EEE 391 Matlab MP2

Section: 2
Utku Kurtulmuş - 21903025

PART1-Examples of Recursive Systems

First Order:

1) **Bank Account Balance**: When we put money to bank our money gets larger and larger according to an interest rate but the total amount we earn changes according to our current money while the interest rate remains constant. Hence we can model our system in the following way:
y[n] is the current bank account, and y[n + 1] is the bank account after the interest rate is applied. And the interest rate is k. We have the following formula: y[n + 1] = y[n] + k*y[n]. To make this formula in the general IIR system form we delay the system with 1 unit: y[n] = y[n-1] + ky[n-1] Then y[n]= (k + 1)y[n -1].This is a first order system where N = 1. and a_1 coefficients = {(k+1)}. We don't have the input x[n] in the calculation hence M = 0, we don't have b_k coefficients.

2) **Population Growth at Constant Rate:** This example is very similar to bank account balance example hence we will keep the explanation simple. y[n] = (k + 1)y[n-1]. First order system where N = 1. and a_1 coefficients = {(k+1)}

3) **Predicting the Length of the next Cpu Burst for SJF Algorithm:** In Computers where the operating system prefers to use algorithms like SJF we need to predict to next cpu burst of a process. y[n] is the predicted cpu burst and x[n] is the real cpu burst length. We will use exponential averaging, a is the weight factor between 1 and 0. We have the following formula: y[n + 1] = a*x[n] + (1 - a)*y[n]. We can change the formula to the form: y[n] = a*x[n -1] + (1 - a)y[n -1]. This is a first order system where N = 1, and a_1 coefficients = {(1-a)}. This time M = 1, and b_k coefficients = {0,a}.

Second Order:

1) **Fibonacci Sequence:** The first thing that comes to my mind as a second order recursive system is the famous Fibonacci Sequence. since we use in both math and cs applications while we are learning about recursion.
y[n] = y[n - 1] + y[n -2]. This is a second order system where N = 2. and a_1 coefficients = {1,1}. We don't have the input x[n] in the calculation hence M = 0, we don't have b_k coefficents.

2) **Financial Stock Prediction:** In economics, to predict the future price of a stuck, the previous two prediction values and a drift term can be used. This helps us to analyze and make long-term predictions about the stock prices. In this formula we are not using the previous stock prices, but the previous stock predictions. y[n] = (½)(y[n-1] + y[n-2]) + ε . This is a second order system where N = 2. and a_1 coefficients = {½ ,½ }. We don't have the input x[n]= in the calculation hence M = 0, we don't have b_k coefficents. In short therm also the previous stock market prices can be used in similar way.

3) **Mechanical Oscillation System:** In mechanics, mass-spring-damper system oscillations can be modeled as: y[n] = 2 * y[n-1] - y[n-2] + (k / m) * (x[n] - 2 * x[n-1] + x[n-2]), where y[n] is the displacement of the mass at time n, x[n] is the external force at time n, k and m are the spring constant and mass of the system respectively. This is a second order system where N = 2. and a_1 coefficients = {2 , -1}. We have also the input in this equation, M = 2, and b_k coefficients = {k/m , -2k/m, k/m}.

PART2
Q1)

The formula:
y[n] indicates the total amount of water on each day in the reservoir.
In this implementation, we assumed that the water coming 12 hours late is also lost by half of the day.
y[0] = 5000
y[n] = 0.97*y[n-1] + 0.985*(180) + 65 = 0.97*y[n-1] + 242.3

Analysis:
We can use the substitution method:
y[n] = 0.97*y[n-1] + 242.3,
y[n - 1] = 0.97*y[n-2] + 242.3,
y[n - 2] = 0.97*y[n-3] + 242.3,

y[n] = 0.97*(0.97*y[n-2] + 242.3) + 242.3,

y[n] = (0.97)^2*y[n-2] + (0.97)*242.3 + 242.3

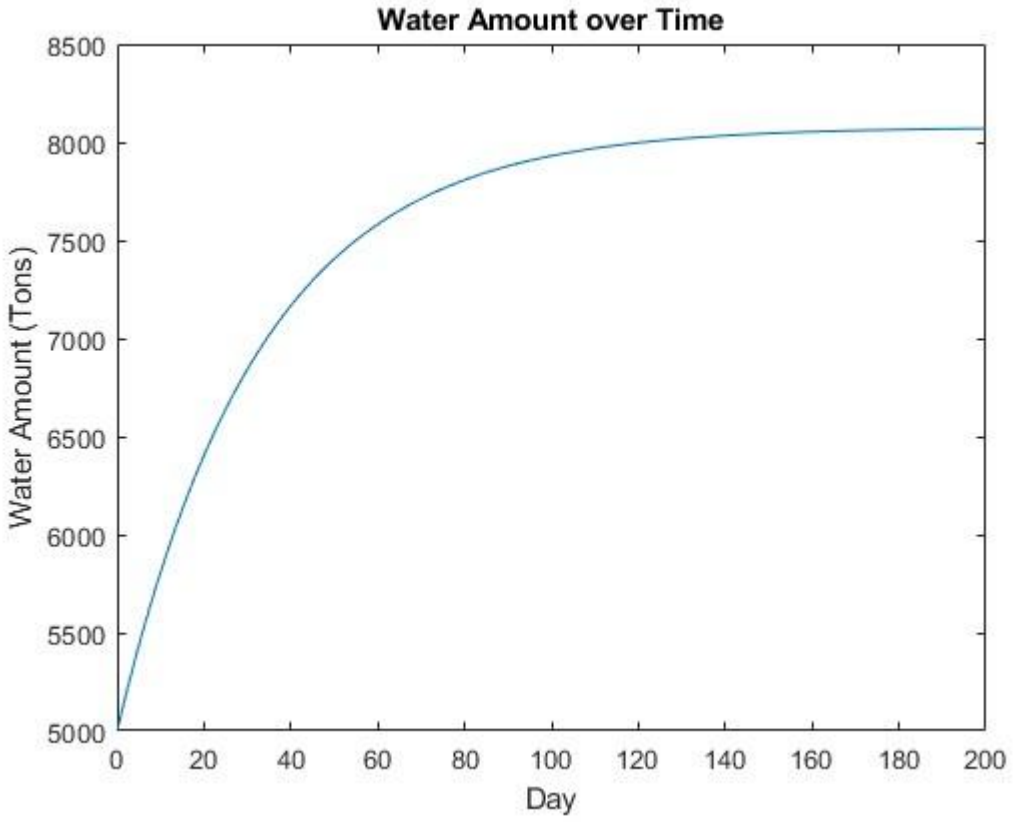
y[n] = (0.97)^2*(0.97*y[n-3] + 242.3) + (0.97)*242.3 + 242.3
y[n] = (0.97)^3*y[n-3] + (0.97)^2*242.3 + (0.97)*242.3 + 242.3
y[n] = (0.97)^3*y[n-3] + 242.3((0.97)^2 + (0.97) + 1)

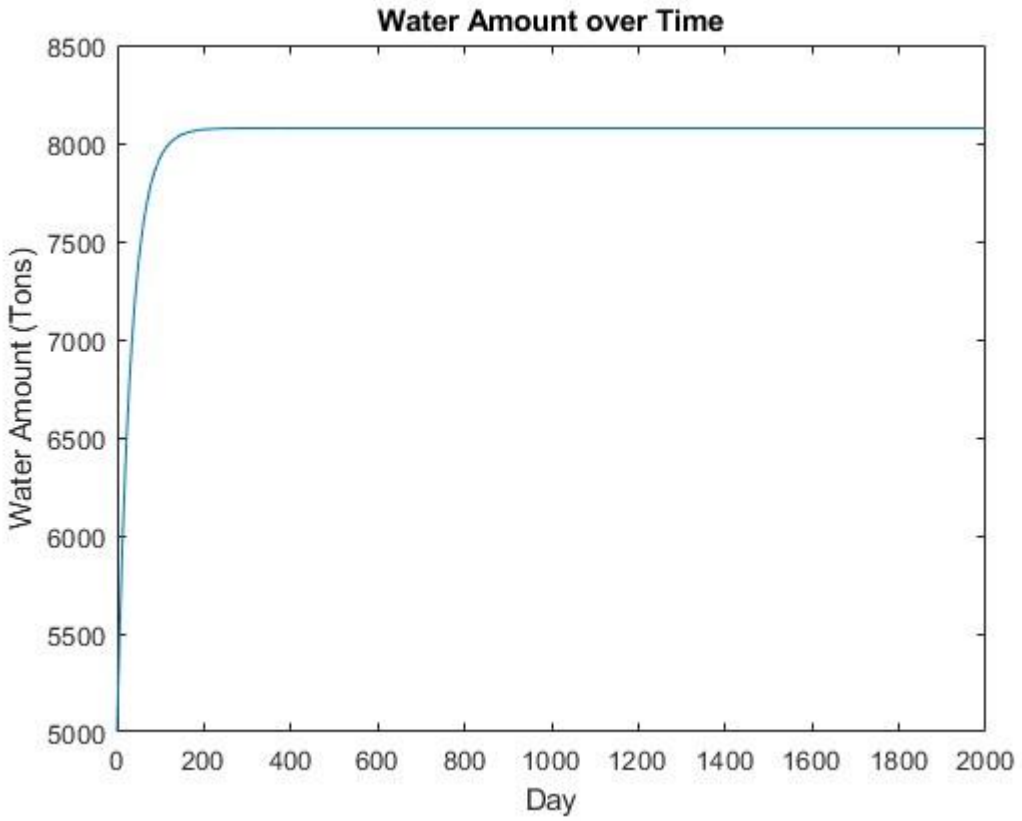
We have seen the pattern: for the first part:
(0.97)^3*y[n-3] will get to ⇒ (0.97)^n*y[0]

and
242.3((0.97)^2 + (0.97) + 1) will get to ⇒
242.3((0.97)^(n-1) + (0.97)^(n-2) … + 1)
using the geometric sum formula ⇒ 242.3(1-0.97^n)/0.03

hence we have the solution:
y[n] = (0.97)^n*y[0] +242.3(1-0.97^n)/0.03, substitute y[0]
⇒ y[n] = (0.97)^n*5000 +242.3(1-0.97^n)/0.03

Matlab Plots:





Is The System Stable?

From the matlab plot of the system we can see that the plot seems to converge as the n values gets larger and larger. Moreover, we already know that this system is a first order IIR system and the absolute value of a_1 < 1, which is 0.97. And this will make the system stable, as it was proven in the lecture.

Q2)

The formula:

y[0] = 40000 n = 0

y[n] = 1.09*y[n-1] + 1000 n = 1, 3, 5 … 2k -1 where k is an positive integer.

y[n] = 1.09*y[n-1] - 2500 n = 2, 4, 6 … 2k where k is an positive integer.

Analysis:

We can use a similar approach to first solution and use the substitution method, but this time there will be two alternating cases and formulas:

Assume n = odd:
y[n] = 1.09*y[n-1] + 1000,
since n is odd, n-1 is even:
y[n - 1] = 1.09*y[n-2] -2500
substitute y[n-1]:
y[n] = 1.09*(1.09*y[n-2] -2500) + 1000
y[n] = (1.09)^2 * y[n-2] - (1.09)2500 + 1000
continue:
y[n - 2] = 1.09*y[n-3] + 1000
y[n] = (1.09)^2 * (1.09*y[n-3] + 1000) - (1.09)2500 + 1000
y[n] = (1.09)^3 * y[n-3] +(1.09)^2 * 1000 - (1.09)2500 + 100

We have seen the pattern for odd n:

$$y[n] = (1.09)^n * y[0] + 1000 * \sum_{k=0}^{(n-1)/2} (1.09)^{2k} - 2500 \sum_{k=1}^{(n-1)/2} (1.09)^{2k-1}$$

using geometric summation formula we have:

$$\sum_{k=0}^{(n-1)/2} (1.09)^{2k} = \sum_{k=0}^{(n-1)/2} ((1.09)^2)^k = (1 - ((1.09)^2)^{(n+1)/2}) / (1 - (1.09)^2)$$

$$\sum_{k=1}^{(n-1)/2} (1.09)^{2k-1} = (1.09) * \sum_{k=1}^{(n-3)/2} ((1.09)^2)^{k-1} = (1.09) * \sum_{k=0}^{(n-3)/2} ((1.09)^2)^k$$

$$= (1.09) * ((1 - ((1.09)^2)^{(n+1)/2}) / (1 - (1.09)^2))$$

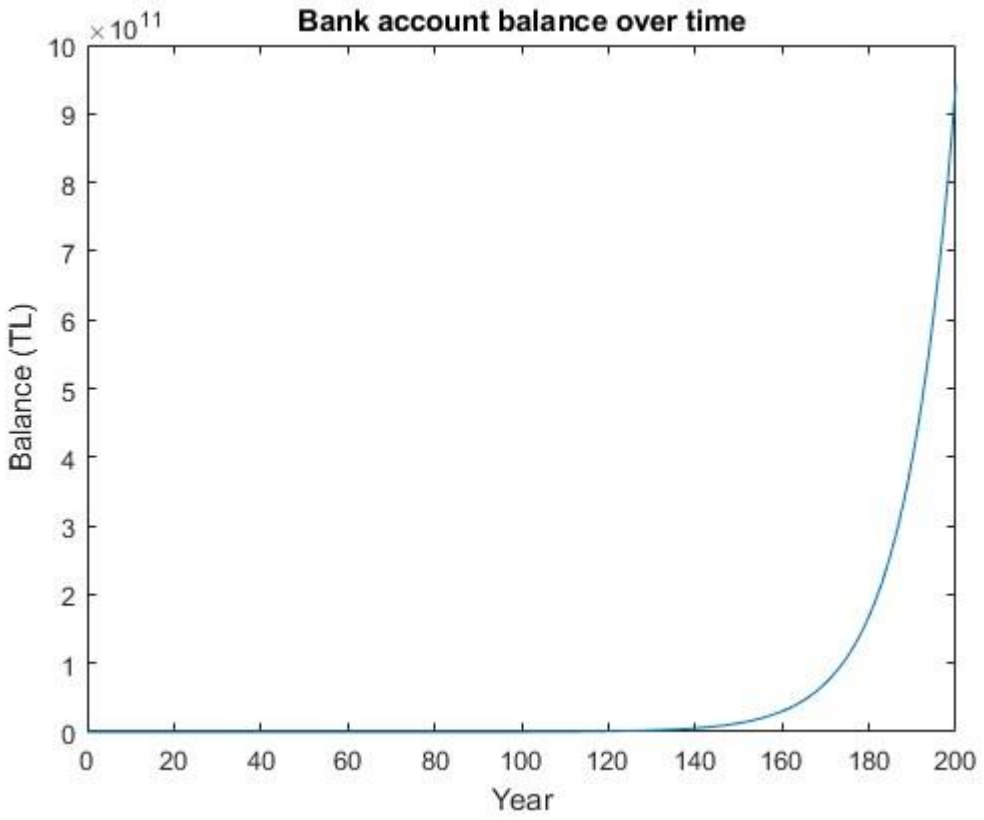
hence for odd n:

$$y[n] = (1.09)^n * y[0] + 1000 * ((1 - ((1.09)^2)^{(n+1)/2}) / (1 - (1.09)^2)) - 2500 (1.09) * ((1 - ((1.09)^2)^{(n+1)/2}) / (1 - (1.09)^2))$$

For even n, the order of 1000 and 2500 will switch:

$$y[n] = (1.09)^n * y[0] - 2500 * \sum_{k=0}^{(n-1)/2} (1.09)^{2k} + 1000 \sum_{k=1}^{(n-1)/2} (1.09)^{2k-1}$$

Matlab Plots:



Is The System Stable?

From the matlab plot of the system we can see that the plot seems to diverge to positive infinity as the n values gets larger and larger. Moreover, we already know that this system is a first order IIR system and the absolute value of a_1 > 1, which is 1.09. And this will make the system unstable, as it was proven in the lecture.

Matlab Code:

```
%part 1
%{
The formula:
   y[0] = 5000
   y[n] = 0.97*y[n-1]  + 0.985*(180)  + 65
%}
n = 200; %number of days
y = zeros(n + 1,1); %amount of water on each day
y(1) = 5000; %initial amount of water (tons)
% Calculate water amount for each day
for i = 2:n+1
   y(i) = y(i-1)*(0.97) + 0.985*(180) + 65;
end
% Plot balance vs. year
plot(0:n, y)
xlabel('Day')
ylabel('Water Amount (Tons)')
title('Water Amount over Time')
%part 2
%{
The formula:
   y[0] = 40000 n = 0
   y[n] = 1.09*y[n-1] + 1000 n = 1, 3, 5 … 2k -1 where  k is an positive
integer.
   y[n] = 1.09*y[n-1] -  2500  n = 2, 4, 6 … 2k  where  k is an positive
integer.
%}
% Set initial conditions
n = 200;          % number of years
y = zeros(n+1,1);  % balance array
y(1) = 40000;      % initial deposit (TL)
% Calculate balance for each year
for i = 2:n+1
   if mod(i,2) == 1  % odd year
       y(i) = y(i-1)*(1+0.09) + 1000;
   else               % even year
       y(i) = y(i-1)*(1+0.09) - 2500;
   end
end
% Plot balance vs. year
figure;
plot(0:n, y)
xlabel('Year')
ylabel('Balance (TL)')
title('Bank account balance over time')
```