Osnovno o Matlabu

Jure Slak

Uvod

MATLAB, okrašava za "matrix laboratory" je okolje za numerično računanje. Razvoj se je začel v sedemdesetih, prva izdaja je bila leta 1984, zadnja izdaja pa je bila ob času pisanja dokumenta pred 19 dnevi.

Okolje Matlab je namenjeno numeričnemu računanju in jezik je izrazito skriptni. Jeziku se pozna, da je bil nadgrajevan skozi čas in ima nekaj okornosti, vendar tudi precej dobrih lastnosti, med drugim enostavnost.

Odprtnokodni bratranec Matlaba je Octave[1].

Osnove

Komentarje se piše s '`%`', prelom vrstice pa s tropičjem '`...`'.

Spremenljivke

Spremenljivk z razliko od Jave ali C++ ni potrebno deklarirati.

Spodaj je primer nekaj deklaracij.

```
x = 6;
y = 'abc';
z = 1:8
w = [1, 2, 4; 5, 6, 7; 6, 8, 9];  % vejice niso obvezne
b = 6 < 7;
```

Te po vrsti naredijo decimalno število x, niz znakov y, vrstični vektor z, matriko w in logično vrednost b. Matlab izpiše vrnjene vrednosti vseh stavkov, ki se ne začnejo s podpičjem. Zgornja koda bi npr. izpisala

```
z =
  1     2     3     4     5     6     7     8
```

Sintaksa $a : b : c$ naredi vrstični vektor z vrednostmi od a do c s korakom b. Lahko je prazen.

Obstajajo tudi več kot dvodimenzionalne matrike, imenovane tenzorji.

Osnovne funkcije za konstrukcijo vektorjev, matrik in tenzorjev so `ones`, `zeros`, `eye`, `diag` in `rand`.

Indeksiranje

Indeksi v tabelah se začnejo z 1, matrike pa se indeksira z okroglimi oklepaji. Spodaj je nekaj primerov. Indeksira se lahko z enim indeksom, vektorjem indeksov ali logično masko. Uporabi se lahko tudi `end` za indeksiranje od konca.

[1] `https://www.gnu.org/software/octave/`

```
y(2)  % b
z(4)  % 4
w(3, 2)  % 8
w(:, 2)  % drugi stolpec, tj. [2; 7; 8]
w(2, :)  % druga vrstica, tj. [5, 6, 7]
mask = z > 4  % [0 0 0 0 1 1 1 1]
z(mask)       % [5 6 7 8]
indeksi = [1, 6, 5, 8, 1];
z(indeksi)    % 1, 6, 5, 8, 1]
z(2:end-3)  % [2 3 4 5]
z(end-3)    % 5
```

Če dostopamo do elementa, ki ne obstaja dobimo napako, če pa priredimo element, na indeks izven meja matrike, se matrika avtomatsko razširi.

```
z(10)  % Error: Index exceeds matrix dimensions.
z(11) = -3;  % z = [1 2 3 4 5 6 7 8 0 0 -3]
z(10)  % 0
```

Matrike lahko indeksiramo tudi samo z enim indeksom, v tem primeru se matrika obnaša kot da bi vse stolpce zložili od leve proti desni v visok vektor.

Drugi tipi

Nekateri drugi osnovni tipi v Matlabu so: `single`, `int8`, `uint8`, `int16`, `uint16`, `int32`, `uint32`, `int64`, `uint64`, `char`, `struct`, `cell`.

Tip `struct` je ekvivalent POD razredu v Javi in `struct` v C-ju. Najlažje ga naredimo kar tako, da določenim poljem za piko neke spremenljivke nastavimo vrednost.

```
s.x = 7;
s.beseda = 'nekaj';
```

Tip `cell` predstavlja polje, ki lahko vsebuje različne tipe ali sezname različnih dolžin. Tako lahko naredimo npr.

```
C = {34, 'abc', [1, 2]};
C{1}  % izpiše 34
```

Operacije

Matlab podpira aritmetične operatorje `+-*/\^'`, ki delujejo na številih, vektorjih, matrikah in tenzorjih. Plus in minus sta običajna, krat pomeni množenje matrik, strešica pomeni potenciranje matrik, levo (`A\B`) in desno (`B/A`) deljenje pa predstavljata reševanje sistema $Ax = B$ in $xA = B$. Primer: $4 / 5 = 0.8$ in $4 \setminus 5 = 1.2$. Opuščaj `'` označuje konjugirano transponiranje.

Vsaka izmed zgornjih matričnih operacij ima tudi verzijo s piko, torej `.* ./ .\ .^`, ki naredijo operacije po elementih matrike. Primer:

```
[1 2; 3 4].^4  % [1 16; 81 256]
[1 2; 3 4]^4   % [199 290; 435 634]
```

Operacija `.'` pomeni običjano transponiranje.

Poleg atrimetičnih pozna tudi relacijske operatorje `== ~= >= <= > <`, ki delujejo tudi na vekorjih in matrikah po elementih. Podobno kot C++ ali Java pozna tudi logična operatorja `&& ||` in logične operatorje po elementih `~ & |`. Kot v Javi in C++ sta `&&` in `||` *short-circuting*.

Matlab pozna tudi operator vertikalne in horizontalne konkatenacije, oz. grajenja matrik. Če imamo dve matriki A in B kompatibilne velikosti, potem `[A B]` naredi bločno matriko, kjer je A zraven B, `[A; B]` pa bločno matriko, kjer je A nad B.

Več o operacijah in posebnih znakih je na voljo v Matlab dokumentaciji.[2]

Kontrolne strukture

Matlab pozna stavke `if`, `switch` in `while`, z enakim pomenov kot Java ali C++. Stavek `for` je podoben foreach zanki `for(x : v)` iz Jave ali C++. Zanka `for` v Matlabu iterira čez dano strukturo. Primer:

```
if all(z > 20)
    disp('Vsi elementi z so večji od 20');
elseif any(z > 20)
    disp('Vsaj en element z je večji od 20');
else
    disp('Noben element z ni večji od 20');
end
k = 5;
while k < 9
    k = k + 0.5;
end
v = [1, 3, 7, 15]
for x = v
    fprintf('x = %d\n', x);
end
for i = 1:10
    switch i
        case 1
            disp('i je 1')
        otherwise
            disp('i ni 1')
    end
end
```

Standardno delujeta tudi stavka `break` in `continue`.

Funkcije se definira v svoji datoteki, ki mora imeti ime enako imenu funkcije. Če naredimo datoteko `funkcija.m`, lahko notri naredimo funkcijo

```
function [v1, v2] = funkcija(p1, p2, p3)
% Opis funkcije, ki se pokaže v pomoči.
```

[2] `https://mathworks.com/help/matlab/matlab_prog/matlab-operators-and-special-characters.html`

```
    v1 = 'abc';
    if p1 < p2, v2 = 4; else v2 = 5; end
end
```

Za vrnitev iz funkcije pred koncem se uporablja `return`. Vrednosti vrnemo tako, da nastavimo izhodne parametre (`v1` in `v2` v zgornjem primeru).

Funkcija lahko tudi izve, koliko parametrov je uporabnik podal na vhodu ali izhodu (ni namreč obvezno podati vseh, če jih ne uporabljamo) z ukazoma `nargin` in `nargout`.

Zgornjo funkcijo bi lahko poklicali kot

```
[v1, v2] = funkcija(4, 5, 6);  % polni klic
v1 = funkcija(4, 3);  % tretjega parametra ne rabimo, vzamemo samo prvi izhod
[~, v2] = funkcija(4, 5)  % vzamemo samo drugi izhod
```

V datoteki se lahko naredi tudi anonimne funkcije, kot npr. `f = @(x, y) 2*x+y`. Ekvivalent v Javi bi bil `(x, y) -> 2*x+y`,
v C++ pa `[](double x, double y) { return 2*x+y; }`.

Funkcije se pogosto definira tako, da so *vektorizirane* ali *matricizirane*, kar pomeni da delujejo tudi za argumente, ki so vektorji ali matrike in se obnaša enako, kot če bi jo z zanko izvedli na vsakem elementu posebej. To je uporabno, ker je krajše in bolj pregledno, še posebej pa, ker je izvajanje funkcije na celi matriki elementov hitrejše kot zanka. Primer:

```
f = @(x, y) x*y + 2*x^2
fvec = @(x, y) x.*y + 2*x.^2

X = rand(100, 100);
Y = rand(100, 100);

tic
Z = fvec(X, Y);
toc   % Elapsed time is 0.000927 seconds.

tic
Z = zeros(size(X));
for i = 1:100
  for j = 1:100
    Z(i, j) = f(X(i, j), Y(i, j));
  end
end
toc  % Elapsed time is 0.005532 seconds.
```

Risanje grafov

Za risanje grafov imamo veliko različnih funkcij, ki rišejo različne tipe grafov. Večina jih ima zelo podobno sintakso:
`plot(podatki, 'Moznost1', Vrednost1, 'Moznost2', Vrednost2)`

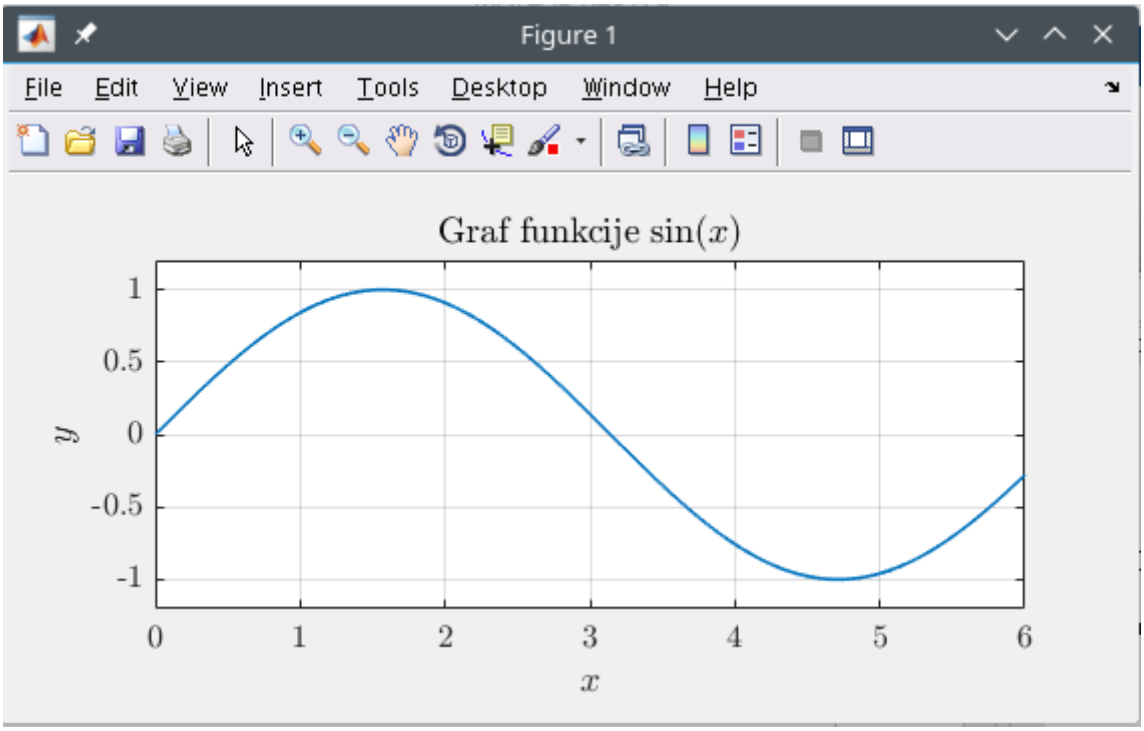
Funkcija `plot` se uporablja za risanje 2D podatkov, obstajajo pa še `scatter`, `plot3`, `scatter3`, `quiver`, `surf`, `trisurf`, `contour`, `histogram`, `pie`, `bar`, ...

Ko je graf narisan, ga lahko oblikujemo do najmanjših podrobnosti. Glavne funkcije so `title`, `xlabel`, `xticks`, `xlim` (ter y in z verzije), `axis`, `grid`, `box`.

Spodnja koda nariše graf funkcije sinus kot na sliki 1.

```
x = 0:0.01:6;
y = sin(x);
plot(x, y, '-');
grid on
box on
xlabel('$x$')
ylabel('$y$')
title('Graf funkcije $\sin(x)$')
axis equal
xlim([0 6])
ylim([-1.2, 1.2])
```

Poleg x in y podatkov sprejme `plot` kot tretji argument še specifikacijo tipa črte, sestavljene iz tipa črte, tipa oznake in tipa barve. Osnovne barve so `rgbcmykw`, osnovne črte so `-`, `--`, `:` in `-.`, osnovne oznake pa so `+o*.xsdhp<>v^`. Vrsti red znakov ni pomemben, niz `'bo-'` pa npr. predstavlja polno črto, podatki so označenimi z krogci, vse skupaj pa je modre barve.



Slika 1: Graf funkcije sinus narisan s priloženo kodo.

.