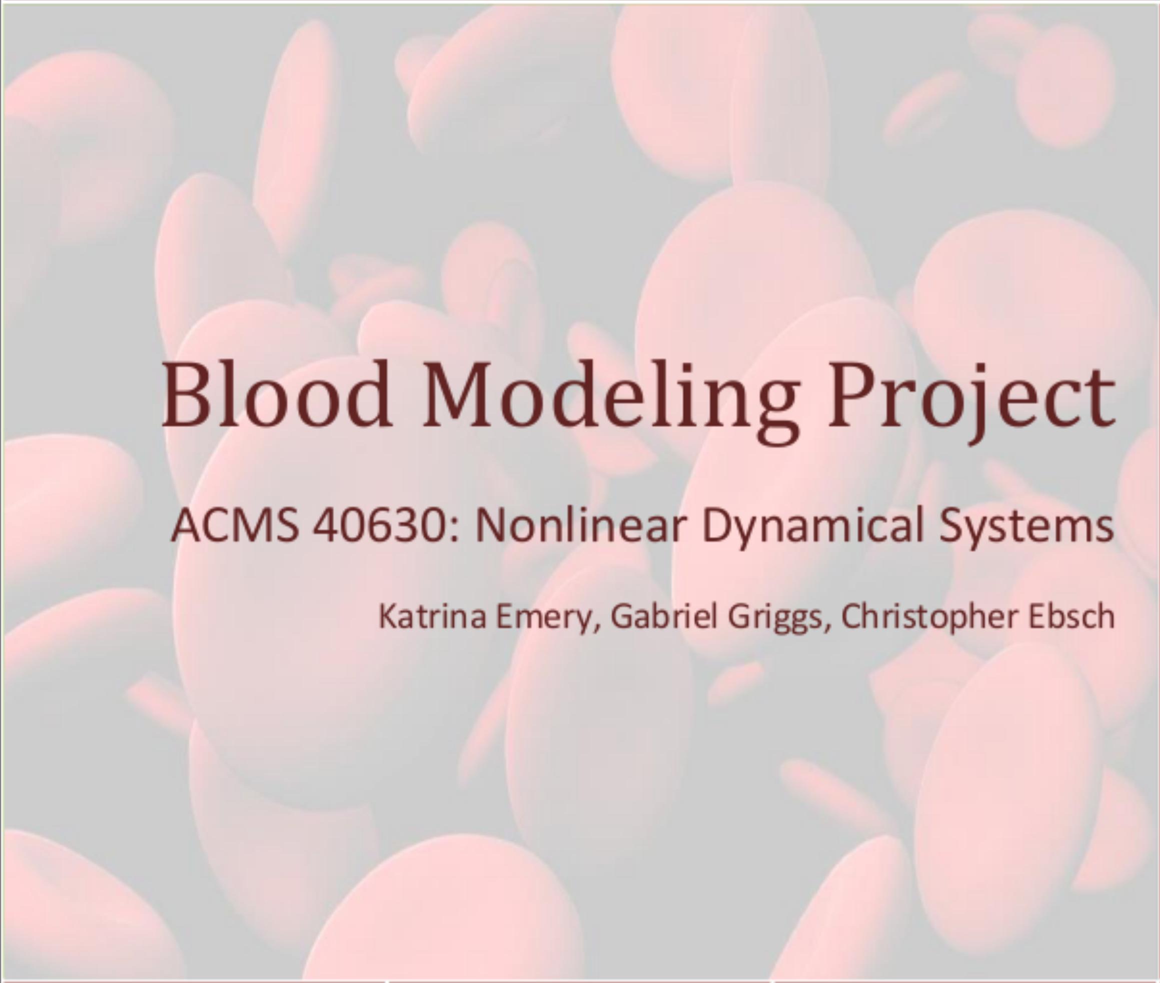
THE DREAM TEAM

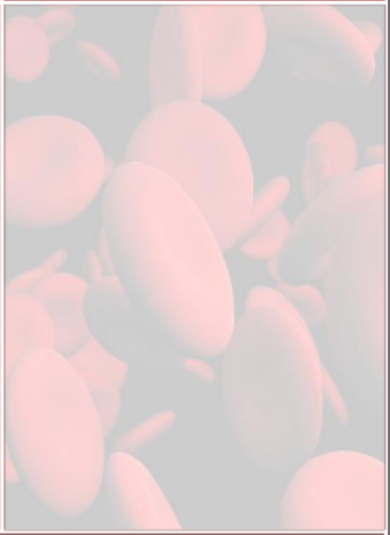


Blood Modeling Project

ACMS 40630: Nonlinear Dynamical Systems

Katrina Emery, Gabriel Griggs, Christopher Ebsch

2014



APPLIED & COMPUTATIONAL MATHEMATICS & STATISTICS

Introduction

The blood cell population model is an excellent example of a discrete dynamical system taken from biology. It helps illustrate "how mathematical modeling can be used to explain the behavior and possible origins of dynamical disease."[1] *This project will attempt to illustrate the significance of this model as well as investigate its discrete dynamics in terms of stability and chaos.*[2] [3]

On average, the human body contains 5 liters of blood consisting of red blood cells, white blood cells, and platelets. There also exist several types of red and white blood cells. This presents an issue from a mathematical modeling perspective because the "cellular and biochemical processes involved in [the cells] dynamics vary considerably."[4] Regardless of the challenges it presents, studying blood is extremely beneficial because it can be used to diagnose certain diseases in the human population such as "anemia, chronic renal failure, acute hemorrhaging and marrow failure."[5]

Red blood cells, or erythrocytes, tend to fluctuate in a healthy human body: woman having 4.2-5.4 per μL and men 4.7-6.1 per μL.[6] It is known that blood cell counts "can oscillate and even display chaotic behavior" which is not unexpected, for "all kids of nonlinear phenomena abound in the human body."[7] Through application of a blood cell model, these red blood cells can be counted to measure hematologic conditions such as anemia.[8][9] It is also helpful to point out that most blood cells are produced by "primitive stem cells in bone marrow."[10] The cells are then destroyed through processes such as natural aging, infection, or disease. This cyclic process is represented in the model as a nonlinear phenomenon.

[1]William B. Gearheart, "A Blood Cell Population Model Dynamical Disease, and Chaos," Department of Mathematics; California State University, Fullerton, n.d. Web. 12 Mar. 2014. ¡`http://users.mat.unimi.it/users/naldi/cell.pdf`¿

[2]Lynch, Stephen, "Analysis of a Blood Cell Population Model," International Journal of Bifurcation and Chaos 15, no. 7 (2005), 2311.

[3]Note: we have been asked to specify the work distribution for this project. Katrina contributed the background research, research on the applications of this model, the powerpoint presentation, the relevant section write-up and helped with compiling the report. Chris focused on the iterative numerics and trajectories, the bifurcation diagrams and wrote up the relevant sections. Gabe contributed the stability analysis of the fixed points, assisted in researching applications and historical significance, wrote up the relevant sections and compiled the final report.

[4]Lynch, 2311.

[5]Lynch, 2312.

[6]Steven H. Strogatz, "12.2: The Henon Map." In *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*, Reading, MA: Addison-Wesley Pub., 1994, 50.

[7]Lynch, 2311.

[8]Anemia is a condition marked by a deficiency of red blood cells or of hemoglobin in the blood, resulting in pallor and weariness (`www.nhblbi.nih.gov`). It is also defined as, "Any condition resulting in a significant decrease in total body erythrocyte mass" (Lynch, p. 285).

[9]Strogatz, 59.

[10]Lynch, 2312.

A Blood Cell Population Model

A simple blood cell population model was first developed by Polish mathematician, Andrzej Lasota, in 1977.[11] The model he considered was this:

$$c_{n+1} = c_n - d_n + p_n \tag{1}$$

where c_n denotes the red cell count at time n, and d_n and p_n represent the number of cells destroyed and produced in the interval, respectively. Furthermore, it is assumed that a constant fraction of cells is destroyed on each iteration, thus $d_n = ac_n$ where $0 < a \leq 1$, and a represents the "destruction coefficient." Also, consider the equation used by Lasota:

$$p_n = bc_n^r e^{-sc_n} \tag{2}$$

where b, c, r, and s are positive constants. Together, these functions give us the blood cell iterative equation used in this project[12]:

$$\begin{gathered} c_{n+1} = (1-a)c_n + bc_n^r e^{-sc_n} \text{ where} \\ 0 < a \leq 1 \text{ and } b, r, s > 0. \end{gathered}$$

The typical parameters used in the model are $b = 1.1x10^6$, $r = 8$, and $s = 16$. The analysis of this model is presented in the following sections of this project. Fixed points of period one and the stability of these points will be examined. To supplement the linear stability analysis, bifurcation diagrams will also be plotted as parameters change.[13] This will be implemented through MATLAB.

Brief Summary

Blood Cell Modeling can be utilized in order to explain the behavior and possible origins of dynamical diseases in the human population, such as anemia. A stability analysis can be carried out for a simple blood cell population model, as this project will attempt to demonstrate. Furthermore, bifurcation diagrams can be used to "establish how the models dynamics evolve as parameters vary." Please refer to the following sections.

[11]Strogatz, 59.
[12]Lynch, 2312.
[13]Lynch, 2314.

Consider the blood cell iterative equation $c_{n+1} = (1-a)c_n + bc_n^r e^{-sc_n}$.

Iterative Numerics

Question 1: Run iterative numerics and describe different types of trajectories.

Answer:

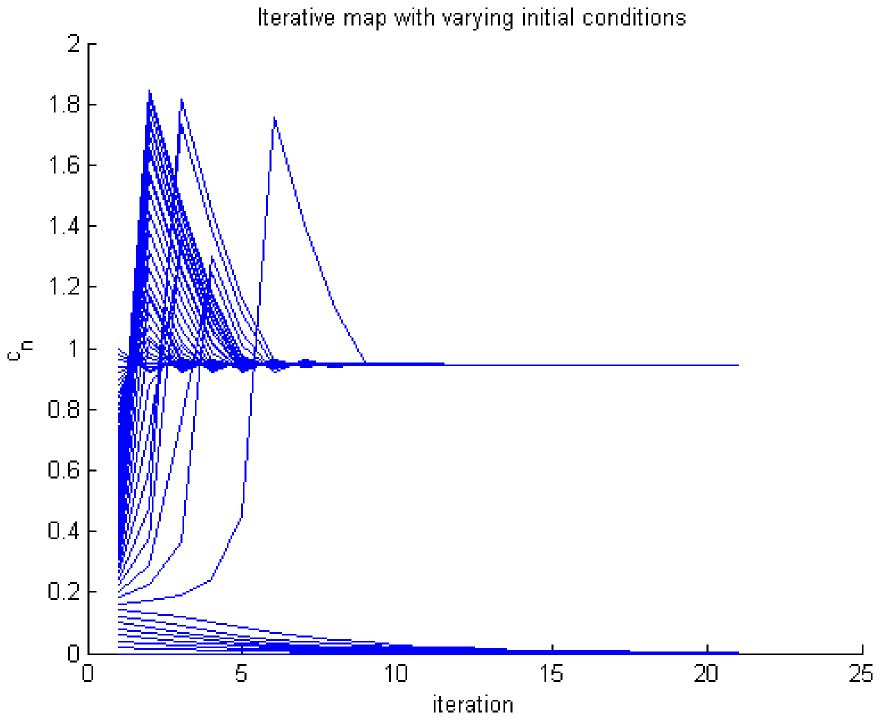
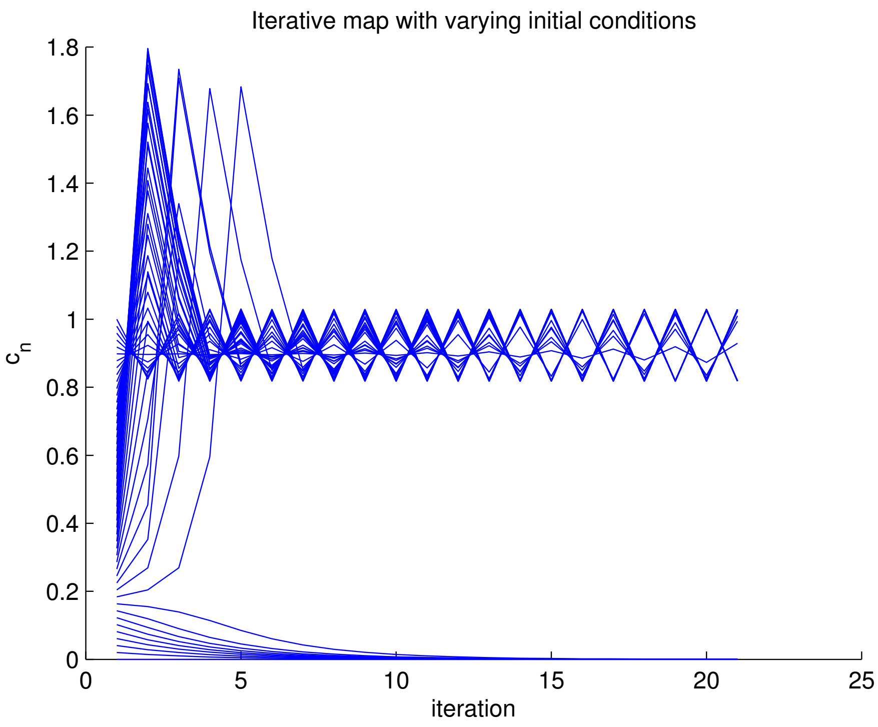


Figure 1: Map with 20 different initial conditions, with a = 0.2.

Figure 1 plots the map for 20 different initial conditions using typical parameters as described by Lasota in 1977, namely: $b = 1.1 * 10^{-6}$, $r = 8$, $s = 16$. Other possible parameter regimes have also been discussed by Steven Lynch. In general, $a \in [0, 1]$, but the 'a' parameter was chosen in this case to be in a region where the system can converge to one of two stable fixed points depending on the initial conditions.



center

center

Figure 2: Map with 20 different initial conditions with parameter a being changed to 0.3.

The only change made from figure 1 to figure 2 was the parameter 'a'. It now lies in a region with one stable solution and a 2-periodic solution. Figure 3 captures all of the behaviours that the system exhibits upon varying 'a' and the initial condition. At the bottom of the bifurcation diagram we can see that a single stable solution that goes to 0. As 'a' increases we see orbits of periods 2, 4, and 8 emerge. Then, undoubling of orbits occurs until $a \approx .7$ at which point we see bistability takes place. Figure 4 zooms in on the bistable region. Period doubling begins again at $a \approx .715$ before the system then goes to chaos, but sporadic instances of high periodicity orbits occur for $a \in (.725, .84)$. Figure 5 shows the system eventually becoming a single stable solution at $c_n = 0$ for large enough 'a' and any initial condition.

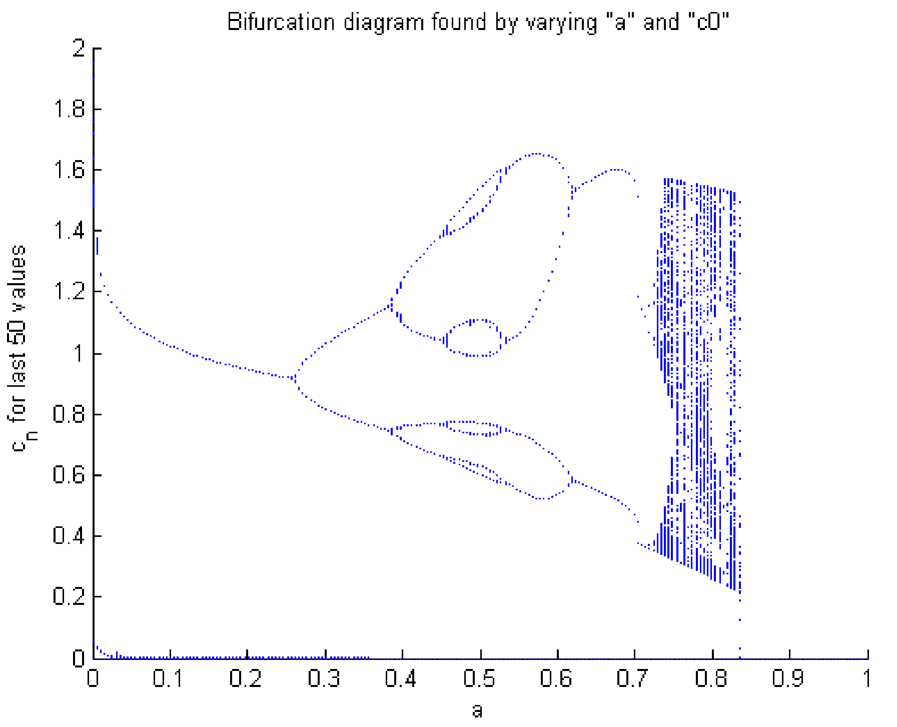


Figure 3: Behaviors of the system exhibited by varying initial conditions.

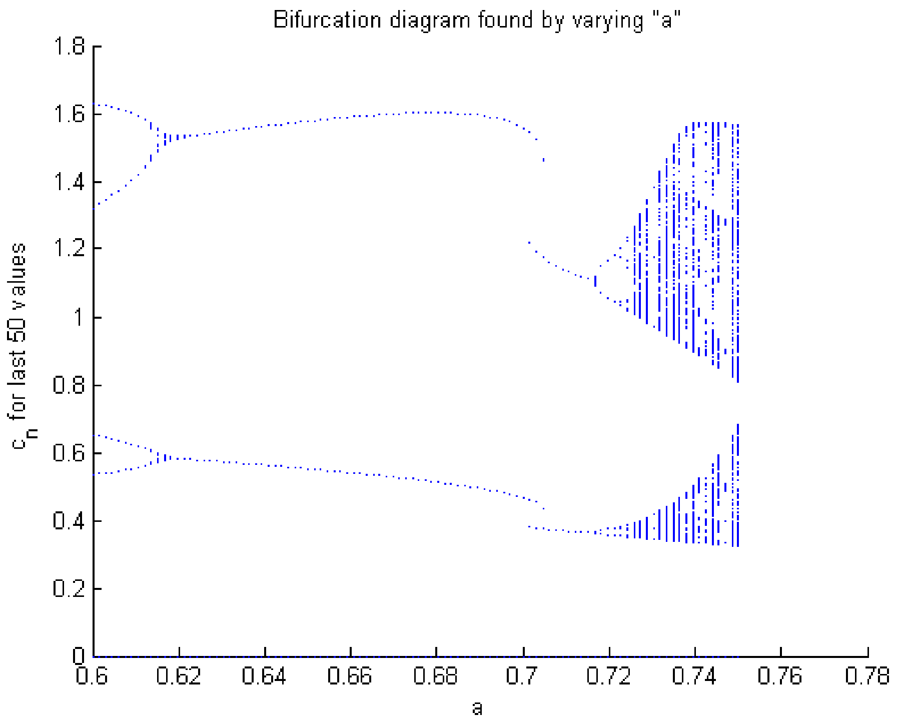


Figure 4: Bistable Region

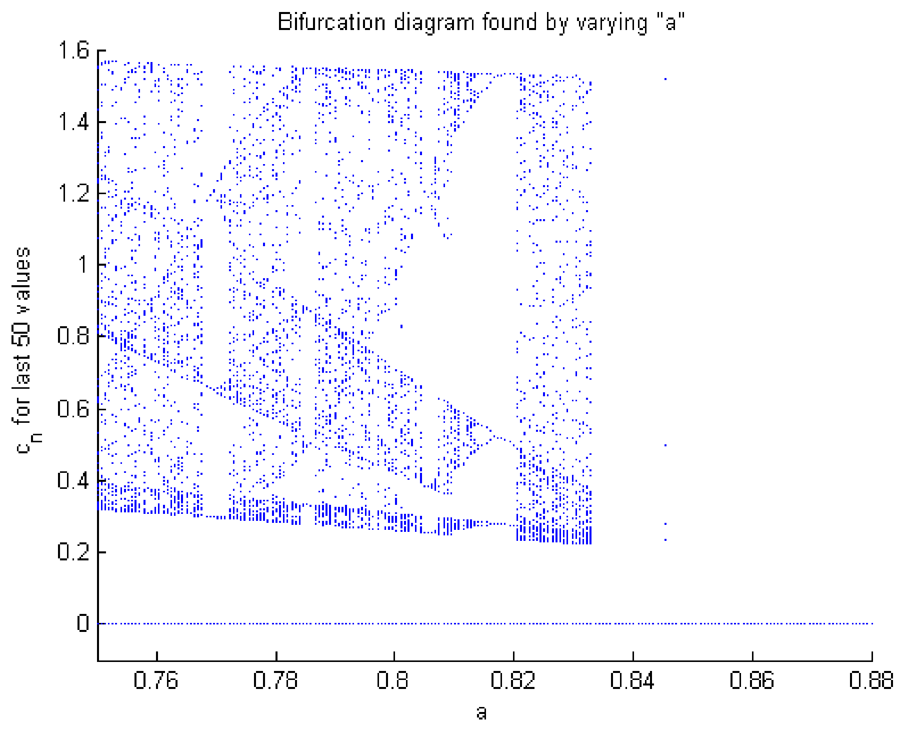


Figure 5: Chaotic to Stable Behavior

Blood Cell Fixed Point Calculations

Consider the blood cell iterative equation $c_{n+1} = (1-a)c_n + bc_n^r e^{-sc_n}$.

Question 2: Assuming that $b = 1.1 * 10^6$, $r = 8$ and $s = 16$ show that there are two stable and one unstable fixed points of period one when $a = 0.2$.

Answer: This question involves both a) finding the fixed points and b) determining the stability of these fixed points.

Finding Fixed Points: Because we are solving for *period one*, we can solve (a) by finding the intersection points of the line $y = x$ with $f(x) = (1-a)x + (1.1 * 10^6)x^8 e^{-16x}$. Usually we can determine these intersection points using numerical algorithms such as Newton's Method or the Secant Method. In this case, however, we found it difficult to get these methods to converge, so we used graphical estimates.

If we needed to approximate these roots numerically, we would take a Taylor Expansion of the function and use this in Newton's Method. In our case, however, graphical estimation with MATLAB's 'data cursor' yielded fixed points that fulfilled the requirements of the derivative test — that is, at these fixed points the absolute value of the derivative was greater than or less than 1 with the expected results.

Using the following MATLAB script, the functions and their derivatives were defined symbolically with the 'symfun' function and plotted against $y = x$:

```
% Set Parameters
a = .2; b = 1.1*10^6; r = 8; s = 16;
% Define Functions and Derivatives
syms x
f = symfun((1-a)*x+b*x^(r)*exp(-s*x),[x]); derivf = diff(f);
a = .3; g = symfun((1-a)*x+b*x^(r)*exp(-s*x),[x]); derivg = diff(g);
y = symfun(x,[x]);
% Plot F(x) vs Y(x)
h1 = ezplot(f,[-.01,1]); hold on; h2 = ezplot(y,[-.01,1]);
set(h1,'color','r','linestyle','--');
% Set Legend
legend('f(x)','y = x'); xlabel('x'); ylabel('f(x) and y(x)');
title('f(x) = (1-a)*x+b*x^(r)*exp(-s*x) against y(x) = x')
hold off
% Plot G(x) vs Y(x)
h3 = ezplot(g,[-.01,1]); hold on; h4 = ezplot(y,[-.01,1]);
set(h3,'color','r','linestyle','--')
% Set Legend
legend('g(x)','y = x');; xlabel('x'); ylabel('g(x) and y(x)');
title('g(x) = (1-a)*x+b*x^(r)*exp(-s*x) against y(x) = x')
```

With $a = 0.2$, this yielded the follow figure which shows that there are at least three fixed points of period one, represented by each intersection:

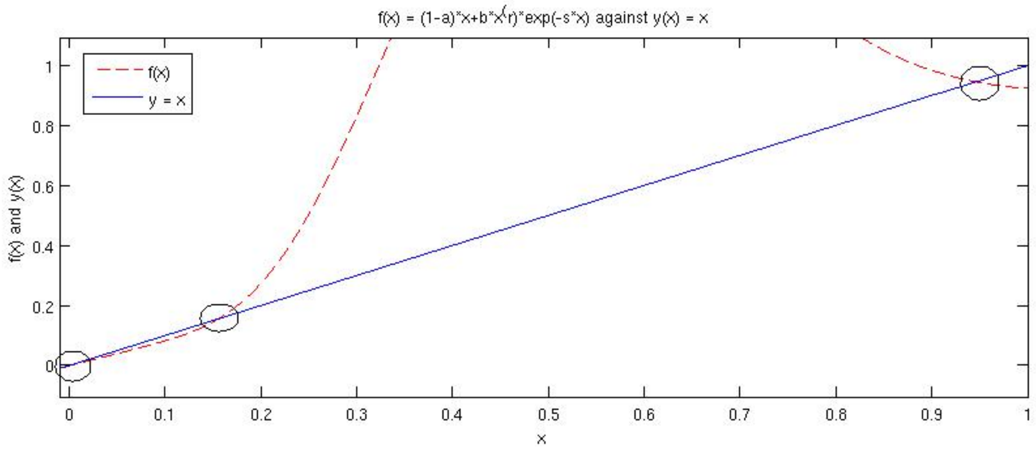


Figure 6: Here we can see that there are at least three fixed points of period one as there are three intersections. Note: the graph has cut off the top of the function for the sake of seeing the intersection points. The behavior of the function is simply concave and reprsents something close to a parabola in this window. For a further picture, see the attached graph at the end of this section.

Using MATLAB's 'data cursor' we can identify these points as $x*_1 = 0$, $x*_2 = 0.16$ and $x*_3 = 0.94$. The next step in our analysis is to find the stability at these points.

Determining Stability: Stability at these fixed points is determined by calculating the absolute value of the derivative at the fixed points. If this value is less than 1, then the fixed point is stable; if it is greater than 1, then the fixed point is unstable. In other words,

$$abs(f'(x*)) < 1 \rightarrow \text{fixed point is stable}$$
$$abs(f'(x*)) > 1 \rightarrow \text{fixed point is unstable}$$

For this system in general, we find that the derivative is:

$$\begin{aligned}\frac{df}{dx*}((1-a)x* + bx*^{r}e^{-sx*}) &= (1-a) + (r)bx*^{r-1}e^{-sx*} + (-s)bx*^{r}e^{-sx*}\\ &= (1-a) + bx*^{r-1}e^{-sx*}(r - sx*)\\ &= (1-a) + bx*^{r}e^{-sx*}\frac{r-sx*}{x*}\end{aligned}$$

Furthermore, with this result, we find that:

$$\frac{df}{dx*} < 1 \text{ when } a > bx*^{r-1}e^{-sx*}(r - sx*)$$

We could use this result to determine stability at fixed points for any given set of parameters at any given fixed point. In our case, however, we have used MATLAB's symbolic toolbox to calculate the derivatives, seen as 'derivf = diff(f)' in the code above. We can then calculate the absolute value of the derivatives at those points yielding the following results:

Fixed Point	Abs(f'(x*))	Stability
0	0.8000	stable
0.16	2.0418	unstable
0.94	0.6759	stable

Thus, we have shown that there are two stable fixed points and one unstable fixed point within this system when $a = 0.2$.

Question 3: Assuming that $b = 1.1 * 10^6$, $r = 8$ and $s = 16$ show that there are one stable and two unstable fixed points of period one when $a = 0.3$.

Answer: We use the same techniques as the last problem in order to show this to be true.

Here is the graph that is generated by plotting $y = x$ against our $g(x) = (1-a)x + (1.1 * 10^6)x^8e^{-16x}$ where $a = 0.3$:

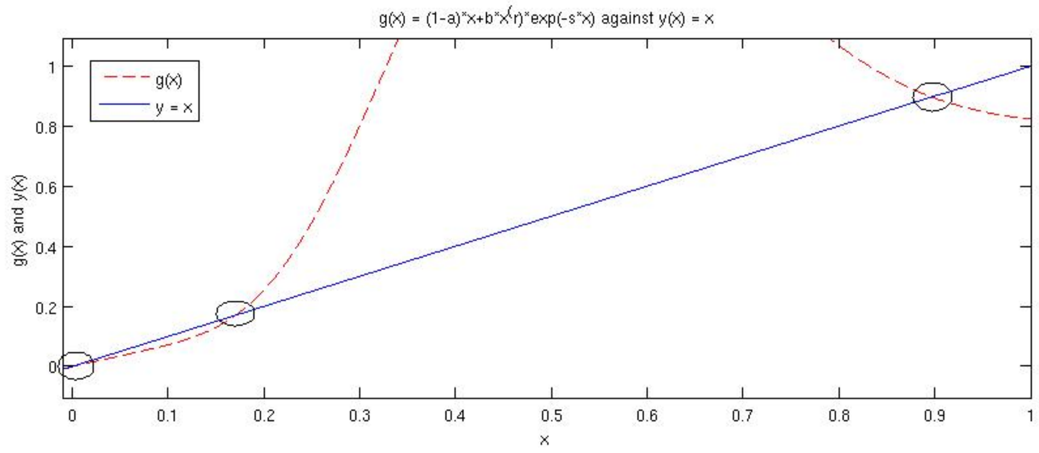


Figure 7: Here, again, we can see that there are at least three fixed points of period one as there are three intersections.

Upon analysis with MATLAB's 'data cursor', it was found that the fixed points seem to differ slightly with values of $x*_1 = 0$, $x*_2 = 0.17$ and $x*_3 = 0.89$. Taking the derivatives at these points yields the expected results of two unstable fixed points and one stable fixed point:

Fixed Point	Abs(g'(x*))	Stability
0	0.7000	stable
0.17	2.2700	unstable
0.89	1.2859	unstable

Complete Diagram of Function from $x = -0.1$ to 1

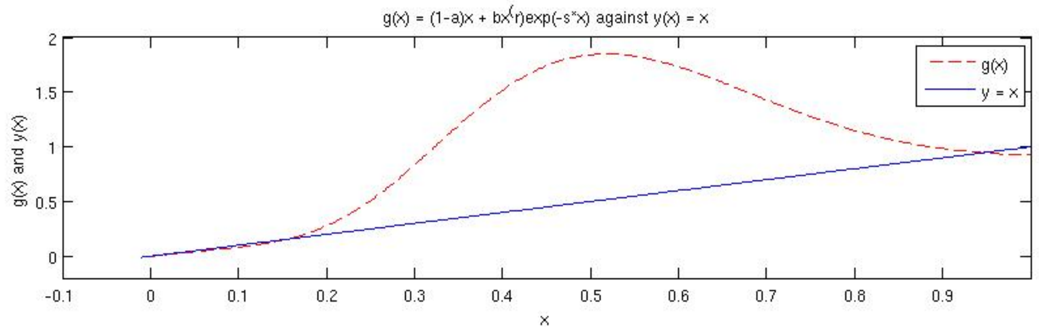


Figure 8: Here we see the complete graph of the function, particularly the parabolic aspect which was cut off in the previous pictures. Furthermore, our function appears to lie underneath the diagonal as Cn approaches infinity. This suggests that these may be the only fixed points for this system of period one, although there are fixed points of other periods.

Bibliography

`Dictionary.com`. “Bifurcate.” *Dictionary.com.* `http://dictionary.reference.com/browse/bifurcate`. (accessed March 26, 2014)

Weisstein, Eric W. “Bifurcation.” From MathWorld—A Wolfram Web Resource. `http://mathworld.wolfram.com/Bifurcation.html`

Gearhart, William B. , and Mario Martelli. “A Blood Cell Population Model, Dynamical Diseases, and Chaos.” Department of Mathematics; California State University, Fullerton. `http://users.mat.unimi.it/users/naldi/cell.pdf` (accessed March 12, 2014).

Lynch, Stephen. “Analysis of a Blood Cell Population Model.” International Journal of Bifurcation and Chaos 15, no. 7 (2005): 2311-2316.

Lynch, Stephen. “Chapter 12.” *Dynamical Systems with Applications Using Mathematica.* Boston, MA: Birkhäuser, 2007. N. pag. Print.

Strogatz, Steven H. “12.2: The Henon Map.” In *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering.* Reading, MA: Addison-Wesley Pub., 1994.

“What Is Anemia?.” - NHLBI, NIH. `https://www.nhlbi.nih.gov/health/health-topics/topics/anemia/` (accessed March 22, 2014).

MATLAB Appendix: Scripts for Generating Bifurcation Diagrams and Iterative Numerics

```
function bloodModelVaryac0(steps,b,r,s)

% Pressing "Run" button uses the below values as the inputs
%   for the function.
if(nargin==0)
    steps=200;
    b=1.1*10.^6;
    r=8;
    s=16;
end

ICs=10; % Number of initial conditions
c0=linspace(0,1,ICs); % Initial conditions tested between 0 and 1.5
runs=70; % Number of different ‘a’ values to test
a=linspace(.6,.75,runs).’; % Linearly spaced vector of ‘a’ values
cn=zeros([1, steps]);

%Starts clock on main loop
t=cputime;

% Loop through initial condition values
for k=1:ICs
    cn(1)=c0(k);
    % Loop through the ‘a’ values
    for j=1:runs
        % Loop through the first (steps-50) steps
        for i=1:steps-50
        cn(i+1)=(1-a(j)).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
    end

    % Plots new data on the same graph
    hold on

    % Loop through final steps while plotting bifurcation diagram.
    % Plotting just these steps ensures that we only use values that
    %   are in the periodic orbits that the map converges to for a given
    %   ‘a’.
    for i=steps-50:steps
        cn(i+1)=(1-a(j)).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
        plot(a(j), cn(i),‘-’, ‘MarkerSize’,2);
    end
    end
end
% Determines how long the main loop takes to run
loopTime=cputime-t;
display(loopTime);

xlabel(‘ a ’);
ylabel(‘ c_n for last 50 values’)
title(‘Bifurcation diagram found by varying ‘‘a’’’);

end

---

function bloodModelr5(c0,steps,b,r,s)

% Pressing ‘‘Run’’ button uses the below values as the inputs
%   for the function.
if(nargin==0)
    c0=1; % Initial condition
    steps=100; % Number of iterations of the map
    b=1.1*10.^6; % Value of parameter given in part (b)
    r=4.9;         % "                                  "
    s=16;        % "                                   "
end

a=.2; % Value given in part (b)
% a=.3;  % Value given in part (c)
cn=zeros([1,steps]); % Create a vector to hold value at each step
cn(1)=c0; % Initial condition

% Main loop to generate iterative map
for i=1:steps
    cn(i+1)=(1-a).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
end

% Simple plot of result
figure
plot(cn(:));
end

---
function bloodModelSimple(c0,steps)
```

```
if(nargin==0)
    c0=linspace(0,1,50);
    steps=20;
end
a=.3;
b=1.1*10.^6;
s=16;
r=8;
cn=zeros([1 steps]);
xlabel('iteration');
ylabel('c_n');
title('Iterative map with varying initial conditions');
for k=1:50
    cn(1)=c0(k);
    for j=1:steps
        cn(j+1)=(1-a).*cn(j)+b*cn(j).^r*exp(-s*cn(j));
    end
    hold on
    plot(cn,'-');
end
end

---
function bloodModelSinglea(c0,steps,b,r,s)

if(nargin==0)
    c0=1.5;
    steps=1000;
    b=1.1*10.^6;
    r=8;
    s=16;
end

runs=200;
a=linspace(0,1,runs).';
cn=zeros([1, steps]);
cn(1)=c0;

for j=1:runs

    for i=1:steps-100
        cn(i+1)=(1-a(j)).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
    end
    hold on
    for i=steps-100:steps
```

```
        plot(a(j), cn(i),'-', 'MarkerSize',2);
    end
end
end


---

function bloodModelVaryA(c0,steps,b,r,s)

% Pressing "Run" button uses the below values as the inputs
%   for the function.
if(nargin==0)
    c0=1.5;
    steps=300;
    b=1.1*10.^6;
    r=8;
    s=16;
end

runs=500; % Number of different 'a' values to test
a=linspace(0,1,runs).'; % Linearly spaced vector of 'a' values
cn=zeros([1, steps]);
cn(1)=c0;

% Loop through the 'a' values
for j=1:runs

    % Loop through the first (steps-50) steps
    for i=1:steps-50
        cn(i+1)=(1-a(j)).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
    end

    % Plots new data on the same graph
    hold on

    % Loop through final steps while plotting bifurcation diagram.
    % Plotting just these steps ensures that we only use values that
    %   are in the periodic orbits that the map converges to for a given
    %   'a'.
    for i=steps-50:steps
        cn(i+1)=(1-a(j)).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
        plot(a(j), cn(i),'-', 'MarkerSize',2);
    end
end
```

```
xlabel(‘ a ’);
ylabel(‘ c_n for last 100 n values’)
title(‘Bifurcation diagram found by varying "a"’);

end


---

function bloodModelVaryac0(steps,b,r,s)

% Pressing "Run" button uses the below values as the inputs
%   for the function.
if(nargin==0)
    steps=200;
    b=1.1*10.^6;
    r=8;
    s=16;
end

ICs=10; % Number of initial conditions
c0=linspace(0,1,ICs); % Initial conditions tested between 0 and 1.5
runs=70; % Number of different ‘a’ values to test
a=linspace(0,1,runs).’; % Linearly spaced vector of ‘a’ values
cn=zeros([1, steps]);

%Starts clock on main loop
t=cputime;

% Loop through initial condition values
for k=1:ICs
    cn(1)=c0(k);
    % Loop through the ‘a’ values
    for j=1:runs
        % Loop through the first (steps-50) steps
        for i=1:steps-50
        cn(i+1)=(1-a(j)).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
    end

    % Plots new data on the same graph
    hold on

    % Loop through final steps while plotting bifurcation diagram.
    % Plotting just these steps ensures that we only use values that
    %   are in the periodic orbits that the map converges to for a given
    %   ‘a’.
    for i=steps-50:steps
        cn(i+1)=(1-a(j)).*cn(i)+b*cn(i).^r*exp(-s*cn(i));
        plot(a(j), cn(i),‘-’, ‘MarkerSize’,2);
    end
    end
end
% Determines how long the main loop takes to run
loopTime=cputime-t;
display(loopTime);

xlabel(‘ a ’);
ylabel(‘ c_n for last 50 values’)
title(‘Bifurcation diagram found by varying ‘‘a’’’);

end

---

function bloodModelvaryr(c0,steps,b,s)

% Pressing "Run" button uses the below values as the inputs
%   for the function.
if(nargin==0)
    c0=.5; % Initial condition
    steps=300; % Number of iterations of the map
    b=1.1*10.^6; % Value of parameter given in part (b)
    s=16;        % "                                  "
end

rVals=300;
r=linspace(0,10,rVals);
% a=.2; % Value given in part (b)
a=.3;  % Value given in part (c)
cn=zeros([1,steps]); % Create a vector to hold value at each step
cn(1)=c0; % Initial condition

t=cputime;
% Main loop to generate iterative map
for k=1:rVals

    for i=1:steps-100
        cn(i+1)=(1-a).*cn(i)+b*cn(i).^r(k)*exp(-s*cn(i));
    end
    hold on
    for i=steps-100:steps
        cn(i+1)=(1-a).*cn(i)+b*cn(i).^r(k)*exp(-s*cn(i));
        plot(r(k),cn(i),'-', 'MarkerSize',2);
    end
end
loopTime=cputime-t;
display(loopTime);
xlabel(' r ');
ylabel(' c_n for last 50 values')
title('Bifurcation diagram found by varying "r"');
end

---

function bloodModelVarys(c0,steps,b,r)

% Pressing "Run" button uses the below values as the inputs
%   for the function.
if(nargin==0)
    c0=.5; % Initial condition
    steps=200; % Number of iterations of the map

    b=1.1*10.^6; % Value of parameter given in part (b)
    r= 8;        % "                                 "
    % s=16;        % "                                 "
end

sVals=200;
s=linspace(8,20,sVals);
% a=.2; % Value given in part (b)
a=.3;  % Value given in part (c)
cn=zeros([1,steps]); % Create a vector to hold value at each step
cn(1)=c0; % Initial condition

t=cputime;
% Main loop to generate iterative map
for k=1:sVals

    for i=1:steps-20
        cn(i+1)=(1-a).*cn(i)+b*cn(i).^r*exp(-s(k)*cn(i));
    end
    hold on
    for i=steps-20:steps
        cn(i+1)=(1-a).*cn(i)+b*cn(i).^r*exp(-s(k)*cn(i));
        plot(s(k),cn(i),'-', 'MarkerSize',2);
    end
end
loopTime=cputime-t;
display(loopTime);
xlabel(' r ');
ylabel(' c_n for last 20 values')
title('Bifurcation diagram found by varying "s"');
end

---

function c0findBifurcation
    format long
    upper=.2;
    lower=.1;
    steps=50;
    a=.3;
    b=1.1*10.^6;
    s=16;
    r=8;
    cn=zeros([1 steps]);
    c0=.15;
    tol = 1*10.^(-13);
    c0temp=1;
    while abs(c0temp-c0) > tol

        cn(1)=c0;
        for j=1:steps-1
            cn(j+1)=(1-a).*cn(j)+b*cn(j).^r*exp(-s*cn(j));
        end
        if cn(steps)< .5
            c0temp=c0;
            lower=c0;
            c0=(upper+c0)/2;
        else
            c0temp=c0;
            upper=c0;
            c0=(lower+c0)/2;
        end
    end
    display(c0);
```