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)

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
  view(az,30)
  axis tight
  axis off
  frame = getframe(gcf);
  aviobj =
addframe(aviobj,frame);
end

aviobj = close(aviobj);
```

Automaty komórkowe

Druga symulacja:

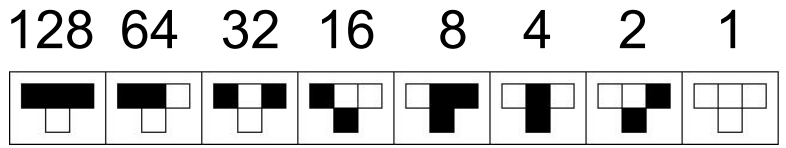
Ewolucja identycznych elementów:
jak proste reguły prowadzą
do złożonych zachowań

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

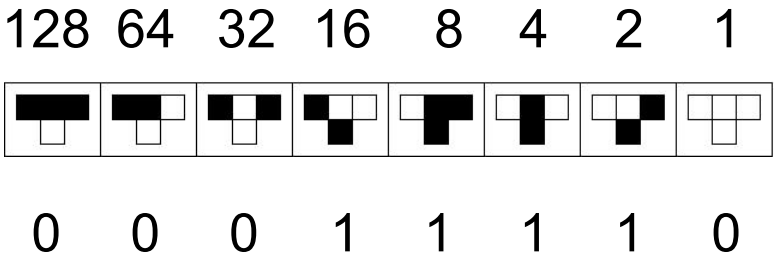


Kodowanie reguły

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.