Sveučilište u Rijeci

Odjel za informatiku

Kratki uvod u MATLAB

Edvin Močibob

edvin.mocibob@gmail.com

5. studenog 2015.

Sadržaj

Uvod

MATLAB (***mat****rix* ***lab****oratory*) označava programski jezik i interaktivno okruženje za programske izračune. Popularan je alat u matematici, inženjerstvu i znanosti. Osim u akademskom okruženju ima široku primjenu i u industriji. Podržan je za sve popularne operacijske sustave: Microsoft Windows, GNU/Linux i Mac OS X. Više informacija dostupno je na Wikipediji[1] i službenoj web stranici[2].

U nastavku slijedi kratki uvod u sintaksu MATLAB-a prezentiran (većinom) kroz primjere[3]. MATLAB ima kvalitetnu dokumentaciju sa primjerima i preporučeno je njezino korištenje uz ovaj dokument. Blokovi koda se neće opširno objašnjavati te pretpostavlja se da čitatelj ima neko bazično znanje iz programiranja.

Kontaktiranje autora u vezi grešaka te unaprjeđenja sadržaja dokumenta moguće je preko email adrese na naslovnoj stranici. LaTeX kod kojim je generiran ovaj dokument trenutno se nalazi na `https://github.com/emocibob/Kratki-uvod-u-MATLAB` pod MIT licencom.

[1] `https://en.wikipedia.org/wiki/MATLAB`

[2] `http://www.mathworks.com/products/matlab/`

[3] Prezentirani kod testiran je u MATLAB verziji R2014a na Windows 7 operacijskom sustavu.

1 Osnove

Jednolinijski komentari počinju sa `%`. Izlaz naredbe neće se prikazati ako naredba završi sa `;`.

```
x = 5; % inicijaliziraj varijablu x na 5
```

Brojeve i vrijednosti varijabla možemo ispisati u komandnoj liniji samim njihovim navođenjem.

```
10 % ispiši broj 10
x  % ispiši vrijednost spremljenu u x
```

Slijedi generiranje vektora koji sadrži 4 vrijednosti od 1 do 10 (granice uključne). Sve susjedne vrijednosti su međusobno jednako udaljene.

```
linspace(1, 10, 4) % rezultat sadrži 1, 4, 7 i 10
```

Isti rezultat dobijemo sljedećom naredbom. Ovdje ne odredimo eksplicitno broj elementa nego inkrement[4] za svaki sljedeći element – u ovom slučaju 3.

```
[1:3:10]
```

Slijede primjeri dohvaćanja elemenata vektora `v` korištenjem indeksa. Valja zapamtiti da indeksi elementa počinju od 1.

```
v(1)        % prvi element
v(1:10)     % prvih 10 elemenata
v(1:2:50)   % svaki drugi element, od 1. do 50.
v([9 9 9]) % tri puta vrati 9. element
```

U sljedećem kodu potencira se broj i vektor. Primijetimo `.` tj. operator točke. Ako je on ispred nekog drugog operatora, drugi operator će izvršiti izračun element po element (tzv. *pointwise* operacija[5]).

```
a = 5
b = [1, 2, 3, 4]
a .^ 3 % 125
b .^ 3 % izlazni vektor sadrži 1, 8, 27 i 64
```

MATLAB će brojeve sa puno znamenki zapisati u znanstvenom formatu. U primjeru je tim formatom spremljen broj 2.1×10^{-8}.

```
k = 2.1e-8
```

Duže naredbe je zbog preglednosti moguće napisati u više linija sa tri točke.

```
n = 1 - 1/12 + 1/123 - 1/1234 + 1/12345 - 1/123456 ...
    + 1/1234567 - 1/12345678 + 1/123456789
```

Relacijski operatori u MATLAB-u su sljedeći: `==` je jednako, `<` manje, `>` veće, `<=` manje ili jednako, `>=` veće ili jednako, `~=` različito od.

[4] Ako je inkrement 1 možemo izostaviti drugi argument, tj. pišemo `[a:b]` umjesto `[a:1:b]`.

[5] Tako nam je, na primjer, bitno koristimo li `.*` ili samo `*` kod množenja matrica. Primjeri u sekciji Matrice.

Modulo operator[6] implementiran je funkcijom `mod`.

```
ostatak = mod(29, 3) % daje 2
```

Dolje je naveden primjer[7] za MATLAB funkcije `any` (da li je ikoji element različit od 0) i `all` (da li su svi elementi različiti od 0).

```
v = [1.5 0.9 0.7 0.12 0.1 0.05];
any(v > 1) % ikoji element veći od 1? true
all(v > 1) % svi elementi veći od 1? false
```

Za logičke operatore koriste se `&` (logičko i), `|` (logičko ili) i `~` (komplement).

2 Dokumentacija i radni prostor

Naredba `lookfor` služi za pretraživanje funkcija iz komandne linije. Pretraživanje može potrajati pošto MATLAB ima bogatu biblioteku funkcija.

```
lookfor label
```

Pristup dokumentaciji određene funkcije može se ostvariti iz komandne linije sa `help`. Nakon naredbe slijedi ime funkcije za koju tražimo dokumentaciju.

```
help doc
```

Za bogatiji prikaz dokumentacije određene funkcije koristi se `doc`. Osim naredbom `doc`, novi prozor sa dokumentacijom možemo otvoriti tako da kliknemo na ime funkcije (da li u komandnoj liniji ili u MATLAB skripti) nakon čega odaberemo `F1` na tipkovnici.

```
doc help
```

MATLAB radni prostor (*workspace*) čine trenutno spremljene varijable. Samim izvršavanjem koda MATLAB automatski sprema imena[8] i pripadne vrijednosti varijabli.

Cijeli radni prostor može se obrisati narednom `clear` (bez argumenata). Pojedinačne varijable brišemo tako ih navodimo nakon `clear`.

```
clear a b
```

Slijedi primjer spremanja radnog prostora koristeći `save`. Radni prostor biti će spremljen u datoteci pod imenom `vjezba1_data.mat`.

```
save('vjezba1_data')
```

Sadržaj komandne linije brišemo naredbom `clc`.

[6] Ostatak cjelobrojnog dijeljenja.

[7] Argument za obje funkcije, tj. rezultat `v > 1`, je vektor sa *true*/*false* (1/0) vrijednostima za pojedine elemente `v`.

[8] Varijabla `ans` sadrži vrijednost zadnjeg izračuna koji nije spremljen u nijednu varijablu.

3 Matrice

Generiranje 2×3 matrice koja sadrži samo jedinice.

```
ones(2, 3)
```

Inicijalizacija matrice `A` dimenzija 3×4.

```
A = [1 2 3 4; 10 20 30 40; 100 200 300 400]
```

Transponiranje `A`.

```
A'
```

Zbroj elemenata `A` po stupcima.

```
sum(A) % izlaz 1x4 matrica
```

Zbroj svih elemenata.

```
sum(sum(A))
```

Najveći element iz `A`.

```
max(max(A))
```

Najmanji elementi iz pojedinih redaka.

```
min(A, [], 2) % min traži po 2. dimenziji - recima
```

Matrica tipa *logical* koja prikazuje koji su elementi `A` veći od 30. Izlaz ima iste dimenzije kao `A`, na mjesta elemenata većih od 30 nalazi se 1 (*true*) a na ostalim 0 (*false*).

```
A > 30
```

Broj elemenata `A` većih od 30.

```
sum(sum(A > 30))
```

Vrati matricu `P` (tipa *logical*) koja sadrži na kojim mjestima u `A` se nalaze parni brojevi. Nakon toga ispiši te parne brojeve iz A.

```
P = mod(A, 2) == 0;
A(P)
```

Matrica `B` koja sadrži elemente `A` povećane za 10.

```
B = A + 10
```

Pojedinim elementima `A` možemo pristupiti sa `A(i, j)` gdje je `i` redak a `j` stupac elementa. Za raspone koristimo `:`. Prisjetimo se, indeksiranje počinje od 1.

```
A(2, 1)   % element u drugom retku i prvom stupcu
A(2, :)   % drugi redak
A(:, 3:4) % treći i četvrti stupac
```

Uvećaj treći stupac 10 puta.

```
A(:, 3) = A(:, 3) * 10;
```

Nova matrica kojoj je osim elementa `A` (dimenzije 3×4) dodan još jedan redak. Moguće koristiti i `vertcat`[9].

```
[A; -1 -2 -3 -4]
```

Nova matrica kojoj je osim elementa `A` (dimenzije 3×4) dodan još jedan stupac. Moguće koristiti i `horzcat`[10].

```
[A [5; 55; 555]]
```

Jedinična matrica[11] `B` istih dimenzija kao `A`.

```
B = eye(size(A))
```

Inicijaliziraj i zatim matrično pomnoži matrice `C` i `D`.

```
C = [1 2; 3 4];
D = [10 20; 30 40];
C * D
```

Množenje matrica `C` i `D` po elementima.

```
C .* D
```

Primjer pretvaranja matrice u vektor.

```
A(:)
```

4 Polja ćelija

Matrice ne mogu sadržavati istovremeno različite tipove podataka (npr. brojeve i tekst). Kada imamo potrebu za tim možemo koristiti polja ćelija (*cell arrays*).

Inicijaliziacija se može izvršiti pomoću vitičastih zagrada. Što se dimenzija tiče, svi reci moraju imati isti broj elemenata. U primjeru vidimo da polja ćelija mogu sadržavati druga polja ćelija.

```
C = {'Tekst', 2, {'XYZ', 111}; 4, 5, 10}
```

Referenciranje ćelija moguće je pomoću običnih zagrada (gdje se pristupa ćelijama) ili pomoću vitičastih zagrada (gdje se pristupa sadržaju ćelija).

U primjeru prvo pristupamo drugom retku polja `C` i zatim pristupamo vrijednostima istog retka.

```
C(2, :)
C{2, :}
```

[9] `vertcat(A, [-1 -2 -3 -4])`
[10] `horzcat(A, [5; 55; 555])`
[11] Sadrži jedinice na dijagonali.

Prazna ćelija dimenzija 0×0 može se inicijalizirati na sljedeći način.

```
C2 = {}
```

Slijedi primjer u kojem se dodaje element u postojeće polje ćelija. Dimenzije polja će se automatski povećati ako se referencira ćelija izvan postojećih dimenzija.

```
C2{4, 4} = 9
```

Samo povećanje dimenzija bez dodavanja ikakve nove vrijednosti vrši se na sljedeći način.

```
C2{5, 6} = []
```

Naredba `cell2mat` se koristi za pretvaranje polja ćelija (ili samo djela polja ćelija) u obični vektor tj. matricu. Uvjet je da svi elementi koji se pretvaraju budu istog tipa.

Sljedeća naredba pretvara drugi redak polja ćelija `C` u vektor[12].

```
cell2mat(C(2, :))
```

5 Grananja i petlje

Slijedi primjer sa sintaksom za `if` grananje.

```
n = 9;
if mod(n, 2) == 0
    disp('Broj paran.')
else
    disp('Broj neparan.')
end
```

Ispiši `for` petljom brojeve od 1 do 10 (granice uključne).

```
for i = 1:10
    i
end
```

Ista stvar u `while` petlji.

```
i = 1;
while i <= 10
    i
    i = i + 1;
end
```

Ispiši `for` petljom svaki drugi broj od 1 do 10 (tj. ispiši 1, 3, 5, 7 i 9).

```
for i = 1:2:10
    i
end
```

[12] Isti efekt postigla bi naredba `[C{2, :}]`.

Primjer prolaska kroz vektor `for` petljom.

```
for el = [5 2 36 1 9]
   el
end
```

Sljedećim blokom koda[13] inicijalizira se matrica `A` dimenzija 3×4 kojoj se zatim petljom svaki element poveća za 10.

```
A = [1 2 3 4; 10 20 30 40; 100 200 300 400]
for i = 1:3
    for j = 1:4
        A(i, j) = A(i, j) + 10;
    end
end
```

6 Pseudo-slučajni brojevi

Generiraj broj između 0 i 1.

```
rand
```

Generiraj 2×2 matricu sa slučajnim vrijednostima između 0 i 1.

```
rand(2, 2)
```

Generiraj cijeli broj u rasponu od 10 do 20 (granice uključne).

```
randi([10, 20], 1)
```

Napravi matricu `A` dimenzija 5×5 koja sadrži slučajne cijele brojeve u rasponu od -100 do 100.

```
A = randi([-100, 100], 5, 5)
```

Ako želimo da funkcije koje generiraju slučajne vrijednosti daju uvijek isti rezultat potrebno je postaviti konstantan *seed* (nenegativni cijeli broj) za generator slučajnih vrijednosti. To činimo pomoću `rng`.

```
rng(999)
```

Recimo da imamo 4×2 matricu `X` kojoj želimo slučajno ispremiješati retke. Prvo bi sa `randperm` generirali indekse (tj. redoslijed) redaka i zatim bi `X` proslijedili te indekse.

```
X = [121 11; 144 13; 169 13; 196 14]
idx = randperm(length(X))
X(idx, :)
```

[13] Primjer služi kao ilustracija ugnježđivanja petlji. Ako bi željeli svaki element `A` povećati za 10, to bi učinili jednostavno sa `A = A + 10`.

7 Stringovi i ispis teksta

Inicijaliziraj novi string.

```
s = 'Neki tekst'
```

Vrati duljinu stringa.

```
length(s)
```

Provjeri ako je `s` string.

```
isstr(s)
```

Pretvori broj u string.

```
num2str(125)
```

String u broj.

```
x = str2num('521')
```

Konkatenacija stringova pomoću uglatih zagrada.

```
tekst = [s, ' i na primjer broj ', num2str(125), '.']
```

Stringove možemo ispisivati sa `disp`. Istu funkciju možemo koristiti i na brojevima.

```
disp(tekst)
disp(7)
```

Za ispise gdje se kombiniraju različiti formati (stringovi, cijeli i decimalni brojevi itd.) može se koristiti `fprintf`.

```
fprintf('Umjesto %f koristimo %d\n', 5.67, 5)
fprintf('String od prije glasi: %s\n', tekst)
```

Za ispis znaka `%` u `fprintf` potrebno ga je navesti dva puta.

8 Korisnički definirane funkcije

Korisnički definirane funkcije spremaju se u datoteke sa nastavkom `.m`.
Slijedi primjer funkcije koje ne vraća vrijednost.

```
function parnost(n)
if mod(n, 2) == 0
    disp('Broj paran.')
else
    disp('Broj neparan.')
end
end
```

Primjer kada funkcija vraća vrijednost. Konkretno, vrijednost koja se vraća je `b`.

```
function b = kvadriraj(a)
b = a * a;
end
```

Funkcije mogu odjednom vraćati više vrijednosti.

```
function [b, c] = kvadrat_i_kub(a)
b = a * a;
c = b * a;
end
```

Prethodno navedenu funkciju bi pozivali na primjer ovako.

```
[kv, kub] = kvadrat_i_kub(11)
```

9 Korisnički unos

Korisnika se može tražiti unos na sljedeći način. U primjeru se unos sprema u `x`.

```
x = input('Unesite broj: ')
```

Složeniji upit za unos u kojem ispisujemo neke varijable možemo postići na sljedeći način.

```
fprintf('Nova vrijednost (prijašnja %d): ', x)
x = input('')
```

Osim brojčanih moguć je unos i tekstualnih vrijednosti.

```
odg = input('Unesite ime: ', 's')
```

10 Grafovi

Ovako bi nacrtali funkciju sinus na intervalu od -5 do 5. Ako imamo samo jedan graf `figure` nije obavezan. Rezultat bi trebao izgledati kao slika 1 u sekciji Prilozi.

```
x = linspace(-5, 5, 101);
figure % otvara novi prozor za graf
plot(x, sin(x))
```

Sljedeći kod pokazuje kako učiniti da se prozor sa grafom ne otvori. Dodatno, graf se spremi kao slika pod imenom `sinus_graf.png`.

```
x = linspace(-5, 5, 101);
fig = figure('visible', 'off');
plot(x, sin(x))
saveas(fig, 'sinus_graf', 'png')
```

Na jednom grafu možemo imati više funkcija odjednom. To radimo tako da jednostavno navodimo argumente svake funkcije jedan za drugim. U donjem primjeru korišteni su i argumenti za stiliziranje linija (`g` i `b` su boje, a `--` i `:` vrste linija). Rezultat slika 2.

```
x = linspace(-5, 5, 101);
figure
plot(x, sin(x), 'g--', x, cos(x), 'b:')
```

Isti rezultat (slika 2) moguće je dobiti korištenjem naredbe `hold`.

```
x = linspace(-5, 5, 101);
figure
plot(x, sin(x), 'g--')
hold on
plot(x, cos(x), 'b:')
```

Slijedi primjer sa detaljnijim stiliziranjem linije i točaka grafa. Izlaz koda u slici 3.

```
x = linspace(-5, 5, 21);
y = x .* x;
figure
plot(x, y, '--go', ... % izgled linije i oblik točaka
    'LineWidth', 2, ... % debljina linije
    'MarkerSize', 6, ... % veličina točaka
    'MarkerEdgeColor', 'r', ... % boja ruba točaka
    'MarkerFaceColor', 'w') % unutrašnja boja točaka
```

Naslov grafa, oznake osi i legendu možemo dodati sljedećim kodom. Rezultat slika 4.

```
x = linspace(-5, 5, 101);
figure
plot(x, sin(x), 'g--', x, cos(x), 'b:')
title('Vrijednost sin(x) i cos(x) na [-5, 5]')
xlabel('Os x')
ylabel('Vrijednost funkcije')
legend('sin(x)', 'cos(x)')
```

U sljedećem primjeru prikazan je stupčasti graf koji vizualizira neke *random* podatke. Naslov grafa, oznake osi itd. bi se po potrebi postavile kao za `plot`. Izlaz slika 5.

```
rng(999) % postavljenje seed-a za random brojeve
x = 1500:20:1620;
podaci = randi([1, 100], 1, 7);
figure
bar(x, podaci)
```

U sljedećem primjeru prikazano je kako se može proizvoljno imenovati svaki stupac stupčastog grafa. Rezultat slika 6.

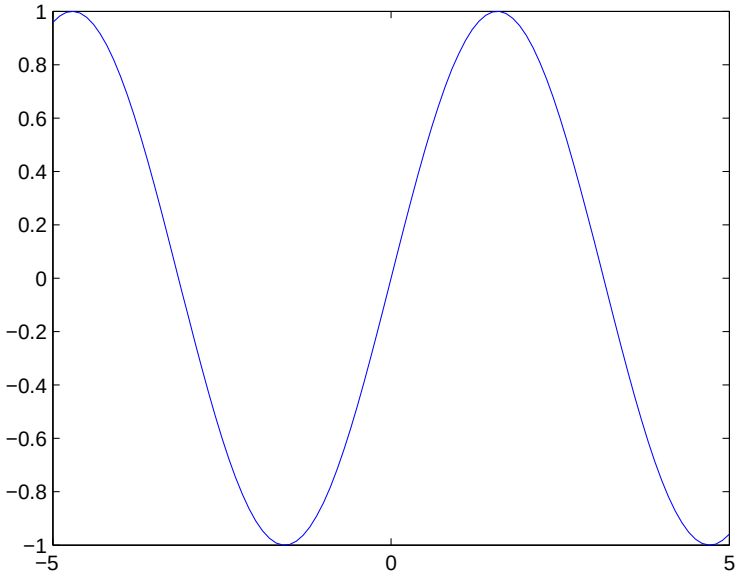
```
rng(999)
podaci = randi([1, 100], 1, 7);
stupci = {'A'; 'B'; 'C'; 'D'; 'E'; 'F'; 'G'};
figure
bar(podaci)
set(gca, 'xticklabel', stupci)
```

Sljedeći blok koda daje primjer kako postaviti više (pod)grafova u jedan *figure*. Korištena je naredba `subplot` koja dijeli *figure* na koordinatni sustav[14] u kojemu smještamo grafove. Rezultat vidimo na slici 7.

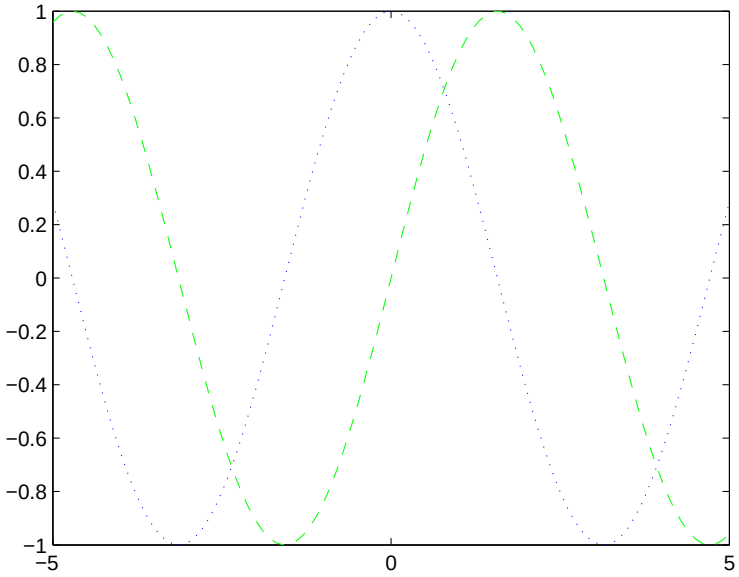
```
% podaci
x = [0:0.1:10];
a = cos(7*x);
b = cos(3.5*x);
c = cos(x);
% grafovi
figure
subplot(2, 2, 1)
plot(x, a)
title('Graf br. 1')
subplot(2, 2, 2)
plot(x, b)
title('Graf br. 2')
subplot(2, 2, [3, 4])
plot(x, c)
title('Graf br. 3')
```

[14] `subplot(m, n, p)` dijeli *figure* u `m` × `n` koordinatni sustav u kojem smještamo pojedini graf na poziciju (ili više njih) `p`.

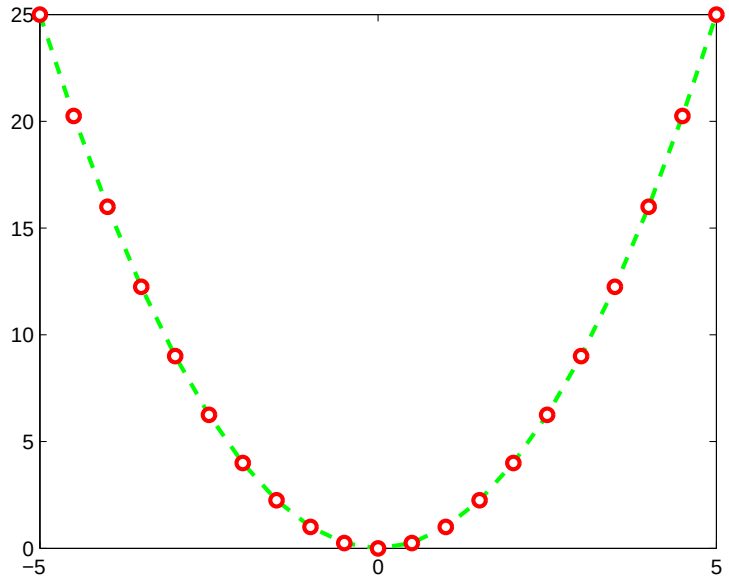
Prilozi



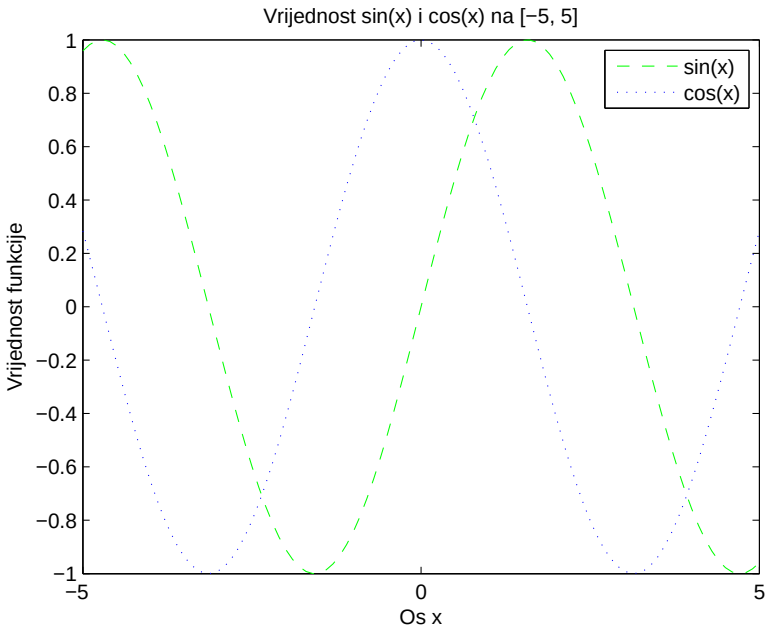
Slika 1: Graf funkcije sinus na intervalu $[-5, 5]$.



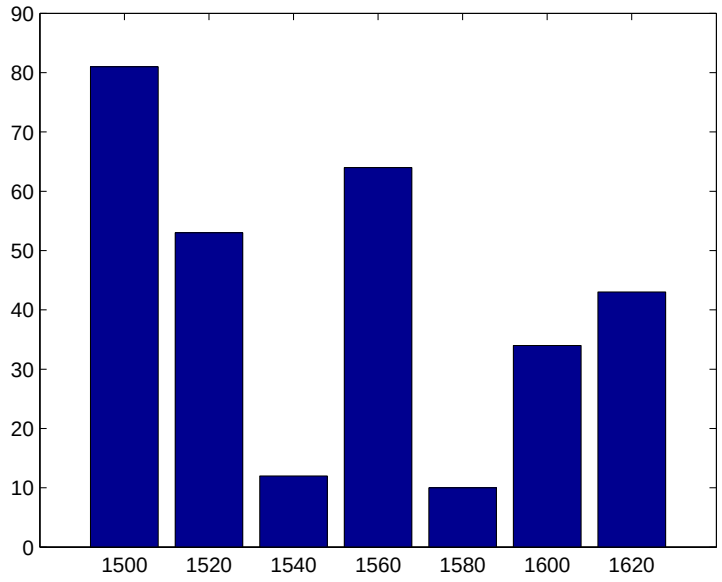
Slika 2: Graf funkcija sinus i kosinus na intervalu $[-5, 5]$.



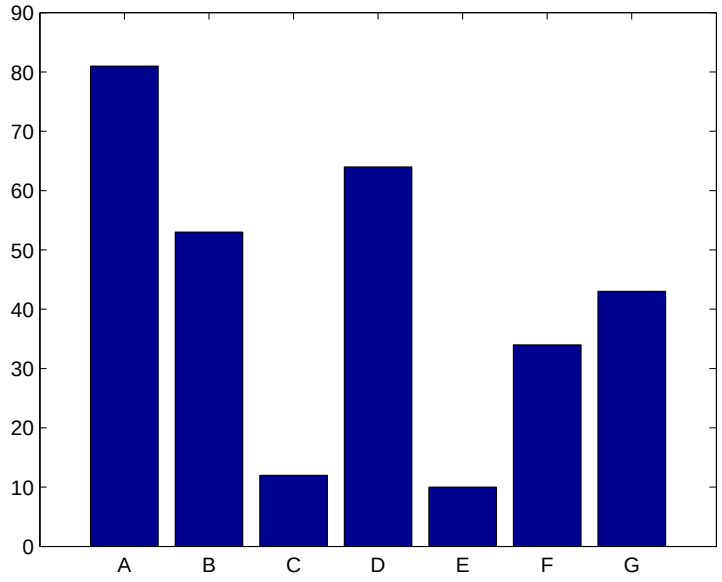
Slika 3: Primjer stiliziranog grafa.



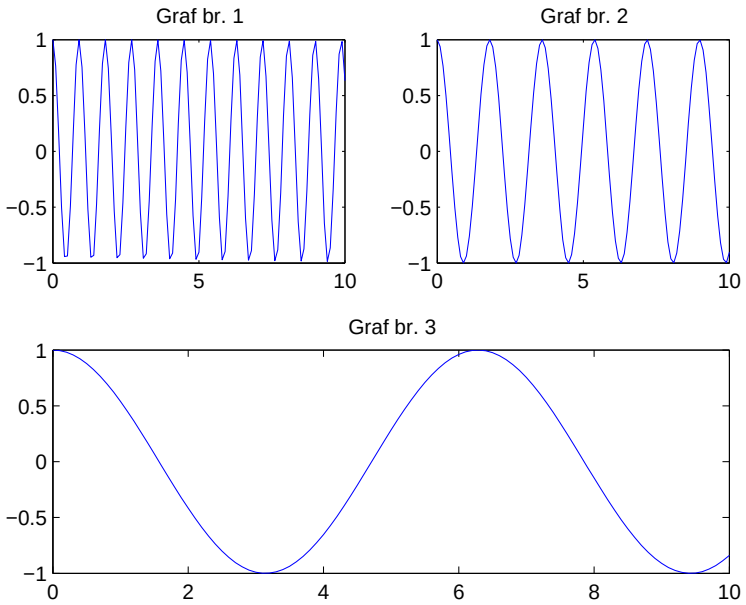
Slika 4: Označeni graf funkcija sinus i kosinus na intervalu $[-5, 5]$.



Slika 5: Primjer stupčastog grafa.



Slika 6: Primjer stupčastog grafa sa proizvoljnim imenima stupaca.



Slika 7: Primjer više grafova na jednoj slici.