Kratki uvod u Matlab

Zoran Pasarić

Geofizički zavod
Zagreb, 2004

Sadržaj

1 Uvod

Cilj je ovog "Kratkog uvoda" objasniti zainteresiranom početniku što je Matlab i kako je nastao, kako je ustrojen, koje su mu jake, a koje slabe strane (a tih je malo) te potom opisati osnovne elemente Matlabovog programskog jezika. Kao i s drugim programskim jezicima, usvajanje Matlaba vezano je za usvajanje odgovarjućeg načina razmišljanja, što dakako zahtijeva motiv, volju i nešto praktičnog rada. Ovaj tekst želi poslužiti kao početna točka na tom putu i tom cilju podredjena je duljina teksta, te količina detalja.

Matlab je napredni programski sustav, odnosno interaktivno radno okruženje za tehničko računanje i vizualizaciju. Možda je najbolje odmah reći da se radi o *profesionalnom* alatu koji, posljedično, nije namijenjen posvemašnjim laicima. To svakako ne znači da je "ulazak" u Matlab jako težak, ali znači da treba napustiti lošu naviku beskrajnog klikanja mišem, te umjesto toga usvojiti osnove ustroja i upotrebe računala (npr. da se hardware računala, najgrublje rečeno, sastoji od centralne procesne jedinice (CPU) i radne memorije (RAM) s jedne, te ulazno/izlaznih (I/O) jedinica s druge strane; da je diskovni prostor organiziran u direktorije i datoteke u obliku stabla, pri čemu je *datoteka* najmanja imenovana cjelina s kojom operacijski sustav barata; da je osnovni alat za komuniciranje s računalom tekstualni editor – a *ne* tekstprocesor – kojim se učinkovito stvaraju i mijenjaju tekstualne datoteke - naravno koristeći tipkovnicu, a ne miša; itd). Temeljna, i za početnike pomalo neugodana osobina Matlaba jest da se radi o *programskom jeziku*, u kojem se može napraviti gotovo sve, ali korisnik mora reći što i kako! Činjenica da se radi o programskom jeziku visoke razine je olakotna, no svejedno, bilo kakvo prethodno poznavanje programiranja dobro je došlo. Konačno, poput svih sustava namijenjenih profesionalnoj upotrebi, Matlab je vrlo brižljivo osmišljen i višestruko će nagraditi početni(čki) trud. Matlab uvijek misli *za* korisnika i daje (najčešće vrlo dobre) odgovore na pitanja kojih korisnik najčešće nije ni svjestan. Takodjer, kada se čini da je rješenje nekog problema složeno ili nezgrapno, odgovor (i to najčešće elegantan!) nadje se promišljanjem na način sukladan Matlabu, pri čemu se svaki put ponovo otkrije unutarnja uskladjenost i promišljenost.

Počeci Matlaba sežu u rane 70-te godine kada je Steve Moler, stručnjak za numeričku analizu, napisao prvu (fortransku) verziju sustava. Cilj je bio stvoriti sučelje kojim bi se moglo učinkovito koristiti fortranske programe, a bez stvarnog poznavanja Fortrana. Danas je Matlab komercijalni proizvod tvrtke Mathworks (`http://www.mathworks.com`), u cijelosti pisan u programskom jeziku C, koji se masovno koristi kao razvojno-istraživački alat u industriji (npr. avioni, automobili, ...) te obrazovanju i znanosti.

U srcu Matlaba nalazi se pojam *matrice*, o čemu govori i samo ime Mat-

lab koje potječe od engleskih riječi MATrix LABoratory. Matrica je jednostavan matematički objekt, pravokutna tablica brojeva, koja se prirodno javlja u najrazličitijim područjima i situacijama, dok jezgru Matlaba čini skup funkcija za jednostavno, prirodno i efikasno manipuliranje matricama. Upravo zahvaljujući moćnom matričnom *engine-u* Matlab se sve više širi i u specijalizirana područja, o čemu govore brojni novi *toolbox-i*. Danas postoji nekoliko slobodno dostupnih sustava analognih Matlabu, npr. Octave (`http://www.octave.org/`), SciLab (`http://scilab.org`), čije su jezgre sukladne Matlabovoj (i posljedično jednako moćne). Nažalost, svi oni značajno zaostaju u nadgradnji, npr. u dodatnim paketima te u grafičkom podsustavu, iako ima naznaka da SciLab postaje dostojan, premda ne i posve kompatibilan suparnik.

Navedimo neke prednosti Matlaba u usporedbi s "klasičnim" programskim jezicima poput Fortran-a ili C-a.

- Interaktivno sučelje omogućava brzo eksperimentiranje i mami na igru (Matlab je intrepretirani jezik za razliku od, npr. Fortrana, koji je prevodjen).
- Potrebna je minimalna briga oko podatkovnih struktura (praktički nema deklaracija varijabli i polja).
- Matlab omogućava brzo i lako programiranje (zahvaljujući moćnom matričnom konceptu).
- Ugradjeni grafički podsustav omogućava jednostavnu, kvalitetnu i brzu vizualizaciju.
- Programi pisani u Matlab-ovom jeziku (tzv. m-datoteke) su obične tekstualne datoteke i stoga su potpuno prenosive izmedju različitih operacijskih sustava/platformi.
- Postoje mnogobrojni dodatni paketi (*toolbox*-i), tj. skupine m-datoteka za razna specijalna područja.
- Postoje mnogobrojne m-datoteke i čitavi paketi koje autori, ujedno i korisnici, stavljaju na slobodno raspolaganje putem Interneta.

Navedimo i jedini(?), neveliki nedostatak: Programi pisani u Matlabu, koji je interpretiran, nužno se sporije izvršavaju od onih pisanih u npr. Fortanu ili C-u, koji su prevodjeni. Spomenuti nedostatak najčešće nije od značaja, naročito u današnje doba sve jeftinijih i bržih računala, a obilno je nadomješten brzim i lakim programiranjem. Vrlo često moguće ga je ublažiti

dosljednim razmišljanjem i programiranjem u duhu Matlaba (tj., sustavnim korištenjem matrica i vektora). Na kraju ostaje mogućnost korištenja fortranskih ili C programa unutar Matlaba, te korištenje Matlabovog prevodioca. Naravno, najzahtijevniji kodovi, npr. meteorološki ili oceanografski numerički modeli, i dalje ostaju u fortranskom dvorištu.

2 Pregled sustava i programskog jezika

2.1 Radni prozor i prostor, varijable i funkcije

Rad u Matlab-u odvija se putem radnog prozora, a unutar radnog prostora. Radni prozor služi za unos naredbi, te ispis rezultata i najvažniji je (a do verzije 5 i jedini) dio korisničkog sučelja. Radni, pak, prostor zamišljamo kao dio radne memorije dodijeljen Matlabu prilikom pokretanja, koji sadrži korisnikove varijable. Mogli bi reći da se kroz radni prozor "gleda" u radni prostor.

Varijable i funkcije osnovni su elementi Matlab-a, pri čemu se varijable dijele na korisničke i ugradjene (interne) (npr. `pi`$= \pi$, ili `i` za imaginarnu jedinicu). Slično se dijele i funkcije na vanjske i ugradjene, pri čemu su prve programi pisani u Matlab-u (bez obzira da li su ih pisali korisnici ili su došli s Matlabom), a druge su dio samog Matlaba pisane u C-u, pri čemu izvorni kod nije dostupan. S korisničke strane, medju njima nema razlike. Funkcije rade nad varijablama koje se trenutno nalaze u radnom prostoru ili kreiraju nove varijable. Po izvršenju odredjene funkcije sve zatečene varijable ostaju u radnom prostoru (za razliku od izvršavanja fortranskih i sličnih programa) i može ih se pregledati, crtati, te dalje koristiti. Detaljnije o funkcijama (tj. skriptama i funkcijama u užem smislu), te interakciji s radnim prostorom bit će više rečeno kasnije.

Osnovne naredbe za upravljanje radnim prostorm su:

- `who`
- `who <popis_varijabli>`

  daje kratki ispis svih (prva verzija) ili zatraženih (druga verzija) varijabli iz radnog prostora.

- `whos`
- `whos <popis_varijabli>`

  daje detaljni ispis varijabli s pripadnom veličinom i tipom (jedna od najvažnijih naredbi!!).

- `clear`
- `clear all`

- `clear <popis varijabli>`

  briše sve (prve dvije inačice), ili samo zadane varijable (treća inačica) iz radnog prostora.

- `save <ime_datoteke> <popis varijabli>`

  sprema zadane varijable u datoteku zadanog imena na tvrdi disk. Za kasnije učitavanje služi naredba `load <ime_datoteke>`.

- exit

  zatvara Matlab. Radni prostor se briše i oslobadja, a sadržaj mu se gubi.

2.2 Sustav pomoći

Počev od verzije 6 Matlab ima raskošan sustav pomoći u HTML formatu s ugradjenim preglednikom (engl. *browser*), koji se pokreće klikom miša. Tu se nalazi kompletna dokumentacija u samo-razumljivom obliku. Ipak i dalje postoji jednostavan i moćan stari sustav pomoći koji koristi naredbenu liniju. Tu nalazimo naredbe:

- `help`
- `help <ime_funkcije>`

  daje sadržaj sustava pomoći (prva verzija) ili minimalne, nužne informacije o tome što radi i kako se poziva tražena funkcija.

- `lookfor <ključna_rijec>`

  daje popis svih funkcija čiji opisi u prvom retku sadrže traženu ključnu riječ. U sprezi s naredbom `help` pretstavlja pomalo spartanski, ali vrlo jednostavan i efikasan put za brzo nalaženje tražene informacije. Počev od šeste verzije Matlaba, upotrebljivost je nešto umanjena jer skoro svaki upit s `lookfor` rezultira kojom stotinom nadjenih funkcija.

- `which <ime_funkcije>`

  ispisuje cjelovitu stazu na disku za zadanu funkciju, tj. za odgovarajuću m-datoteku.

2.3 Tipovi podataka i aritmetika

Osnovni tip podatka u Matlabu je **(kompleksni) broj u dvostrukoj preciznosti** (engl. *double array (complex)*). Na 32-bitnoj PC-arhitekturi to znači da se računa s približno 16 značajnih znamenki. Iako postoji još 15-ak

različitih tipova podataka, svekoliko računanje obavlja se isključivo u osnovnom tipu, dok eksplicitno zadavanje tipa varijable skoro da i ne postoji. Ako se u računu s realnim brojevima kao rezultat pojave kompleksni brojevi, Matlab nastavlja račun u kompleksnom području bez posebnog zahtjeva od strane korisnika. Počev od sedme verzije, Matlab poznaje i cjelobrojnu, te aritmetiku jednostruke preciznosti.

Matlab se drži danas opće prihvaćenog IEEE standarda za aritmetiku pokretnog zareza (engl. *floating point*, kratko FP), čiji detaljni opis nadilazi okvire ovog teksta (vidi npr. V. Hari, `http://www.math.hr/ rogina/2001096/greske.pdf`). Ipak recimo da standard zahtijeva da ako su x i y FP-brojevi, te $\circ$ jedna od operacija $+, -, *, /$ u FP-aritmetici, onda rezultat $z = x \circ y$ treba biti jednak onom koji bi se dobio matematički egzaktnim računom i naknadnim zaokruživanjem u FP-broj. Isto se zahtijeva i za operaciju drugog korijena. Nadalje, svaka FP-operacija mora rezultirati FP-brojem, eventualno posebne vrste (npr. $\infty, -\infty$, NaN). Uz IEEE aritmetiku vezano je nekoliko internih varijabli:

- `realmin = 2.2251e-308` je najmanji broj u FP-sustavu brojeva.
- `realmax = 1.7977e+308` je najveći broj u FP-sustavu brojeva.
- `eps = 2.2204e-16` je relativna udaljenost izmedju dva susjedna FP-broja.
- `inf` znači $+\infty$, a nastaje ako egzaktni rezultat premašuje `realmax`. Slično vrijedi za -inf.
- NaN znači *Not a Number*, tj. neodredjenu numeričku vrijednost. Nastaje iz neodredjenih izraza $0/0$, $\infty - \infty$ i sl.

`NaN` je iznimno koristan objekt. Npr., budući da svaka operacija s `NaN` rezultira opet s `NaN`, korisno je na početku programa razna polja inicijalizirati `NaN`-ovima. Jedini način da vrijednost `NaN` nestane je da bude eksplicitno zamijenjena realnom vrijednošću. Takodjer, velika je korist od `NaN`-va kod crtanja, jer se takve vrijednosti ne crtaju, tj. automatski se preskaču.

Proizvoljan niz znakova okružen jednostrukim navodnicima, npr. `'pero'` predstavlja **tekstualni** tip podatka (engl. *character array*). Takve podatke Matlab tretira kao slovčane matrice, nad kojima je moguće vršiti sve smislene operacije, a koriste se kad god treba zadati ili prenijeti tekstualni sadržaj (npr. opis slike ili koordinatnih osi, odabir neke opcije pri pozivu funkcije i sl.).

2.4 Matrice, vektori i računske operacije

Kako je već rečeno, osnovni objekt u Matlabu je matrica, tj. pravokutna tablica brojeva koja se izravno zadaje kao niz brojeva u *uglatim zagradama*. Pri tom se retci razdvajaju s točka-zarezom, a elementi unutar retka bjelinom (ili zarezom). Npr:

```
>> A = [1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15]
A =
     1   2   3   4   5
     6   7   8   9  10
    11  12  13  14  15
```

zadaje matricu reda 3×5 od 3 retka i 5 stupaca, koja bi se u matematici pisala s:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{bmatrix}.$$

Snaga Matlaba nalazi se u velikim mogućnostima manipulacije matricama. Npr.:

• Izgradnja novih matrica izvlačenjem elemenata iz postojećih:

Element iz drugog retka i trećeg stupca je skalar, tj. matrica reda 1×1:

```
>> A(2,3)
ans =
   8
```

Elementi iz drugog retka, te petog i trećeg stupca čine redak-vektor, tj. matricu reda 1×2:

```
>> A(2,[5 3])
ans =
   10 8
```

Podmatrica sačinjena iz elemanata u presjeku prvog i drugog retka, te trećeg i četvrtog stupca je reda 2×2:

```
>> A([1 2],[3 4])
ans =
     3  4
     8  9
```

• Transponiranje:

```
>> B = A'
B =
     1   6  11
     2   7  12
     3   8  13
     4   9  14
     5  10  15
```

• Ljepljenje (konkatenacija) matrica i vektora se vrši na intuitivan način (kako to zamišlja mali Ivica):

```
>> a = [7 7 7 7 7]
a =
    7  7  7  7  7
>> [a; A]
ans =
    7   7   7   7   7
    1   2   3   4   5
    6   7   8   9  10
   11  12  13  14  15
>> b = [8 8 8]'
b =
    8
    8
    8
>> [A b]
ans =
    1   2   3   4   5   8
    6   7   8   9  10   8
   11  12  13  14  15   8
```

Vidimo da znak "`;`" znači "stavi ispod", a bjelina "stavi pored". Pokušajte isto napraviti u Fortranu. Ukoliko dimezije matrica ne omogućavaju matematički ispravno slaganje, Matlab će javiti pogrešku:

```
>> [b; A]
???  Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
```

Primijetimo internu varijablu `ans`. Ona uvijek sadrži rezultat zadnje izvršene operacije i može se ravnopravno koristiti u sljedećoj opearciji. Ako se rezultat ne imenuje izravno, ispisuje se preko varijable `ans`.

• Operator dvotočke ":" jedan je od najčešće korištenih. Njime se zadaje raspon indeksa:

```
>> [2:9]
ans =
    2  3  4  5  6  7  8  9
```

Malo općenitiji oblik je:

```
>> [2:3:19]
ans =
    2  5  8 11 14 17
```

u kojem srednji broj ima značenje koraka. Slijede primjeri s indeksiranjem redaka i stupaca matrica.

```
>> A(3,[2:4])
ans =
    12  13  14
```

tj. izvlače se elementi iz trećeg retka, te drugog do četvrtog stupca. Isto se može zadati s

```
>> A(3,2:4)
ans =
    12  13  14
```

Ako se stavi samo dvotočka, tj.

```
>> A(3,:)
ans =
    11  12  13  14  15
```

dobiju se *svi* (u ovom slučaju) stupci. Izraz `A(:,:)` isto je što i `A`. No `A(:)` daje *vektor-stupac* koji se sastoji od stupaca matrice `A` i koji u našem primjeru ima 15 elemenata. Inače elemente (2-dimenzionalne) matrice je moguće osim pomoću dvostrukog, dohvatiti i korištenjem jednostrukog indeksa koji napreduje odozgo prema dolje i s lijeva u desno *po stupcima*. Na primjer:

```
>> A(3,4)
ans =
   14
>> A(12)
ans =
   14
```

Osim što ukazuje na fortransko porijeklo i na prvi pogled djeluje kao "komplikacija", dualnost u indeksiranju je vrlo korisna i jedan je od primjera promišljenosti Matlaba.

Unutar pojedine matrice moguće je indeks zadnjeg retka ili stupca zamijeniti varijablom `end`. Npr.

```
>> A(:,2:end-1)
ans =
     2   3   4
     7   8   9
    12  13  14
```

čime smo izvukli sve retke matrice `A`, te stupce od drugog do pretposljednjeg. Uočimo da nije potrebno pamtiti koliko stupaca ima matrica `A`. To za nas radi Matlab.

• Matlab, kako smo već rekli ne zahtijeva izravno deklariranje varijabli. Štovoše, varijable tijekom rada, osim vrijednosti, mogu mijenjati i veličinu i tip. Ovo je posve protivno Fortranu i od korisnika s fortranskim iskustvom

zahtijeva izvjesnu prilagodbu u načinu razmišljanja. Na primjer, ako se zahvati element matrice izvan najvećeg retka ili stupca, Matlab će automatski matricu redimenzionirati, te napuniti nulama odgovarajuće elemente.

```
>> c=[1 2 3]
c =
    1  2  3
>> c(5)=-7
c =
    1  2  3  0  -7
>> c=[1 2 3;4 5 6]
c =
    1  2  3
    4  5  6
>> c(5,4)=-7
c =
    1  2  3   0
    4  5  6   0
    0  0  0   0
    0  0  0   0
    0  0  0  -7
```

Ipak, matricu koja će se računati postepeno u petlji, dobro je na početku zadati u konačnoj veličini. Time se izbjegava stalno redimenzioniranje i značajno ubrzava račun.

Želimo li sve brojeve u postojećoj matrici (npr. `c` zamijeniti jednim te istim brojem (npr. 7), nije dobro napisati `c = 7`. Time bi postojeću matricu `c` (ovdje reda 5×4) zamijenili skalarom 7 (tj. matricom reda 1×1). Umjesto toga treba koristiti:

```
>> c(:)  = 7
c =
    7  7  7  7
    7  7  7  7
    7  7  7  7
    7  7  7  7
    7  7  7  7
```

Navedimo još neke, vrlo korisne funkcije za rad s matricama:

- `size(A)`,

  za matricu A reda $m \times n$ vraća vektor `[m n]` njenih dimenzija.

- `length(A)`

  vraća veću od dvije dimenzije matrice A.

– `repmat(A,m,n)`

je matrica dobivena tako da se svaki element matrice od `m` redaka i `n` stupaca zamijeni matricom `A`. Na primjer

```
>> A = [1 2; 3 4]

A =
 1  2
 3  4

>> B = repmat(A,2,3)

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

>> repmat(NaN,2,6)

ans =
 NaN  NaN  NaN  NaN  NaN  NaN
 NaN  NaN  NaN  NaN  NaN  NaN
```

– `zeros(m,n)`

daje isto što i `repmat(0,m,n)`.

```
>> zeros(2,6)

ans =
 0  0  0  0  0  0
 0  0  0  0  0  0
```

– `ones(m,n)`

daje isto što i `repmat(1,m,n)`.

```
>> ones(2,6)

ans =
 1  1  1  1  1  1
 1  1  1  1  1  1
```

– `A1 = reshape(A,m1,n1)`

je iznimno upotrebljiva funkcija koja gradi novu matricu preslagivanjem po stupcima zadane, postojeće matrice. Naime, elementi matrice A se iscrpuljuju po stupcima i njima se puni matrica $A1$, takodjer po

stupcima. Pri tom je $A1$ reda $m1 \times n1$, a ukupan broj elemenata mora ostati nepromijenjen.

```
>> A = [1 2 3 4 5 6; 7 8 9 10 11 12]

A =
 1  2  3  4   5   6
 7  8  9  10  11  12

>> A1 = reshape(A,3,4)

A1 =
 1  8   4  11
 7  3  10   6
 2  9   5  12
```

Osnovne računske operacije prirodno se vrše nad matricama i vektorima. Pri tom razlikujemo operacije koje se vrše prema pravilima matrične algebre (kratko "u matričnom smislu"), te operacije koje se vrše po elementima (kratko "u smislu polja", engl. *array*). Oznake su sljedeće:

Operacija	U matričnom smislu	U smislu polja
zbrajanje	+	+
oduzimanje	-	-
moženje	*	.*
dijeljenje	/ ili \	./ ili .\
potenciranje	^	.^

Zbrajanje (oduzimanje) u matričnom smislu, istovjetno je zbrajanju (oduzimanju) po elementima.

```
>> a = [1 2 3], b = [4 5 6]
a =
    1  2  3
b =
    4  5  6
>> c = a-b
c =
    -3  -3  -3
```

Množenje u matričnom smislu je definirano ako se broj stupaca prve matrice podudara s brojem redaka druge matrice:

```
>> a = [1 2; 3 4; 5 6]
a =
    1  2
    3  4
    5  6
```

```
>> b = [1; 1]
b =
    1
    1
>> a*b
ans =
     3
     7
    11
>> b*a
???  Error using ==> *
Inner matrix dimensions must agree.
```

Matrično dijeljenje zapravo znači množenje inverznom matricom, odnosno rješavanje sustava linearnih jednadžbi. Obzirom da množenje matrica, za razliku od skalara nije komutativno, razlikujemo moženje inverznom matricom s lijeva i s desna.

Matlab	matematika	
	matrice	skalari
`A\B`	$A^{-1}B$	$\frac{B}{A}$
`A/B`	AB^{-1}	$\frac{A}{B}$

Potenciranje u smislu matrica je definirano za kvadratne matrice.

Operacije u smislu polja, definirane su na prirodan način za operande (matrice) istog reda, ili ako je jedna od njih skalar. Na primjer:

```
>> a = [1 2 3], b = [4 5 6]
a =
    1  2  3
b =
    4  5  6
>> c = a.*b
c =
    4  10  18
>> a./2
ans =
    0.5000  1.0000  1.500
>> a.\2
ans =
    2.0000  1.0000  0.6667
>> a.^2
ans =
```

```
    1   4   9
```
ali
```
>> a^2
???  Error using ==> ^
Matrix must be square.
```
rezultira greškom, budući da matrica `a` nije kvadratna.

U Matlab su kao interne ugradjene sve elementarne, te mnoge specijalne matematičke funkcije. Sve su definirane u punoj matematičkoj općenitosti u kompleksnoj ravnini, a primjenjene na matricu djeluju po elementima. Npr.
```
>> a = [1 2 3]
a =
    1   2   3
>> sin(a)
ans =
    0.8415   0.9093   0.1411
```

2.5 Logički i relacijski operatori

Osnovni logički, te relacijski operatori u Matlabu popisani su u sljedećoj tablici.

relacijski operatori		logički operatori	
`==`	jednakost	`&`	i
`~ =`	različitost	`|`	ili
`>`	veće od	`~`	negacija
`<`	manje od	`xor`	isključivo ili
`>=`	veće ili jednako	`all`	svi istiniti
`<=`	manje ili jednako	`any`	barem jedan istinit

Poput osnovnih računskih operacija, i oni su u Matlabu poopćeni na matrice, pri čemu se svaka numerička vrijednost različita od nule smatra *istinom*, dok je nula *laž*. Promotrimo logički izraz `x = (a<b)`, pri čemu su `a` i `b` matrice ili skalari. Vrijednost izraza `x` je opisana sljedećom tablicom:

`a`	`b`	`x`	
skalar	skalar	skalar	$x = \begin{cases} 1 & \text{ako je } a < b, \\ 0 & \text{inače.} \end{cases}$
matrica (reda $m \times n$)	skalar	matrica (reda $m \times n$)	$x(i,j) = \begin{cases} 1 & \text{ako je } a(i,j) < b, \\ 0 & \text{inače.} \end{cases}$
matrica (reda $m \times n$)	matrica (reda $m \times n$)	matrica (reda $m \times n$)	$x(i,j) = \begin{cases} 1 & \text{ako je } a(i,j) < b(i,j), \\ 0 & \text{inače.} \end{cases}$

Na primjer:

```
>> a = [1 5; 0 -3]
a =
    1   5
    0  -3
>> a>4
ans =
    0  1
    0  0
>> b = [0 -inf; 3 1]
b =
    0  -inf
    3    1
>> a>b
ans =
    1  1
    0  0
```

Ista pravila vrijede i za logičke operatore. Kako će kasnije biti vidljivo iz primjera, kombiniranje matrica s relacijskim i logičkim operatorima omogućava vrlo učinkovit rad s podacima. Ipak, postoji potreba za nekim posebnim logičkim funkcijama, čija imena obično počinju upitnom riječi `is`. Neke su navedene u sljedećoj tablici.

`isequal(A,B)`	logički skalar, i to 1 ako su A i B jednaki u matričnom smislu (tj. ako imaju isti red i iste elemente), a 0 inače.
`isfinite(A)`	logička matrica istog reda kao i A, sadrži 1 tamo gdje A sadrži konačnu, (tj. različitu od NaN ili inf) vrijednost, a 0 inače.
`isnan(A)`	logička matrica istog reda kao i A, sadrži 1 tamo gdje A sadrži NaN, a 0 inače.
`isinf(A)`	Slično kao gore.

Na primjer:

```
>> isequal([1 2; 3 4],'iviCA')
ans =
   0
```

Dakle, matrica $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ nije isto što i tekst 'iviCA'.

```
>> isnan([-2 2; NaN NaN])
ans =
    0  0
    1  1
```

Najuže povezana s logičkim i relacijskim operatorima je važna i često korištena funkcija `find`:

– `X = find(A)`

  daje *vektor indeksa* X, onih elemenata (logičkog) vektora A koji nisu nula, tj. koji su *istiniti*.

Naredba se koristi ako je potrebno iz vektora izdvojiti ili pak promijeniti sve elemente koji udovoljavaju nekom uvjetu. Izdvojimo na primjer sve elemente koji su izmedju -3 i 2:

```
>> A=[-10 0 -4 1 NaN Inf 1.5 1]
A =
   -10.0000 0 -4.0000 1.0000 NaN Inf 1.5000 1.0000
>> X = find(-3<=A & A<=2)
X =
   2 4 7 8
```

Obratite pažnju da je argument funkcije `find` logički vektor dobiven logičkim operatorom & iz dvaju logičkih vektora dobivenih, pak, relacijskim opratorom <=. Izdvojimo nadjene elemente u zaseban vektor:

```
>> A1 = A(X)
A1 =
   0 1.0000 1.5000 1.0000;
```

Zamijenimo nadjene elemente unutar `A` dvostrukim, negativnim vrijednostima:

```
>> A(X) = -2.*A(X)
A =
   -10 0 -4 -2 NaN Inf -3 -2
```

Sve rečeno vrijedi na isti način i kada se naredba primjeni na na 2-dimenzionalnu matricu, tj. opet se dobiva *vektor* indeksa onih elemenata matrice koji su različiti od nule, s tim da se koristi *jednodimenzionalno* indeksiranje, kako je već bilo objašnjeno. Na primjer:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
A =
    1  2  3
    4  5  6
    7  8  9
>> X = find(A==3 | A>7)
X =
    6
    7
    9
>> A1 = A(X)
A1 =
```

```
     8
     3
     9
>> A(X) = 999
A =
     1   2   999
     4   5    6
     7  999  999
```

2.6 Kontrola tijeka programa

Upravljanje tijekom programa vrši se u Matlabu istim naredbama kao i u (ne previše starom) Fortranu ili C-u. Spomenimo ukratko `if`, `for` i `while` naredbe. Prva od njih, tj. `if` prikazana shematski u raznim varijantama na donjoj slici, omogućava grananje u programu. Ovisno o uvjetima (<uvjet_1>, <uvjet_2>, itd.) izvršavaju se odgovarajući blokovi (skupine) naredbi (<naredbe_1>, <naredbe_2>, itd.), a ostali se preskaču. Uvjeti su najčešće logički skalari, no mogu biti i matrice. Napomenimo da Matlab ne poznaje naredbu bezuvjetnog skoka, goto.

```
if <uvjet>
      naredbe_1
else
      naredbe_2
end
```

ili

```
if <uvjet_1>
      naredbe_1
elseif <uvjet_2>
      naredbe_2
elseif <uvjet_3>
⋮
else
      naredbe
end
```

ili

```
if <uvjet>
      naredbe
end
```

Program(čić) na donjoj slici ispituje predznak varijable `x`. Za ispis se koristi naredba `disp`:

```
if x<0
      disp 'x je negativan'
elseif x==0
      disp 'x je nula'
else
      disp 'x je pozitivan'
end
```

Naredbe `for` i `while`, takozvane for i while petlje omogućavaju ponavljanje odredjenog bloka naredbi više puta. U for-petlji je broj ponavljanja poznat unaprijed, dok u while-petlji on ovisi o zadanom uvjetu. Opći oblik naredbi je prikazan na slici, dolje:

```
for <var>=<izraz>
      naredbe
end
```

```
while <uvjet>
      naredbe
end
```

Broj izvršavanja for-petlje zadan je brojem *stupaca* izraza <izraz> koji je najčešće redak-vektor. U i-tom prolazu varijabla <var> poprimi vrijednost i-tog stupca, te se izvrše naredbe izmedju `for` i `end`.

Kod while-petlje naredbe izmedju while i end se izvršavaju dok je ispunjen uvjet <uvjet>. Posljedično, da bi se dogodilo (barem) jedno izvršavanje, <uvjet> na početku mora biti ispunjen.

Po završetku petlji program se nastavlja izvršavati prvom naredbom iza odgovarajuće `end` naredbe. Prijevremeni izlazak iz petlje moguć je korištenjem naredbe `break`.

Dijelovi koda na donjoj slici prikazuju upotrebu for- i while-petlji.

```
for i1=[1:10]
     x(i1) = i1^2
end
```

```
x = 1;
while x>0
      xmin = x;
      x = x/2;
end
x
xmin
```

Osim što predstavlja korektan primjer for-petlje, predstavlja i primjer kako se for-petlja u Matlabu ne smije koristiti. Gornju petlju treba naime

zamijeniti potenciranjem po elementima, tj.

```
>> x = [1:10]^2
x =
    1   4   9  16  25  36  49  64  81  100
```

Primjer s while-petljom bi trebao dati `realmin` tj. najmanji pozitivni broj u FP-sustavu brojeva. U stavrnosti `xmin` će biti manji od `realmin`, no to su već finese IEEE aritmetike.

2.7 m-datoteke

M-datoteke su obične tekstualne datoteke koje sadrže Matlabove naredbe (programe), te imaju nastavak "m". Postoje dvije vrste m-datoteka: skripte i funkcije (u užem smislu). Zajedničko im je sljedeće:

- % je znak za komentar. Matlab ignorira sve što se u nekoj liniji nalazi iza znaka %.
- Prve linije u datoteci koje počinju s % sadrže opis programa. On je prekinut prvom linijom koja ne počinje s %, a ispisuje se na ekran naredbom `help` <ime_datoteke> (bez nastavka "m").
- Format linije je slobodan, pri čemu jedna linija može sadržavati više naredbi odvojenih zarezom ili točka-zarezom. Treba paziti da se rezultat svake naredbe koja ne završava tocka-zarezom odmah ispisuje na ekran - što se najčešće ne želi.
- ... je znak za nastavak linije.

Skripta se izvršava pozivom po imenu, bez prenošenja varijabli, a radi s varijablama iz (glavnog) radnog prostora. Može se donekle usporediti s glavnim programom u Fortranu. Za funkcije u užem smislu karakteristično je sljedeće:

- Prva naredba koja nije komentar mora biti deklaracija funkcije oblika:

  `[iz`$_1$`,iz`$_2$`,...,iz`$_n$`] = ime_funkcije(ul`$_1$`,ul`$_2$`,...,ul`$_1$`)`,

  pri čemu su s lijeve strane popisani izlazni, a s desne strane ulazni argumenti. (Ovdje su zarezi obavezni na obje strane, dok su kod poziva funkcije obavezni samo s desne strane.)
- Jedna datoteka sadrži samo jednu funkciju (postoji mala iznimka), pri čemu ime datoteke (bez nastavka) mora biti istovjetno imenu funkcije.
- Povratak u pozivni program ostvaruje se naredbom `return`.

- Funkcija se izvršava u svom *lokalnom* radnom prostoru. Sve varijable unutar funkcije su lokalne, tj. postoje samo unutar radnog prostora funkcije, koji pak postoji samo dok se funkcija izvršava.

- Veza s radnim prostorom pozivnog programa (funkcije) ili s osnovnim radnim prostorom (ako je skripta pozvala funkciju) ostvaruje se isključivo putem ulaznih, te izlaznih argumenata.

Iako se funkcije mogu usporediti s potprogramima u Fortranu, postoji bitna razlika u tretiranju argumenata - razlika koja u najvećoj mjeri odredjuje filozofiju obaju sustava. U Matlabu se, naime, prenošenje varijabli u funkciju, te iz funkcije obavlja "po vrijednosti" (engl. *by value"*), tj. kod poziva se ulazne varijable fizički kopiraju u radni prostor funkcije, a isto se dešava s izlaznim argumentima kod povratka. Funkcija, dakle, radi s kopijama varijabli, dok izvorne varijable ostaju netaknute. Posljedično, broj, kao ni tip, bilo ulaznih, bilo izlaznih argumenata funkcije u Matlabu nije fiksan, što značajno olakšava programiranje. Moguće je, na primjer, unutar funkcije ispitati broj i tip ulaznih argumenata te, ovisno o nalazu, izvršiti odredjeni postupak, što su već elementi objektnog programiranja. U Fortranu se prenose samo adrese varijabli, tj. pokazivači, a ne i same vrijednsti (engl. prenošenje "by reference"). Posljedično, i broj i tip argumanata moraju se podudarati i fiksni su, no zato se Fortranski programi izvode jako brzo.

Kao primjer, navedimo (unutarnju) Matlabovu funkciju `min`. Ako je ulazni argument vektor (stupac ili redak) funkcija vraća njegov minimalni element:

```
>> min([0 7 -4 2]')
ans =
   -4
```

Ako je ulazni argument matrica, funkcija `min` nalazi minimalni element u svakom stupcu i sve zajedno ih vraća kao jedan vektor-redak:

```
>> A = [1 2 3; 6 5 4; 9 7 8; 10 11 12]'
A =
    1   6   9   10
    2   5   7   11
    3   4   8   12
>> min(A)
ans =
   1 4 7 10
```

No, funkcija `min` može imati i dva izlazna argumenta, u kojem slučaju drugi argument vraća indekse nadjenih minimalnih vrijednosti:

```
>> [M indM] = min(A)
```

```
M =
   1 4 7 10
indM =
   1 3 2 1
```

2.8 Grafički podsustav

Osim matričnog koncepta i pripadnog *engine*-a, druga uporišna točka Matlaba je izvrstan grafički podsustav. Osnovne grafičke funkcije, koje su odmah dostupne, omogućavaju relativno lako crtanje različitih slikovnih prikaza. S druge strane, moguće je i detaljno upravljanje svim aspektima prikaza (npr. veličina i položaj slike na papiru, položaj tikova na osima, položaj promatrača, polažaj i vrsta izvora svjetla, itd., itd.)

Ovdje ćemo se ograničiti na nešto detaljniji opis dviju osnovnih i najkorisnijih funkcija: `plot` i `contour`, od kojih prva služi za crtanje 2D-krivulja, a druga za crtanje izolinija. Postoji, dakako, i cijeli niz funkcija za 3-dimenzionalno crtanje, no s takvim prikazima valja biti suzdržan, jer su najčešce manje informativni od prethodna dva. Na kraju ćemo ukratko izložiti osnovnu strukturu i unutarnji ustroj grafičkog podsustava, poznavanje kojeg nije prijeko potrebno, ali je vrlo korisno za ugodan rad.

Naredba **plot** se javlja u više varijanti:

- `plot(x,y)`,

  pri čemu su `x` i `y` vektori (stupci ili retci) duljine m, spaja linijom točke $(x(i), y(i)), i = 1, \ldots, m$.

- `plot(y)`,

  pri čemu je `y` vektor spaja linijom točke $(i, y(i))$, tj. crta y naspram svojih indeksa.

- `plot(x,y,'`*opis*`')`,

  radi isto kao gore, s tim da je '*opis*' niz znakova koji odredjuju vrstu i boju linije, te upotrijebljene simbole. Na primjer `plot(x,y,'--r*')` znači da će linija biti isprekidana, crvena, te će svaka točka $(x(i), y(i))$ biti označena zvjezdicom.

- `plot(x1,y1,'`*opis1*`', x2,y2,'`*opis2*`')`,

  crta dvije krivulje na istom grafu, prema zadanim opisima. Analogno se može crtati više krivulja.

Značenje pojedinih slova u opisu krivulje vidi se iz donje tablice.

Vrsta linije	
-	puna
--	crtkana
:	točkasta
-.	crtkano-točkasta

Boja	
r	crvena (red)
g	zelena (green)
b	plava (blue)
c	cijan (cyan)
m	magenta (magenta)
y	žuta (yellow)
k	crna (black)
w	bijela (white)

Simbol	
·	točka
o	krug
x	znak x
+	plus
*	zvjezda
s	kvadrat
d	dijamant
^	trokut/gore
v	trokut/dolje
>	trokut/u desno
<	trokut/u lijevo

Naredba `plot` prima i matrične argumente.

– `plot(x,Y)`,

pri čemu je `x` vektor–stupac duljine m, a `Y` matrica reda $m \times n$, crta n krivulja, kombinirajući `x` sa stupcima matrice `Y`. Krivulje se crtaju u raznim bojama, ciklički.

– `plot(Y)`,

pri čemu je `Y` matrica, crta stupce od `Y` naspram pripadnog indeksa.

– `plot(X,Y)`,

pri čemu su `X` i `Y` matrice reda $m \times n$, crta n krivulja, kombinirajući odgovarajuće stupce zadanih matrica.

Općenito, ako je argument kompleksna veličina, plot naredba crta samo realni dio. Imaginarni dio se, dakle, ignorira. Slučaj kada se zada samo jedan kompleksni argument je iznimka:

– `plot(Z)`,

pri čemu je `Z` kompleksan vektor (ili matrica), radi isto što i `plot(real(Z), imag(Z))`.

Osim funkcije `plot` postoje i funkcije `semilogy`, `semilogx`, `loglog` za crtanje u logaritamskoj skali, te mnoge druge funkcije za crtanje posebnih vrsta dvodimenzionalnih grafova, npr. `polar`, `bar`, `hist`, `quiver`, `compass`.

Navedimo neke jednostavne funkcije za opisivanje, te ako je potrebno, i doradu postojeće slike/grafa:

– `title('tekst_naslova')`

  stavlja zadani tekst u naslov iznad grafa.

– `xlabel('opis_osi')`
– `ylabel('opis_osi')`

  stavlja zadani tekst uz x- odnosno y-os.

– `text(x,y,'neki_tekst')`

  ispisuje zadani tekst s početkom u točki (x, y).

– `gtext('neki_tekst')`

  ispisuje zadani tekst na mjestu koje se zadaje klikom miša.

– `axis([xmin,xmax,ymin,ymax])`

  zadaje raspon x-, te y-osi.

– `axis equal`

  izjednačava broj podatkovnih jedinica po jedinici duljine za x- i y-os (tako da se kružnica crta bez deformacije).

– `grid on`
– `grid off`

  postavlja ili uklanja mrežu ortogonalnih linija preko grafa (radi lakšeg očitavanja).

Naknadnim pozivanjem neke grafičke funkcije (poput `plot` ili `contour`) postojeća slika će, u pravilu, biti obrisana i zamijenjena novom. Ako se, pak, želi crtati "preko" postojeće slike, treba koristiti naredbu `hold on`. Suprotan učinak ima naredba `hold off`.

Pomoću naredbe `subplot` lako se crta više grafova (*axis*) na jednoj te istoj slici (*figure*). Preciznije, `subplot(m,n,i)` će podijeliti sliku na $m \times n$ pravokutnika i smjestiti sljedeći graf u i-ti po redu pravokutnik, pri čemu se broji s lijeva u desno i odozgo na dolje.

Funkcija **contour**, koja crta izolinije funkcija dvije varijable, takodjer se javlja u više različitih oblika.

– `contour(x,y,Z)`,

  pri čemu je `x` vektor-redak duljine m, `y` vektor-stupac duljine n, a `Z` matrica reda $m \times n$ crta izolinije funkcije $z(x_i, y_j) = z_{ij}$ zadane matricom `Z`.

– `contour(Z)`,

  crta izolinije funkcije $z(i,j) = z_{ij}$.

– `contour(x,y,Z,v)`,

  pri čemu su `x`, `y`, te `Z` kao gore, a `v` je vektor-redak, crta izolinije zadane vrijednostima u `v`.

Oznake izolinija (labele) neće se automatski nacrtati, već treba upotrijebiti funkciju `clabel`. Na primjer:

```
[C h] = contour(x,y,Z,v)
clabel(C,h,v(1:2:end)
```

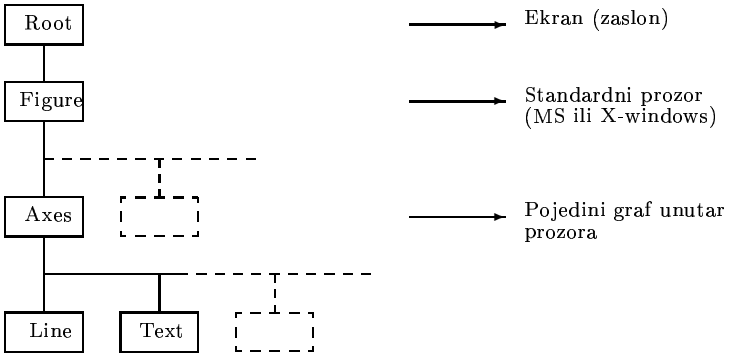
će nacrtati izolinije od `Z` zadane vektorom `v`, i potom označiti svaku drugu izoliniju.

Posve analogno radi i funkcija `contourf`, s tim da se umjesto izvlačenja samih izolinija odgovarajućim bojama ispune prostori medju njima.

Za potpuno razumijevanje gornjeg primjera (s `clabel`), a i šire, potrebno je malo zaviriti ispod površine. Grafički podsustav Matlaba organiziran je hijerarhijski u obliku stabla, prema donjem dijagramu. Glavni (najstariji) objekt je korijen (engl. *root*) koji predstavlja zaslon (ekran) računala. Njegova djeca su objekti tipa "Figure", tj. pojedini prozori koje Matlab otvara za prikaz grafičkih sadržaja. Svaki *figure* u pravilu sadrži više različitih tipova djece, od kojih je najvažniji objekt "axes", koji predstavlja jedan zaseban graf, odredjen trima osima, x, y i z. Svaki *axes* u pravilu posjeduje više djece raznih tipova od kojih su ponajvažniji objekti tipa "line" i "text". U svakom trenutku, i za svaki tip zna se koji je objekt aktivan. Tako će, npr. funkcija plot uzeti aktivni *figure* i, unutar njega, aktivni *axes*, te mu dodati (nacrtati) novo dijete, objekt tipa *line*. Ako u trenutku poziva ne postoji objekt tipa *figure*, bit će kreiran bez posebnog zahtjeva, a isto vrijedi i ako postojeći *figure* nema djece tipa *axes*.

Svaki objekt ovisno o tipu posjeduje niz svojstava (atributa) čije vrijednosti detaljno i potpuno odredjuju što će se i kako nacrtati. Grafičkom bjektu, odnosno njegovim atributima, pristupa se pomoću ručke (engl. *handle*), čija je vrijednost realni broj koji jednoznačno identificira zadani objekt. Obično se ručki pridjeli simboličko ime. Na primjer, naredba `sl1 = figure` otvara novi prozor (objekt tipa *figure*) s ručkom `sl1`, koji ujedno postaje aktivan. Naredba, pak, `figure(sl1)` će učiniti traženi prozor aktivnim i izbaciti ga u prvi plan na zaslonu.

Primitivne naredbe (naredbe niže razine) za upravljenje atributima grafičkih objekata su `set` i `get`:



Slika 1: Djelomična struktura grafičkog podsustava

- `set(h,'svojstvo1',vrijednost1)`,

  pri čemu je h ručka (*handle*), postavlja `svojstvo1` traženog objekta na `vrijednost1`. Ime svojstva se stavlja u navodne znakove, a vrijednost može može biti tekstualna, poput `'on'` ili `'off'`, a može biti i broj ili vektor brojeva, ovisno o traženom svojstvu.

- `set(h,'svojstvo1','svojstvo2',...)`,

  ispisuje sve moguće vrijednosti traženih svojstava.

- `set(h)`,

  ispisuje sve atribute (svojstva) objekta `h` s njihovim mogućim vrijednostima.

- `get(h,'svojstvo1','svojstvo2',...)`,

  ispisuje trenutne vrijednosti `svojstva1`, `svojstva2`, ... traženog objekta `h`.

- `get(h)`,

  ispisuje sve atribute (svojstva) objekta `h` s njihovim trenutnim vrijednostima.

Postoje tri simbolička imena povezana s ručkama trenutno aktivnih grafičkih objekata:

ime	dolazi od (engl.)	objekt na koji pokazuje
`gcf`	*get current figure*	trenutno aktivna slika
`gca`	*get current axis*	trenutno aktivan graf
`gco`	*get current object*	zadnji objekt na kojeg je kliknuto mišem

Svrha im je omogućiti jednostavan pristup grafičkom objektu, bez da posebno imenujemo ili tražimo odgovarajuću ručku.

Ispis slike ne papir moguć je, osim putem padajućeg menu-a, i naredbom print. Na primjer,

- `print`

  će ispisati trenutno aktivnu sliku na standardni (engl. *default*) pisač.

- `print -deps2c 'ime_datoteke.eps'`

  će trenutno aktivnu sliku zapisati u zadanu datoteku u Postscript 2 formatu u boji.

Na kraju navodmo jedan primjer u kojem se crtaju dva grafa (*axes*), na istoj slici (*figure*). Prvi zauzima gornju, lijevu četvrtinu slike, a sadrži graf funkcije

$$y(x) = \sin(2\pi x) \cdot \exp(-x/2), \qquad x \in [0, 2],$$

punom linijom, te funkcije $2-y(x)$ iscrtan simbolima "+". Drugi graf zauzima donju polovinu slike, a sadrži izolinije funkcije

$$Z(x, y) = \exp\left(x^2 + y^2\right)/2 \cdot \sin(2\pi x) \cdot \cos(2\pi x) zax, y \in [0, 2].$$

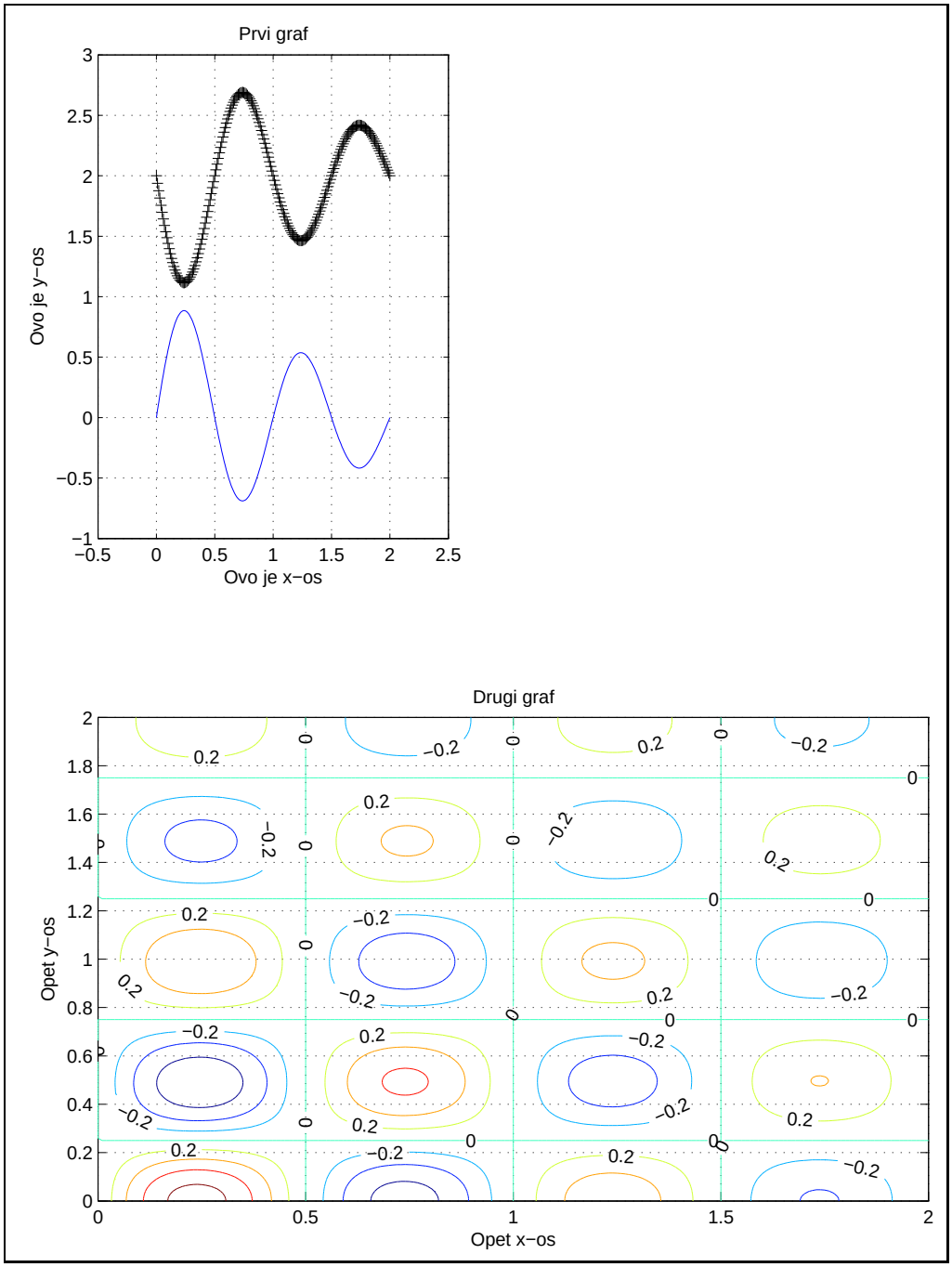
Naredba `meshgrid` iz vektora `x` i `y` priprema matrice `X` i `Y`, pogodne za računanje funkcije (matrice `Z`) korištenjem računskih operacija u smislu polja. Na taj način se funkcija $Z(x, y)$ za sve parove (x, y) izračuna jednom jedinom naredbom, bez korištenja `for` petlje.

```
% -- zadaje vektor-stupac x od 0 do 2 s korakom 0.01
    x = [0:1/100:2]';
% -- racuna vektor-stupac y
    y = sin(2*pi*x).*exp(-x/2);
% -- priprema crtanje u gornjoj lijevoj cetvrtini
    subplot(2,2,1);
% -- crta prvu krivulju
    plot(x,y);
% -- cuva prvu krivulju i crta drugu, crnom bojom i simbolom +
    hold on; plot(x,2-y,'k+');
% -- oznacava osi
    xlabel('Ovo je x-os'); ylabel('Ovo je y-os');
% -- naslov
    title('Prvi graf');
% -- crta mrezu
    grid on;
% -- eksplicitno postavlja raspon x-osi (na aktivnom grafu)
    set(gca,'XLim',[-0.5 2.5]);

% -- priprema racunanje funkcije dvije varijable
    x1 = x; y1 = x1;
    [X Y] = meshgrid(x1,y1);
    Z = exp(-sqrt(X.^2+Y.^2)/2).* sin(2*pi*X).*cos(2*pi*Y);
% -- priprema crtanje u donjoj polovici slike
    subplot(2,1,2);
% -- crta izolinije i priprema labele
    [C h] = contour(X,Y,Z);
% -- ispisuje trazene labele
    clabel(C,h,[-0.2 0 0.2]);
% -- oznacava osi, itd.
    xlabel('Opet x-os'); ylabel('Opet y-os');
    title('Drugi graf');
    grid on;
% -- postavlja 'tikove' na x-osi
    set(gca,'XTick',[0 0.5 1 1.5 2]);

% -- maksimalno razvlaci sliku na papiru u 'portrait' orjentaciji
    orient tall
% -- ispis u datoteku u postscript formatu
    print -deps2c uvod01.eps
```

Rezultat je sljedeća slika (prenesena ovdje iz Postscripta).



2.9 Razno

Na kraju navodimo neke korisne činjenice koje nisu prethodno iznešene ili su razasute po tekstu.

- Pregled korištenja interpunkcijskih znakova u Matlabu.
  - **Zarez** "," razdvaja naredbe u istom retku, ali ne sprječava ispis usputnih rezultata na ekran.
  - **Zarez** "," razdvaja argumente funkcija.
  - **Zarez** "," razdvaja elemente vektora i matrica, no za tu svrhu bolje je koristiti bjelinu.
  - **Bjelina** " " razdvaja elemente unutar vektora i matrica.
  - **Točka-zarez** ";" na kraju naredbe sprječava ispis na ekran.
  - **Točka-zarez** ";" unutar matrica označava prelazak u novi red.
  - **Dvotočka** ":" označava raspon vrijednosti, ili pak sve stupce ili retke matrice.
- Matlab koristi više raznih vrsta datoteka i one se, kako je uobičajeno, prepoznaju po nastavku unutar imena. Najčešći tipovi su:
  - ime.m

    je tekstualna datoteka koja sadrži program pisan u Matlabu.
  - ime.mat

    je binarna datoteka, tj. standardni Matlabov format za spremanje varijabli i njihovih vrijednosti. Takve datoteke se direktno učitavaju naredbom `load ime.mat`.
  - ime.fig

    je binarna datoteka, tj. standardni format Matlaba za spremanje slika. Učitava se naredbom `open ime.fig`.
- Izvodjenje svake naredbe/funkcije unutar Matlaba, u svakom se trenutku može prekinuti s ctrl-C (tj. tako da se istovremeno stisnu tipke Control i C).
- Korisno je imati na umu da Matlab u pravilu poštuje konvenciju rada po stupcima (što dolazi iz matematike i činjenice da se piše s lijeva u desno), pa kad god je to moguće, podatke treba organizirati u stupce.