Wprowadzenie do środowiska MATLAB z zastosowaniami w modelowaniu i analizie danych

Daniel Wójcik
Instytut Biologii Doświadczalnej PAN
Szkoła Wyższa Psychologii Społecznej

d.wojcik@nencki.gov.pl

tel. 022 5892 424

http://www.neuroinf.pl/Members/danek/swps/matlab_html

1. Wprowadzenie do środowiska MATLAB. Elementy interfejsu graficznego, dostępne narzędzia, system pomocy, darmowe odpowiedniki MATLABA, elementarne obliczenia i wykresy.
2. Tablice i macierze. Własności, generacja, operacje na macierzach, interpretacja.
3. Grafika. Podstawowe wykresy w MATLABie, edycja wykresów, przygotowanie wykresów do publikacji i prezentacji, wykresy trójwymiarowe, przetwarzanie obrazów, animacje.
4. **Programowanie. Sterowanie programem, struktury danych, skrypty i funkcje.**
5. Tworzenie interfejsów graficznych do skryptów MATLABa.
6. Modelowanie deterministyczne. Układy z czasem dyskretnym i ciągłym. Oscylacje i chaos. Szukanie rozwiązań i wizualizacja.
7. Wybrane metody numeryczne. Interpolacja i ekstrapolacja. Dopasowywanie funkcji.
8. Liczby losowe. Generacja i zastosowanie w symulacjach stochastycznych.
9. Elementy statystycznej analizy danych w MATLABie.
10. Praca nad własnymi problemami (różne zastosowania)

Programowanie

- Sterowanie programem
- Struktury danych
- Skrypty i funkcje.

M-pliki

Jeżeli wykonujemy skomplikowane zadanie, wymagające użycia wielu komend, wygodnie jest wpisać wszystkie komendy do pliku, zwanego M-plikiem.

Tworzymy w edytorze plik mfile1.m i wpisujemy zawartość prawej kolumny, a następnie po nagraniu pliku w linii komend piszemy

```
mfile1
```

lub klikamy F9 w edytorze

mfile1.m

```
z = peaks;
zplot = z;

% Do the peaks:
clf
subplot(221)
ind = find(z<0);
zplot(ind) = zeros(size(ind));
mesh(zplot)
axis tight

% Do the valleys:
subplot(222)
ind = find(z>0);
zplot = z;
zplot(ind) = zeros(size(ind));
mesh(zplot)
axis tight
```

M-pliki

Linie zaczynające się od znaku % są komentarzami i są ignorowane.

Zmienne utworzone danym skryptem pozostają w przestrzeni roboczej Matlaba:

```
>>   clear
>>   whos
>>   mfile1
>>   whos
```

```
Name          Size  Bytes Class
ind        1544x1   12352 double array
z            49x49 19208 double array
zplot        49x49 19208 double array
```

Skrypty mogą też operować na zmiennych w przestrzeni roboczej Matlaba.

Skrypty mogą uruchamiać inne skrypty

Funkcje

Funkcje to m-pliki, których można użyć do rozszerzania języka Matlaba. Mogą przyjmować argumenty wejściowe i wyjściowe.

Wiele funkcji Matlaba jest implementowanych jako m-pliki. Napisz:

```
type mean
```

Funkcje używają zmiennych lokalnych, które nie pojawiają się w głównej przestrzeni roboczej Matlaba

Plik `kwadratowe.m`

```
function x = kwadratowe(a,b,c)
% KWADRATOWE Znajduje pierwiastki
% rownania kwadratowego.
%
% X = KWADRATOWE(A,B,C) zwraca oba
% pierwiastki rownania kwadratowego
%    y = A*x^2 + B*x + C.
% Pierwiastki sa zwracane w macierzy
% X = [X1 X2].

% na podstawie skryptu A. Knighta,
% kwiecien 2007
delta = 4*a*c;
mian = 2*a;
pierwyz = sqrt(b.^2 - delta);
% Pierwiastek wyznacznika
x1 = (-b + pierwyz)./mian;
x2 = (-b - pierwyz)./mian;
x = [x1 x2];
```

Funkcje

m-pliki funkcji mają format

```
function [output] = function_name(input)
```

np.

```
function x = kwadratowe(a,b,c)
```

przykłady składni:

```
function [xx,yy,zz] = kula(n)
function fajnywykres
function a = listy(x,y,z,t)
```

Nazwa funkcji powinna być taka sama jak nazwa pliku, w którym jest zdefiniowana.

Komentarze po słowie `function` są wyświetlane komendą `help`. Np.

```
help kwadratowe
```

Komentarze i puste linie mogą pojawiać się gdziekolwiek w pliku.

Kontrola wykonania

MATLAB ma 4 rodzaje wyrażeń służące do kontroli wykonania:

if, else, elseif	wykonaj wyrażenia w oparciu o wynik testu logicznego
switch, case, otherwise	wykonaj grupę wyrażeń w oparciu o wynik testu logicznego
while, end	wykonaj wyrażenia nieokreśloną liczbę razy w oparciu o wynik testu logicznego
for, end	wykonaj wyrażenia określoną liczbę razy

if, else, elseif

Podstawowa forma wyrażenia if to:

```
if test
    statements
end
```

`test` jest wyrażeniem, które przyjmuje wartości albo 1 (prawda) albo 0 (fałsz). Wyrażenia pomiędzy wyrażeniami `if` oraz `end` są wykonywane, jeżeli `test` jest prawdą. Jeżeli `test` jest fałszem, wyrażenia te będą zignorowane i program będzie wykonywany od pierwszej linijki następującej po wyrażeniu `end`. Wyrażenie `test` może być wektorem albo macierzą. W tym wypadku wszystkie elementy muszą wynosić 1, żeby wyrażenia zostały wykonane. Dalsze testy można przeprowadzić korzystając z `else, elseif`

```
if test
   statements1
elseif test2
   statements2
else
   statements3
end
```

switch

Podstawowa forma wyrażenia switch:

```
switch test
   case result1
      statements
   case result2
      statements
   .
   .
   .
   otherwise
      statements
end
```

Odpowiednie wyrażenia są wykonywane, jeżeli test jest równy odpowiedniemu wynikowi. Jeżeli żaden z podanych wyników nie ma miejsca, wykonywane są wyrażenia podane po otherwise.

Jeżeli te same wyrażenia mają być wykonywane dla różnych wyników, należy użyć nawiasów klamrowych:

```
switch x
   case 1
      disp('x is 1')
   case {2,3,4}
      disp('x is 2, 3 or 4')
   case 5
      disp('x is 5')
   otherwise
      disp('x is not 1, 2,...
           3, 4 or 5')
end
```

pętla while

Podstawowa forma pętli while:

```
while test
        statements
end
```

Wyrażenia są wykonywane wielokrotnie dopóki wartość testu wynosi 1. Na przykład żeby znaleźć pierwszą liczbę dla której 1+2+· · ·+n jest większe od 1000 możemy napisać:

```
n = 1;
while sum(1:n)<=1000
    n = n+1;
end
```

Szybkim sposobem na „wykomentowanie” fragmentu kodu jest otoczenie go komendami `while 0` i `end`. Tak otoczony fragment nigdy nie będzie wykonany.

pętla for

Podstawowa forma pętli for:

```
for index = start:przyrost:stop
    statements
end
```

Przyrost można ominąć, wtedy przyjmuje się 1. Przyrost może być dodatni lub ujemny. Przy pierwszym przebiegu przez pętlę indeks będzie miał wartość start. W każdym kolejnm przebiegu indeks będzie miał wartość zwiększoną o przyrost dopóki nie przekroczy wartości stop. Ten przykład generuje widoki funkcji peaks z różnych kątów:

```
clf; colormap(gray)
plotnum = 1;
z = peaks(20);
for az = 0:10:350
  subplot(6,6,plotnum)
  surfl(z),shading flat
  view(az,30)
  axis tight
  axis off
  plotnum = plotnum + 1;
end
```

pętla for

Indeks pętli for może być wektorem albo macierzą. Jeżeli jest wektorem, pętla będzie wykonana tyle razy ile jest elementów w wektorze, przy czym w kolejnych przebiegach indeks przyjmuje kolejne wartości wektora. Jeżeli indeks jest macierzą pętla zostanie wykonana tyle razy ile jest kolumn macierzy, przy czym wartości indeksu w kolejnych przebiegach będą równe kolumnom macierzy. Na przykład:

```
>> q = pascal(3)
q=
      1       1       1
      1       2       3
      1       3       6
>> for i = q,i,end
i=
      1
      1
      1

   i =
       1
       2
       3

   i =
       1
       3
       6
```

Przykład: generacja filmu

- film 1: ewolucja rozkładu prawdopodobieństwa
- film 2: Obracający się wykres funkcji peaks
- film 3: zależne od czasu rozwiązanie równania kabla

Film 1:

tendencyjność
monety
w ujęciu
bayesowskim

```
bias = 0.75;      dane=rand(1,700);
dane(find(dane < bias))= 0;
dane(find(dane > 0))=1;
n=length(dane);
H = 0:0.01:1;
E = exp((-(H-0.5).^2)/(0.1.^2));

aviobj = avifile('mymovie75.avi','fps',15);

i=0;
while i < n
    if i < 50 i = i+1; else i=i+10; end

    r=length(find(dane(1:i) == 0));
    wykres1 = (H.^r).*(1-H).^(i-r);
    wykres2 = (H.^r).*(1-H).^(i-r).*E;
    plot(H, wykres1 / max(wykres1));
    hold on;
    plot(H, wykres2/ max(wykres2), 'r');
    hold off;
    frame = getframe(gcf);
    aviobj = addframe(aviobj,frame);
end

aviobj = close(aviobj);
```

Film 2:

Obracający się wykres funkcji peaks

```
clf;
colormap(gray)
plotnum = 1;
z = peaks(20);

aviobj =
avifile('peaks.avi','fps',15);

for az = 0:1:359
  surfl(z),shading flat
  view(az,30)
  axis tight
  axis off
  frame = getframe(gcf);
  aviobj =
addframe(aviobj,frame);
end

aviobj = close(aviobj);
```