Wprowadzenie do środowiska MATLAB z zastosowaniami w analizie danych EEG

Daniel Wójcik
Instytut Biologii Doświadczalnej PAN

d.wojcik@nencki.gov.pl

tel. 022 5892 424

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

Programowanie

- Sterowanie programem
- 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 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 %interp
  view(az,30)
  axis tight
  axis off
  frame = getframe(gcf);
  aviobj = addframe(aviobj,frame);
end

aviobj = close(aviobj);
```

Automaty komórkowe

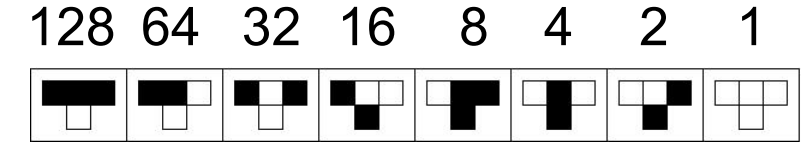
Ewolucja identycznych elementów:
jak proste reguły prowadzą
do złożonych zachowań

Rozważmy 1D automat komórkowy:

- Mamy “świat” składający się z L kolejnych “komórek”.
- Każda komórka (o numerze n) przyjmuje jeden z dwóch stanów, 0 lub 1.
- Wszystkie komórki zmieniają stan jednocześnie.
- Stan przyszły (w chwili t+1) komórki n zależy od stanu obecnego (w chwili t) tej komórki oraz jej dwóch sąsiadów: n-1 oraz n+1.
- Przyjmujemy okresowe warunki brzegowe, czyli komórka o numerze L jest lewym sąsiadem komórki o numerze 0.

Przykład

reguła 30

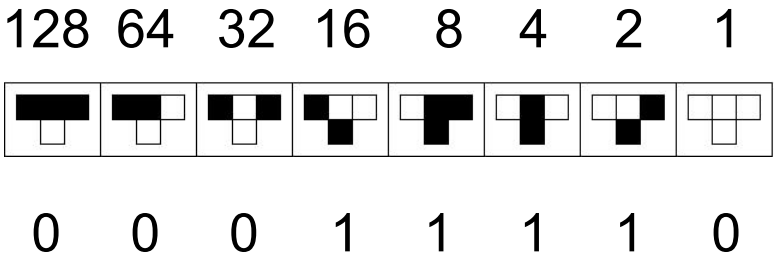


Kodowanie reguły

Każdemu układowi stanów komórki *n* i jej sąsiadów *n+1* i *n-1* w chwili *t* przypisujemy liczbę jak na rysunku obok

Kodem reguły jest suma liczb kodujących te trójki stanów, po których w chwili *t+1* stan komórki *n* ma być 1

reguła 30



kod reguły = 16+8+4+2 = 30

Przygotowanie

Zacznijmy od stworzenia nowego skryptu:

```
edit automaty
```

Tworzymy świat

Nasz “świat” ma długość L, możemy więc myśleć o nim jako o wektorze. Ewolucja dodaje wymiar czasowy: w każdej chwili t mamy nowy stan układu. Możemy myśleć o tym jako o wielu wektorach, albo jako o 2D macierzy, w której jest jeden wymiar przestrzenny i jeden czasowy:

```
universe = zeros(nr_of_cells+2,max_time);
```

Dlaczego +2? Komórki o numerach 1 i L+2 pomogą nam poradzić sobie z warunkami brzegowymi.

Stan początkowy

Wybierzmy losowy stan początkowy.

```
universe(2:nr_of_cells+1,1) = ...
      floor(2*rand(nr_of_cells,1));
```

Co robi ta komenda?
A ten trick?

```
universe(1+find(rand(nr_of_cells,1)<0.3),...
                                    1) = 1;
```

Program testowy

```
nr_of_cells = 50;
max_time = 100;
prob = 0.9;

universe = zeros(nr_of_cells+2,max_time);

universe(2:nr_of_cells+1,1) = ...
                floor(2*rand(nr_of_cells,1));

for time=2:max_time
  universe(1+find(rand(nr_of_cells,1)<prob),...
             time) = 1;
end

imagesc(universe)
```

Jeżeli wiemy już jak działa ten program testowy, wróćmy do wersji nam potrzebnej:

```
nr_of_cells = 50;
max_time = 100;

universe = zeros(nr_of_cells+2,max_time);

universe(2:nr_of_cells+1,1) = ...
                floor(2*rand(nr_of_cells,1));

for time=2:max_time
  % tu musimy wstawić reguły ewolucji
end

imagesc(universe)
```

Musimy teraz zaprogramować ewolucję. Jest wiele sposobów. Oto jeden z prostszych:

```
for n=2:nr_of_cells+1
  if (universe(n-1,time-1) == 0 & ...
      universe(n  ,time-1) == 0 & ...
      universe(n+1,time-1) == 0 )
          universe(n,time) = 0;
  end
  if (universe(n-1,time-1) == 0 & ...
      universe(n  ,time-1) == 0 & ...
      universe(n+1,time-1) == 1 )
          universe(n,time) = 1;
  end
  % ...
  % w sumie osiem, prawda?
end
```

O mało co nie zapomnieliśmy o warunkach brzegowych!

```
for time=2:max_time
  % warunki brzegowe
  universe(1,time-1) = ...
           universe(nr_of_cells+1,time-1);
  universe(nr_of_cells+2,time-1) = ...
           universe(2,time-1);

  for n=2:nr_of_cells+1
     %...

  end
end
```

Mamy teraz uniwersalny symulator 1D automatów komórkowych. Możemy zmieniać:

- Rozmiar układu,
- Regułę ewolucji (jest ich 256)
- Stan początkowy, itd.

Przykładowe pomysły na eksperymenty

Czy każda reguła zachowuje się tak samo?
Opisz jakościowo zachowanie się różnych reguł,
na przykład porównaj reguły 36, 38, 40 i 52.
Czy wyniki są jakościowo podobne?

Ile jest różnych stanów końcowych (cykli)?
Jak długie są cykle?
Jak zmieniają się te parametry, kiedy rośnie długość sieci?

Automaty wielostanowe:
- Modele ośrodków aktywnych (excitable media?) – spirale
- Modele dyfuzji

Automaty w 2 i 3 wymiarach, np. life.

Ogólny automat komórkowy jednowymiarowy

```
function nextstep = rule(rulenumber,a1,a2,a3)
bit = 4*a1+2*a2+a3;
bits = dec2bin(rulenumber,8);
nextstep = eval(bits(8-bit));
```

```
function A = ca(rule_nr,lat_length,time)
A = zeros(time,lat_length);
A(1,(lat_length+1)/2) = 1;
for i = 1:(time-1)
    A(i+1,1) = rule(rule_nr,A(i,lat_length),A(i,1),A(i,2));
    A(i+1,lat_length) = rule(rule_nr,A(i,lat_length-1),...
                             A(i,lat_length),A(i,1));
    for j = 2:(lat_length-1)
        A(i+1,j) = rule(rule_nr,A(i,j-1),A(i,j),A(i,j+1));
    end
end
```

```
A = ca(30,101,200);
colormap(gray);
imagesc(1-A)
```

Prosta implementacja „Life”

```
function A = ca2(lat_length, time, prob, init_type)
% A naive implementation of "Life"

%initial state
A = zeros(time,lat_length,lat_length);
if (init_type == 1)
    A(1,:,:)=(rand(lat_length,lat_length) < prob);
elseif (init_type == 2)
    mid=floor((lat_length+1)/2);
    A(1, mid  , mid  ) = 1;
    A(1, mid  , mid-1) = 1;
    A(1, mid+1, mid  ) = 1;
    A(1, mid-1, mid  ) = 1;
    A(1, mid+1, mid+1) = 1;
    A(1, mid-1, mid+1) = 1;
end
```

```
for i = 1:(time-1)
    % I do not touch the borders to keep
    % Dirichlet boundary conditions
    for j = 2:lat_length-1
        for k = 2:lat_length-1
            nbhd = squeeze(A(i,j-1:j+1,k-1:k+1));
            howmanynbs = sum(sum(nbhd));
            if (nbhd(2,2) == 0) % dead cell
                if (howmanynbs == 3)
                    A(i+1, j, k) = 1;
                else
                    A(i+1, j, k) = 0;
                end
            else % alive cell, counted in howmanynbs
                if ((howmanynbs < 3) || (howmanynbs >4))
                    A(i+1, j, k) = 0;
                else
                    A(i+1, j, k) = 1;
                end
            end
        end
    end
end
```

Sterownik do ca2

```
% mylife
lat_length = 31;
time=100;
prob=0.1;
init_state = 2;

B=ca2(lat_length,time,prob,init_state);

clf;
colormap(gray)

%aviobj = avifile('my_life.avi','fps',10);

for frm=1:time
  imagesc(1-squeeze(B(frm,:,:)))
  %frame = getframe(gcf);
  %aviobj = addframe(aviobj,frame);
end

%aviobj = close(aviobj);
```

Ćwiczenia z mylife

- Zmień program mylife tak, żeby obrazek był wyświetlany po wyliczeniu każdej iteracji. W tym celu napisz funkcję ca2step, która będzie wyliczała stan układu w t+1 na podstawie stanu układu w t.
- Zmień warunki brzegowe na okresowe w przestrzeni.

Państwa zadaniem będzie teraz:

- Poeksperymentować z programem tak, żeby wszystkie jego elementy były dla Państwa zrozumiałe
- Napisać sprawozdanie z eksperymentów. Sprawozdanie powinno zawierać:
  - Przedstawienie problemu
  - Kod programu z opisem, co on robi
  - Opis przeprowadzonych eksperymentów z rysunkami przedstawiającymi wyniki
  - Podsumowanie zawierające wnioski z przeprowadzonych symulacji

exmedia.m –
a program to evolve a wave
in excitable media