MATHEMATICAL AND COMPUTER MODELING OF NONLINEAR BIOSYSTEMS I

COMPUTER LABORATORY III: : Discrete logistic equation (solutions, bifurcations), two dimensional discrete models

Ph. D. Programme 2013/2014

Project co-financed by European Union within the framework of European Social Fund

Discrete logistic equation

Let us consider the logistic map (discrete logistic equation)

$$x_{n+1} = rx_n(1 - x_n)$$

where $r \in (0, 4]$ and $x_0 \in [0, 1]$.

Value x_n is the ratio of existing population to the maximum possible population at year n.

The above equation became famous because it exhibits chaotic dynamics. In the series of exercises we will try to understand what does it mean.

Exercise 1 - Implementation of logistic map

Implement the logistic map

$$x_{n+1} = rx_n(1 - x_n)$$

as a MATLAB function, with the following features:

- three arguments: r, initial value, number of iterations to perform;
- optional boolean argument stating if the whole trajectory $x_0, x_1, ..., x_n$ or only the last element x_n should be returned. By default the function should return only the last value;
- initial value can be a vector and solution to the logistic map will be calculated for each element of that vector.

Exercise 1 - solution

```
function a = logisticMapVectInit( r, a0, n, traj )
    if nargin<4 %check if argument is passed
        traj = false;
    end
    a0=a0(:);
    if traj %true: return whole trajectory
        a=zeros(length(a0),n+1); %predefine output
        a(:,1)=a0;
        for i=1:n
            a(:,i+1)=r*a(:,i).*(1-a(:,i));
        end
    else
        a=a0;
        for i=1:n
            a=r*a.*(1-a);
        end
    end
end
```

Exercise 1a - Implementation of logistic map

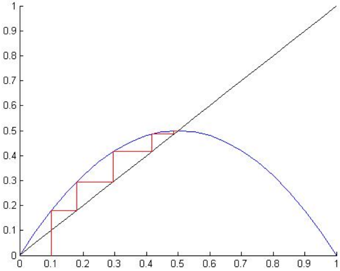
Implement the same function as in Exercise 1, but vectorized with respect to r instead of vectorization with respect to the initial condition.

Exercise 1 - solution

```
function a = logisticMapVectParam( r, a0, n, traj )
    if nargin<4 %check if argument is passed
        traj = false;
    end
    r=r(:); %make vertical vector
    if traj %true: return whole trajectory
        a=zeros(length(r),n+1); %predefine output
        a(:,1)=a0;
        for i=1:n
            a(:,i+1)=r.*a(:,i).*(1-a(:,i));
        end
    else
        a=a0*ones(length(r),1);
        for i=1:n
            a=r.*a.*(1-a);
        end
    end
end
```

Exercise 2 - solution illustration (cobweb plot)

Using the functions implemented in previous exercises write a MATLAB script which solves the logistic equation for specified parameters, initial conditions and number of iterations. Visualize solution using cobweb plot (Verhulst diagram).



cobweb plot

Exercise 2 - solution (part 1)

```
clear all
close all

%%%% Initial settings
r = 3.5;
a0 = 0.1;
n=10;
%%%%

sol=logisticMap(r,a0,n,true);
```

Exercise 2 - solution (part 2)

```
figure(1)
clf
hold on
%plotting RHS
plot(0:0.05:1,logisticMapVectInit(r,0:0.05:1,1));
%plotting auxilary
plot(0:0.05:1,0:0.05:1,'k');
%plotting iterations
init = 0;
for i=2:n+1
    plot([sol(i-1) sol(i-1)],[init sol(i)],'r');
    plot([sol(i-1) sol(i)],[sol(i) sol(i)],'r');
    init = sol(i);
end
hold off
```

Exercise 3 - bifurcation diagram (Feigenbaum tree)

Plot the bifurcation diagram for the logistic map. Procedure description

1. define the uniform mesh for parameter r in the range $[1, 4]$;
2. for each mesh value of r solve the logistic equation starting from $x_0 = 0.5$ up to $n = 200$
3. plot the values of $x_{101}, x_{102}, ..., x_n$ on the y-axis with the x-axis value equal to current r value.

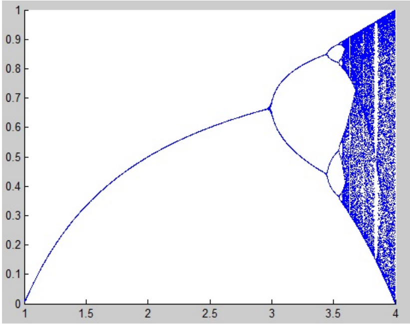
Exercise 3 - naive solution

```
function FeigenbaumTreeNaive
END=1500; %number of mesh points
hold on
for i=1:END,
r = 1 + i*(3/END); a = 0.5;
for j=1:100,
a = r*a*(1-a);
end
for j=1:100,
a = r*a*(1-a);
plot(r,a);
end;
end;
hold off
end
```

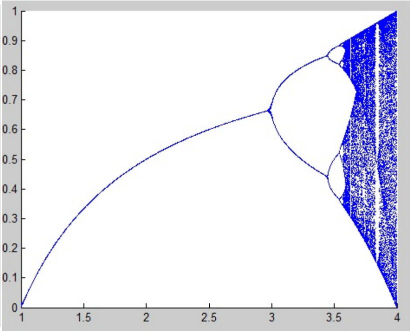
Exercise 3 - solution using previously implemented functions

```
function FeigenbaumTree(r, n)

    Iter = logisticMapVectParam(r,0.5,2*n,true);

    h=scatter(repmat(r,1,n),reshape(Iter(:,n+2:end),1,[]),'.');
    hChildren = get(h, 'Children');
    set(hChildren, 'Markersize', 2)
end
```

Comparizon of solutions - 1500 mesh points for r



Naive: ≈ 76 **seconds**



More efficient: ≈ 1.6 **seconds**

Leslie model

Consider the population of individuals divided into ω age groups $n_0, n_1, ..., n_{\omega-1}$.
We may describe that population using the Leslie model

$$\begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_{\omega-1} \end{bmatrix}_{t+1} = \begin{bmatrix} f_0 & f_1 & f_2 & f_3 & \dots & f_{\omega-1} \\ s_0 & 0 & 0 & 0 & \dots & 0 \\ 0 & s_1 & 0 & 0 & \dots & 0 \\ 0 & 0 & s_2 & 0 & \dots & 0 \\ 0 & 0 & 0 & \ddots & \dots & 0 \\ 0 & 0 & 0 & \dots & s_{\omega-2} & 0 \end{bmatrix} \begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_{\omega-1} \end{bmatrix}_t$$

where parameter f_i describes the average number of offspring from the age class x and s_i is the fraction of individuals that survives from age class i to age class $i+1$.

Exercise 4 - stable regions in Leslie model

Consider the Leslie model with 3 age groups described by the following transition matrix

$$M = \begin{bmatrix} a & b & 0.4 \\ 0.9 & 0 & 0 \\ 0 & 0.6 & 0 \end{bmatrix}$$

where a and b are within the range $[0, 1]$.

Plot the region of values of a and b for which the size of the population described by the above model doesn't go to infinity when time goes to infinity.

Exercise 4 - naive solution (simulations)

```
clear all;
close all;
%stabilizacja
M=[0 0 0.4; 0.9 0 0;0 0.6 0];

a=linspace(0,1,100);
b=linspace(0,1,100);

%1st version - pure simulations
tic
nIt=150;
dY=zeros(length(a),length(b));
tol=1e-4;
for i=1:length(a)
    for j=1:length(b)
        M(1,1)=a(i);
        M(1,2)=b(j);
```

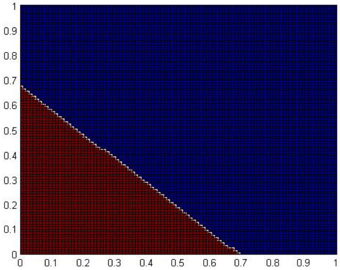
Exercise 4 - naive solution (simulations)

```
        Y=[65; 70; 60];
        for ii=1:nIt
            Yn=M*Y;
            diff=abs(sum(Y)-sum(Yn));
            Y=Yn;
            if Y>1e10
                break;
            end
        end
        dY(i,j)=diff;
    end
end
figure(1)
clf
surface(a,b,double(dY<tol))
toc
```

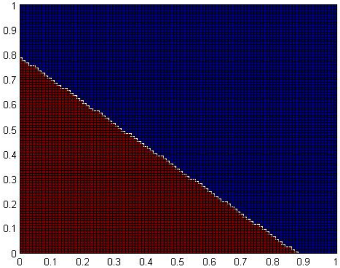
Exercise 4 - solution using eigenvalues

```
tic
mEig=zeros(length(a),length(b));
for i=1:length(a)
    for j=1:length(b)
        M(1,1)=a(i);
        M(1,2)=b(j);

        mEig(i,j)=max(real(eig(M)));
    end
end
figure(2)
clf
surface(a,b,double(mEig<=1));
toc
```

Comprazion of both approaches



Naive: ≈ 16 **seconds**
Highly dependent on settings!!!



More efficient: ≈ 0.4 **seconds**