Sahib Azgar

ME456S Computational Fluid Mechanics

Due: 01/24/2024

Prof Allshouse

Problem 1 - Tent Map: Maps can be more complex than a simple continuous function. One common form is a "piecewise" map. This map performs a different function depending on the value of the input. The map we will use is:

$$x_{n+1} = \begin{cases} 2x_n, & \text{if } x_n \le \frac{1}{2}, \\ 2(1-x_n), & \text{if } x_n > \frac{1}{2}. \end{cases} \tag{1}$$

Starting with $x_0 \in [0,1]$, this map generates a sequence $\{x_n\}_{n\ge1}$ in $[0,1]$.

(a) Derive the fixed points by hand.

(Hand derivation)

$x_{n+1} = 2x_n \rightarrow x_n \le \frac{1}{2}$ $\quad$ $x_{n+1} = 2(1-x_n) \rightarrow x_n > \frac{1}{2}$

$x^* = 2x^*$ $\quad$ $x^* = 2 - 2x^*$

$-x^* = 0$ $\quad$ $+2x^* \quad +2x^*$

$x^* = 0$ $\quad$ $\frac{3x^*}{3} = \frac{2}{3}$

$x^* = \frac{2}{3}$

Fixed Points are $x^* = 0$ & $x^* = 2/3$

(b) Determine the stability of the fixed points analytically.

(Hand derivation)

$F(x^*, \lambda) = 2x_n$ $\quad$ $F(x^*, \lambda) = 2(1-x_n) = 2 - 2x_n$

$F'(x^*, \lambda) = 2$ $\quad$ $F'(x^*, \lambda) = -2$

$F'(x^* = 0) = 2$ $\quad$ $F'(x^* = 2/3) = -2$ $\quad$ $|-2| = 2$ → unstable

$$|F'(x^*, \lambda)| = \begin{cases} <1 \rightarrow x^* \text{ stable} \\ >1 \rightarrow x^* \text{ unstable} \\ =1 \rightarrow x^* \text{ semi-stable} \end{cases}$$

$x^* = 0$ is unstable fixed point, $x^* = \frac{2}{3}$ is unstable fixed point

(c) There is a 2-cycle that exists in this map. Derive the values that are repeated in this 2-cycle. Is this the only 2-cycle in the system?

(Hand derivation)

$$x_{n+1} = \begin{cases} 2x_n & \text{if } x_n \le 0.5 \\ 2 - 2x_n & \text{if } x_n > 0.5 \end{cases}$$

Branch 1

$x_{n+1} = 2(2x_n) = 4x_n \rightarrow 4x^* = x^*$

$x^* = 0$ fixed pt

Branch 2

$x_{n+1} = 2 - 2(2x_n) = 2 - 4x_n \rightarrow 2 - 4x^* = x^*$

$\hookrightarrow 2 = 5x^* \rightarrow x^* = \frac{2}{5}$

Branch 3

$x_{n+1} = 2 - 2(2 - 2x_n)$

$= 2 - 4 + 4x_n = -2 + 4x_n$

$-2 + 4x^* = x^* \rightarrow -2 = -3x^* \rightarrow x^* = \frac{2}{3}$ fixed pt

Branch 4

$x_{n+1} = 2x_n = 2(2 - 2x_n) = 4 - 4x_n$

$4 - 4x^* = x^*$

$4 = 5x^* \rightarrow x^* = 0.8$

$2(0.8) \rightarrow 1.6 > 0.5$ not ≤ 0.5

$\downarrow$

$2 - 2(1.6) < 0.5$ not ≥ 0.5

$\hookrightarrow$ two cycle pt

$2\left(\frac{2}{5}\right) \rightarrow 0.8 > 0.5$

$\hookrightarrow 2 - 2(0.8) = 0.4 < 0.5$ not > 0.5] goes back & forth

two cycle pts

$x^* = 2/5$ & $x^* = 4/5$

Problem 1d) To evaluate the stability of the 2-cycle, we will write a code to perform the tent map. Modify the map script in ps2p1_template.m. Update the script to start the map at one of the values of the 2-cycle and perform the map 60 times.

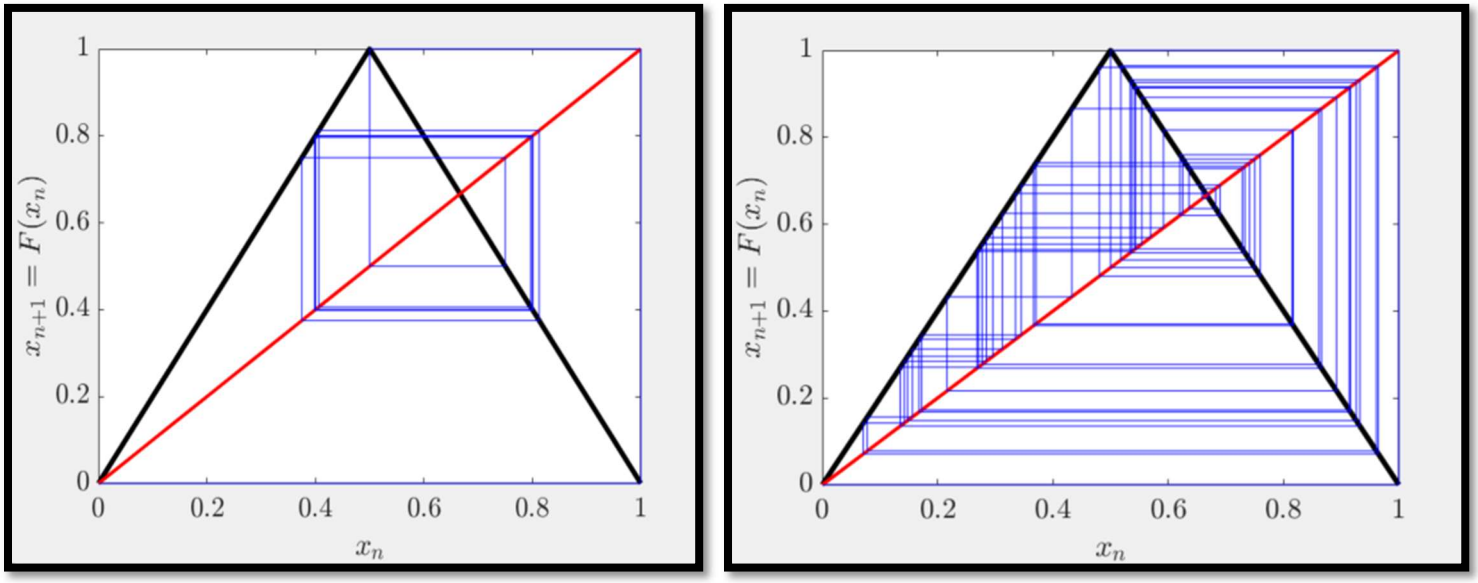

Figure 1: Left Plot shows it starting map on one of two cycles (x*=0.4). Right plot shows it when x(1) = rand.

Script for Problem 1d Below: (HW2P1_D_and_E)

```
clear; close all; clc;

%% Create the mapping function plot and the x_n=x_n+1 line
plot([0 .5 1],[0 1 0],'k','LineWidth',3);
hold on;
plot([0 1],[0 1],'r','LineWidth',2);
xlabel('$$x_n$$','FontSize',24,'Interpreter','latex')
ylabel('$$x_{n+1}=F(x_n)$$','FontSize',24,'Interpreter','latex')
set(gca,'FontSize',16,'TickLabelInterpreter','latex')

%% Set the number of iterations
N = 60;

%% Create the variable x to store the values of the mapping
x = zeros(N,1);

%% Set the initial condition
x(1) = 0.4;

%% Perform a for loop to execute the mapping
for n = 1:N-1
    %% Set the current state
    x_n = x(n); % Use "x_n" for the current state

    %% Perform the piecewise map to calculate x_np1 (x_(n+1))
```

```
    % Use "x_np1" to represent the next state (the x_(n+1) state)
    if x_n <= 0.5
        x_np1 = 2 * x_n;
    else
        x_np1 = 2 * (1 - x_n);
    end

    %% Store the next state in the x-vector
    x(n+1) = x_np1;

    %% Plot the mapping plot
    plot([x_n x_n x_np1],[x_n x_np1 x_np1],'b');
    pause(0.5);
end

%% Plot the state as a function of iteration number (part e addition)
figure;
plot(1:N, x, '-o', 'LineWidth', 2);
xlabel('Iteration Number','FontSize',16);
ylabel('State','FontSize',16);
title('State as a Function of Iteration','FontSize',18);
grid on;
```

Problem 1e) Plot the state as a function of iteration number. Based on the result, is the 2-cycle stable?

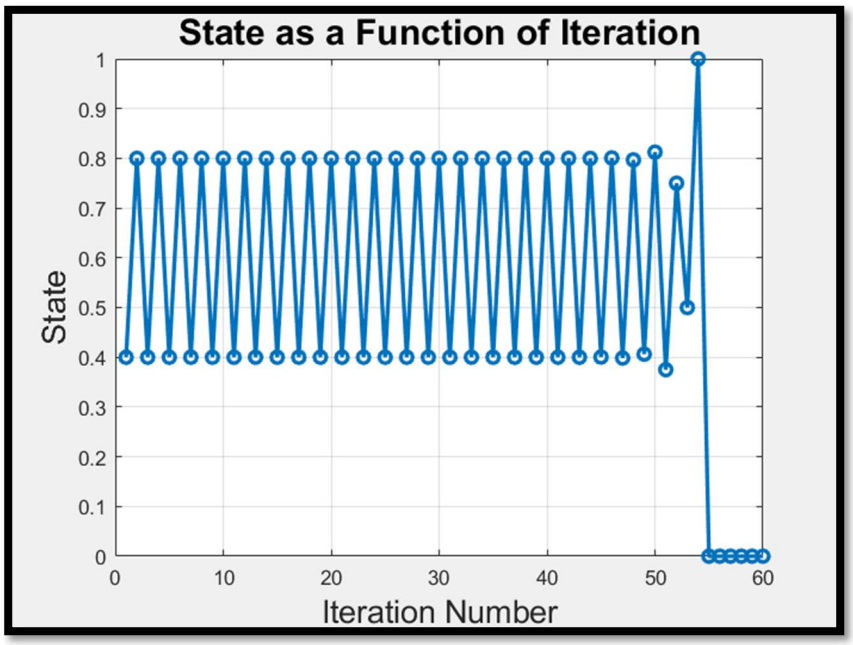

Figure 2: State as a Function of Iteration Plot after 60 times when x(1) = 0.4

Comments: The two cycle is unstable. The graph shows that x*=0.4 and x*=0.8 are oscillating back and forth but at the 50th iteration when the plot goes away from the fixed point, it does not return the

value. Thus, after the 50th iteration, it does not return to the fixed points. This behavior indicates that these fixed points are unstable.

Script for Problem 1e below: (HW2P1_D_and_E)

```
clear; close all; clc;

%% Create the mapping function plot and the x_n=x_n+1 line
plot([0 .5 1],[0 1 0],'k','LineWidth',3);
hold on;
plot([0 1],[0 1],'r','LineWidth',2);
xlabel('$$x_n$$','FontSize',24,'Interpreter','latex')
ylabel('$$x_{n+1}=F(x_n)$$','FontSize',24,'Interpreter','latex')
set(gca,'FontSize',16,'TickLabelInterpreter','latex')

%% Set the number of iterations
N = 60;

%% Create the variable x to store the values of the mapping
x = zeros(N,1);

%% Set the initial condition
x(1) = 0.4;

%% Perform a for loop to execute the mapping
for n = 1:N-1
    %% Set the current state
    x_n = x(n); % Use "x_n" for the current state

    %% Perform the piecewise map to calculate x_np1 (x_(n+1))
    % Use "x_np1" to represent the next state (the x_(n+1) state)
    if x_n <= 0.5
        x_np1 = 2 * x_n;
    else
        x_np1 = 2 * (1 - x_n);
    end

    %% Store the next state in the x-vector
    x(n+1) = x_np1;

    %% Plot the mapping plot
    plot([x_n x_n x_np1],[x_n x_np1 x_np1],'b');
    pause(0.5);
end

%% Plot the state as a function of iteration number (part e addition)
figure;
plot(1:N, x, '-o', 'LineWidth', 2);
xlabel('Iteration Number','FontSize',16);
ylabel('State','FontSize',16);
title('State as a Function of Iteration','FontSize',18);
grid on;
```

Problem 1f) Run the mapping script starting 0.001 away from one of the 2-cycle locations. Plot the state as a function of the iteration number. Comment on the dynamics of the map.

2 cycle locations from part c are x*=0.4 and x*=0.8

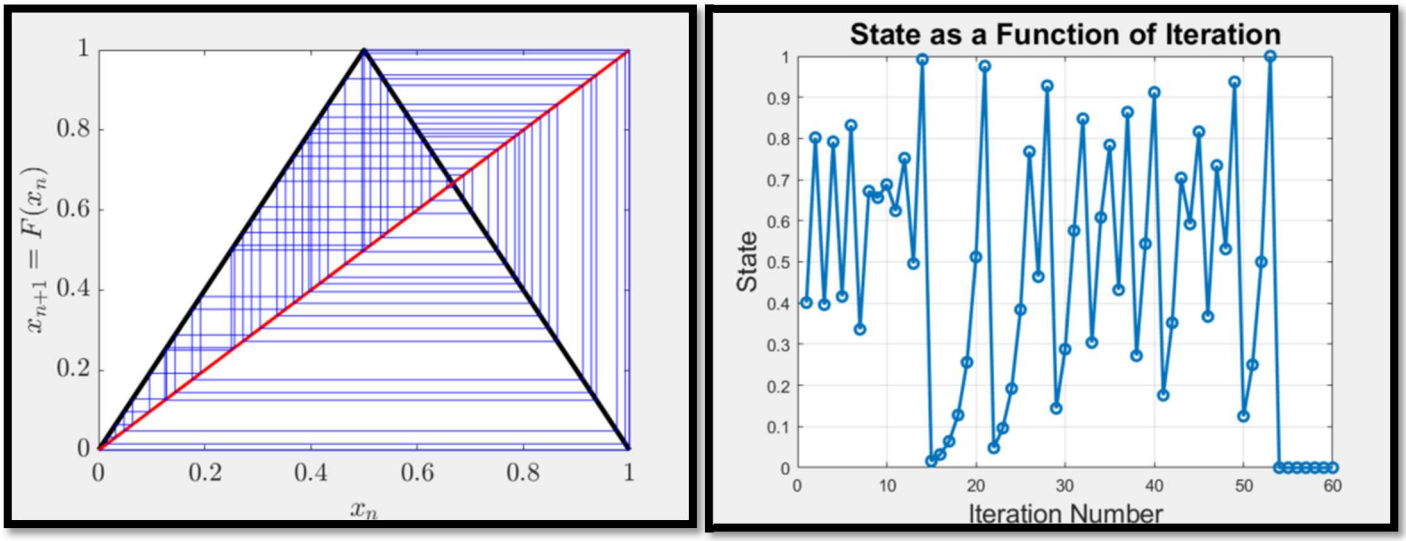

Figure 3: Tent Map and State as a Function of Iteration Plot after 60 times when x*=0.4 was added with 0.001

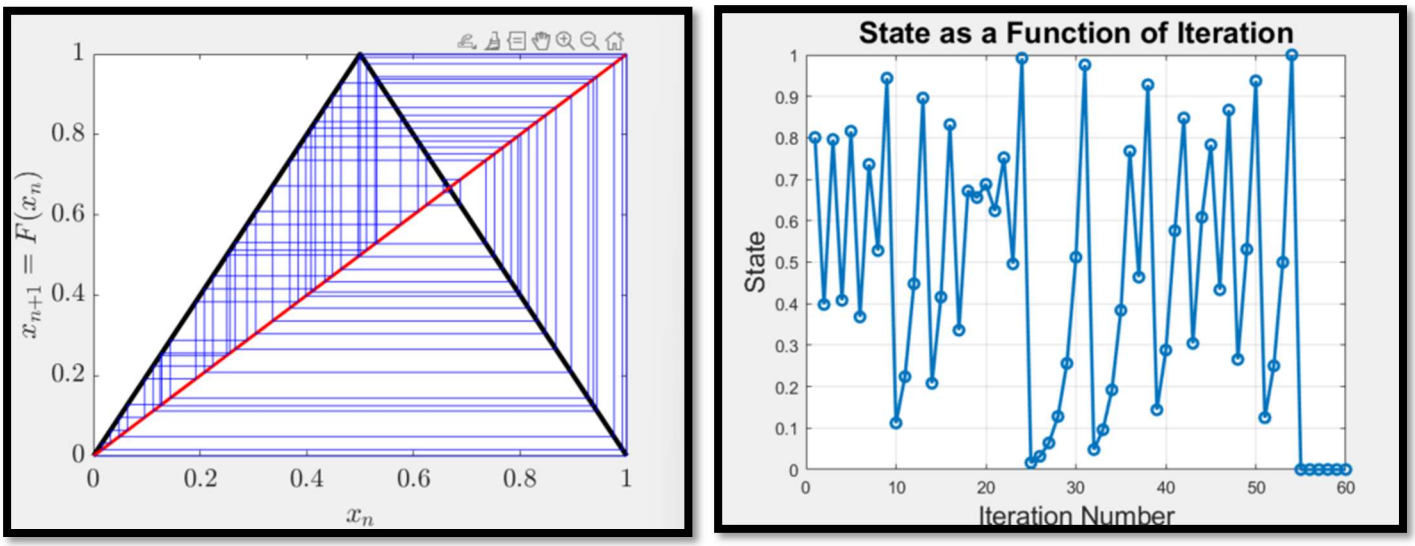

Figure 4: Tent Map and State as a Function of Iteration Plot after 60 times when x*=0.8 was added with 0.001

Comments: The graphs look essentially the same. The graph State vs Iteration Number never hits x*=0.4 and x*=0.8. For when x*=0.4 as a 2-cycle point and it is starting 0.001 away, the graph appears to oscillate but after iteration 53 (was x=1), iteration 54 and onward it stayed at 0. For when

x*=0.8 as a 2-cycle point and it is starting 0.001 away, iteration 54 was at 1 but iteration 55 and onward it is at 0. The graph looks like it descends to chaos very quickly unlike part e's graph.

Therefore, the two cycle points are unstable.

Script for Problem 1f below (HW2P1_F.m)

```
clear; close all; clc;

%% Create the mapping function plot and the x_n=x_n+1 line
plot([0 .5 1],[0 1 0],'k','LineWidth',3);
hold on;
plot([0 1],[0 1],'r','LineWidth',2);
xlabel('$$x_n$$','FontSize',24,'Interpreter','latex')
ylabel('$$x_{n+1}=F(x_n)$$','FontSize',24,'Interpreter','latex')
set(gca,'FontSize',16,'TickLabelInterpreter','latex')

%% Set the number of iterations
N = 60;

%% Create the variable x to store the values of the mapping
x = zeros(N,1);

%% Set the initial condition
x(1) = 0.4 +0.001; % adding 0.001 away prediction descends to chaos

%% Perform a for loop to execute the mapping
for n = 1:N-1
    %% Set the current state
    x_n = x(n); % Use "x_n" for the current state

    %% Perform the piecewise map to calculate x_np1 (x_(n+1))
    % Use "x_np1" to represent the next state (the x_(n+1) state)
    if x_n <= 0.5
        x_np1 = 2 * x_n;
    else
        x_np1 = 2 * (1 - x_n);
    end

    %% Store the next state in the x-vector
    x(n+1) = x_np1;

    %% Plot the mapping plot
    plot([x_n x_n x_np1],[x_n x_np1 x_np1],'b');
    pause(0.5);
end

%% Plot the state as a function of iteration number (part e addition)
figure;
plot(1:N, x, '-o', 'LineWidth', 2);
xlabel('Iteration Number','FontSize',16);
ylabel('State','FontSize',16);
title('State as a Function of Iteration','FontSize',18);
grid on;
```

Problem 2 - Discrete Map with a System Parameter

As we saw in class, a piece-wise map performs a different operation depending on the value of the input. The map that we will use in this problem,

$$x_{n+1} = \begin{cases} 4x_n(\lambda - x_n), & \text{if } x \leq \lambda \\ 4(x_n - 1)(\lambda - x_n), & \text{if } x > \lambda \end{cases} \tag{2}$$

has a system parameter λ that we will allow to vary between 0 and 1.

a. Derive all the fixed points $x^{*,i}$ that exist in the interval $[0, 1]$ for this map. Your fixed points will be a function of λ. For each fixed point, state over what range of λ they exist.

Hand derivation and Comments

If $x \leq \lambda$

$x_{n+1} = 4x_n(\lambda - x_n)$

$x^{*,i} = 4x^{*,i}(\lambda - x^{*,i})$

$x^{*,i} = 4\lambda x^{*,i} - 4(x^{*,i})^2$

$4(x^{*,i})^2 - 4\lambda x^{*,i} + x^{*,i} = 0$

$x^{*,i}(4x^{*,i} - 4\lambda + 1) = 0$

$x^{*,i} = 0$

$4x^{*,i} - 4\lambda + 1 = 0$

$\frac{4x^{*,i}}{4} = \frac{4\lambda - 1}{4}$

$x^{*,i} = \lambda - \frac{1}{4}$

$0 \leq \lambda - \frac{1}{4} \leq 1$

$0 \leq \lambda - 0.25 \quad \lambda - 0.25 \leq 1$

$0.25 \leq \lambda \quad \lambda \leq 1.25$

$\lambda \geq 0.25 \quad [0.25, 1]$

If $x > \lambda$

$x_{n+1} = 4(x_n - 1)(\lambda - x_n)$

$x^* = 4(x^* - 1)(\lambda - x^*)$

$x^* = 4(\lambda x^* - (x^*)^2 - \lambda + x^*)$

$x^* = 4\lambda x^* - 4(x^*)^2 - 4\lambda + 4x^*$

$0 = 4\lambda x^* - 4(x^*)^2 - 4\lambda + 3x^*$

$4(x^*)^2 - 4\lambda x^* + 4\lambda - 3x^* = 0$

$4(x^*)^2 - x^*(4\lambda + 3) + 4\lambda = 0$

$a = 4 \quad b = -(4\lambda + 3) \quad c = 4\lambda$

$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = x^*$

$x^* = \frac{-(4\lambda + 3) \pm \sqrt{(-4\lambda - 3)^2 - 4(4)(4\lambda)}}{2(4)}$

$x^* = \frac{(4\lambda + 3) \pm \sqrt{16\lambda^2 - 40\lambda + 9}}{8}$

$16\lambda^2 - 40\lambda + 9 < 0$

$16\lambda^2 - 4\lambda - 36\lambda + 9 < 0$

$-36 \times -4 = 144, \quad -40$

$4\lambda(4\lambda - 1) - 9(4\lambda - 1) = 0$

$(4\lambda - 9)(4\lambda - 1) = 0$

$4\lambda = 9 \quad 4\lambda - 1 = 0$

$\lambda = 9/4 \quad \lambda = 1/4$

for it not to exist λ between to 1/4 to 9/4

$\lambda < \frac{1}{4}$

$x^{*,i} = 0$

$x^{*,i} = \lambda - \frac{1}{4}$ when $0.25 \leq \lambda \leq 1$

$x^{*,i} = \frac{(4\lambda + 3) + \sqrt{16\lambda^2 - 40\lambda + 9}}{8}$ when $0 \leq \lambda < \frac{1}{4}$

$x^{*,i} = \frac{(4\lambda + 3) - \sqrt{16\lambda^2 - 40\lambda + 9}}{8}$ when $0 \leq \lambda < \frac{1}{4}$

Problem 2b) Write a function discrete_map (x0, lambda) that takes as inputs x0 and λ, performs the mapping 1000 times, then outputs an array containing only the final 100 states.

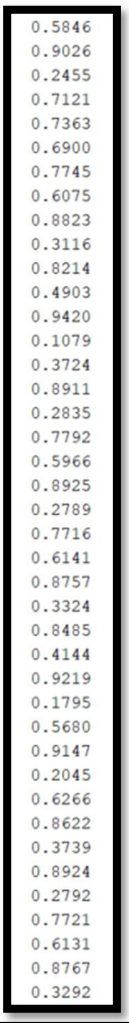

Figure 5: Snippet of part of 100 last numbers when running discrete_map (rand, rand) on command window

Script for Problem 2b below (discrete_map.m)

```
% Problem 2 Part B

%% Define the discrete_map function
function result = discrete_map(x0, lambda)
x_n = x0; % Initialize the current state x_n with the initial condition x0

N = 1000; % Set the total number of iterations
nSave = 100; % Set the number of iterations to save

result = zeros(nSave, 1);  % Initialize the result array to store the final states

    for iter = 1:N % Loop through iterations
```

```
        if x_n <= lambda  % Check the condition and apply the appropriate mapping
            x1 = 4 * x_n * (lambda - x_n);
        else
            x1 = 4 * (x_n - 1) * (lambda - x_n);
        end

        % If we are in the final 100 steps, store the position
        if iter > (N - nSave)
            result(iter-N+nSave) = x1;
        end

        % Update the state for the next iteration
        x_n = x1;
    end
end
```

Problem 2c) Write a script that cycles through λ = 0:0.005:1. For each value of λ, call your function twice, each time starting from

$$x_0^- = \frac{\lambda}{2}, \text{ then from } x_0^+ = \lambda + \frac{1-\lambda}{2}.$$

Use the outputs from each call to your function to create a bifurcation diagram (λ vs. final states). (Hint: for each value of λ, you should be plotting 200 points)

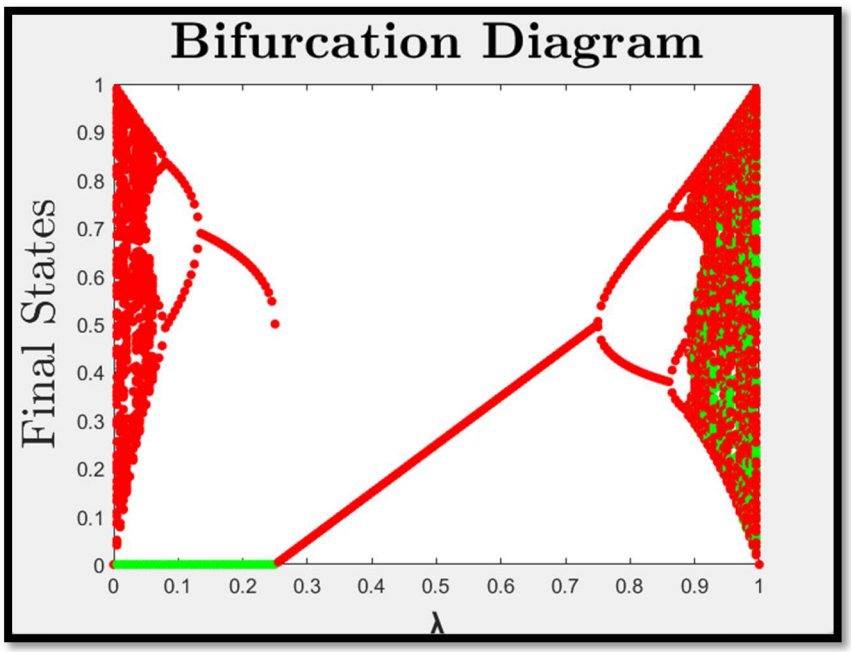

Figure 6: Bifurcation Diagram cycling through λ = 0:0.005:1.

Script for Problem 2c below (HW2P2C.m)

```
clear; close all; clc;

lambda_values = 0:.005:1; % range of lambda from 0 to 1 with increments of 0.005
```

```
for lambda = lambda_values
    initial_neg = lambda/2; %set initial condition of x0 negative
    initial_pos = lambda + (1-lambda)/2; %set initial condition of x0 positive
    trajectory_neg = discrete_map(initial_neg, lambda); % Trajectory for the negative
initial condition on map
    trajectory_pos = discrete_map(initial_pos, lambda); % Trajectory for the positve
initial condition on map

    % Plot unique values for each lambda separately
    figure(1);

    % Plot  values where the initial negative value of the trajectory for a specific
lambda on the graph
    plot(lambda*ones(size(unique(trajectory_neg))), unique(trajectory_neg), '.g',
'MarkerSize', 16);
    hold on;

    % Plot  values where the initial positive value of the trajectory for a specific
lambda on the graph
    plot(lambda*ones(size(unique(trajectory_pos))), unique(trajectory_pos), '.r',
'MarkerSize', 16);
end

% Title and labels for the plot
title('\textbf{Bifurcation Diagram}', 'FontSize', 26, 'Interpreter', 'latex'); %title
of graph
ylabel('Final States', 'FontSize', 24, 'Interpreter', 'latex');
xlabel('λ', 'FontSize', 22, 'Interpreter', 'latex');
```

Problem 2d) Plot each of the fixed points as a function of λ on the bifurcation diagram. Use different colors for each $x_{*,i}$ and plot them only for λ values such that the corresponding fixed point exists. Based on the figure, what is the stability of each fixed point, as a function of λ (i.e., for what values of λ is each fixed point stable/unstable?)?

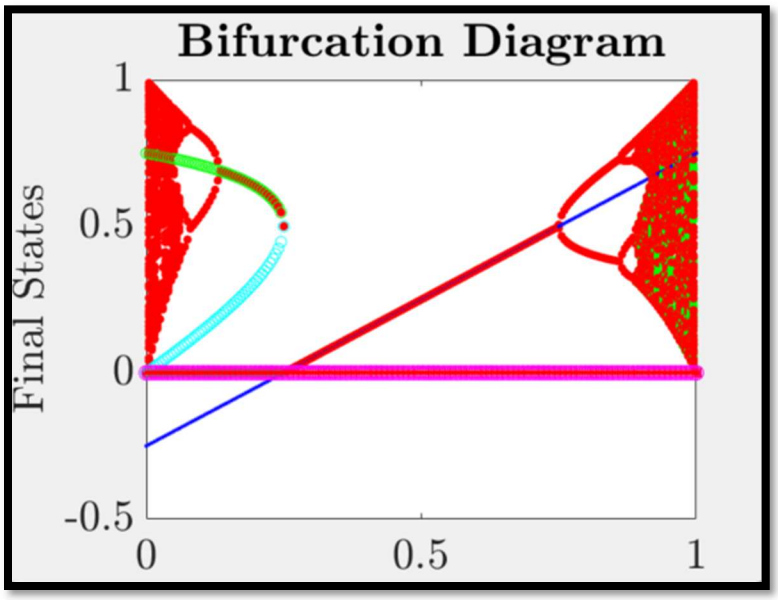

Figure 7: Bifurcation Diagram with different colors.

Comments: At x*0, lambda is stable from the range of 0 to 0.25. At x*=$\lambda - \frac{1}{4}$, it is stable between 0.25 and 0.75 (0.25 is inclusive and 0.75 is not inclusive). For the quadratic and variable named "Quadpos", lambda is stable between 0.135 to 0.25. For the variable "Quadneg" lambda goes from the range 0 to 0.25 but it is unstable.

Script for Problem 2d below (HW2P2D.m)

```
clear; close all; clc;

lambda_values = 0:.005:1;

for lambda = lambda_values
    %initializing variables and functions
    initial_neg = lambda/2;
    initial_pos = lambda + (1-lambda)/2;
    trajectory_neg = discrete_map(initial_neg, lambda);
    trajectory_pos = discrete_map(initial_pos, lambda);

    % Plot unique values for each lambda separately
    figure(1);

    % Plot values where the initial negative value of the trajectory for a specific
lambda on the graph
    plot(lambda*ones(size(unique(trajectory_neg))), unique(trajectory_neg), '.g',
'MarkerSize', 16);
    hold on;

    % Plot values where the initial positive value of the trajectory for a specific
lambda on the graph
    plot(lambda*ones(size(unique(trajectory_pos))), unique(trajectory_pos), '.r',
'MarkerSize', 16);

    % part d
    quadpos = (3 + 4*lambda + sqrt(16*lambda^2-40*lambda+9))/8;
    quadneg = (3 + 4*lambda - sqrt(16*lambda^2-40*lambda+9))/8;
    if lambda <= 1/4
        % Plot the fixed points only for λ values where they exist
        plot(lambda, 0, 'sr', 'MarkerSize', 6); % Fixed point 1
        plot(lambda, lambda-0.25, '.b', 'MarkerSize', 6); % Fixed point 2
        plot(lambda, quadpos, 'og', 'MarkerSize',6) %plotting quadratic positive
fixed point
        plot(lambda, quadneg, 'oc', 'MarkerSize',6) %plotting quadratic negative
fixed point
    else
        % Plot the fixed points only for λ values where they exist
        plot(lambda, 0, 'sr', 'MarkerSize', 6); % Fixed point 1
        plot(lambda, lambda-0.25, '.b', 'MarkerSize', 6); % Fixed point 2
    end
end

%plots from part c
plot(lambda_values, unique(trajectory_neg), 'om', 'MarkerSize', 8);
```

```
plot(lambda_values, unique(trajectory_pos), '.r', 'MarkerSize', 8);
set(gca, 'FontSize', 24, 'TickLabelInterpreter', 'latex');
title('\textbf{Bifurcation Diagram}', 'FontSize', 26, 'Interpreter', 'latex'); %title
of graph
xlabel('λ', 'FontSize', 22, 'Interpreter', 'latex'); %labeling x-axis
ylabel('Final States', 'FontSize', 24, 'Interpreter', 'latex'); %labeling y-axis
```

Problem 3 - Pipe Flow Simulation: In class we are going to walk through a simple simulation of flow down an axially symmetric pipe. If you need a reminder of how we did this in Ansys, you can work through the walkthrough available at `https://confluence.cornell.edu/display/SIMULATION/FLUENT+-+Laminar+Pipe+Flow`. For this homework problem, implement the following changes:

- Double the radius from 0.1 m to 0.2 m
- Double the number of cells in the vertical direction
- Change the fluid viscosity so that the Reynolds number $Re = \rho UD/\mu$ is 1, 10, 100, 1000 (You will run the simulation 4 times).

Download the axial velocity data for each of the 4 simulations. For each simulation, produce a contour plot of the axial velocity in Matlab. Then, in one figure, plot the axial velocity profile at $z = 2$ m down the pipe for all Re. Discuss any differences that you notice.

The function `dataConversion.m` on Canvas can be used to convert your saved files to variables used for plotting.

(4 contour plots and 1 line plot)

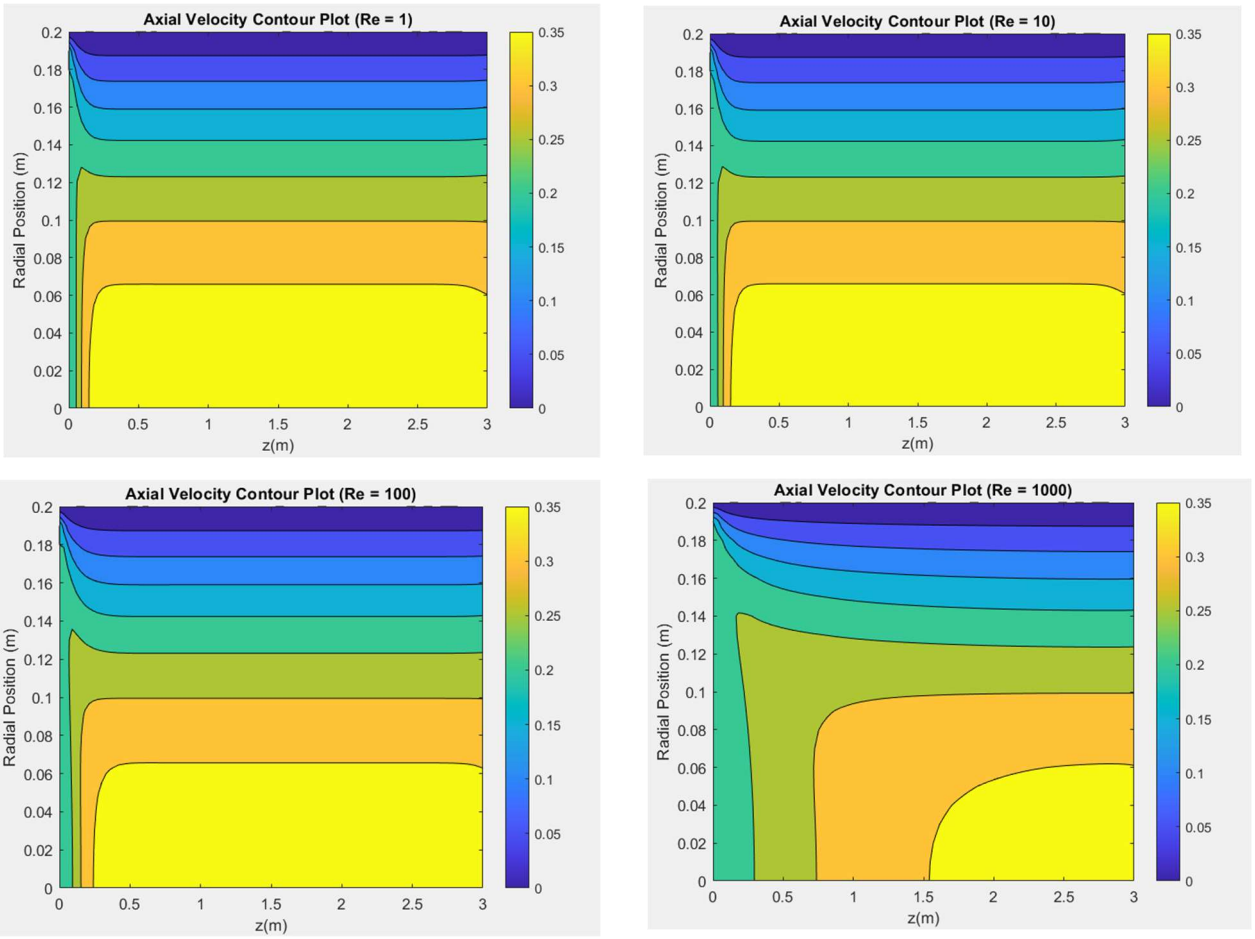

Figure 8: Four Contour Maps of Re = 1, 10, 100, 1000 respectively

Script for Problem 3 (4 Contour Plots only) below (HW2P3.m)

```
clear; close all; clc;

% Making all text files into a vector
txt_N = {'sim_1.txt', 'sim_2.txt', 'sim_3.txt', 'sim_4.txt'};

% Reynolds numbers corresponding to each simulation (added it to code since
% making code for each Re_value would be very tedious.
Re_values = [1, 10, 100, 1000];

%I used a similar for loop from my ME2355 lab since we needed to make multiple plots
from lab reports
for i = 1:length(txt_N)
    data = importdata(txt_N{i}); % Call the plotting function to represent each file.
    % In each loop iteration, the code works with a different file
    % listed in the txt_N collection.

    % Identify the grid locations and the axial velocity from the simulation
    x = data.data(:, 2);
    y = data.data(:, 3);
    axialVel = data.data(:, 4);

    % Create the set of points where we want the data
    x_Vertex = 0:.03:3;
    y_Vertex = 0:.01:.2;

    [xGrid, yGrid] = meshgrid(x_Vertex, y_Vertex);

    % Use interpolation to determine the velocity at the grid points
    axVelGrid = griddata(x, y, axialVel, xGrid, yGrid);

    % Contour Plot
    figure;
    contourf(xGrid, yGrid, axVelGrid); %contourf shades the regions
    colorbar;
    title(['Axial Velocity Contour Plot (Re = ', num2str(Re_values(i)), ')']);
    xlabel('z(m)');
    ylabel('Radial Position (m)');

end
```

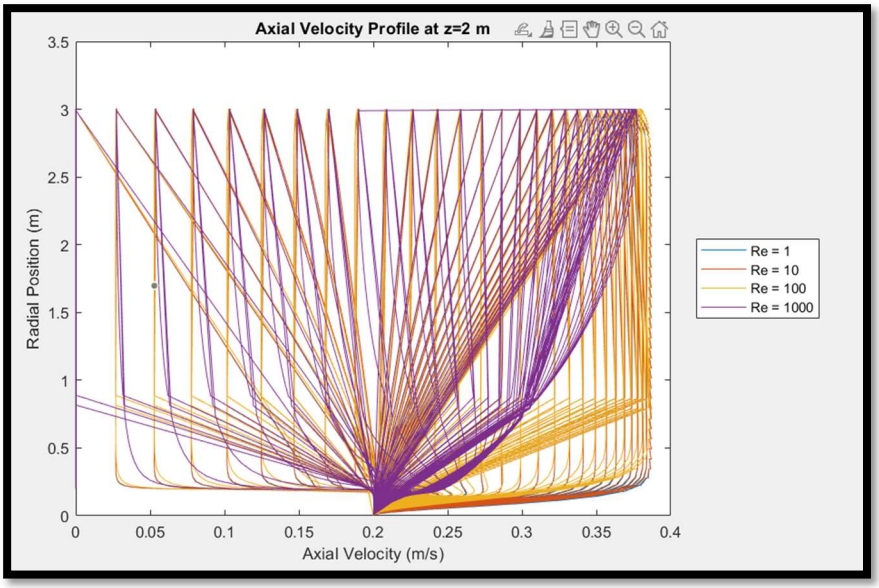

Figure 9: Line Plot of Respective Re Values for 4 Simulations

Comments: Not sure if this code is right because there are a lot more lines than I wanted when plotting. The trend of the graph is suppose to be that all plots with different Reynold's numbers would go in an exponential downward trend. When z=2 m, the graph goes to increase the velocity and downwards radial trend. Could I have some feedback on how to fix my line plot if necessary.

Script for Problem 3 (1 Line Plots only) below (HW2P3LinePlot.m)

```
clear; close all; clc;

% Making all text files into a vector
txt_N = {'sim_1.txt', 'sim_2.txt', 'sim_3.txt', 'sim_4.txt'};

% Create a figure for the combined line plot
figure;

% Iterate through each simulation
for i = 1:length(txt_N)
    data = importdata(txt_N{i}); % Load data from each file

    % Identify the grid locations and the axial velocity from the simulation
    x = data.data(:, 2);
    y = data.data(:, 3);
    axialVel = data.data(:, 4);

    %% Create the set of points where we want the data
    x_Vertex = 0:.03:3;
    y_Vertex = 0:.01:.2;
    axVel = zeros(21,101);
```

```
    % Convert Cartesian coordinates to radial coordinates
    r = sqrt(x.^2 + y.^2);

    % Plot axial velocity vs. radial position on the same graph
    plot(axialVel, r, '-');
    hold on;
end

% Title and labels for the entire plot
title('Axial Velocity Profile at z=2 m');
xlabel('Axial Velocity (m/s)');
ylabel('Radial Position (m)');

% Add a legend indicating the Reynolds number for each line
legend('Re = 1', 'Re = 10', 'Re = 100', 'Re = 1000', 'Location', 'eastoutside',
'Orientation', 'vertical');
```