova u istom grafičkom prozoru

Osim crtanja više krivulja u jednom grafičkom prozoru te crtanja više krivulja u različitim prozorima, moguće je prikazati i više krivulja i više grafova u jednom grafičkom prozoru. To se postiže pomoću naredbe **`subplot`**.

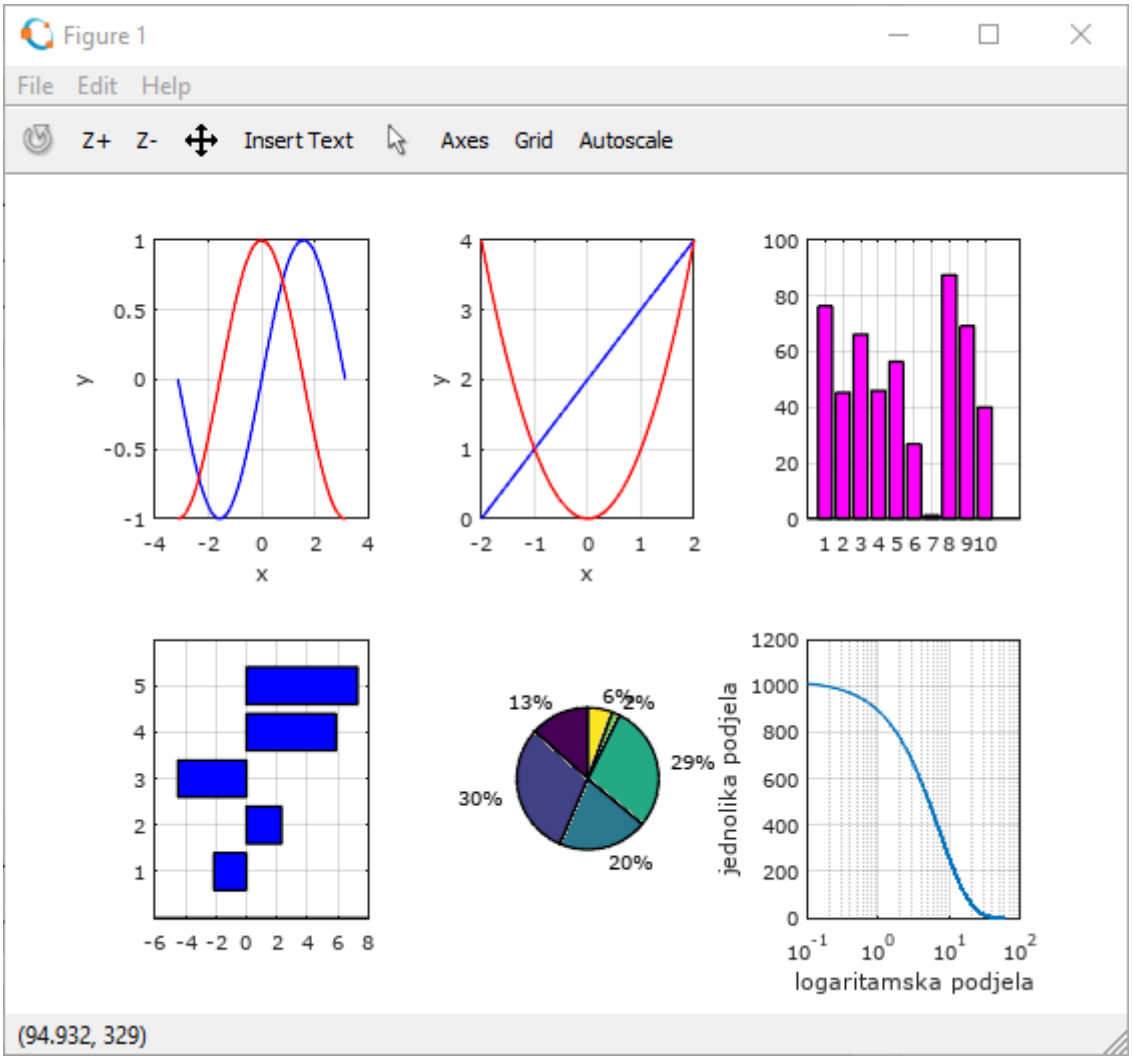
subplot

Naredba **`subplot`** ima najčešći oblik **`subplot(m,n,p)`**, gdje su **`m`**, **`n`** i **`p`** skalari. Brojevi **`m`** i **`n`** određuju podjelu grafičkog prozora na **`m`** vodoravnih i **`n`** uspravnih grafova. Broj **`p`** određuje trenutačno aktivni graf na kojem se prikazuju krivulje.

Primjer 13.23: Pomoću naredbe **`subplot(2,3,1)`** dijeli se grafički prozor na 2*3 grafa i označuje prvi graf kao aktivan. Kasnije se pomoću naredbi **`subplot(2,3,p)`**, gdje **`p`** ide od 2 do 6, aktiviraju pojedini grafovi i prikazuju određeni tipovi grafova. Svih šest grafova prikazani su u istom grafičkom prozoru (slika 13.27). Na prvom i drugom podgrafu prikazuju se krivulje pomoću naredbe **`plot`**. Na trećem podgrafu prikazuje se uspravni vrpčasti graf. Na četvrtom podgrafu prikazuje se vodoravni vrpčasti graf. Na petom podgrafu prikazuje se tortni graf. Na šestom podgrafu prikazuje se graf čija apscisa ima logaritamsku podjelu, a ordinata jednoliku podjelu.

```
close all
clear all
clc

format short
format compact

figure
% Podgraf 1
subplot(2,3,1)
% Definiranje vektora x, y1 i y2
x = linspace(-pi,pi,100);
y1 = sin(x);
y2 = cos(x);
% Prikaz grafova funkcija y1 i y2
plot(x,y1,'b','LineWidth',2)
hold on
plot(x,y2,'r','LineWidth',2)
grid on
xlabel('x')
ylabel('y')

% Podgraf 2
subplot(2,3,2)
% Definiranje vektora x, y1 i y2
x = linspace(-2,2,100);
y1 = x+2;
y2 = x.^2;
% Prikaz grafova funkcija y1 i y2
plot(x,y1,'b','LineWidth',2)
hold on
plot(x,y2,'r','LineWidth',2)
grid on
xlabel('x')
ylabel('y')

% Podgraf 3
subplot(2,3,3)
% Definiranje vekora x i y
x = 1:10;
y = 100*rand(1,10);
% Prikaz uspravnog vrpcastog grafa
bar(x,y,'m')
grid on

% Podgraf 4
subplot(2,3,4)
```

```
% Definiranje vektora x i y
x = 1:5;
y = 20*rand(1,5)-10;
% Prikaz vodoravnog vrpcastog grafa
barh(x,y,'b')
grid on

% Podgraf 5
subplot(2,3,5)
% Definiranje vektora x i prikaz tortnog grafa
x = 20*rand(1,6);
pie(x)

% Podgraf 6
subplot(2,3,6)
% Definiranje vektora x i y
x = linspace(0.1,60,1000);
y = 2.^(-0.2*x+10);
% Prikaz grafa funkcije y=f(x) s logaritamskom podjelom x-osi
semilogx(x,y,'LineWidth',2)
grid on
xlabel('logaritamska podjela')
ylabel('jednolika podjela')
```



Slika 13.27 Više grafova prikazanih u istom grafičkom prozoru

Pitanja za provjeru znanja:

1. Čemu služi naredba `plot`?
2. Ako su `x1`, `y1`, `x2` i `y2` vektori, je li ispravna sintaksa naredbe `plot(x1,y1,x2,y2)`?
3. Čemu služe naredbe `xlabel` i `ylabel`?
4. Čemu služe naredbe `title` i `legend`?
5. Koja je razlika između naredbi `semilogx`, `semilogy` i `loglog`?
6. Čemu služe naredbe `axis auto` i `axis([0 5 0 10])`?
7. Čemu služi naredba `bar(x,y)`?
8. Koja je razlika između naredbi `bar(x,y)` i `barh(x,y)`?
9. Čemu služi naredba `area(x,y)`?
10. Čemu služi naredba `stairs(x,y)`?
11. Čemu služi naredba `stem(x,y)`?
12. Čemu služi naredba `pie`?
13. Koja je razlika između naredbi `plot(x1,y1,x2,y2)` i `plotyy(x1,y1,x2,y2)`?
14. Čemu služi naredba `subplot`?

14. 3D PRIKAZ PODATAKA

Octave može prikazati funkciju dviju neovisnih varijabli u 3D prostoru.

meshgrid

Funkcija **meshgrid** stvara mrežu, koja je potrebna za crtanje funkcija dviju varijabli u 3D prostoru. Oblik funkcije je:

[xm,ym] = meshgrid(xv,yv) – gdje vektori **xv** i **yv** određuju područje u kojem se želi stvoriti mreža, a matrice **xm** i **ym** predstavljaju koordinate točaka mreže. U tim se točkama mreže izračunavaju vrijednosti funkcije dviju varijabli.

Primjer 14.1: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2] i u rasponu [1,7]. Redni vektor **xv** ima pet elemenata, a redni vektor **yv** ima sedam elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 7*5. Matrice **xm** i **ym** dimenzija 7*5 predstavljaju koordinate točaka, odnosno sjecište pravokutne mreže. Ukupan broj pravokutnih ploha koje aproksimiraju plošni prikaz funkcije u ovom je slučaju 6*4=24.

```
>> xv = linspace(-2,2,5)
xv =
    -2    -1     0     1     2
>> yv = linspace(1,7,7)
yv =
     1     2     3     4     5     6     7
>> [xm,ym] = meshgrid(xv,yv)
xm =
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
    -2    -1     0     1     2
ym =
     1     1     1     1     1
     2     2     2     2     2
     3     3     3     3     3
     4     4     4     4     4
     5     5     5     5     5
     6     6     6     6     6
     7     7     7     7     7
>>
```

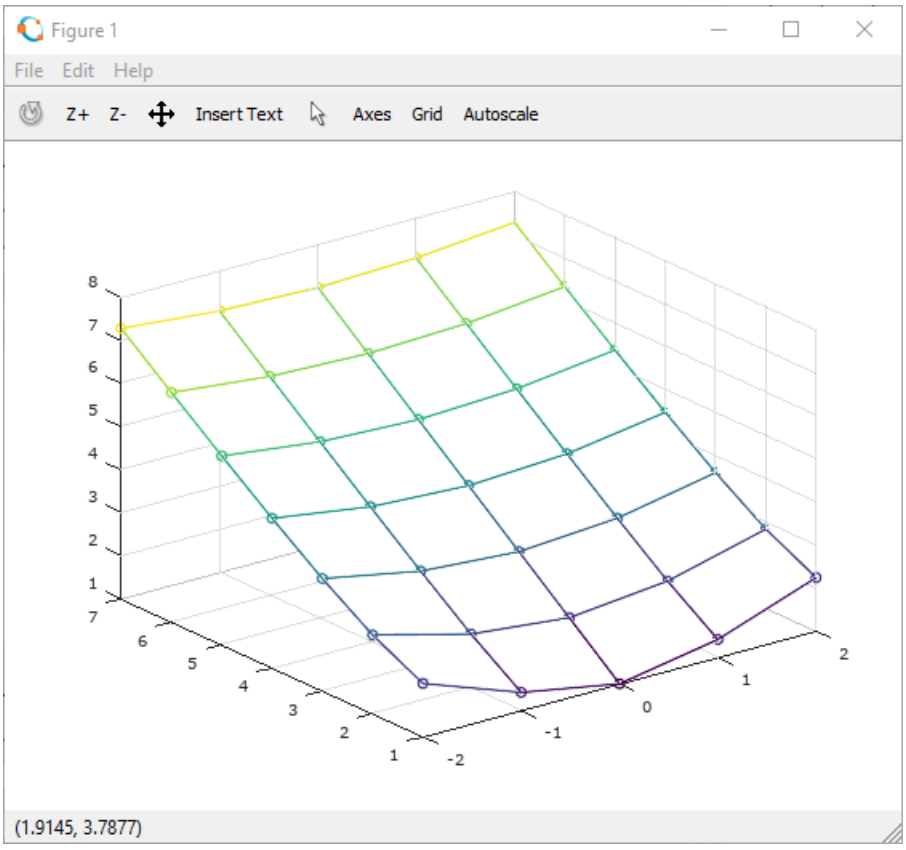
mesh

Naredba **mesh** služi za žičani prikaz (engl. *wire frame*) funkcije dviju varijabli u 3D prostoru. U žičanom prikazu definirane su točke i spojne ravne crte između tih točaka. Nisu definirane plohe pravokutnika omeđenih tim crtama. To znači da se ne mogu određivati obilježja tih ploha, npr. boja. Prikaz nalikuje na model izrađen od žičane mreže. Najčešći oblik naredbe je:

mesh(xm,ym,zm) – gdje su matrice **xm** i **ym** točke koje definiraju mrežu, a matrica **zm** sadrži vrijednosti funkcije za točke u matricama **xm** i **ym**.

Primjer 14.2: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2] i u rasponu [1,7]. Redni vektor **xv** ima 5 elemenata, a redni vektor **yv** ima 7 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 7*5. Funkcija **meshgrid** stvorila je pomoću tih matrica mrežu dimenzija 6*4. Matrice **xm** i **ym** dimenzija 7*5 predstavljaju koordinate točaka, odnosno sjecište pravokutne mreže dimenzija 6*4. Pomoću naredbe **mesh** prikazana je funkcija čije su vrijednosti izračunate za točke u matricama **xm** i **ym**, slika 14.1.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,5)
yv = linspace(1,7,7)
% Mreza
[xm,ym] = meshgrid(xv,yv)
% Funkcija z=f(x,y)
zm = sqrt(xm.^2 + ym.^2)
% Prikaz funkcije z=f(x,y)
mesh(xm,ym,zm,'Marker','o','LineWidth',2)
```

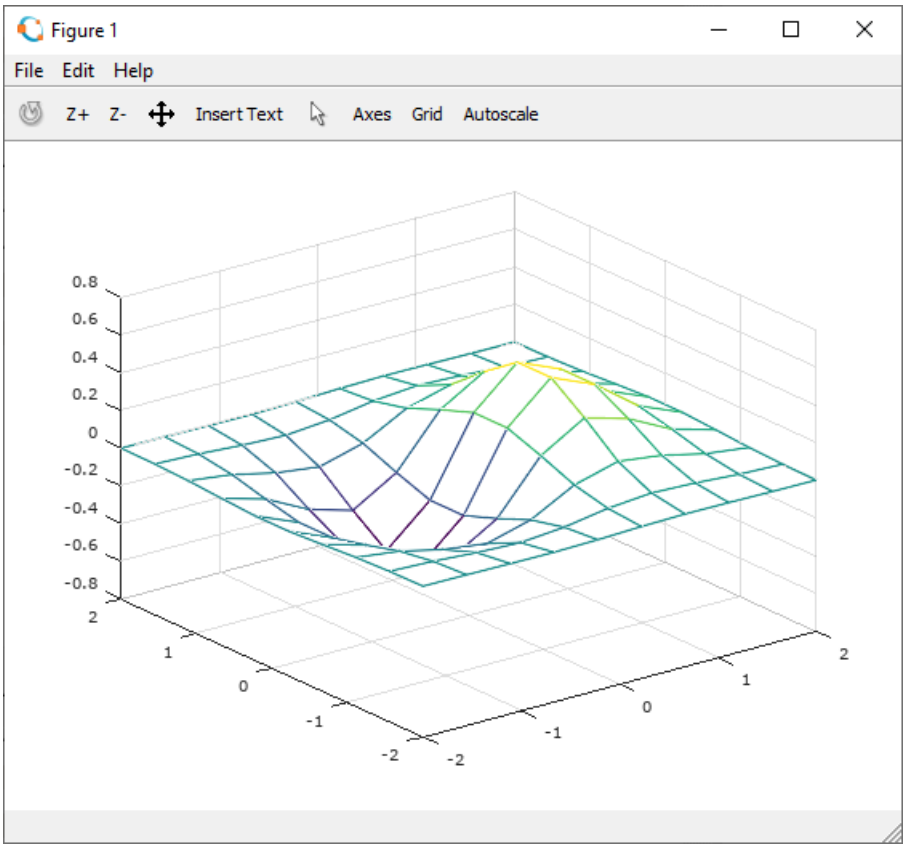


Slika 14.1 Prikaz funkcije pomoću funkcije **meshgrid** i naredbe **mesh**

Primjer 14.3: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 10 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 10*10. Funkcija **meshgrid** stvorila je pomoću tih matrica mrežu pravokutnika dimenzija 9*9. Matrice **xm** i **ym** dimenzija 10*10 su koordinate točaka. Pomoću naredbe **mesh** prikazana je

funkcija definirana pomoću točaka u matricama **xm** i **ym**, slika 14.2. Funkcija z=f(x,y) definirana je pomoću matrice **zm = xm .* exp(-xm.^2 – ym.^2)** koja sadrži vrijednosti funkcije.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,10);
yv = linspace(-2,2,10);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
mesh(xm,ym,zm)
```



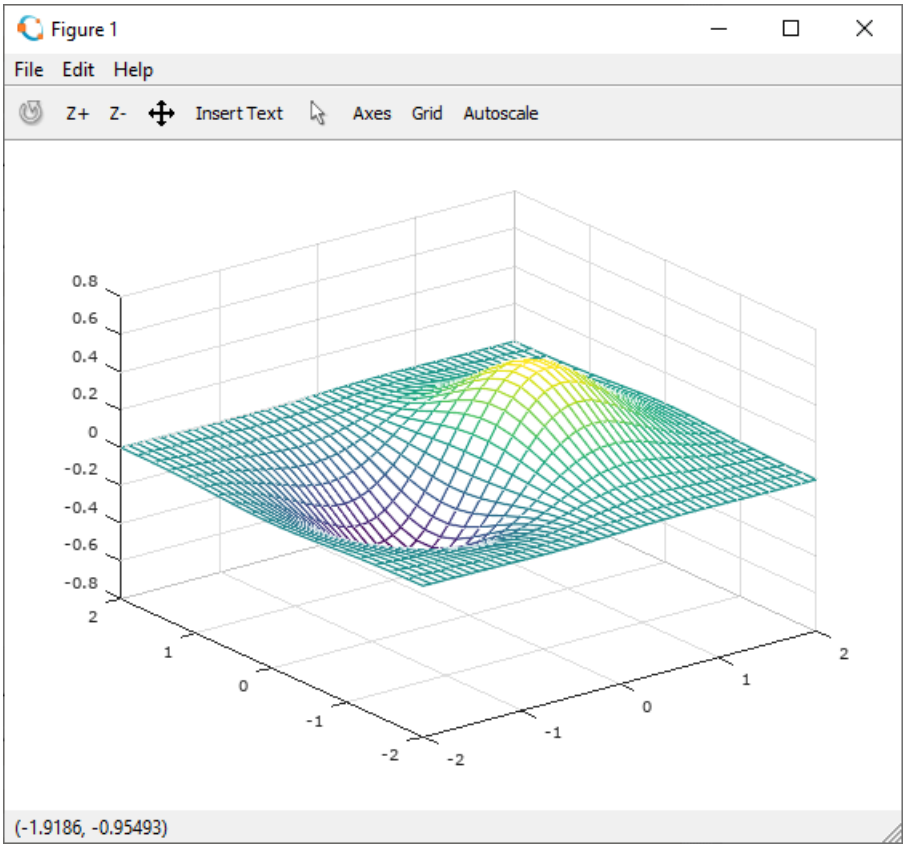
Slika 14.2 Prikaz funkcije pomoću naredbe **mesh** za mrežu gustoće 10*10 točaka

Ako se želi vjerniji prikaz funkcije dviju varijabli u prostoru, potrebno je definirati više točaka mreže. Finija mreža (mreža koja se sastoji od više točaka) stvara se pomoću vektora **xv** i **yv** koji sadrže više elemenata (u istom rasponu) i funkcije **meshgrid**. Funkcija **meshgrid** stvorit će matrice **xm** i **ym** koje se sastoje od više točaka, odnosno definiraju gušću mrežu u istom rasponu.

Primjer 14.4: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 50 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 50*50. Funkcija **meshgrid** stvorila je pomoću tih matrica mrežu dimenzija 49*49. Matrice **xm** i **ym** dimenzija 50*50 su koordinate točaka. Pomoću naredbe **mesh** prikazana je funkcija definirana pomoću točaka u matricama **xm** i **ym**, slika 14.3. Funkcija z=f(x,y) definirana je pomoću matrice **zm =**

`xm .* exp(-xm.^2 - ym.^2)` koja sadrži vrijednosti funkcije. To je isti primjer kao prethodno prikazani, samo je gustoća mreže 5 puta veća. Zbog toga je prikaz funkcije vjerniji.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,50);
yv = linspace(-2,2,20);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
mesh(xm,ym,zm)
```



Slika 14.3 Prikaz funkcije pomoću naredbe **`mesh`** za mrežu gustoće 50*50 točaka

zlabel

Naredba **`zlabel`** služi za opisivanje z-osi kao što naredba **`xlabel`** opisuje x-os, a naredba **`ylabel`** opisuje y-os koje su opisane u poglavlju 2D prikaz podataka. Naredba se koristi u obliku **`zlabel('znakovni niz')`** gdje se pod navodnicima navodi tekst za opisivanje z-osi (znakovna varijabla).

box

Naredba **`box`** služi za stvaranje okvira oko grafa funkcije u 3D prostoru. Naredba ima oblik:

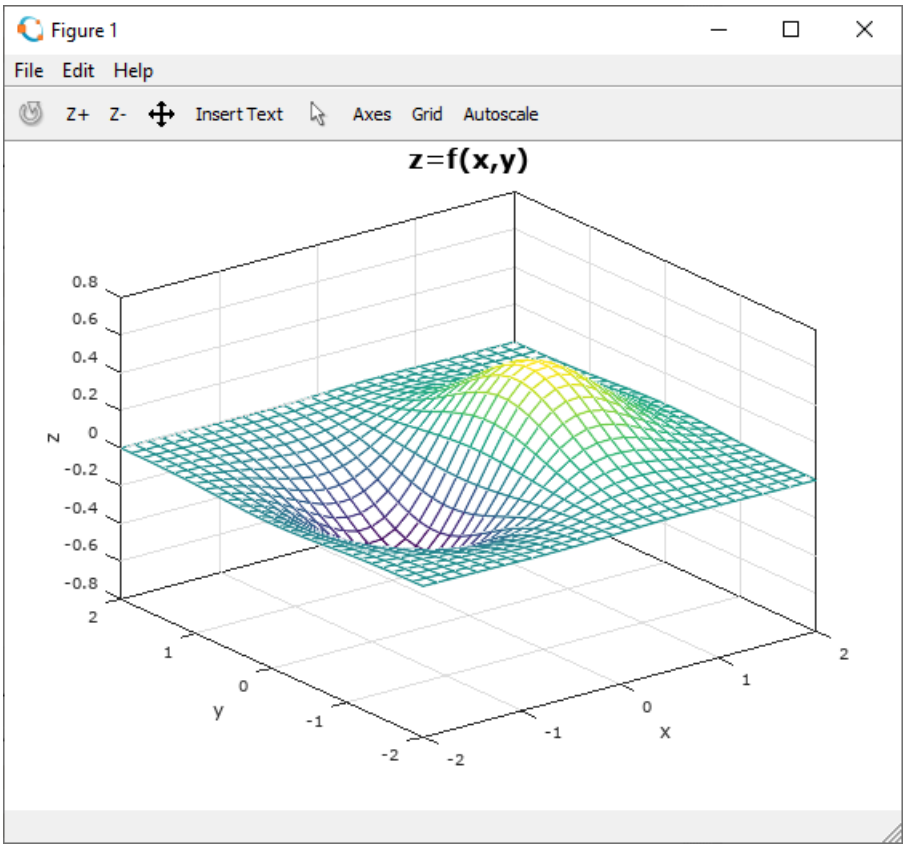
`box on` – uključuje okvir oko grafa

`box off` – isključuje okvir oko grafa

box – mijenja stanje iz jednog u drugi.

Primjer 14.5: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 30 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 30*30. Matrice **xm** i **ym** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **mesh** prikazana je funkcija definirana pomoću točaka u matricama **xm** i **ym**, slika 14.4. Funkcija z=f(x,y) definirana je pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije. Ovdje su korištene naredbe **xlabel**, **ylabel** i **zlabel** za opis koordinatnih osi. Naredba **title** korištena je za postavljanje naslova grafa. Parametri koji se mogu koristiti kod navedenih naredbi opisani su u poglavlju 2D prikaz podataka. Naredba **box on** stvara okvir oko grafa funkcije.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
mesh(xm,ym,zm)
xlabel('x','FontSize',12)
ylabel('y','FontSize',12)
zlabel('z','FontSize',12)
title('z=f(x,y)','FontSize',16)
box on
```



Slika 14.4 Prikaz funkcije pomoću naredbe **mesh**, s označenom x, y i z-osi te natpisom iznad grafa

U naredbi **mesh** moguće je rabiti parametre koji definiraju izgled žičanog prikaza grafa funkcije dviju varijabli. Parametara ima mnogo te se preporučuje pogledati Octaveov sustav pomoći za opis svih parametara. Ovdje će biti prikazani samo neki parametri. U tablicama 14.1 i 14.2 opisane su neke značajke žičanog prikaza grafa funkcije dviju varijabli iscrtanih pomoću naredbe **mesh** koje se često koriste.

Tablica 14.1 Neki parametri koji se koriste kod naredbe **mesh**

Svojstvo	Opis	Moguće vrijednosti svojstva
LineWidth	Debljina crte	Broj
LineStyle	Vrsta crte	Prikazane u tablici 14.2
Marker	Vrsta markera	Prikazane u tablici 14.2
MarkerSize	Veličina markera	Broj
MarkerEdgeColor	Boja obruba markera	Oznaka boje
MarkerFaceColor	Boja ispune markera	Oznaka boje

Oznaka vrste crte i vrste markera prikazane su u tablici 14.2.

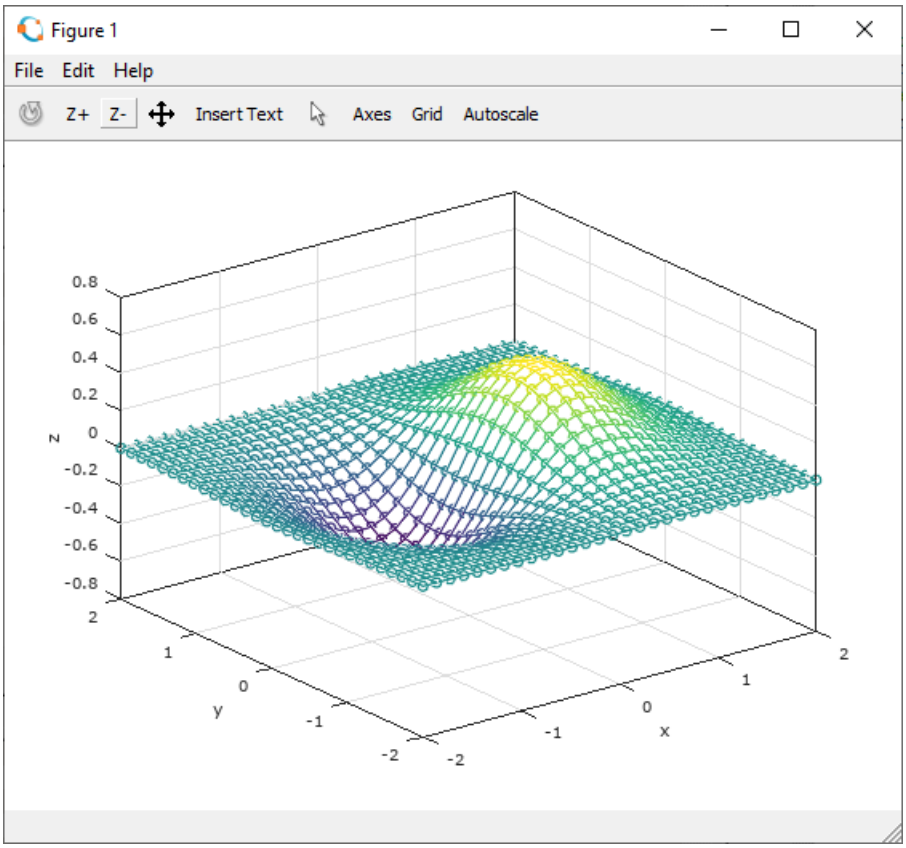
Tablica 14.2 Vrste crta i markera s pripadajućim oznakama

Vrsta crte	Oznaka	Vrsta markera	Oznaka
Puna	**-**	Točka	**.**
Točkasta	**:**	Krug	**o**
Crta-točka	**-.**	X-znak	**x**
Isprekidana	**--**	Plus	**+**
		Zvijezda	*****
		Kvadrat	**s**
		Dijamant	**d**
		Trokut (prema dolje)	**v**
		Trokut (prema gore)	**^**
		Trokut (nalijevo)	**<**
		Trokut (nadesno)	**>**
		Zvijezda s pet krakova	**p**
		Zvijezda sa šest krakova	**h**

Primjer 14.6: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 30 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 30*30. Matrice **xm** i **ym** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **mesh** prikazana je funkcija definirana pomoću točaka u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije, slika 14.5. U naredbi **mesh** mogu se koristiti parametri koji su opisani uz naredbu **plot** u poglavlju 2D prikaz podataka. U ovom se slučaju koristio parametar **'LineWidth'** koji određuje debljinu crta mreže prikazanog grafa, te parametar **'Marker'** koji određuje oznake (markere) na koordinatama točaka (vrhovima mreže) određenih pomoću matrica **xm** i **ym**.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
```

```
mesh(xm,ym,zm,'LineWidth',2,'Marker','o')
xlabel('x')
ylabel('y')
zlabel('z')
box on
```

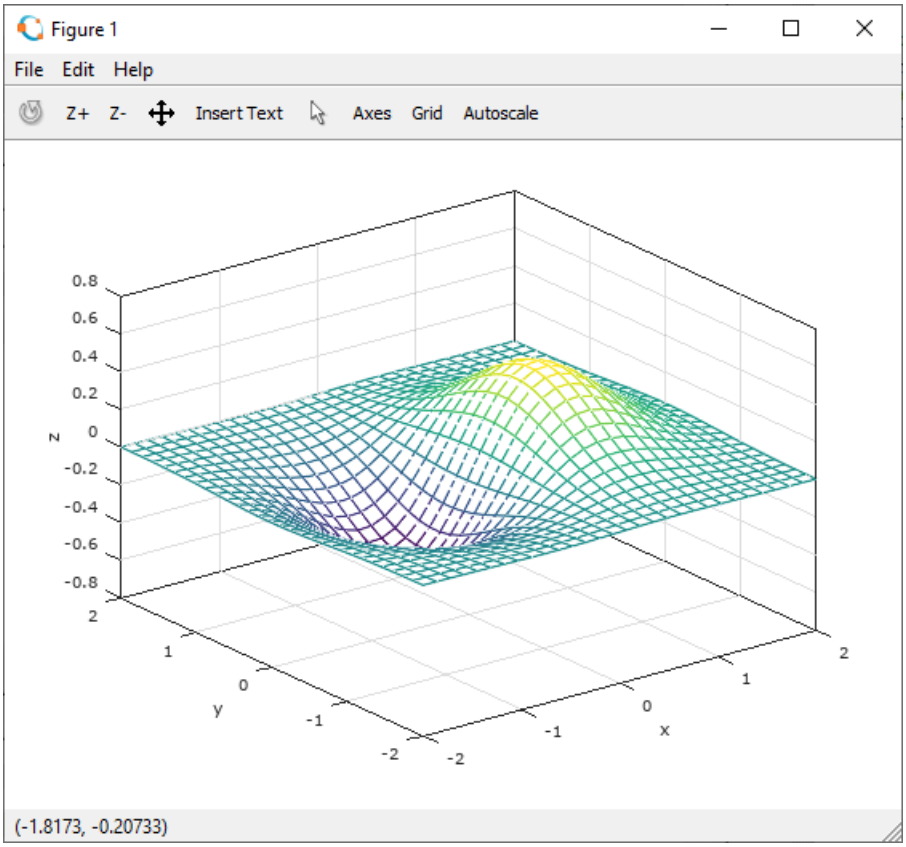


Slika 14.5 Prikaz funkcije s parametrima koji određuju debljinu crte i vrstu markera kod žičanog prikaza funkcije dviju varijabli

Primjer 14.7: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 30 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 30*30. Matrice **xm** i **ym** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **mesh** prikazana je funkcija definirana pomoću točaka u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije, slika 14.6. Uz naredbu **mesh** mogu se koristiti parametri koji su opisani uz naredbu **plot** u poglavlju 2D prikaz podataka. U ovom je slučaju korišten parametar **'LineWidth'** koji određuje debljinu crta mreže prikazanog grafa, te parametar **'LineStyle'** koji određuje vrstu crta.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
```

```
mesh(xm,ym,zm,'LineWidth',2,'LineStyle','-.')
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.6 Prikaz funkcije s parametrima koji određuju debljinu i vrstu crte kod žičanog prikaza funkcije dviju varijabli

hidden

Pomoću naredbe **`hidden`** moguće je prikazati ili ne prikazati dio žičanog prikaza grafa funkcije dviju varijabli. Oblik naredbe je:

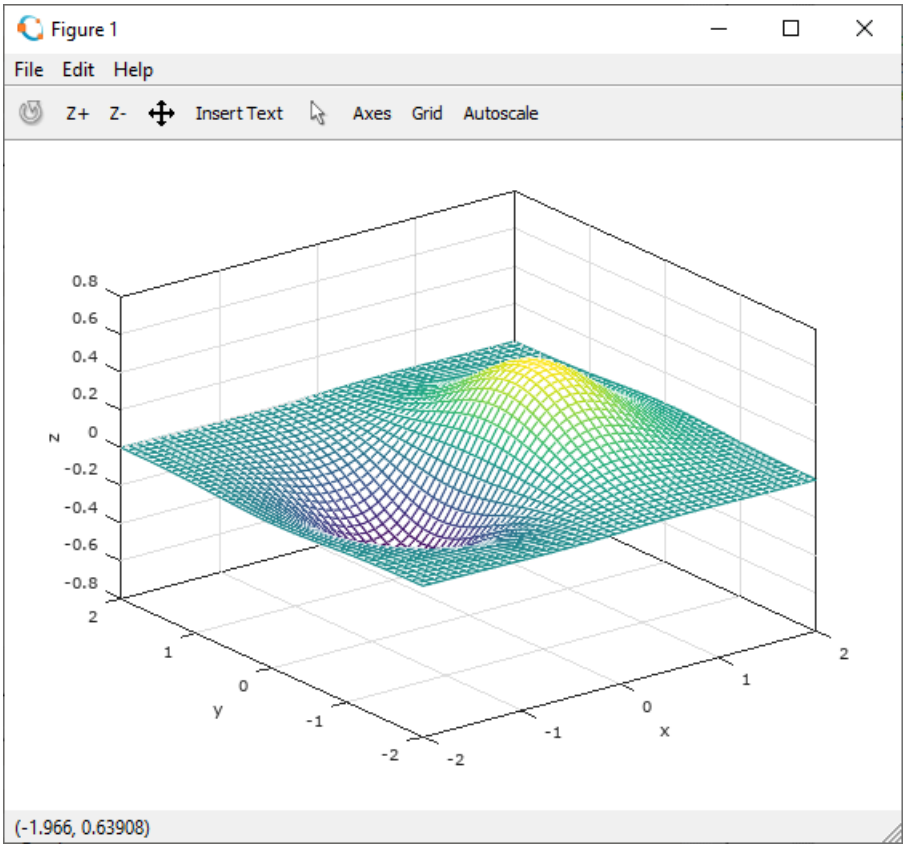
`hidden on` – ne prikazuje se dio žičanog prikaza grafa funkcije dviju varijabli koji se ne vidi iz zadanog kuta gledanja.

`hidden off` – prikazuje se dio žičanog prikaza grafa funkcije dviju varijabli koji se ne vidi iz zadanog kuta gledanja (prikazuje se cijela mreža žičanog prikaza grafa funkcije dviju varijabli). Kod ove vrste prikaza vidi se kroz žičanu mrežu.

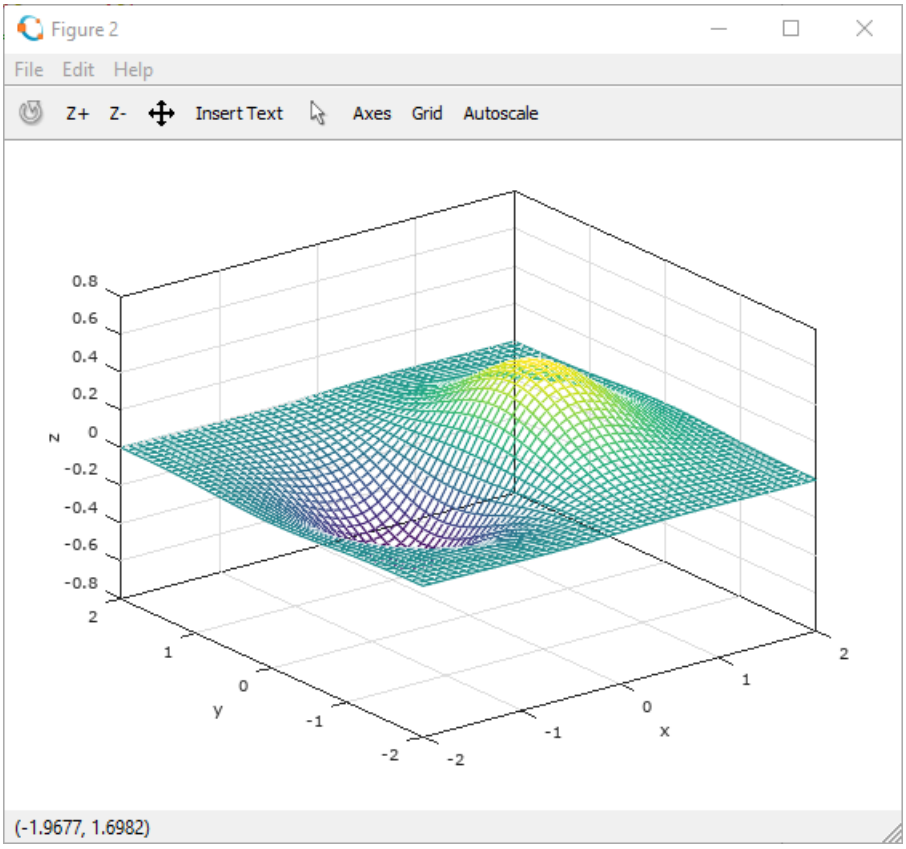
`hidden` – mijenja stanje iz jednog u drugi.

Primjer 14.8: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 50 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 50*50. Funkcija **`meshgrid`** stvorila je pomoću tih matrica mrežu dimenzija 49*49. Matrice **`xm`** i **`ym`** dimenzija 50*50 koordinate su točaka, odnosno vrhovi pravokutne mreže dimenzija 49*49. Pomoću naredbe **`mesh`** prikazana je funkcija definirana pomoću točaka u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije. Program prikazuje dva grafička prozora u kojima je u svakom iscrtana ista funkcija. U prvom grafičkom prozoru, koji je prikazan na slici 14.7, koristi se naredba **`hidden on`** koja skriva dijelove funkcije koji se ne vide iz promatranog kuta gledanja. U drugom grafičkom prozoru, koji je prikazan na slici 14.8, koristi se naredba **`hidden off`** koja prikazuje sve dijelove funkcije koji se inače ne vide pod tim kutom gledanja.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,50);
yv = linspace(-2,2,50);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
figure
mesh(xm,ym,zm)
hidden on
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
mesh(xm,ym,zm)
hidden off
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.7 Prikaz funkcije gdje nisu prikazani dijelovi funkcije koji se ne vide pod promatranim kutom gledanja



Slika 14.8 Prikaz funkcije gdje su vidljivi dijelovi funkcije koji se inače ne vide pod promatranim kutom gledanja ("prozirna" žičana mreža)

surf

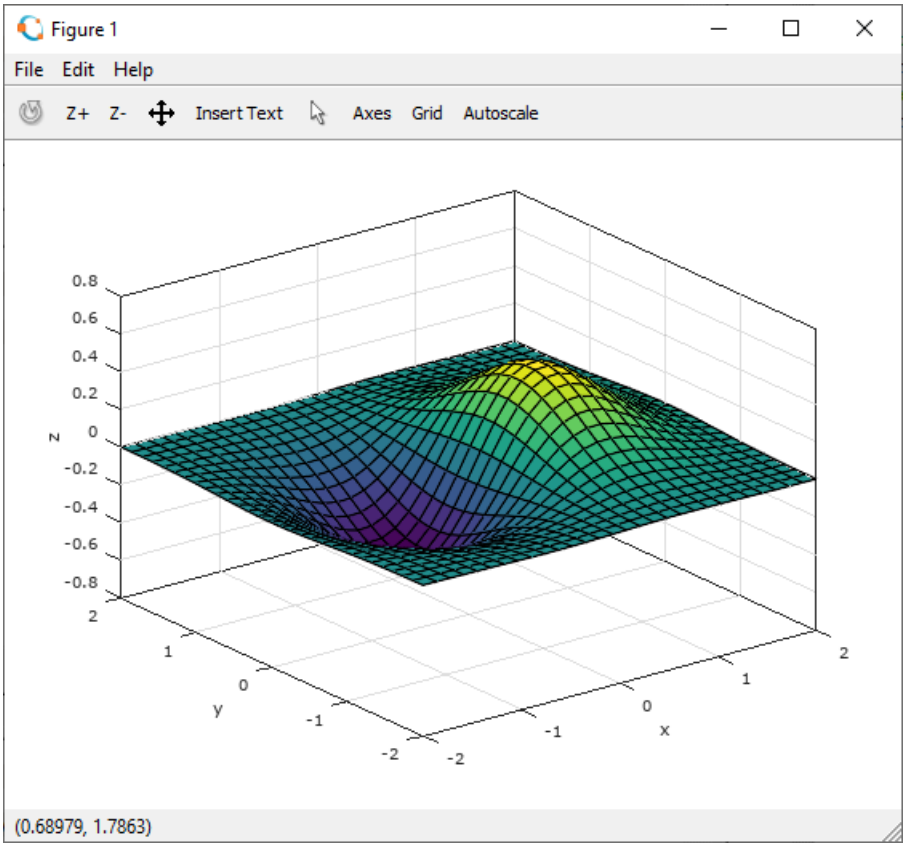
Naredba **surf** služi za plošni prikaz funkcije dviju varijabli u 3D prostoru. Kod plošnog prikaza definirane su i plohe omeđene spojnim crtama točaka. Te je pravokutne plohe moguće, npr. bojiti. Najčešći oblik naredbe je:

surf(xm,ym,zm) – gdje su matrice **xm** i **ym** točke koje definiraju mrežu, a matrica **zm** predstavlja vrijednosti funkcije za točke u matricama **xm** i **ym**.

Primjer 14.9: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 30 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 30*30. Matrice **xm** i **ym** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **surf** prikazana je funkcija definirana pomoću točaka u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije, slika 14.9. Pomoću naredbe **surf** svaki kvadrat ili pravokutnik mreže koji predstavlja oblik funkcije ispunjen je bojom (iz palete boja) koja u ovom slučaju odgovara vrijednosti funkcije.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
```

```
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.9 Prikaz funkcije pomoću naredbe **`surf`** (plošni prikaz funkcije)

colormap

Naredba **`colormap`** određuje paletu boja za aktivni grafički prozor. Grafovi u grafičkom prozoru prikazat će se u odabranoj paleti boja. Oblik naredbe je:

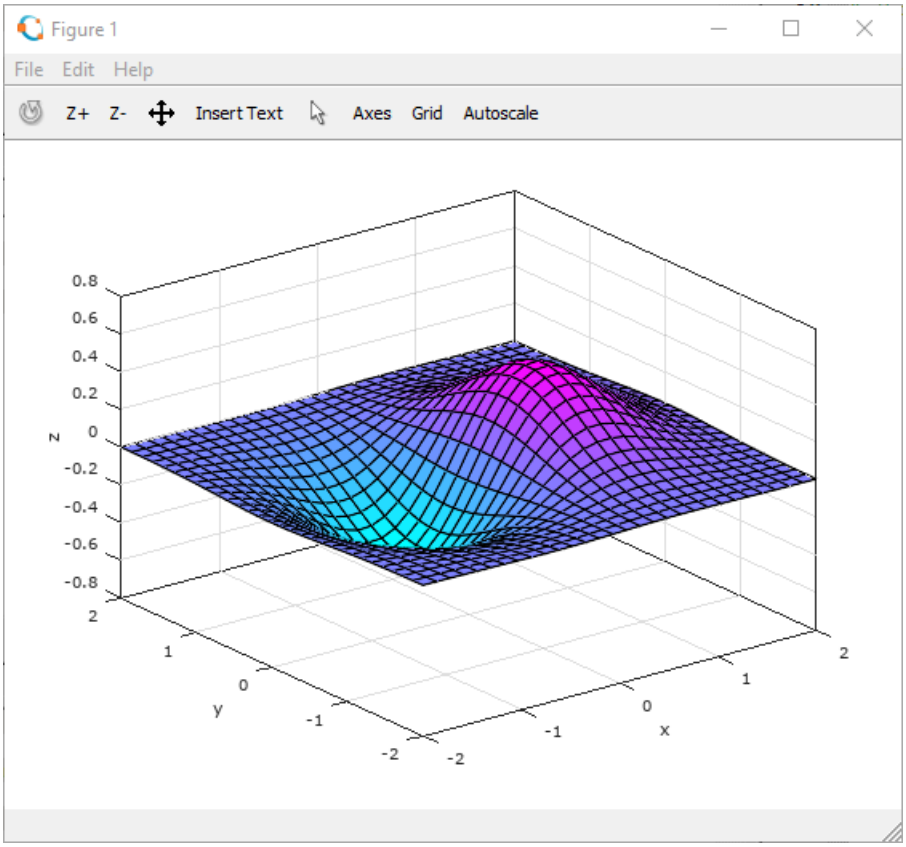
`colormap(map)` – gdje je **`map`** matrica koja predstavlja paletu boja.

Paleta boja **`map`** matrica je koja može imati proizvoljan broj redaka, ali mora imati točno 3 stupca. Prvi stupac predstavlja udio crvene boje, drugi stupac udio zelene boje i treći stupac udio plave boje u boji. Svaki redak matrice **`map`** predstavlja jednu boju. U Octaveu postoji nekoliko definiranih paleta boja: **`jet`**, **`hsv`**, **`hot`**, **`cool`**, **`spring`**, **`summer`**, **`autumn`**, **`winter`**, **`gray`**, **`bone`**, **`copper`**, **`pink`** i **`lines`**.

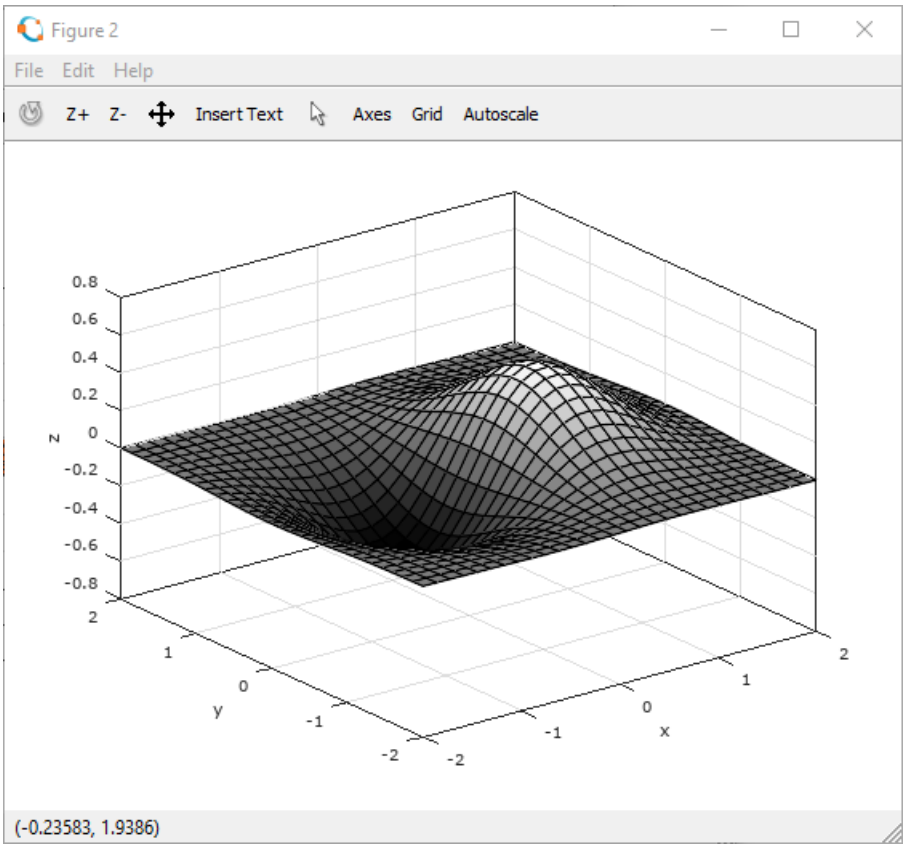
Primjer 14.10: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`surf`** prikazana je funkcija definirana pomoću točaka u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije. U ovom se primjeru otvara 5 grafičkih prozora i u svakom prikazuje ista funkcija. Pomoću naredbe **`colormap`** u svakom se grafičkom prozoru prikazuje funkcija u drugoj paleti boja. Ovdje su korištene palete boja **`cool`** (slika 14.10), **`gray`** (slika 14.11), **`hot`** (slika 14.12), **`hsv`** (slika 14.13) i **`jet`** (slika 14.14).

```
close all
```

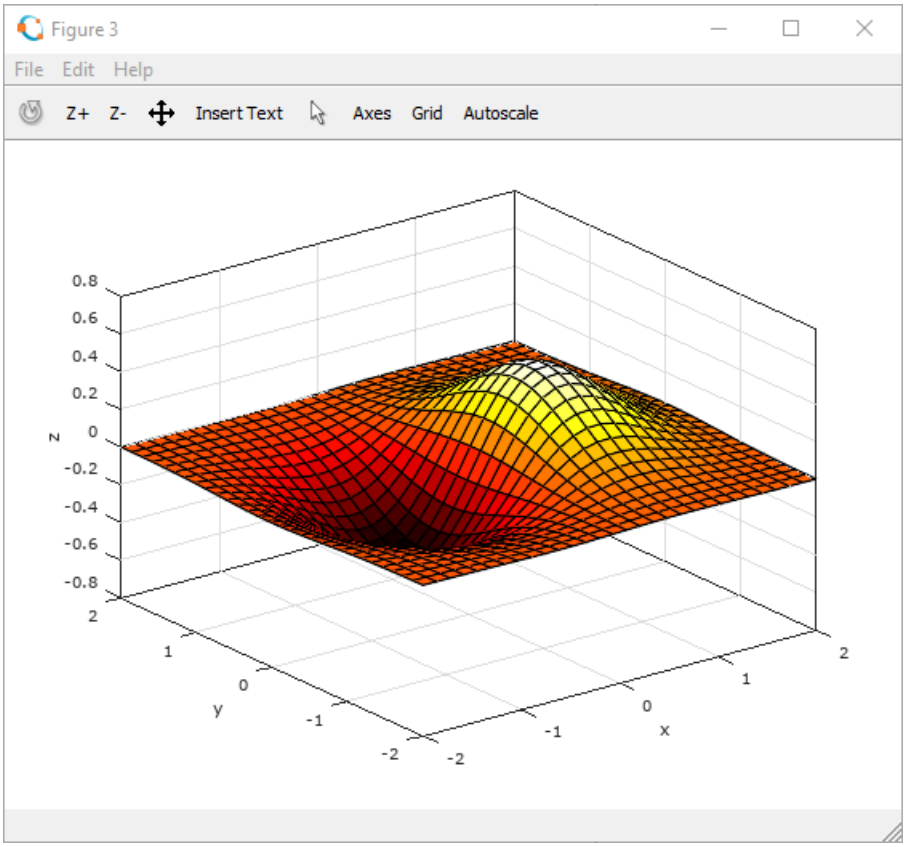
```
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
colormap(cool)
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
colormap(gray)
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
colormap(hot)
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
colormap(hsv)
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
colormap(jet)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



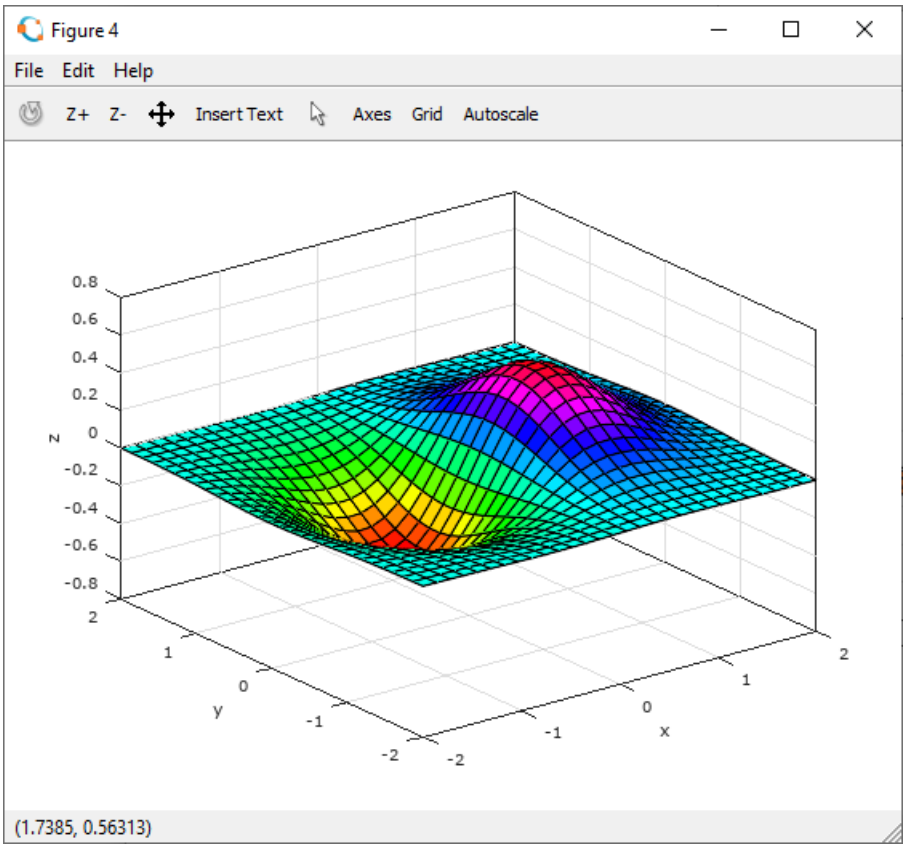
Slika 14.10 Prikaz funkcije pomoću naredbe **`surf`** i palete boja **`cool`**



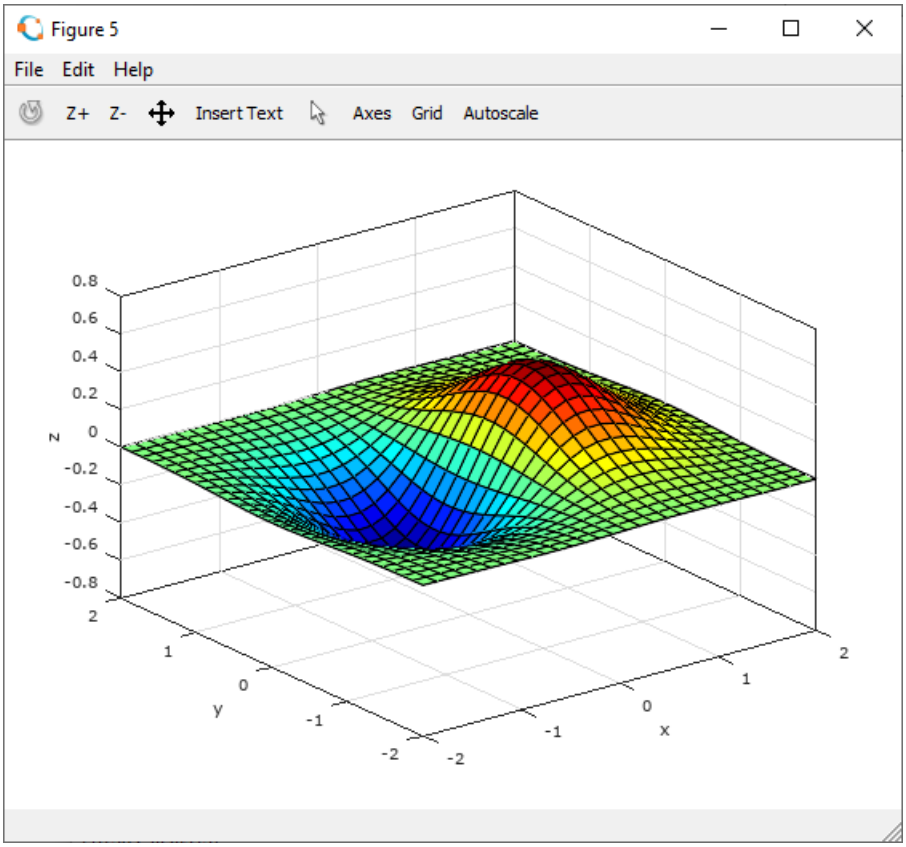
Slika 14.11 Prikaz funkcije pomoću naredbe **`surf`** i palete boja **`gray`**



Slika 14.12 Prikaz funkcije pomoću naredbe **`surf`** i palete boja **`hot`**



Slika 14.13 Prikaz funkcije pomoću naredbe **`surf`** i palete boja **`hsv`**



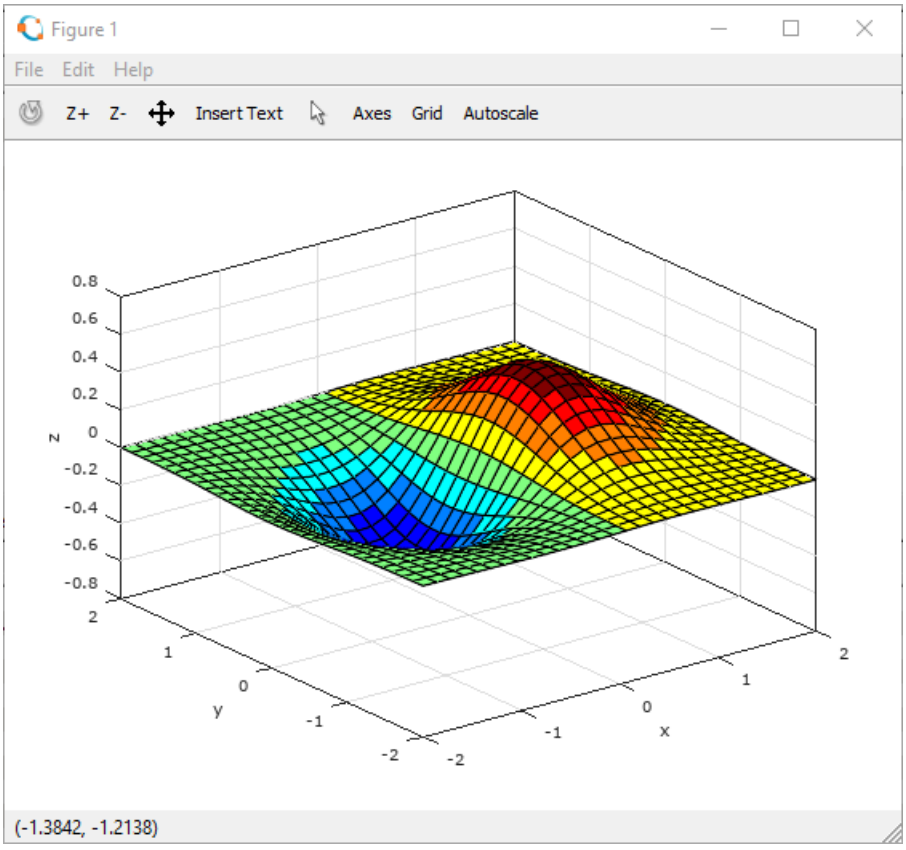
Slika 14.14 Prikaz funkcije pomoću naredbe **`surf`** i palete boja **`jet`**

Kod naredbe **`colormap(map)`** moguće je, uz postojeće palete boja definirane u Octaveu, odrediti i broj boja u paleti boja. Oblik naredbe je **`colormap(map(n))`**, gdje je **`n`** (skalar) željeni broj boja u paleti boja **`map`**. Tada će se ta paleta boja sastojati od **`n`** boja.

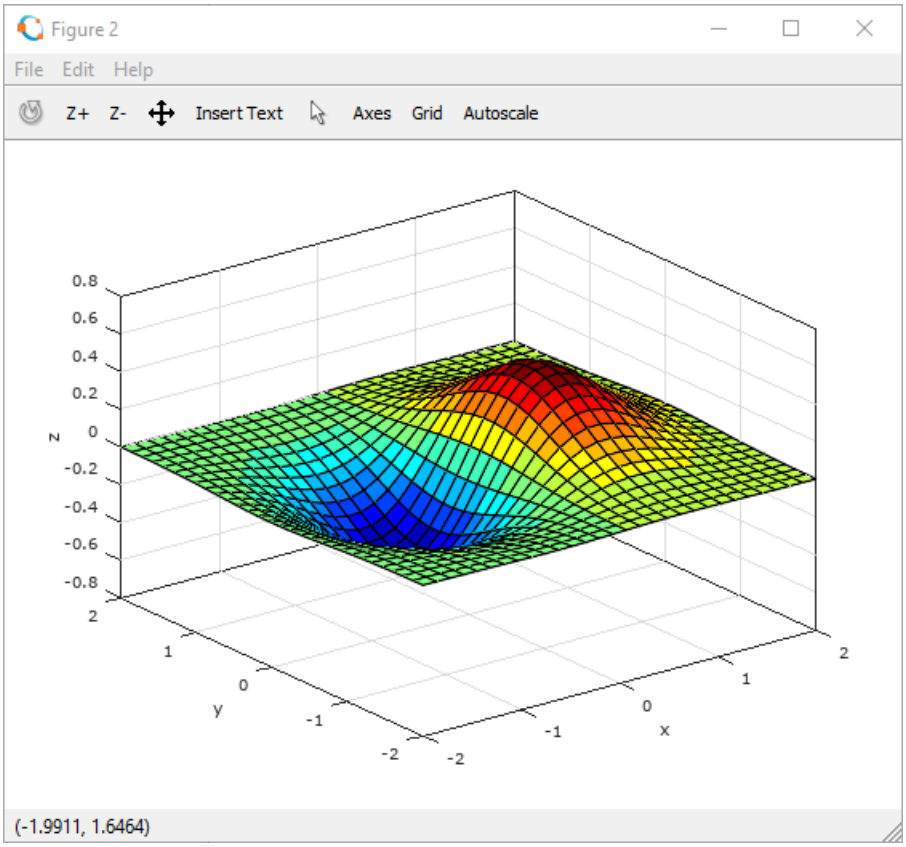
Primjer 14.11: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`surf`** prikazana je funkcija definirana pomoću točaka u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije. U ovom se primjeru otvaraju 3 grafička prozora i u svakom prikazuje ista funkcija. Pomoću naredbe **`colormap`** u svakom se grafičkom prozoru prikazuje funkcija u istoj paleti boja (**`jet`**), ali s različitim brojem boja u paleti boja. U prvom grafičkom prozoru paleta boja sadrži 8 boja (slika 14.15), u drugom grafičkom prozoru paleta boja sadrži 16 boja (slika 14.16), a u trećem grafičkom prozoru paleta boja sadrži 256 boja (slika 14.17).

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
```

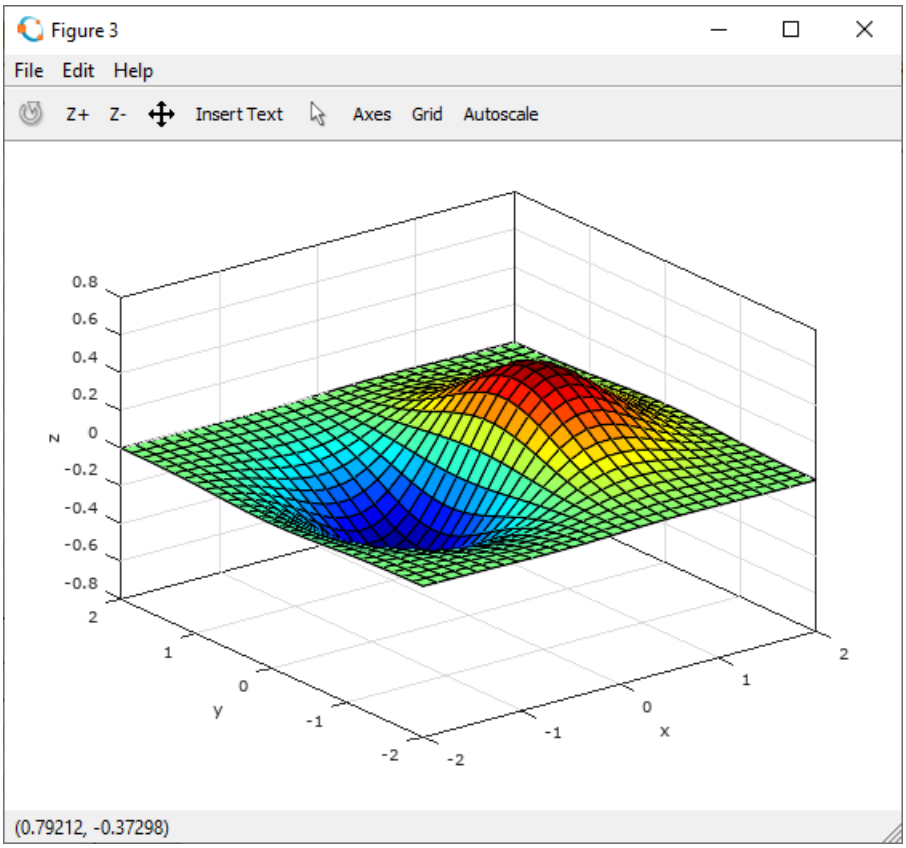
```
colormap(jet(8))
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
colormap(jet(16))
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
colormap(jet(256))
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.15 Prikaz funkcije pomoću naredbe **`surf`** i paleta boja **`jet`** sa 8 boja



Slika 14.16 Prikaz funkcije pomoću naredbe **`surf`** i paleta boja **`jet`** sa 16 boja



Slika 14.17 Prikaz funkcije pomoću naredbe **`surf`** i paleta boja **`jet`** sa 256 boja

shading

Kod plošnog prikaza pomoću naredbe **surf** moguće je odrediti način sjenčanja plohe koja predstavlja funkciju dviju varijabli, pomoću naredbe **shading**. Oblik naredbe je:

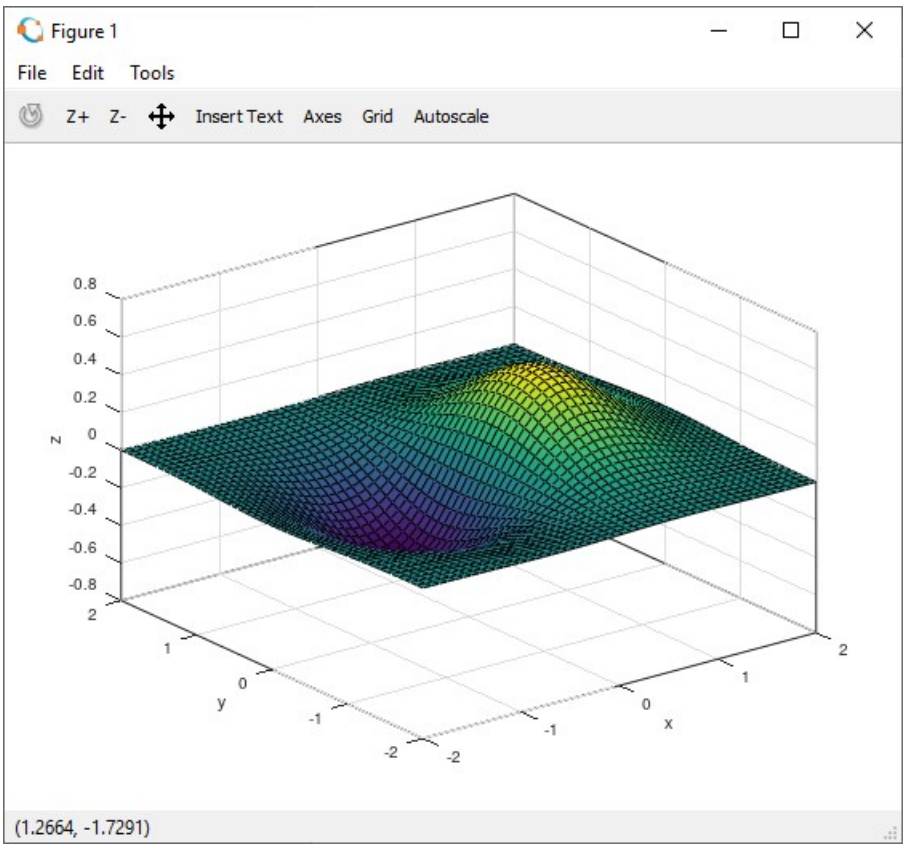
shading faceted – sjenčanje s mrežom

shading flat – sjenčanje bez mreže i

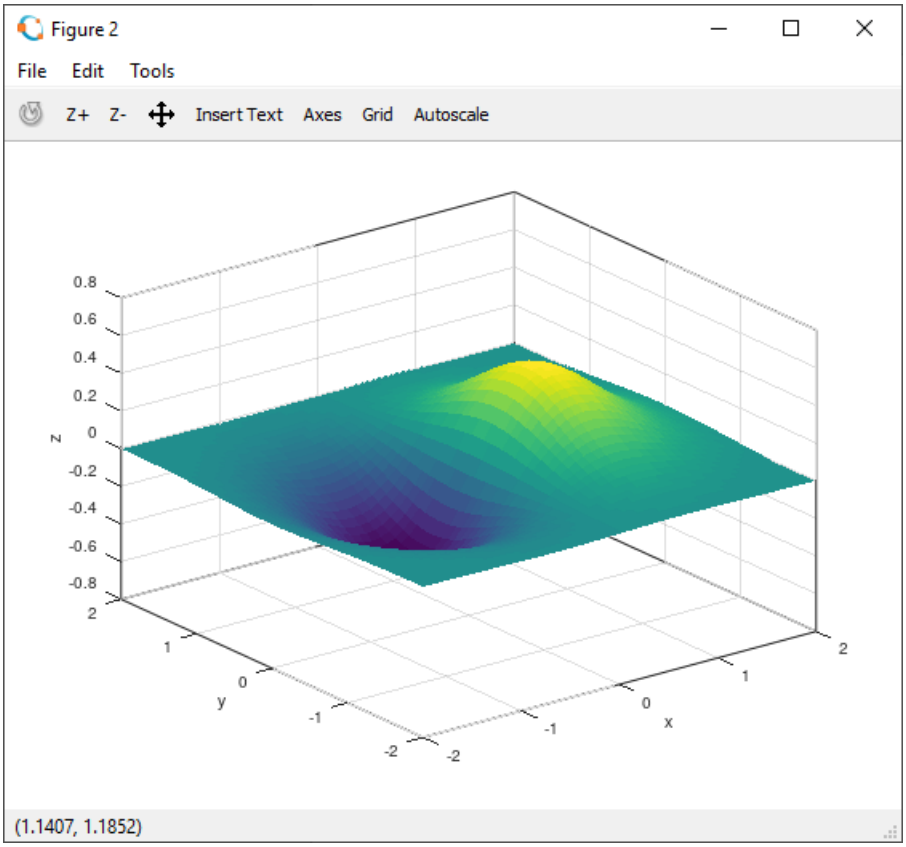
shading interp – sjenčanje bez mreže s interpoliranim prijelazima između boja na plohi.

Primjer 14.12: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 50 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 50*50. Matrice **xm** i **ym** dimenzija 50*50 koordinate su točaka. Pomoću naredbe **surf** prikazana je funkcija definirana pomoću točaka u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije. U ovom se primjeru otvaraju 3 grafička prozora i u svakom prikazuje ista funkcija. U prvom se grafičkom prozoru koristi naredba **shading faceted** koja prikazuje funkciju s iscrtanom mrežom, slika 14.18. U drugom se grafičkom prozoru koristi naredba **shading flat** koja prikazuje funkciju bez iscrtane mreže, slika 14.19. U trećem se grafičkom prozoru koristi naredba **shading interp** koja prikazuje funkciju bez mreže s interpoliranim prijelazima između boja na plohi, slika 14.20.

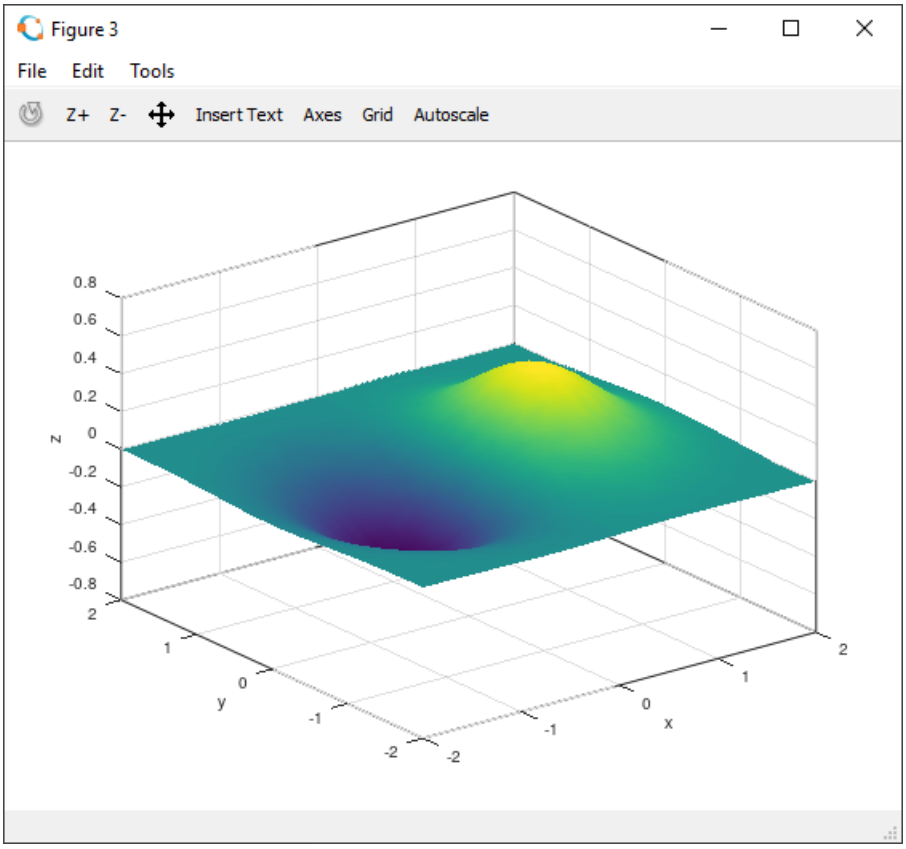
```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,50);
yv = linspace(-2,2,50);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
shading faceted
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
shading flat
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
surf(xm,ym,zm)
shading interp
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.18 Prikaz funkcije pomoću naredbi **`surf`** i **`shading faceted`**



Slika 14.19 Prikaz funkcije pomoću naredbi **`surf`** i **`shading flat`**



Slika 14.20 Prikaz funkcije pomoću naredbi **`surf`** i **`shading interp`**

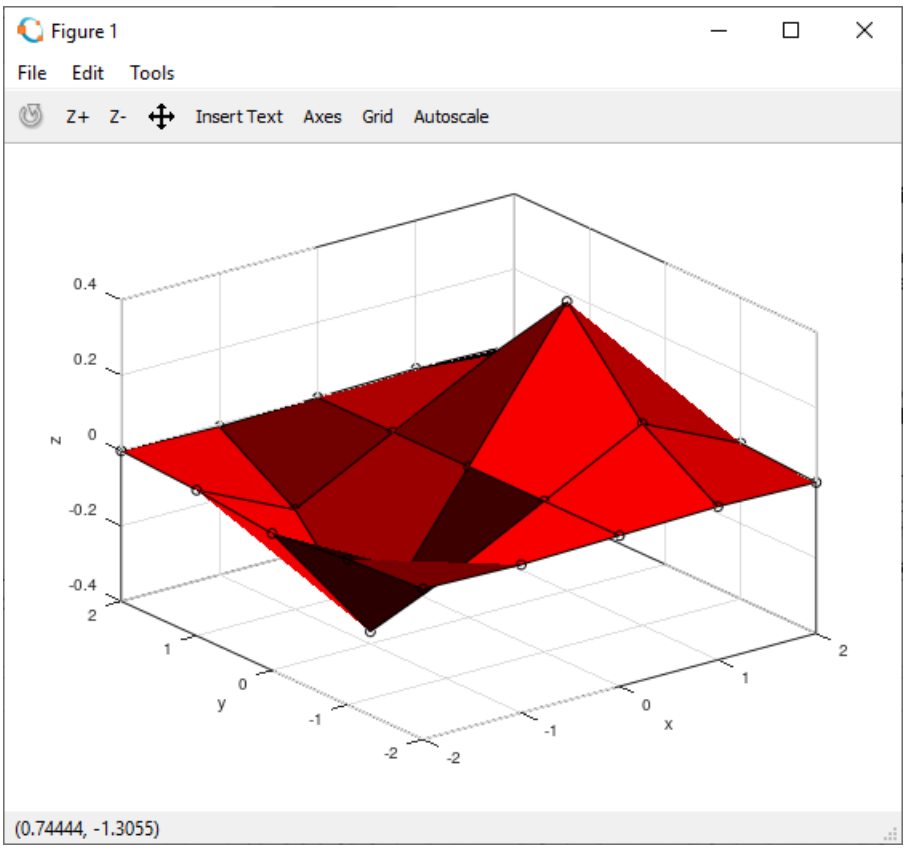
Naredba **`surf`** može imati oblik kod kojeg su definirane boje svakog elementa mreže u plošnom prikazu grafa funkcije dviju varijabli. Oblik naredbe je:

`surf(xm,ym,zm,c)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, matrica **`zm`** predstavlja vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**, a **`c`** je polje dimenzija m*n*3. Polje **`c`** ima dimenzije m*n*3, gdje je n jednak broju elemenata vektora **`xv`** ili **`xv-1`**, m jednak broju elemenata vektora **`yv`** ili **`yv-1`**, a 3 je zbog toga što postoje tri komponente boje (crvena, zelena i plava).

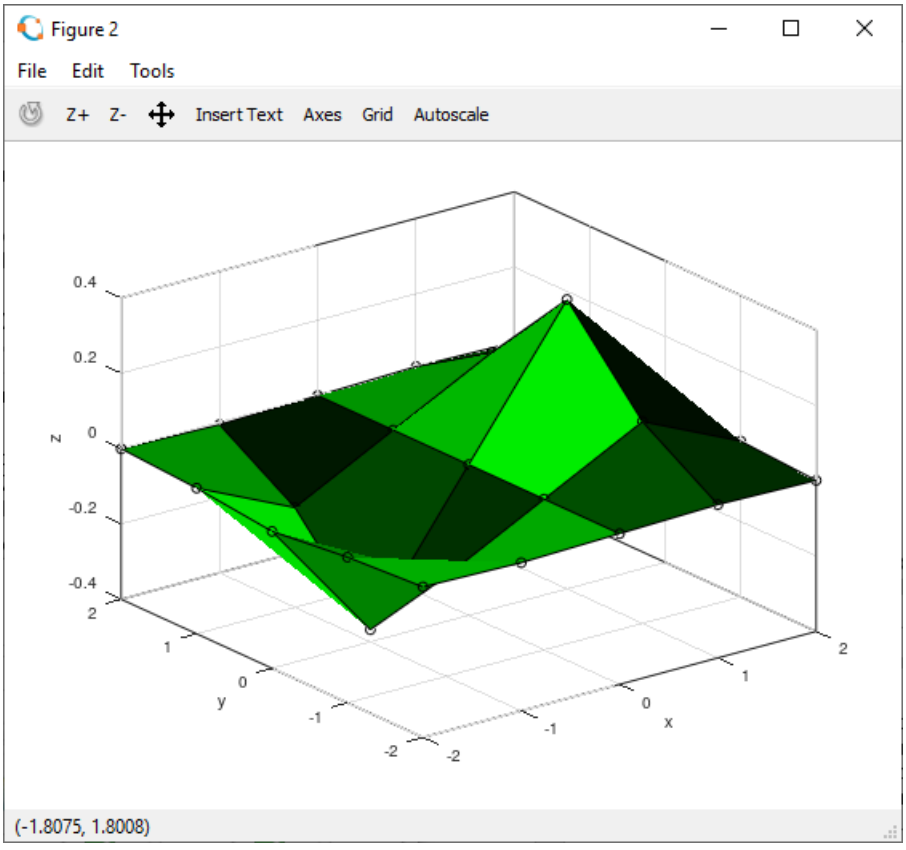
Primjer 14.13: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 5 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 5*5. Matrice **`xm`** i **`ym`** dimenzija 5*5 koordinate su točaka. Pomoću naredbe **`surf`** prikazana je funkcija definirana pomoću točaka u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije. U ovom se primjeru otvaraju 4 grafička prozora i u svakom prikazuje ista funkcija. Ovdje se koristi naredba **`surf(x,y,z,c)`**, pri čemu je polje **`c`** dimenzija 5*5*3 i sadrži komponente boja (crvenu, zelenu i plavu). Te komponente boja određuju boju svakog segmenta mreže. Dimenzija polja **`c`** je 5*5*3 jer je dimenzija mreže 4*4, a postoje tri komponente boje. U prvom je grafičkom prozoru (slika 14.21) polje **`c`** stvoreno pomoću funkcija **`rand`** i **`zeros`**, gdje crvena komponenta boje sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,1], a zelena i plava komponenta boje sadrže nule. U drugom je grafičkom prozoru (slika 14.22) polje **`c`** stvoreno pomoću funkcija **`rand`** i **`zeros`**, gdje zelena komponenta boje sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,1], a crvena i plava komponenta boje sadrže nule. U trećem je grafičkom prozoru (slika 14.23) polje **`c`** stvoreno pomoću funkcija **`rand`** i **`zeros`**, gdje plava komponenta boje sadrži slučajne realne brojeve iz jednolike razdiobe u rasponu [0,1], a crvena i zelena komponenta boje sadrže nule. U četvrtom je grafičkom prozoru (slika 14.24) polje **`c`** stvoreno pomoću funkcije **`rand`**, gdje crvena, zelena i plava komponenta boje sadrže slučajne realne brojeve iz jednolike razdiobe u rasponu [0,1].

```
close all
clear all
clc
```

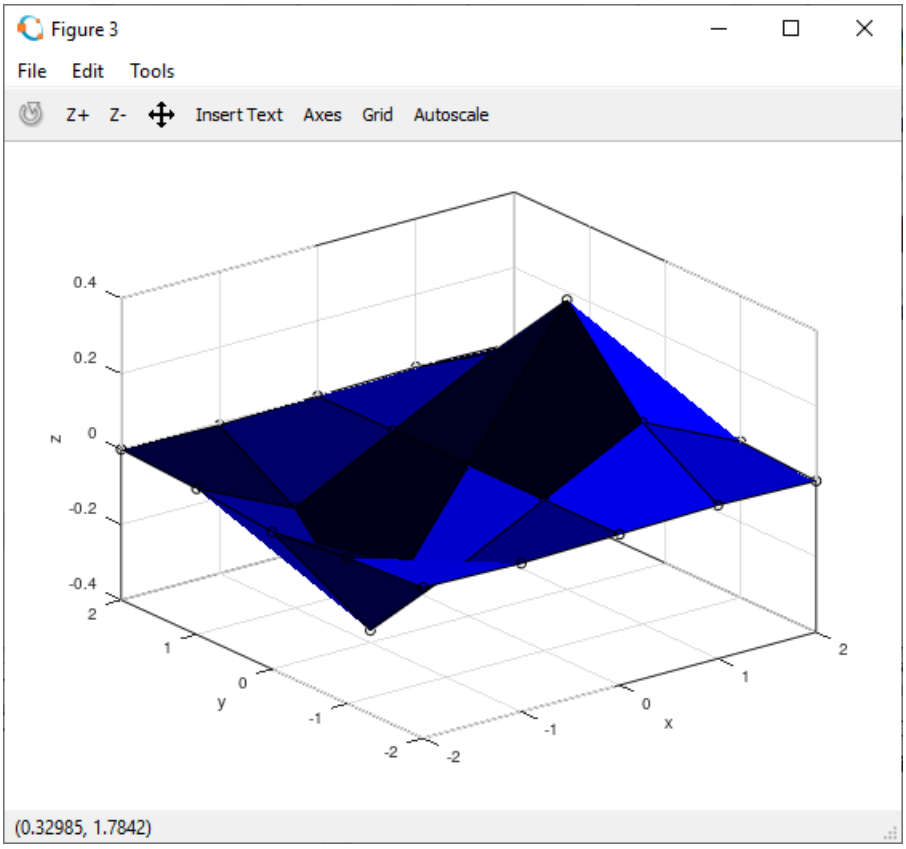
```
format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,5);
yv = linspace(-2,2,5);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
figure
% Boje segmenata mreze
c(:,:,1) = rand(5,5);
c(:,:,2) = zeros(5,5);
c(:,:,3) = zeros(5,5);
surf(xm,ym,zm,c,'Marker','o')
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
% Boje segmenata mreze
c(:,:,1) = zeros(5,5);
c(:,:,2) = rand(5,5);
c(:,:,3) = zeros(5,5);
surf(xm,ym,zm,c,'Marker','o')
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
% Boje segmenata mreze
c(:,:,1) = zeros(5,5);
c(:,:,2) = zeros(5,5);
c(:,:,3) = rand(5,5);
surf(xm,ym,zm,c,'Marker','o')
xlabel('x')
ylabel('y')
zlabel('z')
box on

% Prikaz funkcije z=f(x,y)
figure
% Boje segmenata mreze
c(:,:,1) = rand(5,5);
c(:,:,2) = rand(5,5);
c(:,:,3) = rand(5,5);
surf(xm,ym,zm,c,'Marker','o')
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



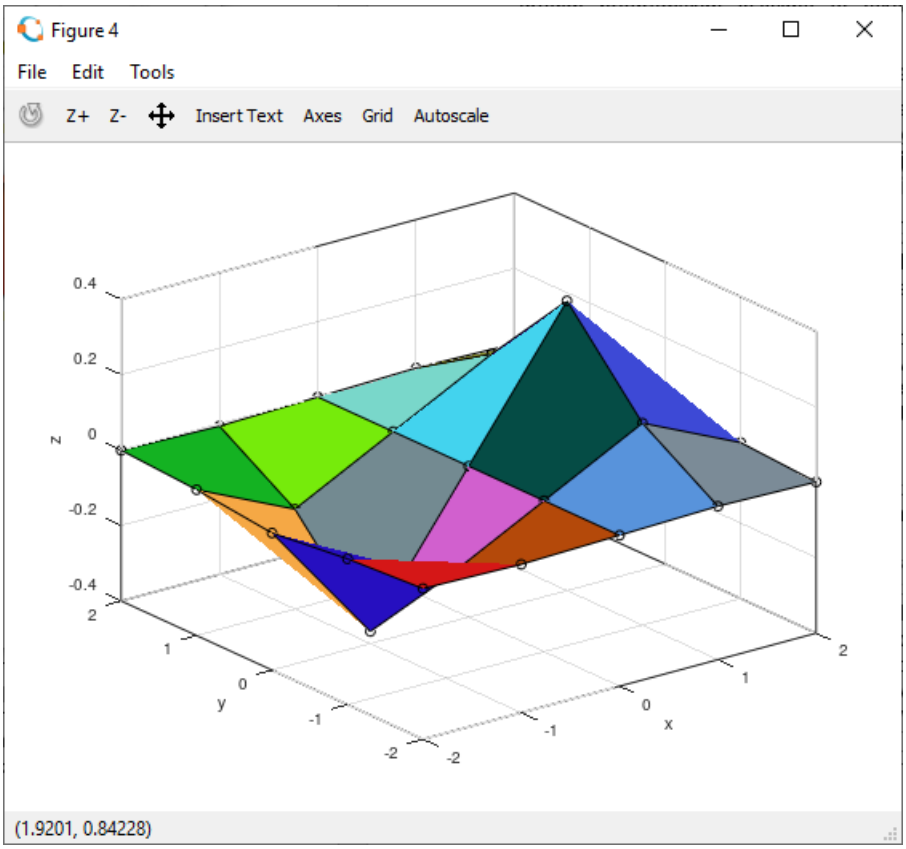
Slika 14.21 Prikaz funkcije pomoću naredbe **`surf`** gdje se koristi polje **`c`**, koje definira boju svakog segmenta mreže, dimenzija 5*5*3 i sadrži samo crvenu komponentu boje



Slika 14.22 Prikaz funkcije pomoću naredbe **`surf`** gdje se koristi polje **`c`**, koje definira boju svakog segmenta mreže, dimenzija 5*5*3 i sadrži samo zelenu komponentu boje



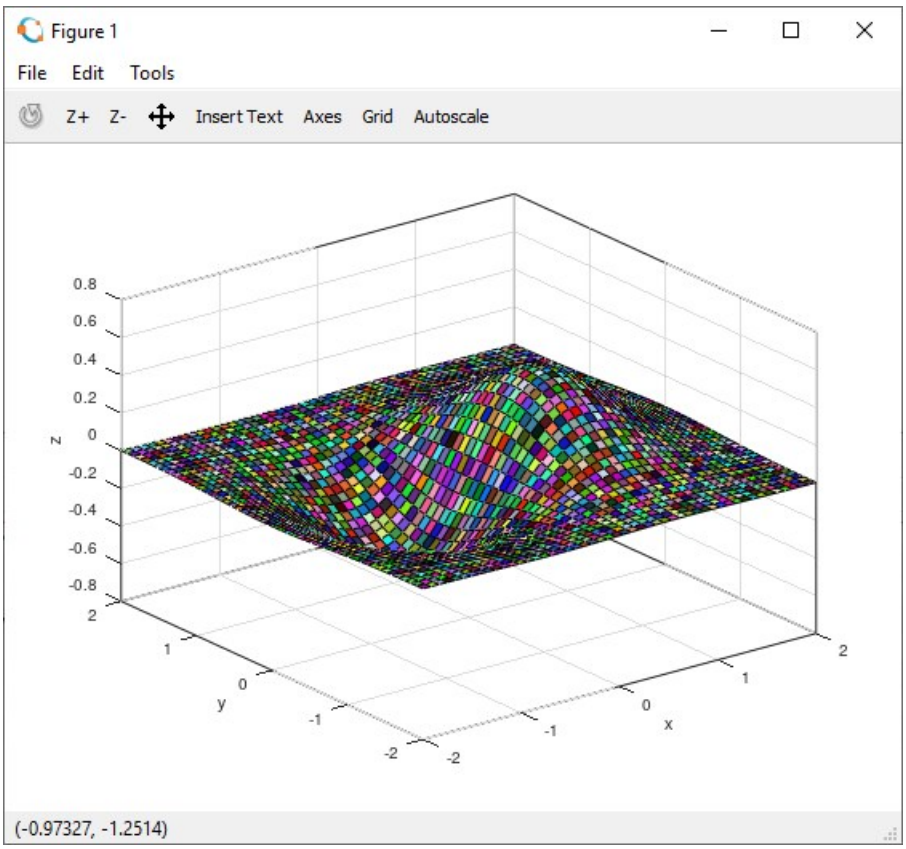
Slika 14.23 Prikaz funkcije pomoću naredbe **`surf`** gdje se koristi polje **`c`**, koje definira boju svakog segmenta mreže, dimenzija 5*5*3 i sadrži samo plavu komponentu boje



Slika 14.24 Prikaz funkcije pomoću naredbe **surf** gdje se koristi polje **c**, koje definira boju svakog segmenta mreže, dimenzija 5*5*3 i sadrži crvenu, zelenu i plavu komponentu boje

Primjer 14.14: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 50 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 50*50. Matrice **xm** i **ym** dimenzija 50*50 koordinate su točaka. Pomoću naredbe **surf** prikazana je funkcija definirana pomoću točaka u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije, slika 14.25. Ovdje se koristi naredba **surf(x,y,z,c)**, pri čemu je polje **c** dimenzija 49*49*3 i sadrži komponente boja (crvenu, zelenu i plavu). Te komponente boja određuju boju svakog segmenta mreže. Dimenzija polja **c** je 50*50*3 jer je dimenzija mreže (broj pravokutnika) 50*50, a postoje tri komponente boje. Polje **c** stvoreno je pomoću funkcije **rand**, gdje crvena, zelena i plava komponenta boje sadrže slučajne realne brojeve iz jednolike razdiobe u rasponu [0,1].

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,50);
yv = linspace(-2,2,50);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
% Boje segmenata mreze
c(:,:,1) = rand(50,50);
c(:,:,2) = rand(50,50);
c(:,:,3) = rand(50,50);
surf(xm,ym,zm,c)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



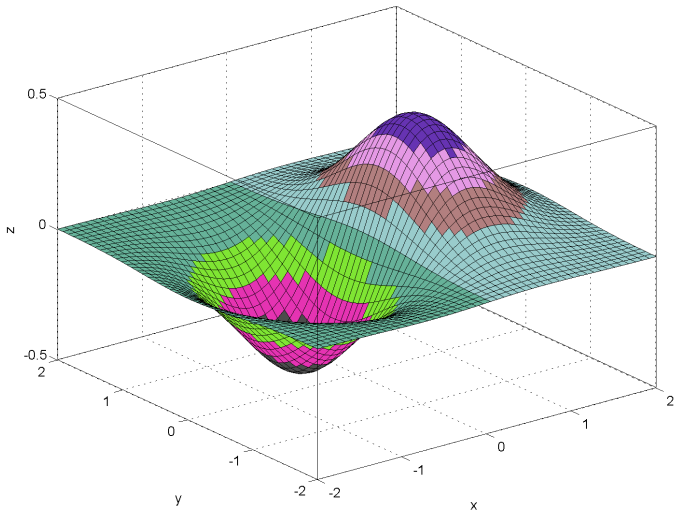
Slika 14.25 Prikaz funkcije pomoću naredbe **`surf`** gdje se koristi polje **`c`**, koje definira boju svakog segmenta mreže, dimenzija 50*50*3 i sadrži crvenu, zelenu i plavu komponentu boje

Osim što je moguće rabiti ugrađene palete boja u Octaveu, pomoću naredbe **`colormap`** moguće je stvoriti i vlastitu paletu boja. Paleta boja mora biti matrica dimenzija m*3, a vrijednosti elemenata matrice moraju biti u rasponu [0,1]. Ako je, npr. paleta boja definirana pomoću matrice **`cm`** dimenzija 8*3, tada se paleta boja postavlja za trenutačno aktivni grafički prozor pomoću naredbe **`colormap(cm)`**.

Primjer 14.15: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 50 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 50*50. Matrice **`xm`** i **`ym`** dimenzija 50*50 koordinate su točaka Pomoću naredbe **`surf`** prikazana je funkcija definirana pomoću točaka u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije, slika 14.26. Ovdje se koriste naredba **`colormap`** i funkcija **`randi`** kako bi se stvorila korisnička paleta boja. Pomoću funkcije **`randi`** definirana je matrica **`cm`** dimenzija 8*3 koja sadrži brojeve 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8 i 0,9, slučajno birane pomoću jednolike razdiobe. Matrica **`cm`** dimenzija 8*3 koristi se u naredbi **`colormap`** za stvaranje korisnički definirane palete koja sadrži 8 boja. Matrica **`cm`** je dimenzija 8*3 jer postoje tri komponente boje (crvena, zelena i plava), a korisnički definirana paleta sadrži 8 boja.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,50);
yv = linspace(-2,2,50);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
```

```
% Prikaz funkcije z=f(x,y)
% Korisnicka paleta boja
cm = randi([2 9],8,3)/10;
surf(xm,ym,zm)
colormap(cm)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.26 Prikaz funkcije pomoću naredbe **`surf`** gdje se koristi korisnički definirana paleta boja koja sadrži 8 boja

axis

Kada se izvrši naredba **`mesh`**, **`surf`**, **`plot3`** ili neka druga naredba za crtanje u 3D prostoru, Octave automatski određuje raspon vrijednosti na osima na temelju vrijednosti funkcije koja se prikazuje. Raspon osi može se promijeniti pomoću naredbe **`axis`**. Naredba **`axis`** ima različite oblike, a ovdje će biti navedene one koje se najčešće koriste:

`axis([xmin xmax ymin ymax zmin zmax])` – određuje najmanju i najveću vrijednost osi na temelju vrijednosti **`xmin`**, **`xmax`**, **`ymin`**, **`ymax`**, **`zmin`** i **`zmax`**

`axis auto` – automatsko iscrtavanje osi

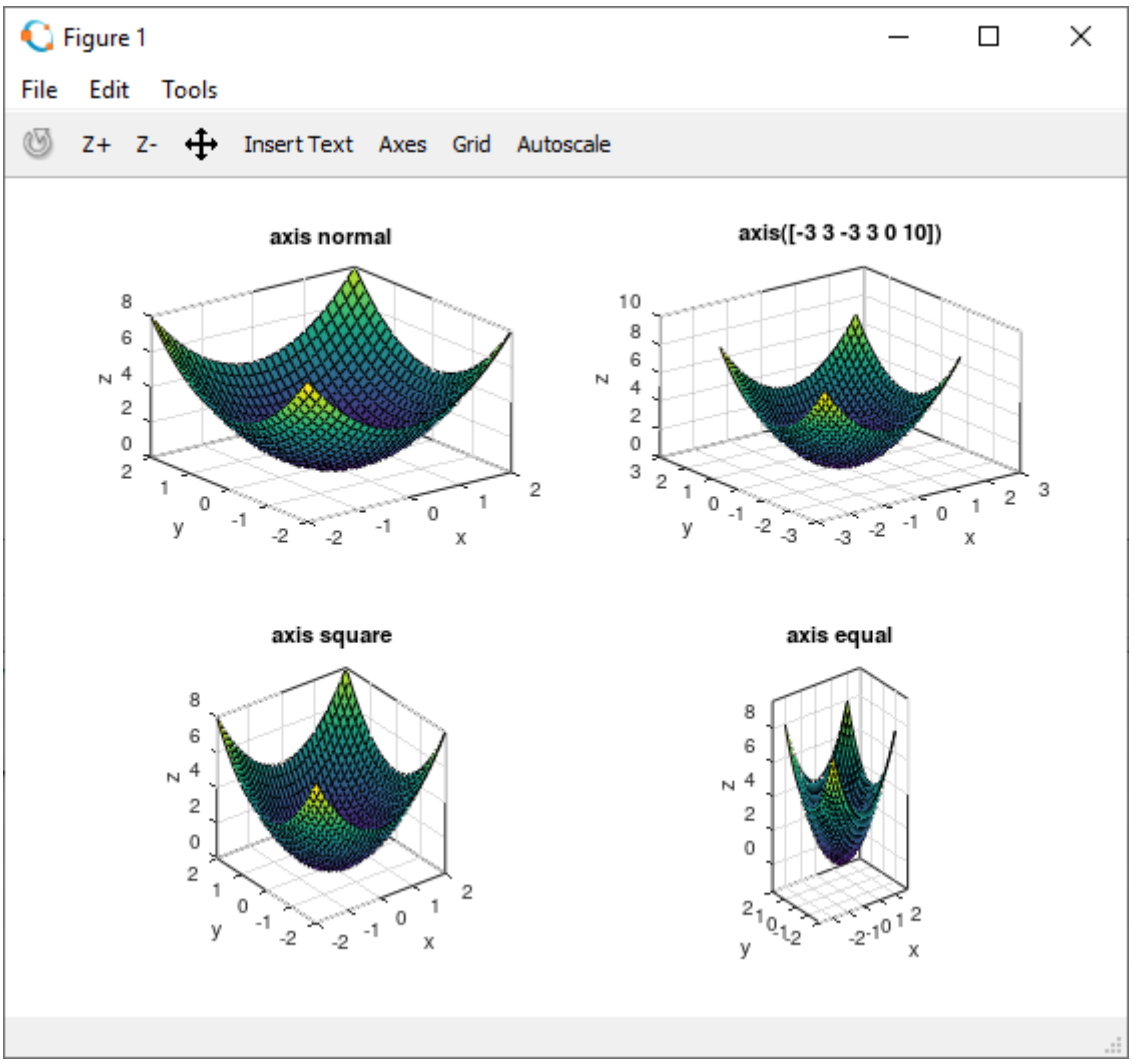
`axis equal` – odnos između x, y i z osi je 1:1:1

`axis square` – kvadratni (u slučaju 2D) ili kockasti (u slučaju 3D) oblik područja osi (raspon x, y i z-osi je jednake duljine).

Primjer 14.16: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`surf`** prikazana je funkcija definirana pomoću točaka u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm.^2 + ym.^2`** koja sadrži vrijednosti funkcije. Pomoću naredbe **`subplot`** u grafičkom prozoru otvaraju se 4 grafa od kojih svaki prikazuje istu funkciju, slika 14.27. Svaka funkcija prikazana je u drugom načinu prikaza koordinatnih osi. Prvi prikaz koristi automatski način. Drugi prikaz koristi naredbu **`axis([-3 3 -3 3 0 10])`** gdje su rasponi x, y i z osi postavljene na određene vrijednosti. U trećem se prikazu koristi naredba **`axis square`** gdje su sve tri osi jednakih duljina i tvore kocku. U četvrtom se prikazu koristi naredba **`axis equal`** gdje su podjele svih osi jednake.

```
close all
clear all
clc
```

```
format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm.^2 + ym.^2;

% Prikaz funkcije z=f(x,y)
figure
subplot(2,2,1)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('axis normal')
box on

subplot(2,2,2)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('axis([-3 3 -3 3 0 10])')
box on
axis([-3 3 -3 3 0 10])

subplot(2,2,3)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('axis square')
box on
axis square

subplot(2,2,4)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('axis equal')
box on
axis equal
```



Slika 14.27 Prikaz iste funkcije u istom grafičkom prozoru sa 4 različita prikaza koordinatnih osi

view

Pomoću naredbe **`view`** određuje se trenutačni pogled promatrača na graf u 3D prostoru. Oblik naredbe je:

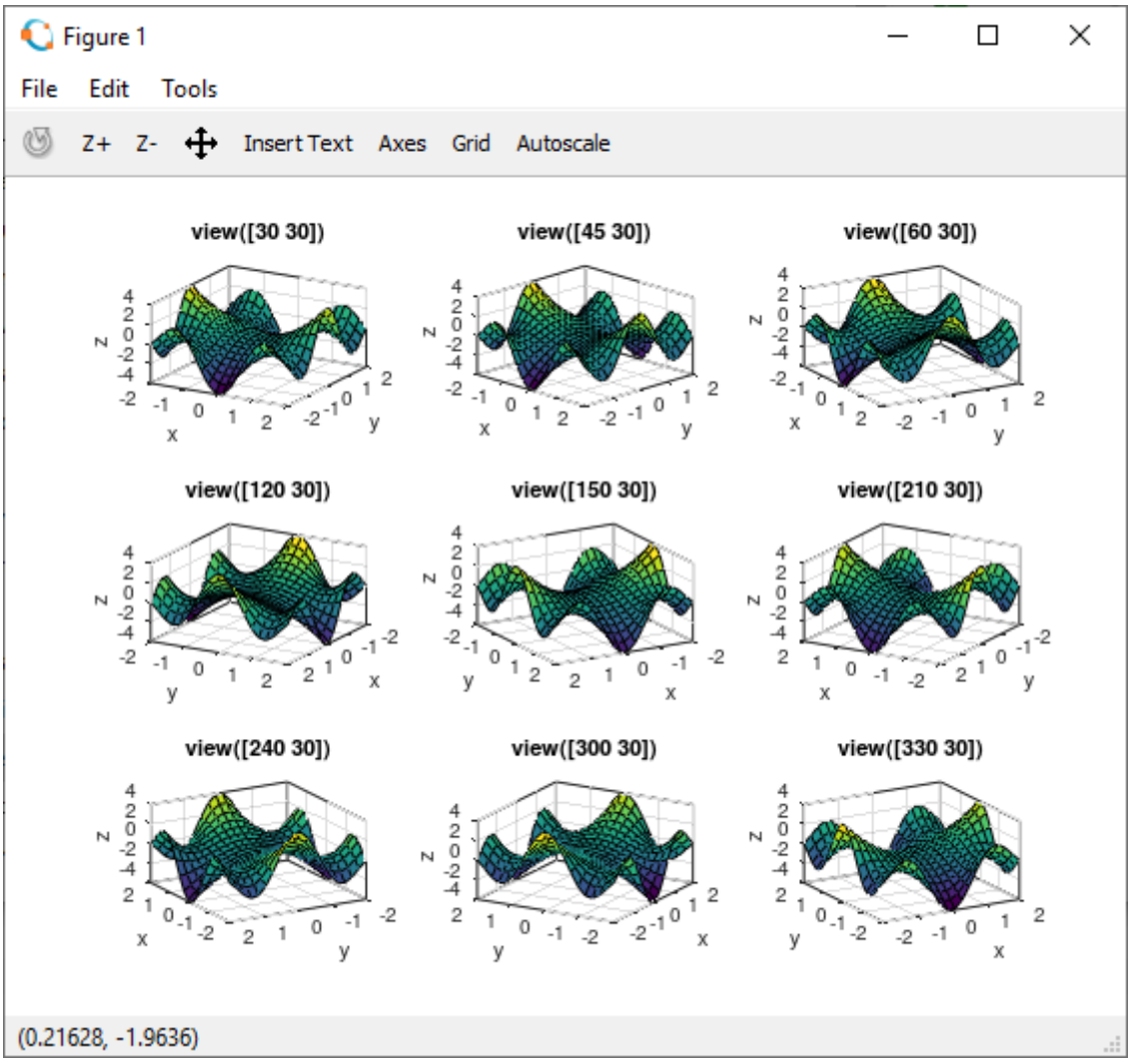
`view([az el])` – gdje je **`az`** azimut (vrijednost u rasponu [0,360]), a **`el`** elevacija (vrijednost u rasponu [0,90]).

Primjer 14.17: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 20 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 20*20. Matrice **`xm`** i **`ym`** dimenzija 20*20 koordinate su točaka. Pomoću naredbe **`surf`** prikazana je funkcija definirana pomoću vrijednosti u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = (sin(xm) + cos(ym)) .* sin(xm+ym)`** koja sadrži vrijednosti funkcije. Pomoću naredbe **`subplot`** u istom se grafičkom prozoru otvara 9 grafova od kojih svaki prikazuje istu funkciju, slika 14.34. Svaka funkcija prikazana je iz drugog kuta s obzirom na azimut pogleda.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,20);
yv = linspace(-2,2,20);
```

```
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = cos(xm.*ym) .* (xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
figure
subplot(3,3,1)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([30 30])')
box on
view([30 30])

subplot(3,3,2)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([45 30])')
box on
view([45 30])

subplot(3,3,3)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([60 30])')
box on
view([60 30])

subplot(3,3,4)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([120 30])')
box on
view([120 30])

subplot(3,3,5)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([150 30])')
box on
view([150 30])

subplot(3,3,6)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([210 30])')
box on
view([210 30])

subplot(3,3,7)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([240 30])')
box on
view([240 30])

subplot(3,3,8)
```

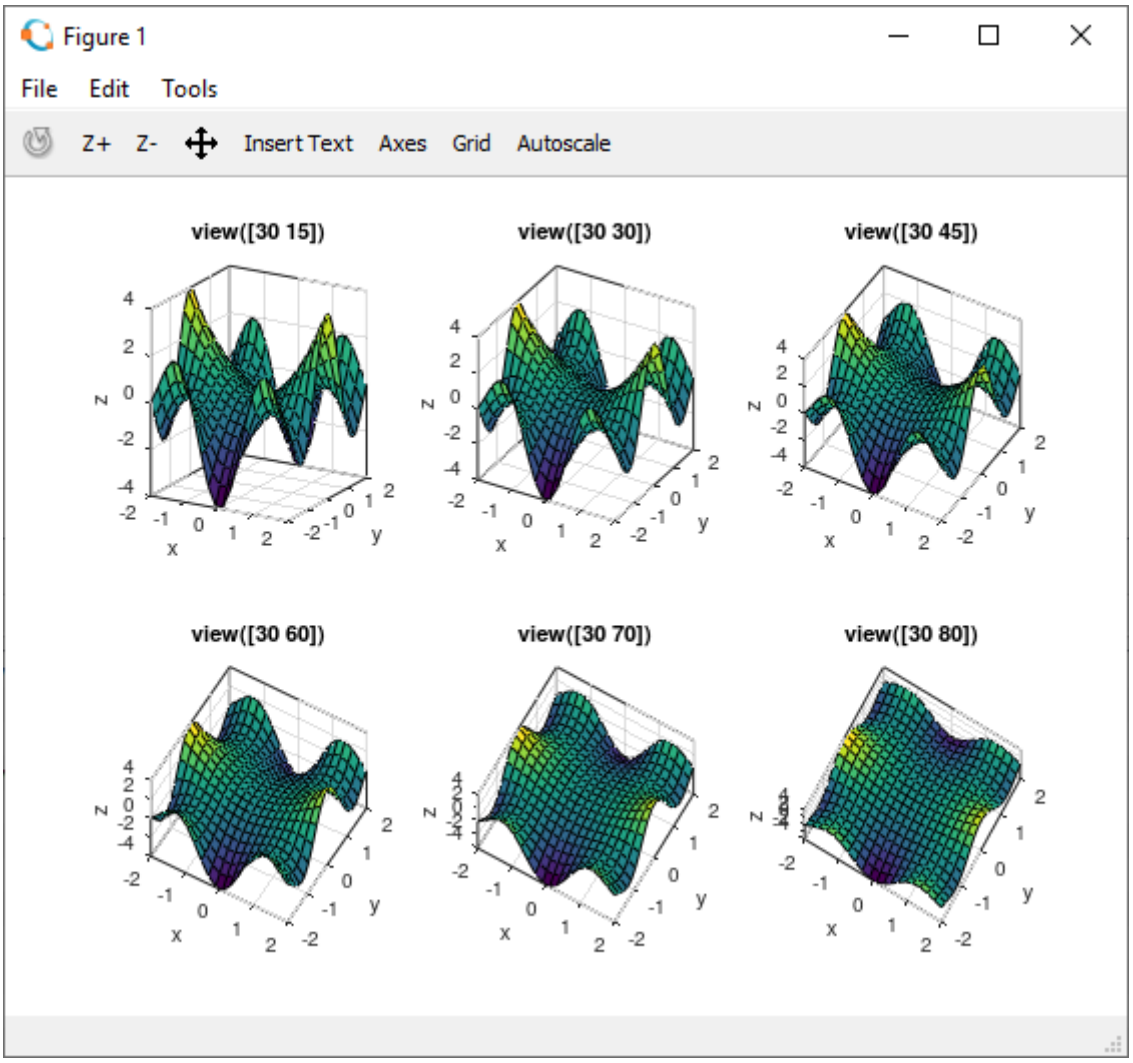
```
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([300 30])')
box on
view([300 30])

subplot(3,3,9)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([330 30])')
box on
view([330 30])
```



Slika 14.28 Prikaz iste funkcije u istom grafičkom prozoru sa 9 različitih pogleda s obzirom na azimut pogleda

Primjer 14.18: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 20 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 20*20. Matrice **xm** i **ym** dimenzija 20*20 su koordinate točaka. Pomoću naredbe **surf** prikazana je funkcija definirana pomoću vrijednosti u matricama **xm** i **ym** te pomoću matrice **zm = (sin(xm) + cos(ym)) .* sin(xm+ym)** koja sadrži vrijednosti funkcije. Pomoću naredbe **subplot** u istom grafičkom prozoru otvara se 6 grafova od kojih svaki prikazuje istu funkciju, slika 14.35. Svaka funkcija prikazana je iz drugog kuta s obzirom na elevaciju pogleda.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,20);
yv = linspace(-2,2,20);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = cos(xm.*ym) .* (xm.^2 - ym.^2);

% Prikaz funkcije z=f(x,y)
figure
subplot(2,3,1)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([30 15])')
box on
view([30 15])

subplot(2,3,2)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([30 30])')
box on
view([30 30])

subplot(2,3,3)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([30 45])')
box on
view([30 45])

subplot(2,3,4)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([30 60])')
box on
view([30 60])

subplot(2,3,5)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([30 70])')
box on
view([30 70])

subplot(2,3,6)
surf(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
title('view([30 80])')
box on
view([30 80])
```



Slika 14.29 Prikaz iste funkcije u istom grafičkom prozoru sa 6 različitih pogleda s obzirom na elevaciju

contour

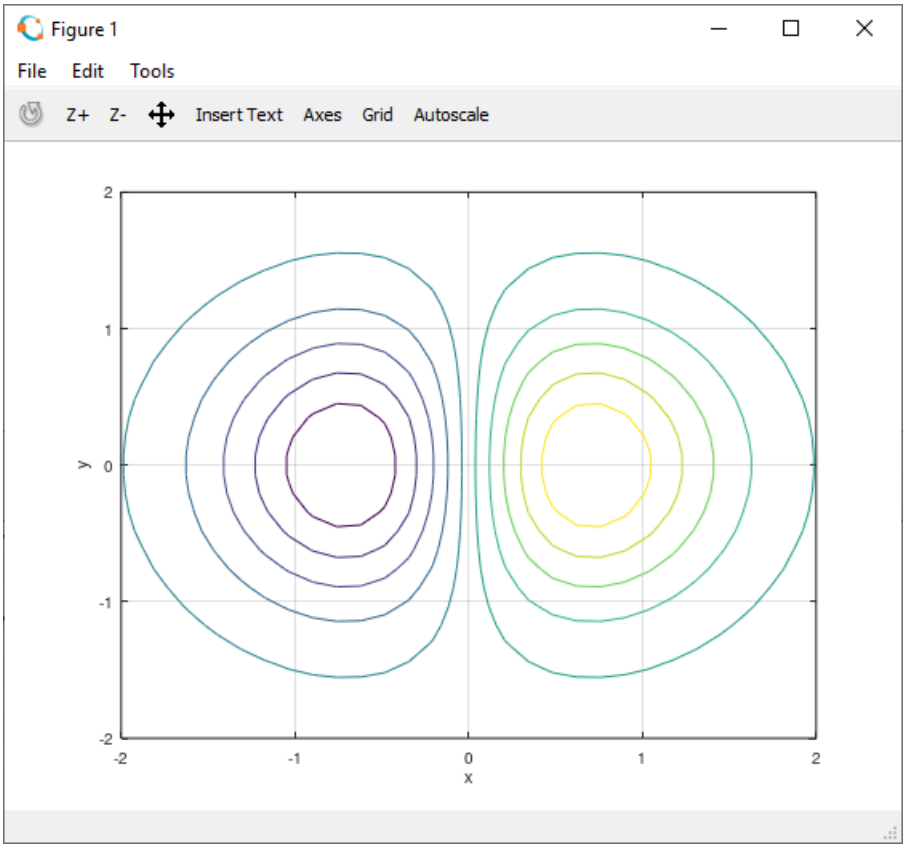
Pomoću naredbe **`contour`** prikazuju se obrisi (konture) funkcije dviju varijabli projicirani na xy-ravninu. Oblik naredbe je:

`contour(xm,ym,zm)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, a matrica **`zm`** predstavlja vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**.

Primjer 14.19: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`contour`** prikazani su obrisi (konture) funkcije definirani pomoću vrijednosti u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije, slika 14.36. Naredba **`contour`** prikazuje projekciju funkcije na xy-ravninu.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
```

```
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
contour(xm,ym,zm)
grid on
xlabel('x')
ylabel('y')
```



Slika 14.30 Prikaz obrisa (kontura) funkcije na xy-ravnini pomoću naredbe **`contour`** (broj obrisa 10)

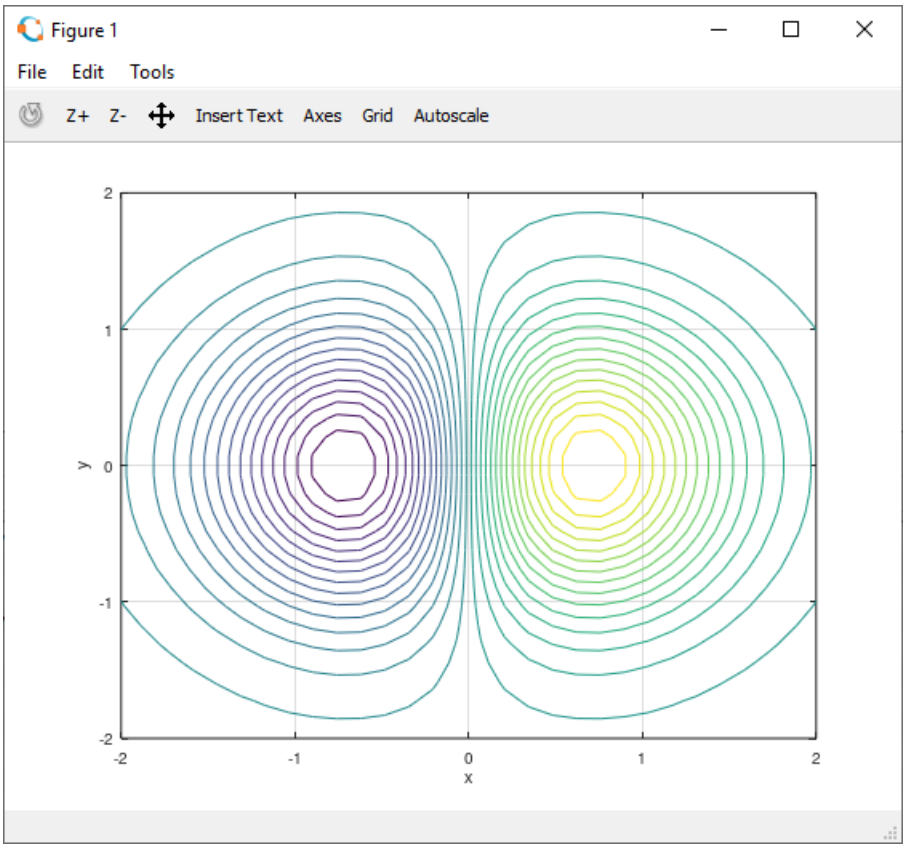
Kod naredbe **`contour`** moguće je odrediti broj obrisa (kontura) funkcije dviju varijabli projiciranih na xy-ravninu. Oblik naredbe je:

`contour(xm,ym,zm,n)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**, a **`n`** skalar koji određuje broj obrisa.

Primjer 14.20: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`contour(xm,ym,zm,n)`** prikazani su obrisi (konture) funkcije definirani pomoću vrijednosti u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije, slika 14.37. Skalar **`n`** određuje broj obrisa (kontura) funkcije. U ovom je primjeru broj obrisa (kontura) funkcije 30.

```
close all
clear all
clc
```

```
format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
contour(xm,ym,zm,30)
grid on
xlabel('x')
ylabel('y')
```



Slika 14.31 Prikaz obrisa (kontura) funkcije na xy-ravnini pomoću naredbe **`contour`** (broj obrisa 30)

contour3

Pomoću naredbe **`contour3`** prikazuju se obrisi (konture, izohipse) funkcije dviju varijabli u 3D prostoru. Oblik naredbe je:

`contour3(xm,ym,zm)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, a matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**.

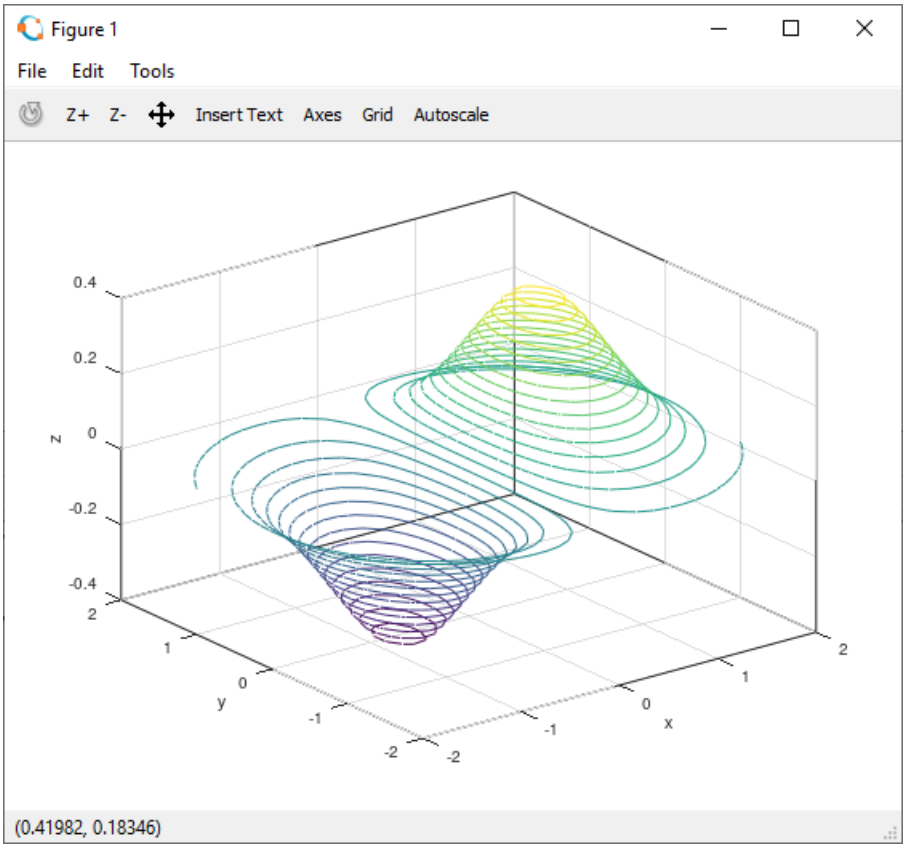
Kod naredbe **`contour3`** moguće je odrediti broj obrisa (kontura) funkcije dviju varijabli u 3D prostoru. Oblik naredbe je:

`contour3(xm,ym,zm,n)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**, a **`n`** skalar koji određuje broj obrisa.

Primjer 14.21: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`contour3(xm,ym,zm,n)`** prikazani su obrisi (konture) funkcije definirani pomoću vrijednosti u

matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije, slika 14.38. Naredba **contour3** prikazuje obrise (konture) funkcije u prostoru. Skalar **n** određuje broj obrisa (kontura) funkcije. U ovom je primjeru broj obrisa (kontura) funkcije 30.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
contour3(xm,ym,zm,30)
grid on
box on
xlabel('x')
ylabel('y')
zlabel('z')
```



Slika 14.32 Prikaz obrisa (kontura) funkcije u prostoru pomoću naredbe **contour3** (broj obrisa 30)

contourf

Pomoću naredbe **contourf** prikazuju se ispunjeni obrisi (konture, izohipse) funkcije dviju varijabli projiciranih na xy-ravninu. Oblik naredbe je:

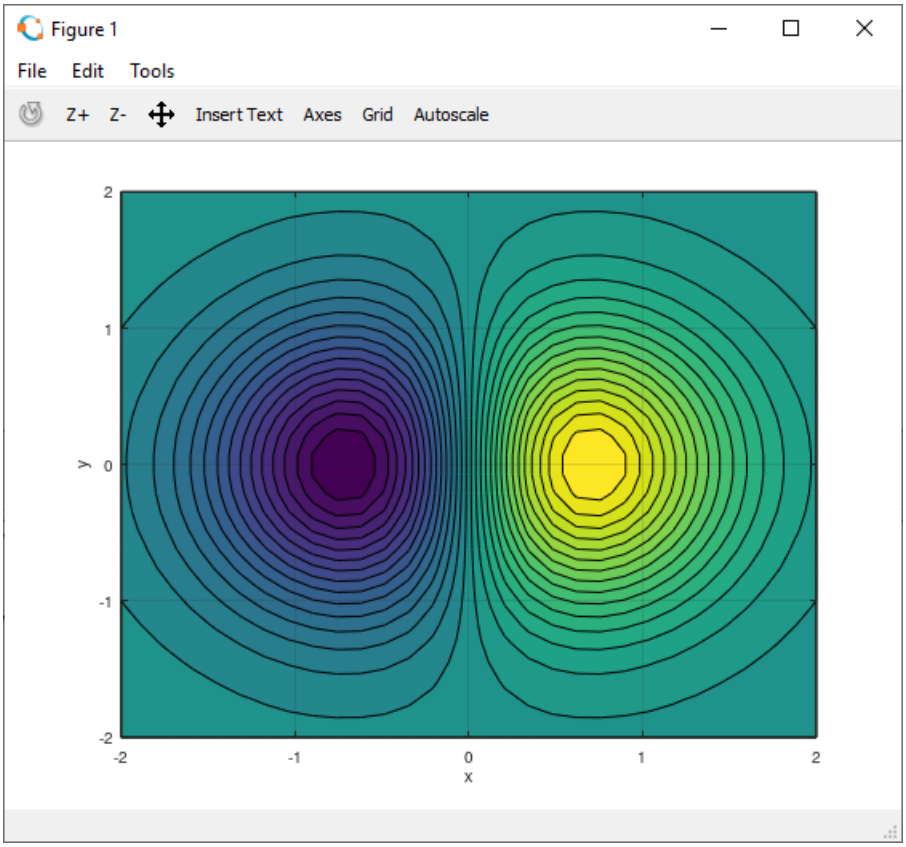
`contourf(xm,ym,zm)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, a matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**.

Kod naredbe **`contourf`** moguće je odrediti broj ispunjenih obrisa (kontura, izohipsi) funkcije dviju varijabli projiciranih na xy-ravninu. Oblik naredbe je:

`contourf(xm,ym,zm,n)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**, a **`n`** skalar koji određuje broj obrisa.

Primjer 14.22: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`contourf(xm,ym,zm,n)`** prikazani su ispunjeni obrisi (konture) funkcije definirani pomoću vrijednosti u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije, slika 14.39. Skalar **`n`** određuje broj ispunjenih obrisa (kontura) funkcije. U ovom je primjeru broj ispunjenih obrisa (kontura) funkcije 30.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
contourf(xm,ym,zm,30)
grid on
xlabel('x')
ylabel('y')
```



Slika 14.33 Prikaz ispunjenih obrisa (kontura) funkcije na xy-ravnini pomoću naredbe **`contourf`** (broj obrisa 30)

clabel

Pomoću naredbi **`contour`** ili **`contourf`** i **`clabel`** moguće je ispisati vrijednosti funkcije na obrise (konture, izohipse). Oblik naredbe je:

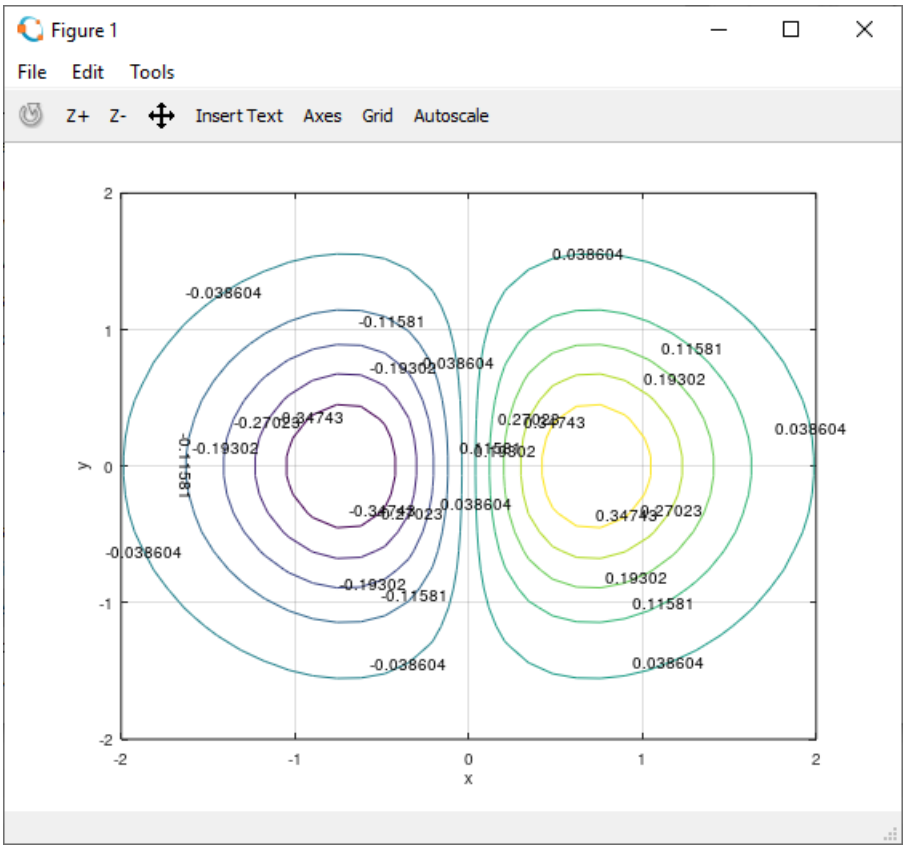
`clabel(c,h)` – gdje je **`c`** matrica dobivena pomoću naredbe **`contour`** ili **`contourf`** koja sadrži obrise (konture) funkcije, a **`h`** je struktura koja sadrži između ostalog vrijednosti funkcije i položaj gdje se ispisuju.

Prije naredbe **`clabel(c,h)`** potrebno je stvoriti matricu **`c`** i strukturu **`h`** pomoću naredbe **`[c,h] = contour(xm,ym,zm,n)`** ili **`[c,h] = contourf(xm,ym,zm,n)`** gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**, a **`n`** skalar koji određuje broj obrisa.

Primjer 14.23: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`contour`** dobiveni su podatci o obrisima (konturama) funkcije definirani pomoću vrijednosti u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije. Zatim su pomoću naredbe **`clabel`** prikazani obrisi (konture) funkcije i ispisane vrijednosti funkcije na svakoj pojedinoj konturi, slika 14.40.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
```

```
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
[c] = contour(xm,ym,zm,10);
clabel(c)
grid on
xlabel('x')
ylabel('y')
```

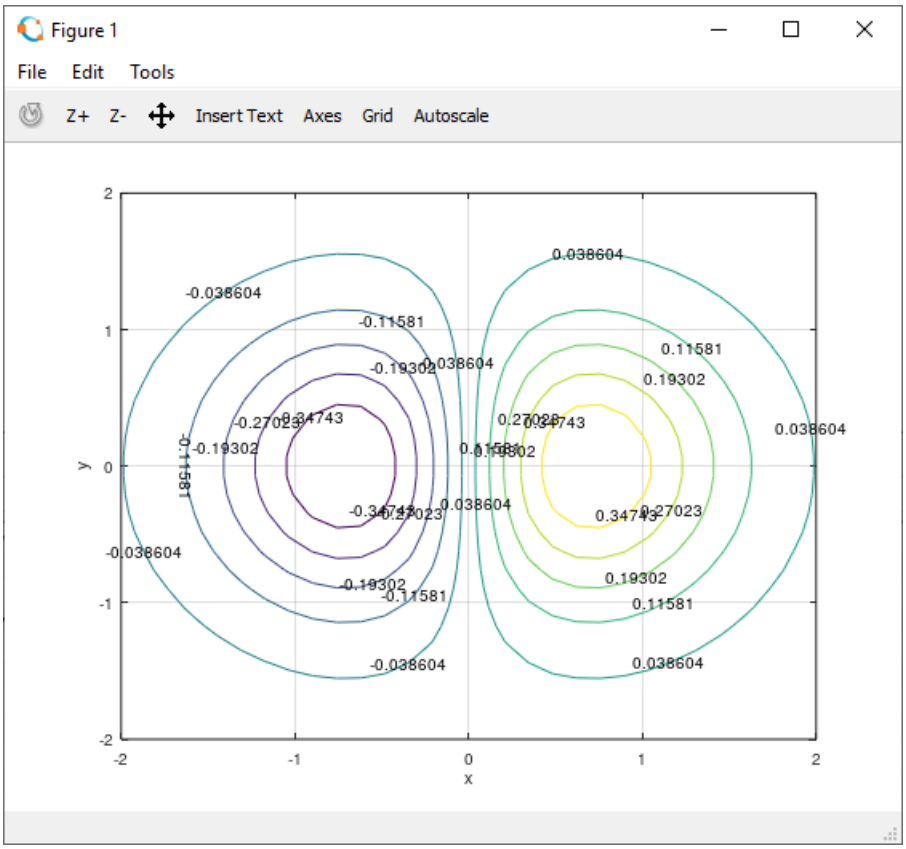


Slika 14.34 Prikaz obrisa (kontura) funkcije na xy-ravnini s ispisanim vrijednostima funkcije na svakoj pojedinoj konturi pomoću naredbe **`clabel`**

Primjer 14.24: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka Pomoću naredbe **`contour`** izračunati su podatci o obrisima (konturama) funkcije definirane pomoću točaka u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije. Zatim su pomoću naredbe **`clabel`** prikazani obrisi (konture) funkcije i ispisane vrijednosti funkcije na svakoj pojedinoj konturi. Vrijednosti funkcije nalaze se na svakom obrisu funkcije ispisane više puta (slika 14.41).

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
```

```
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
[c,h] = contour(xm,ym,zm,10);
clabel(c,h)
grid on
xlabel('x')
ylabel('y')
```

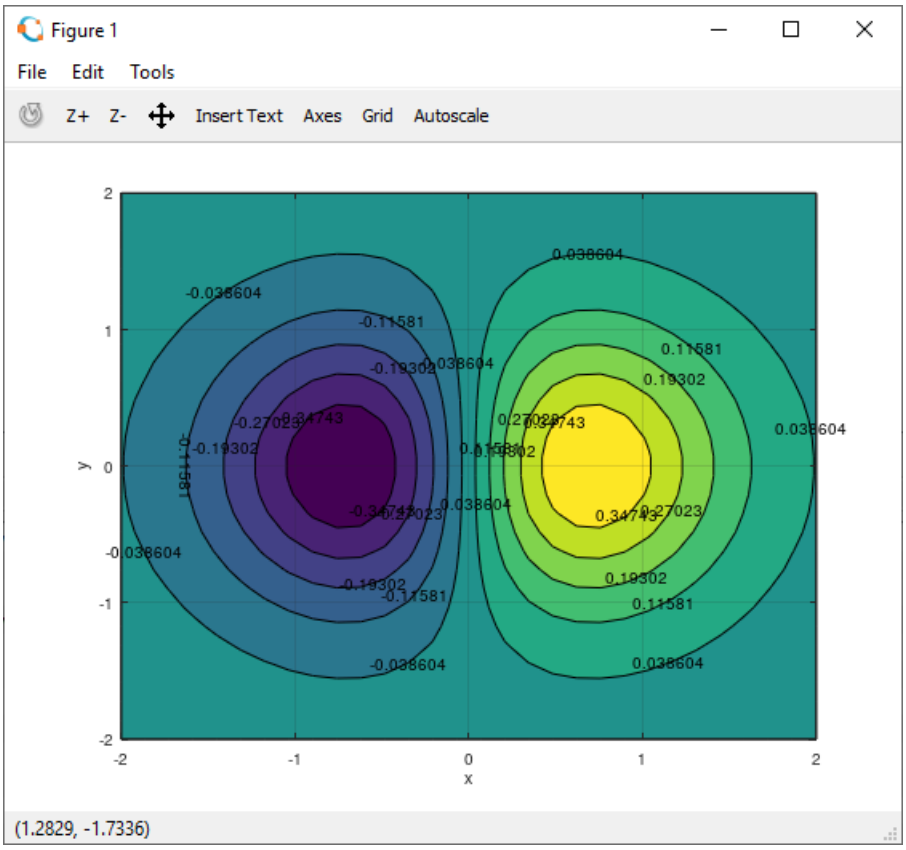


Slika 14.35 Prikaz obrisa (kontura) funkcije na xy-ravnini s vrijednostima funkcije na svakoj konturi ispisanim više puta pomoću naredbe **`clabel`**

Primjer 14.25: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`contourf`** izračunati su podatci o ispunjenim obrisima (konturama) funkcije definirani pomoću vrijednosti u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije. Zatim su pomoću naredbe **`clabel`** prikazani ispunjeni obrisi (konture) funkcije i ispisane vrijednosti funkcije na svakoj pojedinoj konturi. Vrijednosti funkcije nalaze se na svakom obrisu ispisane više puta (slika 14.42).

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
```

```
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
[c,h] = contourf(xm,ym,zm,10);
clabel(c,h)
grid on
xlabel('x')
ylabel('y')
```



Slika 14.36 Prikaz ispunjenih obrisa (kontura) funkcije na xy-ravnini s vrijednostima funkcije na svakoj pojedinoj konturi ispisanim više puta pomoću naredbe **`clabel`**

meshc

Naredba **`meshc`** služi za istodobni žičani i konturni prikaz funkcije dviju varijabli u 3D prostoru. Najčešći oblik naredbe je:

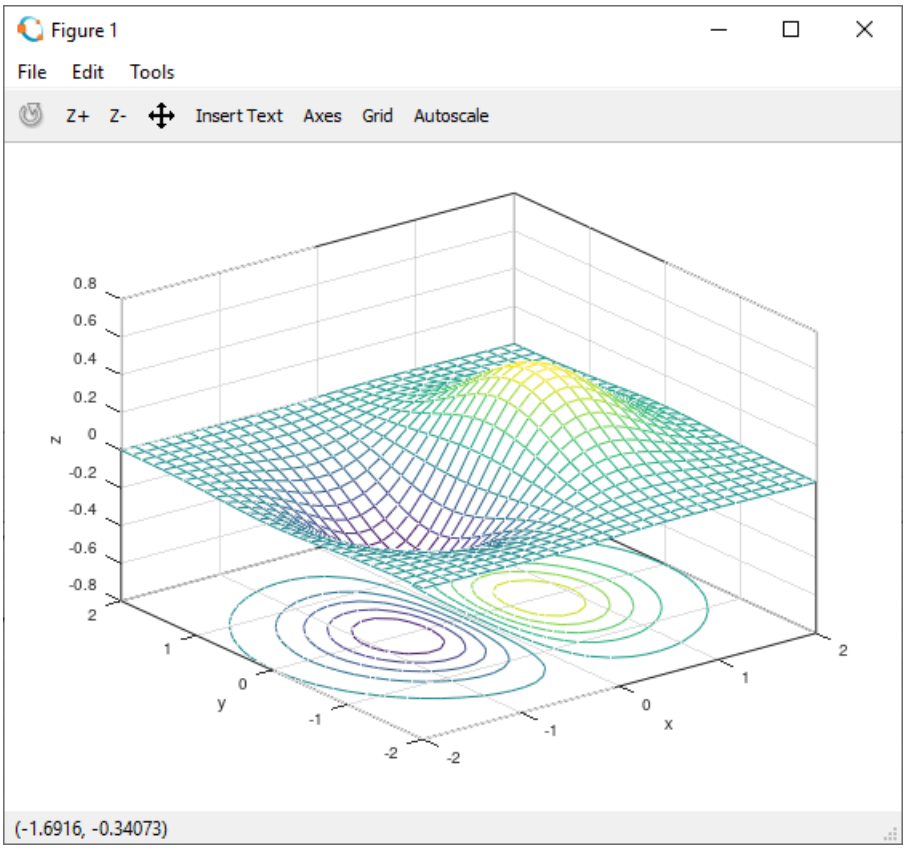
`meshc(xm,ym,zm)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, a matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**.

Obrisi (konture) funkcije prikazuju se na najmanjoj vrijednosti koordinate z-osi u xy-ravnini, odnosno na dnu grafa koji prikazuje žičani prikaz funkcije.

Primjer 14.26: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`meshc`** prikazana je funkcija definirana pomoću vrijednosti u matricama **`xm`** i **`ym`** te pomoću matrice **`zm = xm .* exp(-xm.^2 - ym.^2)`** koja sadrži vrijednosti funkcije, slika 14.43. Naredba **`meshc`**, osim toga, prikazuje i obrise (konture) funkcije u xy-ravnini na najmanjoj vrijednosti koordinate z-osi na grafu.

```
close all
clear all
```

```
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
meshc(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.37 Žičani prikaz funkcije i obrisa (kontura) funkcije u xy-ravnini pomoću naredbe **`meshc`**

surfc

Naredba **`surfc`** služi za istodobni plošni i konturni prikaz funkcije dviju varijabli u 3D prostoru. Najčešći oblik naredbe je:

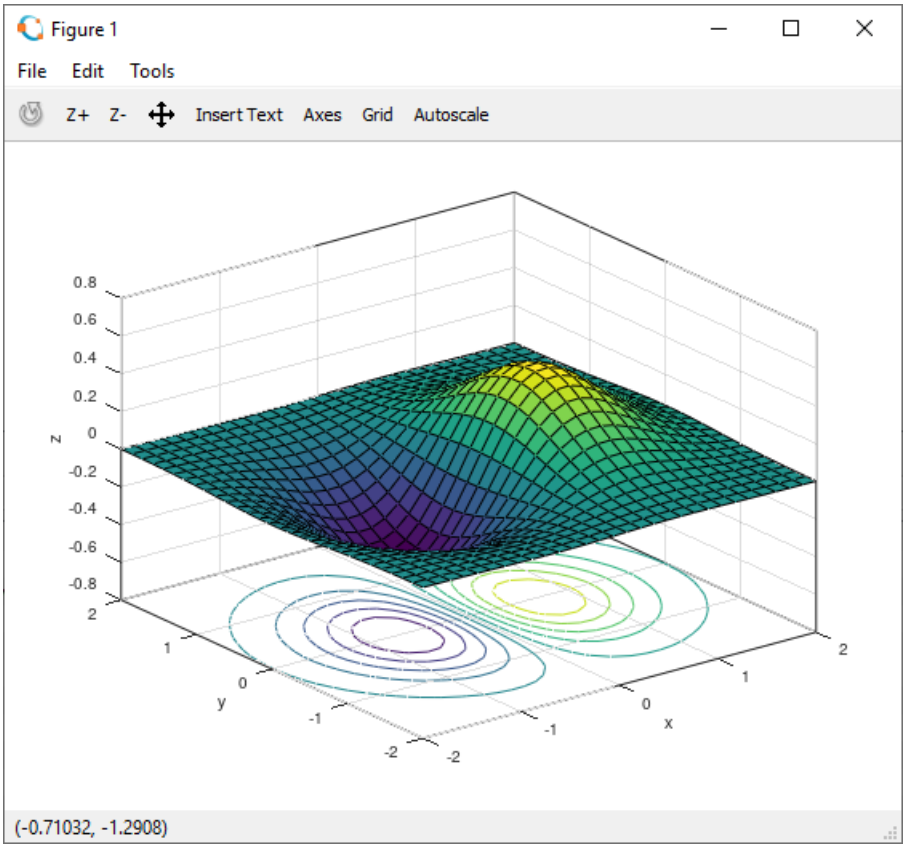
`surfc(xm,ym,zm)` – gdje su matrice **`xm`** i **`ym`** točke koje definiraju mrežu, a matrica **`zm`** sadrži vrijednosti funkcije za točke u matricama **`xm`** i **`ym`**.

Obrisi (konture) funkcije prikazuju se na najmanjoj vrijednosti koordinate z-osi u xy-ravnini, odnosno na dnu grafa koji prikazuje plošni prikaz funkcije.

Primjer 14.27: Definirani su vektori **`xv`** i **`yv`** pomoću funkcije **`linspace`** u rasponu [-2,2]. Redni vektori **`xv`** i **`yv`** imaju 30 elemenata. Pomoću funkcije **`meshgrid`** te vektora **`xv`** i **`yv`** definirane su matrice **`xm`** i **`ym`** dimenzija 30*30. Matrice **`xm`** i **`ym`** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **`surfc`**

prikazana je funkcija definirana pomoću vrijednosti u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije, slika 14.44. Naredba **surfc** prikazuje, osim toga, i obrise (konture) funkcije u xy-ravnini na najmanjoj vrijednosti koordinate z-osi na grafu.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
surfc(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.38 Plošni prikaz funkcije i obrisa (kontura) funkcije u xy-ravnini pomoću naredbe **surfc**

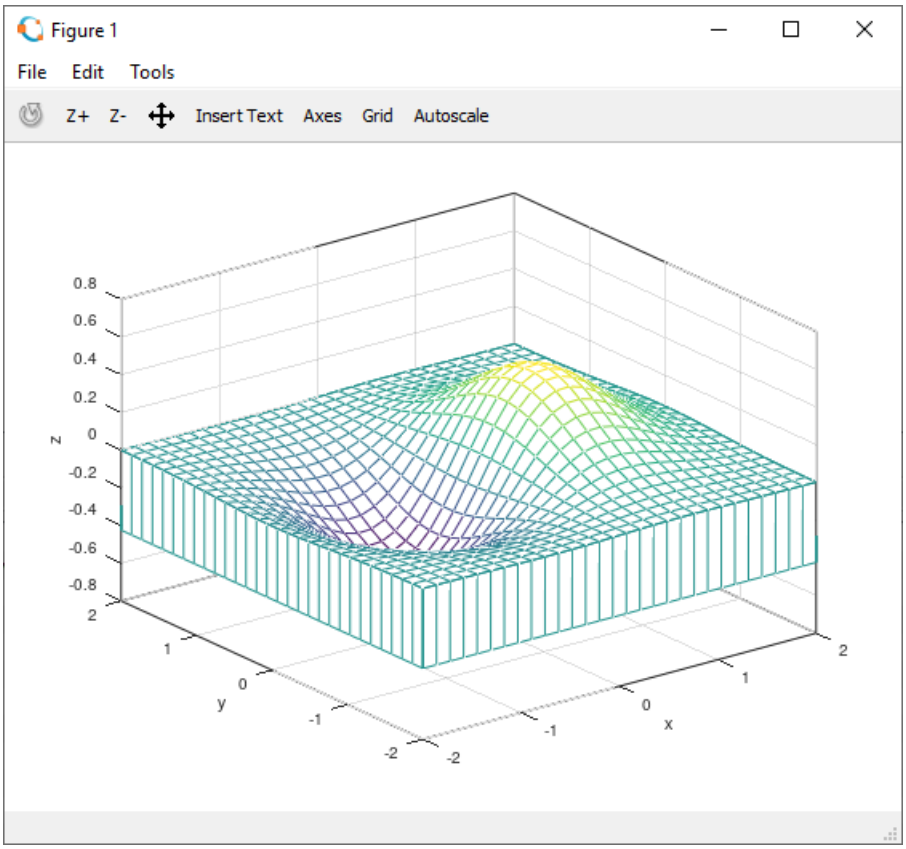
meshz

Naredba **meshz** služi za žičani prikaz funkcije dviju varijabli u 3D prostoru i zavjesu u smjeru z-osi. Najčešći oblik naredbe je:

meshz(xm,ym,zm) – gdje su matrice **xm** i **ym** točke koje definiraju mrežu, a matrica **zm** sadrži vrijednosti funkcije za točke u matricama **xm** i **ym**.

Primjer 14.28: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 30 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 30*30. Matrice **xm** i **ym** dimenzija 30*30 koordinate su točaka. Pomoću naredbe **meshz** prikazana je funkcija definirana pomoću vrijednosti u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije, slika 14.45. Naredba **meshz**, osim toga, prikazuje zavjesu u smjeru z-osi.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,30);
yv = linspace(-2,2,30);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
meshz(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.39 Žičani prikaz funkcije i zavjese u smjeru z-osi pomoću naredbe **meshz**

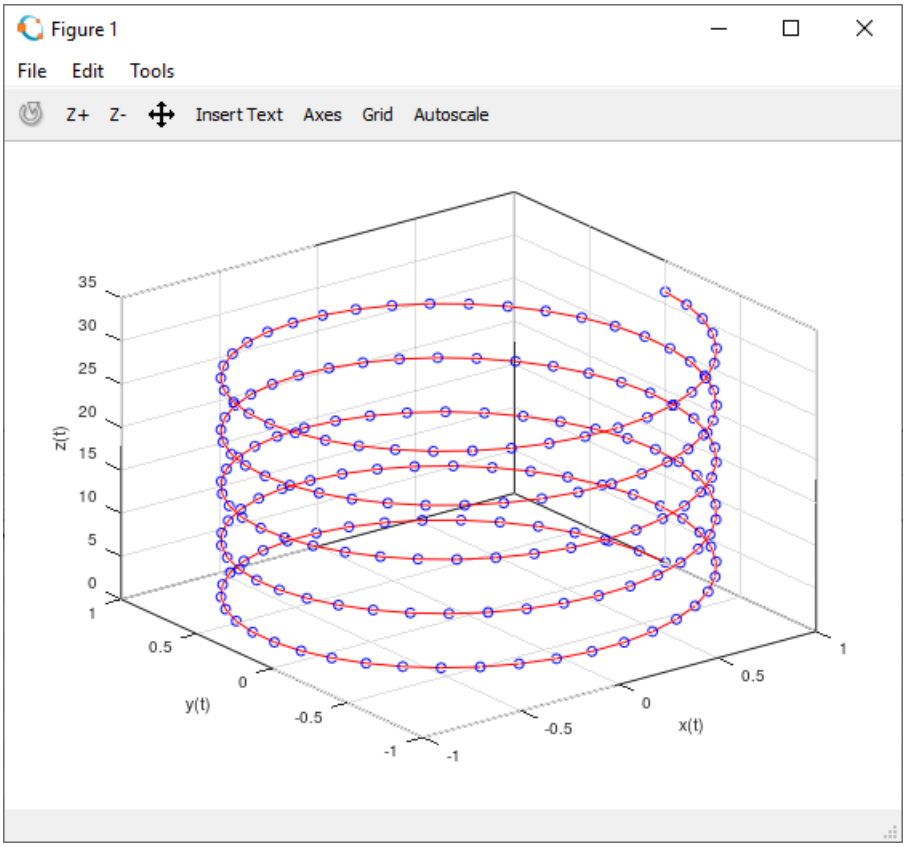
plot3

Naredba **plot3** služi za prikaz točaka i crta u 3D prostoru. Ta naredba odgovara naredbi **plot** u 2D prostoru. Parametri koji se mogu koristiti kod naredbe **plot**, opisani u poglavlju 2D prikaz podataka, mogu se koristiti i kod naredbe **plot3**. Oblik naredbe je:

plot3(x,y,z) – gdje su **x**, **y** i **z** vektori jednakih dimenzija.

Primjer 14.29: Definiran je vektor **t** pomoću funkcije **linspace** u rasponu [0,10*pi]. Redni vektor **t** ima 200 elemenata. Redni vektor **x** sadrži vrijednosti funkcije kosinus za vrijednosti vektora **t**, redni vektor **y** sadrži vrijednosti funkcije sinus za vrijednost vektora **t**, a redni vektor **z** jednak je vektoru **t**. Naredba **plot3** prikazuje točke i crte koje ih povezuju u prostoru, koje su definirane rednim vektorima **x**, **y** i **z**. Kod naredbe **plot3** moguće je koristiti iste parametre kao kod naredbe **plot** koji su opisani u poglavlju 2D prikaz podataka. U ovom su primjeru korišteni parametri **'Marker'** za oblik markera i **'MarkerEdgeColor'** za boju obrisa markera, slika 14.46.

```
close all
clear all
clc

format short
format compact

% Vektori t, x, y i z
t = linspace(0,10*pi,200);
x = cos(t);
y = sin(t);
z = t;
% Prikaz grafa
plot3(x,y,z,'r','Marker','o','MarkerEdgeColor','b')
grid on
xlabel('x(t)')
ylabel('y(t)')
zlabel('z(t)')
box on
```

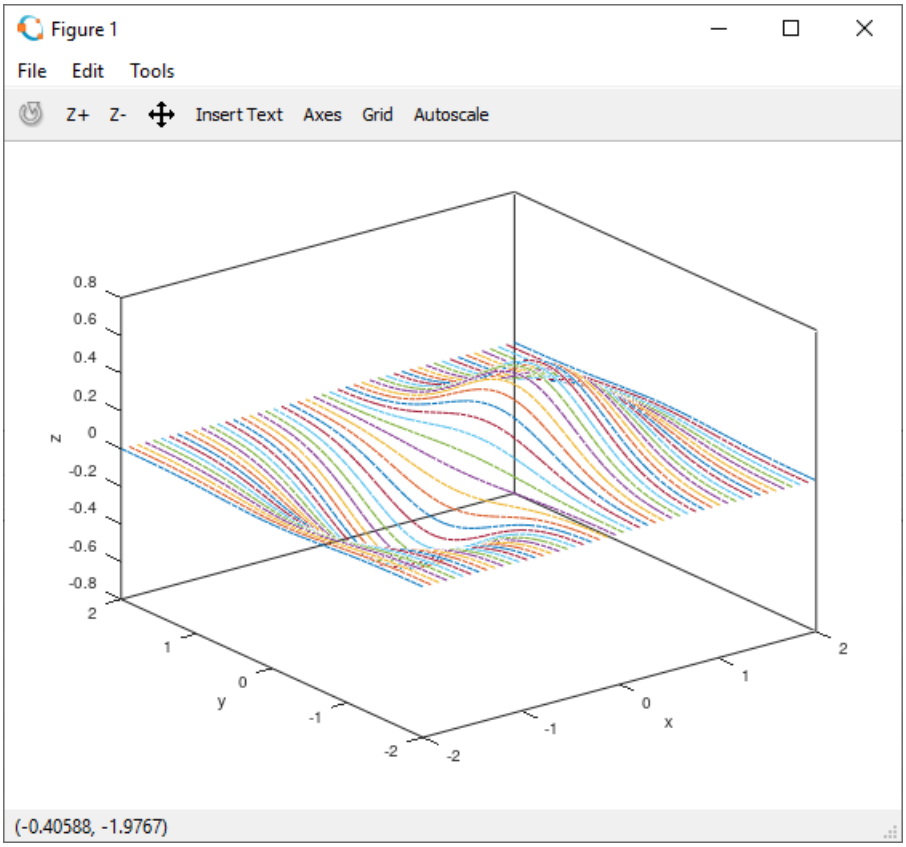


Slika 14.40 Prikaz funkcije `x = cos(t)`, `y = sin(t)` i `z = t` pomoću naredbe `plot3`

Ako su kod naredbe `plot3(xm,ym,zm)` – `xm`, `ym` i `zm` matrice, prikazuju se crte po stupcima tih matrica.

Primjer 14.30: Definirani su vektori `xv` i `yv` pomoću funkcije `linspace` u rasponu [-2,2]. Redni vektori `xv` i `yv` imaju 50 elemenata. Pomoću funkcije `meshgrid` te vektora `xv` i `yv` definirane su matrice `xm` i `ym` dimenzija 50*50. Matrice `xm` i `ym` dimenzija 50*50 koordinate su točaka. Naredba `plot3` prikazuje crte u prostoru koje su definirane matricama `xm`, `ym` i `zm`, slika 14.47.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,50);
yv = linspace(-2,2,50);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
plot3(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
box on
```



Slika 14.41 Prikaz funkcije `zm = xm .* exp(-xm.^2 - ym.^2)` pomoću naredbe `plot3`

pie3

Pomoću naredbe `pie3` crta se tortni graf u 3D prostoru. Oblici naredbe su:

`pie3(x)` – crta tortni graf u 3D prostoru s podatcima sadržanim u vektoru `x`.

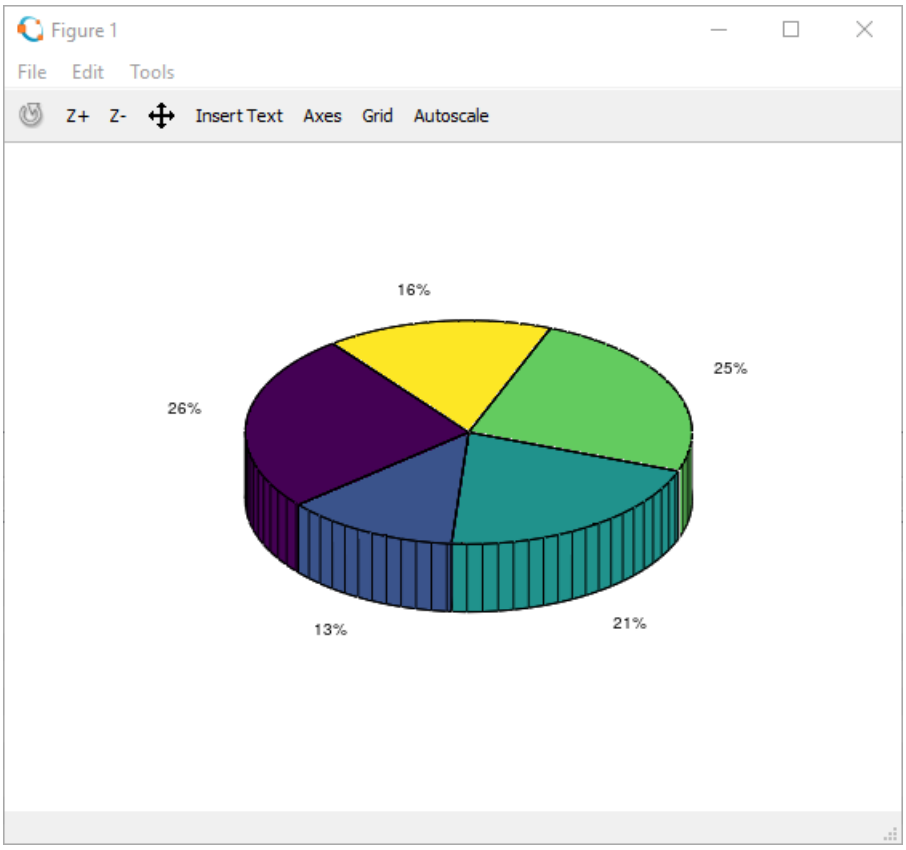
`pie3(x,explode)` – crta tortni graf u 3D prostoru s podatcima sadržanim u vektoru `x`, a vektor `explode` služi za izdvajanje segmenata tortnog grafa. Vektor `explode` mora biti iste dimenzije kao i vektor `x`. Ako su elementi vektora `explode` različiti od 0, odgovarajući elementi vektora `x` bit će izdvojeni segmenti u tortnom grafu. Najčešće vektor `explode` sadrži vrijednosti 0 i 1.

`pie3(x,'znakovni niz')` – crta tortni graf u 3D prostoru s podatcima sadržanim u vektoru `x`, a `'znakovni niz'`, koji mora biti iste dimenzije kao i vektor `x`, sadrži opise segmenata tortnog grafa.

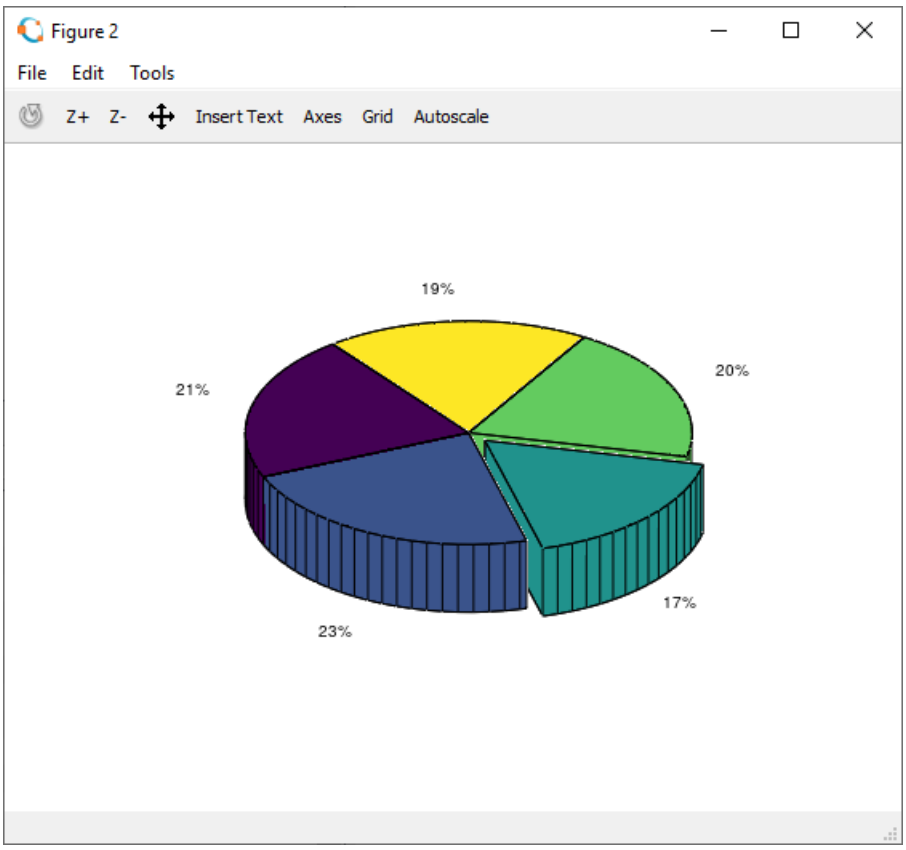
Primjer 14.31: Definiran je redni vektor `x` sa 5 elemenata koji sadrži slučajne realne brojeve u rasponu [0,20]. Program otvara 3 grafička prozora i u svakom od njih crta tortne grafove. U prvom se grafičkom prozoru prikazuje tortni graf sa 5 segmenta i s oznakama segmenata u postotcima, slika 14.53. U drugom se grafičkom prozoru prikazuje tortni graf sa 5 segmenata gdje je treći segment izdvojen, slika 14.54. U trećem se grafičkom prozoru prikazuje tortni graf sa 4 segmenta gdje su segmenti opisani tekstom, slika 14.55.

```
close all
clear all
clc

format short
format compact

figure
x = 20*rand(1,5);
pie3(x)

figure
```

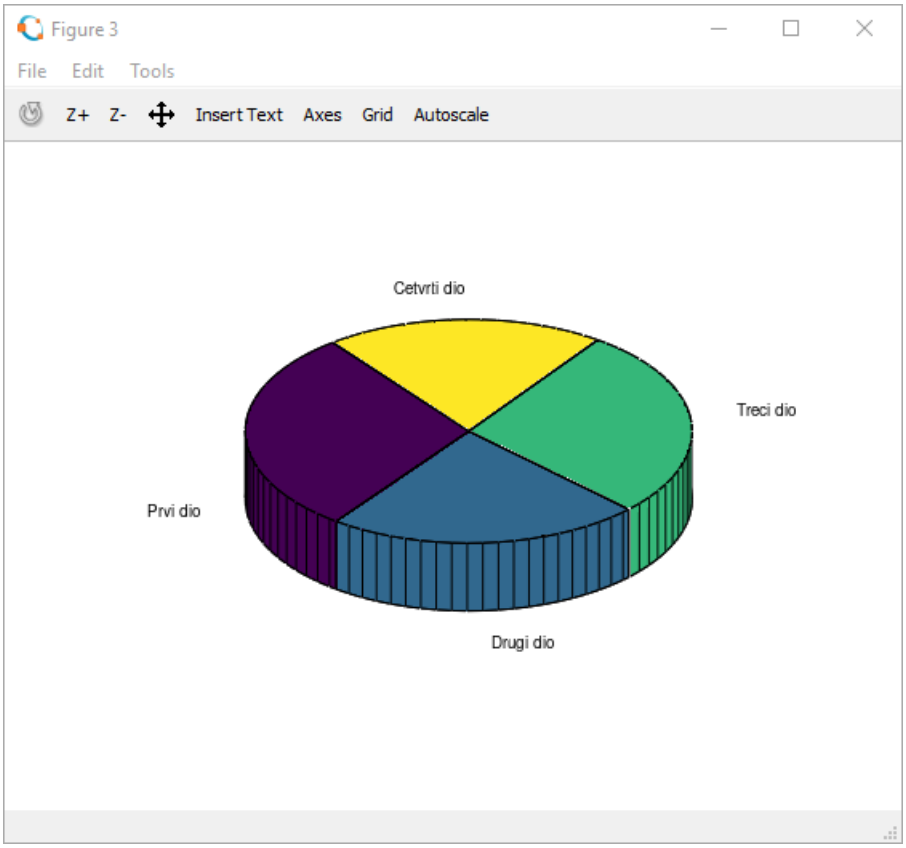
```
x = 20*rand(1,5);
explode = [0 0 1 0 0];
pie3(x,explode)

figure
x = 10*rand(1,4);
labels = {'Prvi dio','Drugi dio','Treci dio','Cetvrti dio'};
pie3(x,labels)
```



Slika 14.42 Tortni graf u 3D prostoru



Slika 14.43 Tortni graf u 3D prostoru s jednim izdvojenim segmentom



Slika 14.44 Tortni graf u 3D prostoru s oznakama segmenata opisanim tekstom

stem3

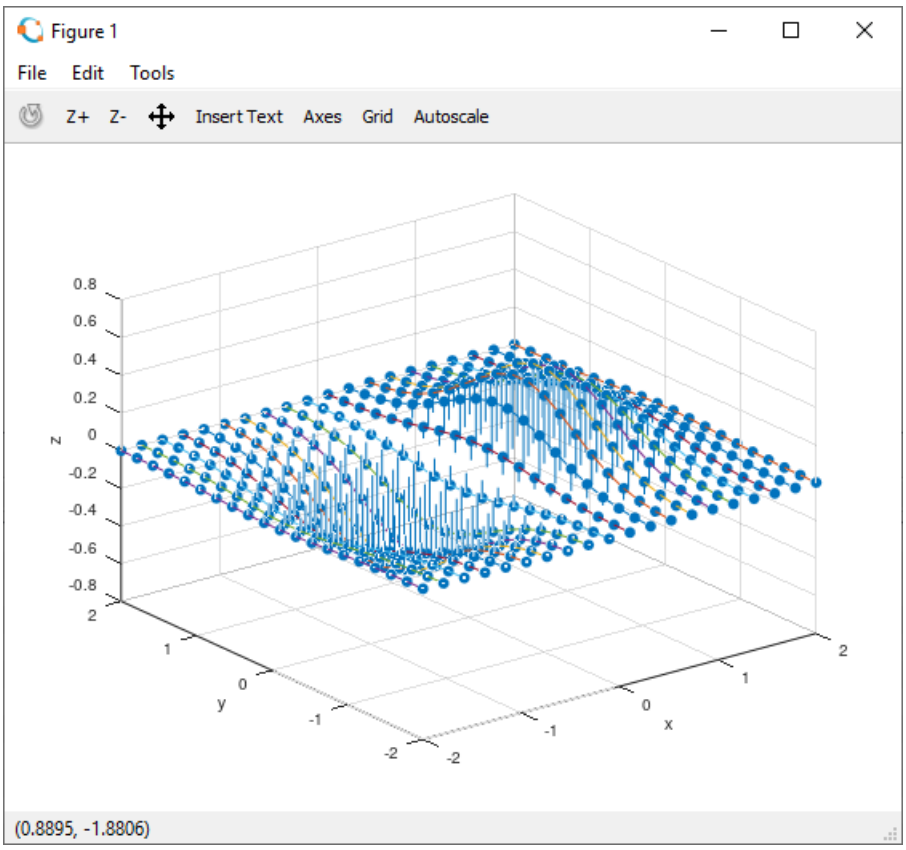
Pomoću naredbe **stem3** crta se graf diskretnih podataka u 3D prostoru. Oblik naredbe je:

stem3(xm,ym,zm) – gdje matrice **xm**, **ym** i **zm** predstavljaju podatke.

stem3(xm,ym,zm,'znakovni niz') – gdje matrice **xm**, **ym** i **zm** predstavljaju podatke, a **'znakovni niz'** su različiti parametri i njihove vrijednosti koji određuju boju, debljinu i stil crte kao kod naredbe **plot**.

Primjer 14.32: Definirani su vektori **xv** i **yv** pomoću funkcije **linspace** u rasponu [-2,2]. Redni vektori **xv** i **yv** imaju 20 elemenata. Pomoću funkcije **meshgrid** te vektora **xv** i **yv** definirane su matrice **xm** i **ym** dimenzija 20*20. Matrice **xm** i **ym** dimenzija 20*20 koordinate su točaka. Pomoću naredbe **stem3** prikazuje se graf diskretnih podataka koji predstavljaju funkciju definiranu pomoću vrijednosti u matricama **xm** i **ym** te pomoću matrice **zm = xm .* exp(-xm.^2 - ym.^2)** koja sadrži vrijednosti funkcije. Parametri koji se mogu koristiti kod naredbe **stem3** jednaki su onima koji se mogu koristiti kod naredbe **stem** koja je opisana u poglavlju 2D prikaz podataka. Na istom su grafu pomoću naredbe **plot3** prikazane crte u prostoru koje su definirane matricama **xm**, **ym** i **zm**, slika 14.56.

```
close all
clear all
clc

format short
format compact

% Vektori xv i yv
xv = linspace(-2,2,20);
yv = linspace(-2,2,20);
% Mreza
[xm,ym] = meshgrid(xv,yv);
% Funkcija z=f(x,y)
zm = xm .* exp(-xm.^2 - ym.^2);
% Prikaz funkcije z=f(x,y)
stem3(xm,ym,zm,'fill')
hold on
plot3(xm,ym,zm)
xlabel('x')
ylabel('y')
zlabel('z')
```



Slika 14.45 Prikaz funkcije diskretnim podatcima pomoću naredbi `stem3` i `plot3`

Pitanja za provjeru znanja:

1. Čemu služi naredba `meshgrid`?
2. Koja je razlika između naredbi `meshgrid` i `mesh`?
3. Koja je razlika između naredbi `mesh` i `surf`?
4. Čemu služe naredbe `xlabel`, `ylabel` i `zlabel`?
5. Čemu služi naredba `colormap`?
6. Je li moguće u Octaveu stvoriti vlastitu paletu boja?
7. Kojom se naredbom određuje način sjenčanja plohe pri 3D prikazu?
8. Čemu služi nareba `view`?
9. Pomoću koje naredbe se prikazuju obrisi funkcije dviju varijabli projicirani na xy-ravninu?
10. Koja je razlika između naredbi `contour`, `contourf` i `contour3`?
11. Čemu služe naredbe `meshc` i `surfc`?
12. Čemu služi naredba `plot3`?
13. Koja je razlika između naredbi `plot` i `plot3`?
14. Koja je razlika između naredbi `pie` i `pie3`?

15. LITERATURA

1. E. Coman,, M.W. Brewster,, S.K. Popuri,, A.M. Raim,, M.K. Gobbert, A Comparative Evaluation of Matlab, Octave, FreeMat, Scilab, R, and IDL on Tara (n.d.) 46.

2. J.W. Eaton,, D. Bateman,, S. Hauberg,, R. Wehbring, GNU Octave: Free Your Numbers (2018).

3. Linge, S. Linge,, H.P. Langtangen, Programming for Computations - MATLAB/Octave, Springer International Publishing, Place of publication not identified, 2016.

4. P.J.G. Long, Introduction to Octave (2005).

5. S. Nagar, Introduction to Octave, Apress, Berkeley, CA, 2018.

6. A. Quarteroni,, F. Saleri,, P. Gervasio, Scientific computing with MATLAB and Octave, 4. ed, Springer, Berlin, 2014.

7. J. Schmidt Hansen, GNU Octave: beginner’s guide ; become a proficient octave user by learning this high-level scientific numerical tool from the ground up, Packt Publ, Birmingham, 2011.

POJMOVNIK