Podsetnik za Matlab

Sadržaj

Tipovi fajlova

Izdvojićemo tri tipa fajlova u Matlabu:

- skript fajlovi
- funkcijski fajlovi
- live script fajlovi

Skript fajlovi su fajlovi koji sadrže niz podataka ili komandi. Ekstenzija ovih fajlova je .m. Oni nemaju ulazne argumente niti izlazne. Obično se koriste za čuvanje nekih podataka. Mi ćemo ih koristiti da u njima čuvamo test primere. Npr.

```
%primer1.m

% interval
x0=0;
xm=3;

%pocetni uslov
y0=0;

%desna strana
f=@(x,y) ones(size(x));
%{
Konstantnu funkciju u matalbu zapisujemo kao vektor iste duzine kao i x kako
bi se
racunske operacije u matlabu pravilno obavile.
%}

%tacno resenje
ye=@(x) x;
```

Ako imamo skript fajl sa imenom primer1.m, možemo ga pozvati u komandnom prozoru (ili drugom skript fajlu, ili funkciji) njegovim imenom (ne navodimo ekstenziju):

```
primer1
```

Tada će svi podaci koji su upisani u njemu biti dostupni i u okruženju u kome smo ga pozvali. Skript fajlovi se mogu učitati i pomoću funkcije run(imef), gde promenljiva imef mora biti string koji predstavlja naziv fajla:

```
run('primer1')
```

Ova funkcija je naročito korisna ako želimo kodove da testiramo na različitim primerima.

Funkcijski fajlovi imaju istu ekstenziju .m, a razlikuju se po tome što mogu imati ulazne i izlazne argumente. Definišu se navođenjem ključne reči function, zatim u istom redu u uglastim zagradama navodimo izlazne argumente, pa znak jednakosti, ime funkcije i u malim zagradama ulazne argumente.

Sve promenljive koje su definisane unutar funkcijskog fajla su privatne, tj. nakon izvršavanja funkcije, nisu dostupne u komandnom prozoru.

Live script fajlovi su relativno novi tip fajlova u matlabu. Prepoznajemo ih po ekstenziji .mlx. Služe za interaktivni prikaz rezultata zajedno sa običnim tekstom, tekstom kucanim u texu, gaficima,... pa ih možemo nazvati Matlab sveskama. Ovaj tekst je pisan u live scriptu. Live script možemo izvesti i kao pdf, tex, word ili html fajl. Svakako vrlo koristan tip fajla.

Live script može biti i live funkcija, tj. u live scriptu se može definisati i funkcija (ili više njih), ali se sav teskt i poyivi funkcija postavljaju na početak fajla, a definicije funkcija na sam kraj fajla.

Pokazivači na funkcije

Krenimo prvo od tipa podataka **funcion handle - pokazivač na funkciju**. Na primer, uzmimo funkciju sin(x). Ako zatražite da vam matlab odštampa njen kod, dobućete odgovor

```
type sin
```

```
'sin' is a built-in function.
```

da je ona ugrađena funkcija, što znači da je programirana u nekom drugom programskom jeziku (u C-u) i instalira se zajedno sa matlabom. Pretpostavimo da je želimo prosleđivati preko neke promenljive, možda nekoj drugoj funkciji. Tada moramo definisati pokazivač na nju, što radimo dodavanjem simbola @ ispred naziva funkcije:

```
f=@sin
```

```
f = function_handle with value:
    @sin
```

Sada sinusu možemo pristupiti preko promenljive f:

```
f(pi/2)
```

```
ans = 1
```

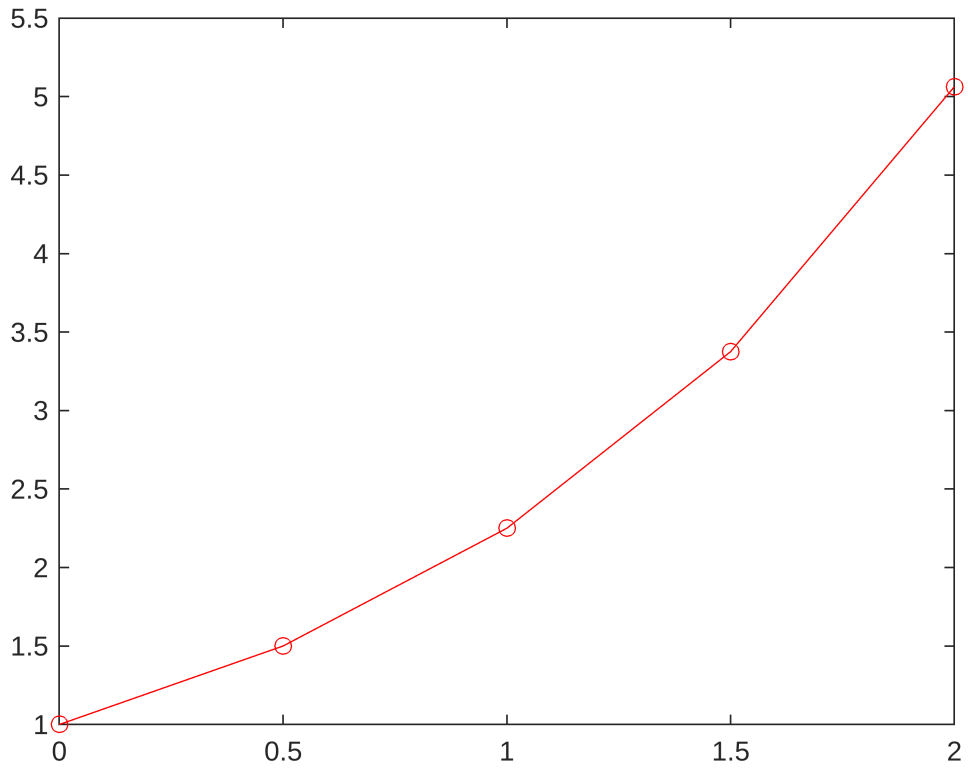
Slično, možemo definisati pokazivač i na funkcijski fajl koji smo sami napravili. Naprimer, imamo funkcijski fajl ojler(imef,h). Pokazivač na njega definišemo analogno

```
test=@ojler
```

```
test = function_handle with value:
    @ojler
```

Sada funkciju ojler možemo pozivati i zamenom reči ojler sa rečju test (isti broj argumenata moramo navesti kao i u originalnoj funkciji)

```
test('primer2',0.5)
```




```
ans = 1×5
    1.0000    1.5000    2.2500    3.3750    5.0625
```

Anonimna funkcija

Anonimna funkcija je takođe pokazivač na funkciju. Razlika je samo u tome što ovde definišemo i nezavisne promenljive, tj. ulazne argumente. Navodimo ih u malim zagradama, odmah nakon symbola @. Na primer, neka je potrebno definisati funkciju $f(x,y) = ysin(x)$, anoninmna funkcija koja joj odgovara je

```
f=@(x,y) y.*sin(x);
```

Vrednost f(pi/2,2) sada možemo dobiti komandom

```
f(pi/2, 2)
```

```
ans = 2
```

Može se dogoditi da nam je jedna od vrednosti fiksirana, na primer x=pi/2, a nama trebaju vrednosti za različito y, tada od f možemo napraviti novu anonimnu funkciju jedne promenljive y:

```
g=@(y) f(pi/2,y);
```

U tom slučaju će matlab definisati novu funkciju g tako što će uzeti definiciju funkcije f i u njoj, na mesto promenljive x učitati pi/2, dok će y ostati slobodna promenljiva. Sada f(pi/2,2) mozemo izračunati i sa g(2)

```
g(2)
```

```
ans = 2
```

Slično, definicija sledećeg oblika će proći bez prijave greške

```
clear % izbrisi sve promenljive iz komandnog prozora
h=@(x) x.^2+a % promenljiva a nema dodeljenu vrednost
```

```
h = function_handle with value:
    @(x)x.^2+a
```

Ali ako probamo da je koristimo, dobićemo kao odgovor grešku.

```
%{
Greška blokirati izvršavanje svih linija koda ispod nje (a hoću još komandi
da
izvršavam),  pa ću grešku "uhvatiti" sa blokom try catch. try će naterati da
se sve
linije koda izvršavaju i pored greške, a catch će upisati sve informacije
o greškama u promenljivu koju sam nazvala greška.
%}
try
    h(1)
catch greska
    disp(greska.message)
end
```

```
Unrecognized function or variable 'a'.
```

Matlab grešku prijavljuje jer promenljiva a nema nikakvu vrednost. Ako joj dodelimo vrednost, npr.

```
a=10;
```

i ponovo definišemo anonimnu funkciju h

```
h=@(x) x.^2+a
```

```
h = function_handle with value:
    @(x)x.^2+a
```

tada će matlab učitati u definiciju funkcije h da je a=10. Pa možemo koristiti h bez ikakvih problema.

```
h(0)
```

```
ans = 10
```

Vektor kao argument anonimne funkcije

Navešću još jedna koristan primer. Recimo da radimo sa vektorom vrstom, želimo da kvadriramo samo poslednji element i tu operaciju treba da uradimo puno puta, na različitim vektorima (vrstama). Takav posao bi onda mogla da radi anonimna funkcija oblika

```
w=@(x) [x(1:end-1), x(end)^2]
```

```
w = function_handle with value:
    @(x)[x(1:end-1),x(end)^2]
```

```
w([1,1,3])
```

```
ans = 1×3
     1     1     9
```

```
w(1:10)
```

```
ans = 1×10
     1     2     3     4     5     6     7     8     9   100
```

```
w(4)
```

```
ans = 16
```

Primetimo, anonimna funkcija ima samo jedan izlazni argument (broj, vektor, matrica,...). Dakle, ako želimo više izlaznih argumenata, onda moramo praviti funkcijski fajl.

Izračunavanje vrednosti integrala u matlabu

Funkcija za određivanje (približne) vrednosti određenog integrala u matlabu je integral(f,a,b), gde je f pokazivač na funkciju, a a i b, krajevi intervala integracije. Na primer, ako želimo izračunati integral sinusa na segmentu od 0 do pi, to možemo uraditi sa komandom

```
integral(@sin, 0, pi)
```

```
ans = 2.0000
```

ili

```
integral(@(x) sin(x), 0,pi)
```

```
ans = 2.0000
```

Možemo definisati i integral složene funkciju, npr.

```
f=@(x) sin(x);
g=@(x) x.^2;
integral(@(x) g(f(x)), 0,pi)
```

```
ans = 1.5708
```

što je isto što i

```
integral(@(x) (sin(x)).^2, 0, pi)
```

```
ans = 1.5708
```

Približna vrednost integrala se u funkciji integral određuje pomoću adaptivnog kvadraturnog pravila.

Pre funkcije integral je za iste namene bila implementirana funkcija quad(f,a,b) pomoću Simpsonovog kvadraturnog pravila. Nju treba izbegavati jer matlab već dugo najavljuje da će je izbaciti iz budućih verzija softvera. Inače, može se koristiti na potpuno identičan način kao i integral:

```
quad(@sin, 0, pi)
```

```
ans = 2.0000
```

```
quad(@(x) sin(x), 0,pi)
```

```
ans = 2.0000
```

```
quad(@(x) g(f(x)), 0,pi)
```

```
ans = 1.5708
```