Introduction

With the logistic map, $x_{n+1} = r \cdot x_n(1 - x_n)$, the stability of fixed points, bifurcations, period-doubling and chaos can be analyzed. It is mainly used to describe the behaviour of a population, with r being the *intrinsic growth rate*. In this assignment the bifurcation diagram of the logistic map will be computed and analyzed. [1]

Syntax

For this function there are several input arguments, firstly, λ_{min} and λ_{max} which define the interval for which the logistic map will be calculated. The initial value x_0, from which calculations will start. N, the number of values of λ between λ_{min} and λ_{max} for which the attractor will be calculated. n, for how many times x_{n+1} is calculated.

The output is the given figure, furthermore, the values of the matrix can be given, however this is not needed.

Description of the Method

First of all, default values are given to some of the decision variables to ensure that giving input variables is optional, except for λ_{min} and λ_{max}. These optional parameters are set to the default values of $N = 2000$, $x_0 = 0.5$, $n = 400$.

Then, λ_{min}, λ_{max} and x_0 are checked for correctness. An interval with the right amount of values is initiated by using the MATLAB function linspace. x is defined as a matrix full of zeroes, with the total length of the previously defined vector as the number of rows, and $n + 1$ columns, so that for each λ, n x_{n+1} values are calculated, as the first value is x_0.

The for-loop iterates over the x matrix incrementing i from 1 to n by 1. For each λ-value, x_{n+1} is calculated by the logistic map:

$$x_{n+1} = r \cdot x_n(1 - x_n)$$

To ensure a nice-looking and well estimated figure, only the 25% last n of the calculated x values are plotted.

Illustration

The figure is produced by commanding a plot of the values of r plotted against the values of the last 25% of x. As displayed, the decision variables are valued at: $\lambda_{min} = 0$, $\lambda_{max} = 4$, $N = 2000$, $x_0 = 0.5$, $n = 400$.

[1] Strogatz, S. H. (2000). Nonlinear Dynamics and Chaos. Cambridge: Westview Press.

The figure shows the bifurcation diagram of the logistic map with the shown parameters. For $r < 1$, the origin is the only fixed point. As r increases, the attractor keeps doubling in the number of fixed points, causing the origin to lose stability. x^* bifurcates from the origin in a **transcritical bifurcation** at $r = 1$, causing a change in stability of the fixed point. The attractor becomes a period-1 cycle if $r > 1$. At $r = 3$ there occurs a **flip bifurcation**, also known as a period-2 cycle. As r increases further and it passes around $r = 3.57$, chaos occurs, which means that x never settles into a fixed point or limit cycle, the period of the periodic cycles goes to ∞. At $r \approx 3.8$ there is a small period of stability, once again followed by chaos.[2]

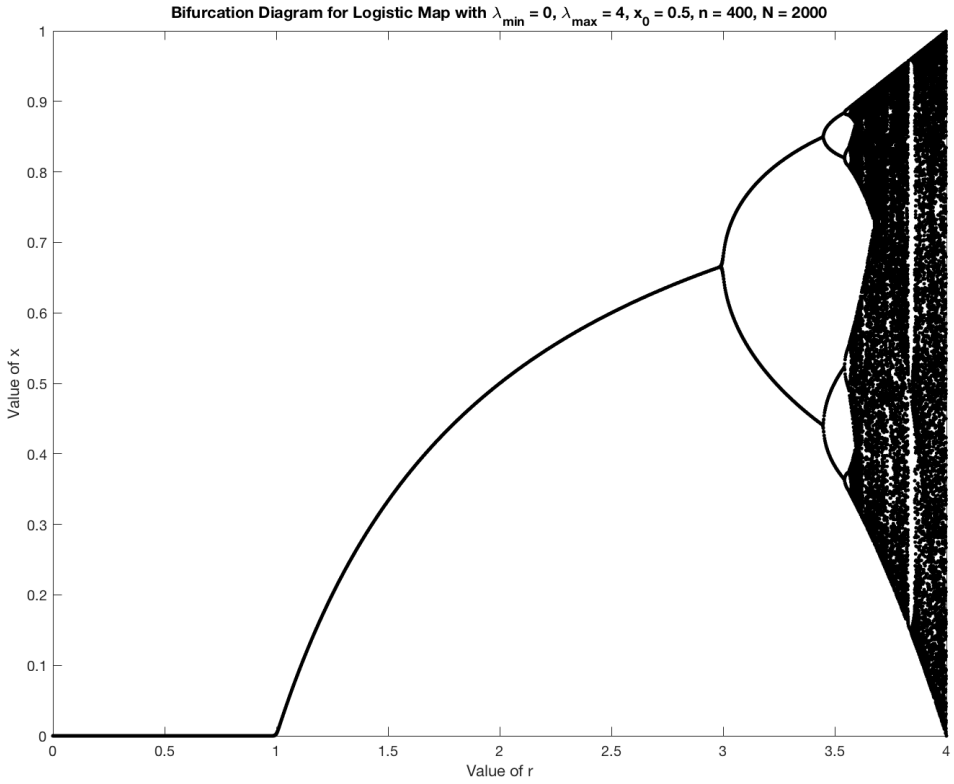

Figure 1: Output of `bf_logistic_map(0,4)`.

[2]Strogatz, S. H. (2000). Nonlinear Dynamics and Chaos. Cambridge: Westview Press.

Questions

Which value of x_0 do you choose?

Choosing x_0 to be either 0 or 1, will produce a straight line at $x_n = 0$, any other value will give a normal bifurcation diagram. The default value has been set to 0.5, which is also used in Figure 1.

Which values of n do you use in the picture?

Choosing the right value of n is a choice between performance and quality of the graph. You do not want a low value, as there will not be enough data points per λ-value. But you also do not want a too high of a value, as this will cost a lot of computational power without much gain. Finding an optimal value is about finding a balance. This was done by trial and error.

Only large n values should be plotted for a good estimation of the convergence. It was chosen to only plot the highest 25% of n values, a balance between amount plotted and amount calculated. For the actual (default) n-value, the options 200, 400, 600 and 800 were compared. With $n = 200$ there was too much scatter and with $n = 800$ it was too cluttered. This led us to choose a default value of $n = 400$, see Figure 2 for an example.

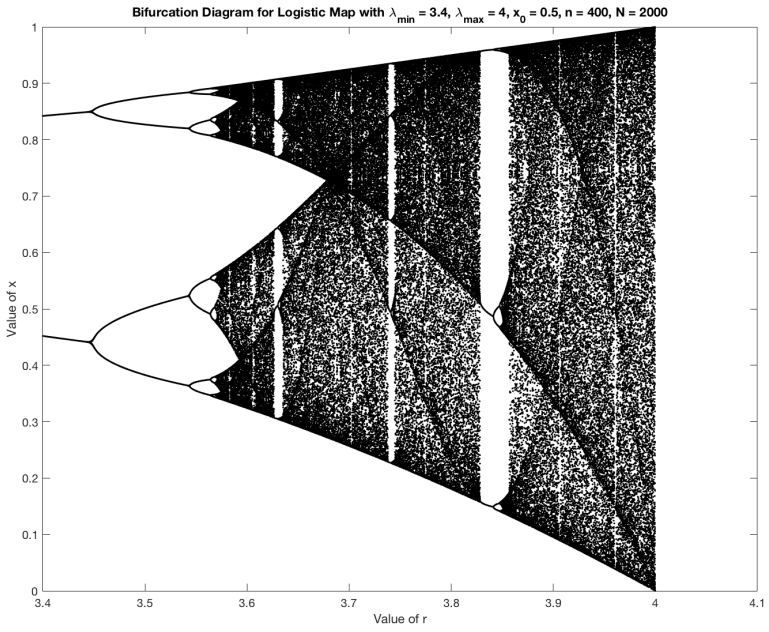

Figure 2: Output of `bf_logistic_map(3.4,4)`.

Performance

Due to our first implementation of the function being written with double for-loops instead of matrix multiplication, it was interesting to compare the new – presumed more efficient – code to the old implementation.

The test setup was a laptop with an Intel® Core™ i5-7360U@2.30 GHz processor running MATLAB R2019a. Both implementations of the code can be found in Appendix I and II. The code was benchmarked with the `tic` and `toc` commands in MATLAB. `tic` starts a timer and `toc` stops it and reports the time taken to complete. The functions were repeated 1,000 times and the resulting times saved in a `txt` file.

```
for i = 1:1000
tic; bf_logistic_map(2,4,2000);toc
end
for i = 1:1000
tic; bf_logistic_map_old(2,4,2000);toc
end
```

The resulting `txt` files were read by R, and saved in variables. A summary of the files can be found below.

```
> summary(new)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1187  0.1260  0.1267  0.1284  0.1283  0.2967
> summary(old)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1199  0.1263  0.1270  0.1274  0.1286  0.1470
> var(new)
[1] 0.0001164112
> var(old)
[1] 4.655628e-06
```

At first glance there does not seem to be a big difference between the performance of the two implementations. Upon further inspection we can observe that average runtime of the old—double for-loop—implementation is lower than that of the new matrix multiplication method – which was presumed to be faster. We can also observe that the variance is lower of the old method.

It is debatable that the double for-loop method is unanimously the best choice for this setup. However, higher values of n and N could be chosen by the user, which have not been extensively benchmarked. A quick test with $n = 8,000$ and $N = 40,000$, which is not really a real use case scenario but just a good measure, shows timings of 3.81 and 11.49 seconds for the new and old method respectively. Now MATLAB's matrix optimization definitely shows. However, this has only been a quick benchmark. Further research needs to be done in optimizing this method.

Appendix I: Function Implementation

```
function [] = bf_logistic_map(lambdaMin, lambdaMax, N, x_0, n)
% This function produces a picture of the "attractor" for a sequence
% of values lambda (r) of the logistic map:
%
%                  x_{n+1} = lambda * x_n * (1 - x_n)
%
%   bf_logistic_map(lambdaMin, lambdaMax, N, x_0, n)
%   tries to plot a bifurcation diagram of the logistic map with variables
%   lambdaMin and lambdaMax, which are the values of interest for the
%   parameter lambda which creates an interval. This interval is divided
%   into N parts. The interval should in between or at maximum [0,4].
%
%   x_0 is the initial value of x_n which is a number in [0,1]. x_n repre-
%   sents the ratio of existing population to the maximum possible
%   population.
%
%   The default values of N, x_0 and n are 2000, 0.5 and 400 respectively.

% check if all parameters are present, if not: fill in default values
if ~exist('N')
    N = 2000;
end;
if ~exist('x_0')
    x_0 = .5;
end;
if ~exist('n')
    n = 400;
end;

% check if parameters are correct
assert(lambdaMin>=0 && lambdaMin<=lambdaMax, ...
    'lambdaMin is not an element of [0, lambdaMax]');
assert(lambdaMax>=lambdaMin, lambdaMax<=4, ...
    'lambdaMax is not an element of [lambdaMin, 4]');
assert(x_0>=0 && x_0<=1, ...
    'x_0 is not an element of [0, 1]');

% create interval of [lambdaMin,lambdaMax] with N steps in between
% create vector of xs for every lambda
lambdas = linspace(lambdaMin, lambdaMax, N+2)';
x = zeros(length(lambdas), n+1);

% set initial x_0 on every row of x
x(:, 1) = x_0;

% apply the formula for every column, based on the previous column
% use point wise multiplication for better performance
for i = 1:n
    x(:, i+1) = lambdas .* x(:,i) .* (1 - x(:, i));
```

```
end;

% only use top 25% of n (large n)
x = x(:, .75*n:n);

% plot lambdas to x, black dots
plot(lambdas, x, 'black.', 'markersize', 2);

% add titles, axis and set window size
title(['Bifurcation Diagram for Logistic Map with ' ...
    '\lambda_{min} = ' num2str(lambdaMin) ...
    ', \lambda_{max} = ' num2str(lambdaMax) ...
    ', x_{0} = ' num2str(x_0) ...
    ', n = ' num2str(n) ...
    ', N = ' num2str(N)]);
xlabel('Value of r');
ylabel('Value of x');
set(gcf, 'Position',  [100, 100, 800, 600])

return;
```

Appendix II: Old Function Implementation (for Benchmark)

```
function [] = bf_logistic_map_old(lambdaMin, lambdaMax, N)

lambdas = linspace(lambdaMin, lambdaMax, N+2);
n = 400;
x = zeros(length(lambdas), n);

for i = 1:length(lambdas)
    x(i,1) = .5;
    for j = 1:n-1
        x(i,j+1) = lambdas(i) * x(i,j) * (1 - x(i,j));
        end
end

x = x(:,[n*.75:n]);

plot(lambdas, x, 'k.', 'markersize', 2);
title('Bifurcation diagram of the logistic map');
xlabel('r');  ylabel('x_n');
set(gca, 'xlim', [lambdaMin lambdaMax]);
set(gcf, 'Position',  [100, 100, 800, 600])

return;
```