Cassandra Cantu
305100205
CAE/MAE M20 Summer A 2020
July 7th 2020

Homework 02

1 The Three Species Problem

1.1 Introduction

The goal in this problem is to develop a script that calculates the time-varying populations of three species given their initial values at time 0. To do this, the continuous functions for these populations had to be approximated to functions the program can work with. After, the results at time 0 to time 12 with 0.5 intervals will be printed to the screen. Then, the time it takes to calculate the results with various time intervals will be found and printed to the screen. Any differences/trends in results and time will be discussed.

1.2 Model and Theory

The Lotka-Volterra equations are a set of continuous differentials equations that describe the growth rate of one population affected by the other population(s). However, since programs cannot compute continuous calculations, the equations were discretized using the forward Euler method. They were then rearranged to a programable way. The applicable equations for this particular problem are:

$Lotka - Volterra\ equations$

$$\frac{dx}{dt} = 0.75x\left(1 - \frac{x}{20}\right) - 1.5xy - 0.5xz$$

$$\frac{dy}{dt} = y\left(1 - \frac{y}{25}\right) - 0.75xy - 1.25yz$$

$$\frac{dz}{dt} = 1.5z\left(1 - \frac{z}{30}\right) - xz - yz$$

$Rearranged\ forward\ Euler\ method\ (applied\ to\ each\ equation\ above)$

$$x_{k+1} = x_k + \frac{dx}{dt}\Delta t$$

Where

- $\frac{dx}{dt}$ = population per unit area of species X
- $\frac{dy}{dt}$ = population per unit area of species Y
- $\frac{dz}{dt}$ = population per unit area of species Z
- x = population of species X at a given time
- y = population of species Y at a given time
- z = population of species Z at a given time
- x_{k+1} = approximate population of species X at time k+1
- x_k = population of species X at a time k
- Δt = change in time

1.3 Methods and Pseudocode

The flow of calculate should be as follows:

1. Set initial conditions of each population, time conditions, and iteration count
2. Assign initial conditions to variable `x_old, y_old, z_old)` (equivalent to x_k in above equation)
    a. To start calculations using these values
3. Print heading for output and population values at time = 0 (initial conditions)
    a. Print heading here so It would not print out repeatedly in `for` loop
    b. Since calculating the population values at time = 0 with the Forward Euler method introduces error, this is to ensure we get the correct initial values
4. Enter `for` loop for `t = 0:delta_t:t_final` and calculate the population values at a given time
    a. Starts at time 0, step size is Δt in above equation, and ends at 12
    b. Calculates with discretized Lotka-Volterra equations using Forward Euler method
5. Update variables using `x_old = x_new;` and etc.
    a. Since the equations depend on the value before it, need to replace the old stored values
6. Prints time-varying populations for delta_t = 0.05 using `if (mod(iter, 100)==0)`
    a. In this case, our `delta_t` = 0.005, so using this `if` statement will give format required

1.4 Calculations and Results

With an initial population of x, y, and z being 2, 2.49, and 1.5 at time 0 respectively, the output, starting at time 0 and ending at time 12 with timestep sizes equaling 0.5 is:

```
Which problem do you want to run? Enter 1 for the three species problem or 2
for the pocket change problem: 1
Time    X      Y      Z
0.0   2.00   2.49   1.50
0.5   0.59   1.43   0.74
1.0   0.26   1.29   0.65
1.5   0.12   1.29   0.65
2.0   0.06   1.33   0.67
2.5   0.02   1.37   0.70
3.0   0.01   1.40   0.72
3.5   0.00   1.43   0.73
4.0   0.00   1.44   0.74
4.5   0.00   1.44   0.75
5.0   0.00   1.44   0.76
5.5   0.00   1.43   0.77
6.0   0.00   1.41   0.79
6.5   0.00   1.37   0.81
7.0   0.00   1.31   0.86
7.5   0.00   1.20   0.94
8.0   0.00   1.03   1.11
8.5   0.00   0.76   1.45
9.0   0.00   0.41   2.19
9.5   0.00   0.11   3.81
10.0  0.00   0.01   6.92
10.5  0.00   0.00   11.63
11.0  0.00   0.00   17.18
```

```
11.5  0.00  0.00  22.19
12.0  0.00  0.00  25.73
```

With an initial population of x, y, and z being 7.1, 5, and 3.6 at time 0 respectively, the output, starting at time 0 and ending at time 12 with timestep sizes equaling 0.5 is:

```
Which problem do you want to run? Enter 1 for the three species problem or 2
for the pocket change problem: 1
Time   X     Y     Z
0.0   7.10  5.00  3.60
0.5   1.47  1.23  0.61
1.0   0.84  0.96  0.43
1.5   0.54  0.93  0.41
2.0   0.34  0.98  0.43
2.5   0.21  1.09  0.46
3.0   0.11  1.22  0.51
3.5   0.05  1.37  0.53
4.0   0.02  1.55  0.53
4.5   0.01  1.80  0.48
5.0   0.00  2.17  0.37
5.5   0.00  2.82  0.23
6.0   0.00  3.95  0.09
6.5   0.00  5.74  0.02
7.0   0.00  8.20  0.00
7.5   0.00  11.14 0.00
8.0   0.00  14.25 0.00
8.5   0.00  17.15 0.00
9.0   0.00  19.57 0.00
9.5   0.00  21.40 0.00
10.0  0.00  22.69 0.00
10.5  0.00  23.55 0.00
11.0  0.00  24.10 0.00
11.5  0.00  24.45 0.00
12.0  0.00  24.66 0.00
```

With an initial population of x, y, and z being 10, 1, and 1 at time 0 respectively, the output, starting at time 0 and ending at time 12 with timestep sizes equaling 0.5 is:

```
Which problem do you want to run? Enter 1 for the three species problem or 2
for the pocket change problem: 1
Time   X     Y     Z
0.0   10.00 1.00  1.00
0.5   9.28  0.05  0.02
1.0   11.03 0.00  0.00
1.5   12.83 0.00  0.00
2.0   14.45 0.00  0.00
2.5   15.83 0.00  0.00
3.0   16.93 0.00  0.00
3.5   17.79 0.00  0.00
4.0   18.42 0.00  0.00
4.5   18.89 0.00  0.00
5.0   19.22 0.00  0.00
5.5   19.46 0.00  0.00
6.0   19.63 0.00  0.00
6.5   19.74 0.00  0.00
7.0   19.82 0.00  0.00
7.5   19.88 0.00  0.00
8.0   19.92 0.00  0.00
8.5   19.94 0.00  0.00
```

```
9.0   19.96 0.00  0.00
9.5   19.97 0.00  0.00
10.0  19.98 0.00  0.00
10.5  19.99 0.00  0.00
11.0  19.99 0.00  0.00
11.5  19.99 0.00  0.00
12.0  20.00 0.00  0.00
```

With the timestep sizes equaling 0.005, the output is:

```
Which problem do you want to run? Enter 1 for the three species problem or 2
for the pocket change problem: 1
Time   X     Y     Z
0.0   2.00  2.49  1.50
0.5   0.59  1.43  0.74
1.0   0.26  1.29  0.65
1.5   0.12  1.29  0.65
2.0   0.06  1.33  0.67
2.5   0.02  1.37  0.70
3.0   0.01  1.40  0.72
3.5   0.00  1.43  0.73
4.0   0.00  1.44  0.74
4.5   0.00  1.44  0.75
5.0   0.00  1.44  0.76
5.5   0.00  1.43  0.77
6.0   0.00  1.41  0.79
6.5   0.00  1.37  0.81
7.0   0.00  1.31  0.86
7.5   0.00  1.20  0.94
8.0   0.00  1.03  1.11
8.5   0.00  0.76  1.45
9.0   0.00  0.41  2.19
9.5   0.00  0.11  3.81
10.0  0.00  0.01  6.92
10.5  0.00  0.00  11.63
11.0  0.00  0.00  17.18
11.5  0.00  0.00  22.19
12.0  0.00  0.00  25.73

Time =

               0.031504322
```

With the timestep sizes equaling 0.05 (> 0.005), the output is:

```
Which problem do you want to run? Enter 1 for the three species problem or 2
for the pocket change problem: 1
Time   X     Y     Z
0.0   2.00  2.49  1.50
0.5   0.54  1.40  0.72
1.0   0.24  1.28  0.64
1.5   0.11  1.30  0.64
2.0   0.05  1.35  0.66
2.5   0.02  1.41  0.68
3.0   0.01  1.47  0.68
3.5   0.00  1.53  0.67
4.0   0.00  1.62  0.64
4.5   0.00  1.77  0.57
5.0   0.00  2.01  0.47
```

```
5.5   0.00  2.46  0.33
6.0   0.00  3.26  0.17
6.5   0.00  4.63  0.05
7.0   0.00  6.69  0.01
7.5   0.00  9.36  0.00
8.0   0.00  12.41 0.00
8.5   -0.00 15.48 0.00
9.0   -0.00 18.23 0.00
9.5   -0.00 20.43 0.00
10.0  -0.00 22.04 -0.00
10.5  -0.00 23.13 -0.00
11.0  -0.00 23.85 -0.00
11.5  -0.00 24.30 -0.00
12.0  -0.00 24.57 -0.00

Time =

               0.023687734
```

With the timestep sizes equaling 0.0005 (< 0.005), the output is:

```
Which problem do you want to run? Enter 1 for the three species problem or 2
for the pocket change problem: 1
Time   X     Y     Z
0.0   2.00  2.49  1.50
0.5   0.59  1.43  0.75
1.0   0.26  1.29  0.65
1.5   0.12  1.29  0.65
2.0   0.06  1.33  0.67
2.5   0.03  1.37  0.70
3.0   0.01  1.40  0.72
3.5   0.00  1.42  0.74
4.0   0.00  1.42  0.75
4.5   0.00  1.41  0.77
5.0   0.00  1.39  0.79
5.5   0.00  1.34  0.83
6.0   0.00  1.26  0.90
6.5   0.00  1.12  1.02
7.0   0.00  0.90  1.26
7.5   0.00  0.58  1.77
8.0   0.00  0.23  2.91
8.5   0.00  0.03  5.27
9.0   0.00  0.00  9.29
9.5   0.00  0.00  14.61
10.0  0.00  0.00  20.03
10.5  0.00  0.00  24.29
11.0  0.00  0.00  27.00
11.5  0.00  0.00  28.51
12.0  0.00  0.00  29.28

Time =

               0.037172101
```

1.5 Discussions and Conclusions

After using a guess-and-check approach, I could not find a balancing point at which all three species coexist peacefully. One will always crowd out the other two over time. I found that the

timing of the calculations with the 0.05 step size was generally less than the timing of the calculations with the 0.005 step size. This was expected as the program does not need to go through as many iterations to reach the final time. In contrast, the timing of the calculations with the 0.0005 step size was generally more than the time of the calculations with the 0.005 step size. This occurred because the program had to go through more iterations to reach the final time. We can also see that for a larger step size, the outputs for X, Y, and Z have more variation than the other step sizes tested. This occurred because the outputs are proportional to the step size (delta_t). Small step sizes are able to follow the Lotka-Volterra continuous functions more closely. Using a large step size, therefore, introduces more error in the calculations.

2 The Pocket Change Problem

2.1 Introduction

The goal in this problem is develop a script that calculates the average number of coins expected to receive in change after a cash transaction. First, the minimum number of coins needed to make each amount of change from 0 to 99 cents is found. Then, the average is calculated by adding these values together and dividing by the number of all change amounts (100). Then, the average number of coins when pennies are eliminated from circulation is calculated. After, a guess-and-check approach was taken to find an optimal monetary system by changing coin denominations.

2.2 Model and Theory

For this problem, we use the US monetary system's coin values. We did not consider 50 cent or dollar coins. The applicable equations for this particular problem are:

$$Q = 25$$
$$D = 10$$
$$N = 5$$
$$P = 1$$

Where

- Q = quarter
- D = dime
- N = nickel
- P = penny

2.3 Methods and Pseudocode

The flow of calculate should be as follows:

1. Initiate total sum of coins
2. Enter `for` loop to calculate number of coins for each starting amount of change from 0 to 99 cents. Define rest of change as starting amount of change.
    a. For loop lets us calculate for definitive number of times
    b. Defining rest of change as starting change makes sure that value is restarted to each iteration
3. Enter `while` loop to check if the remaining change is greater than 0

    a. Checks for any negative change amount & stops program from executing the following `if/elseif` statements if there is any negative change
4. Enter `if/elseif` statements to calculate the minimum number of each type of coin to make up given starting amount of change
    a. Starts from the highest denomination (quarter) and continues through to the lowest denomination (penny) until the remaining change is less than 0
    b. Calculates by subtracting the coin value from the remaining change and then increasing that coin count by 1
5. Outside of both the if statements and while loop but inside the for statement, calculates the total coins per starting amount as well as updates the sum of total coins.
    a. Placed here so all calculations per each starting amount of change have already taken place and the remaining change is already less than 0
6. Calculates the average number of coins through dividing the sum by 100
7. Outside of the for loop, prints out the average number of coins expected to receive in change
    a. Placed here so it will only be printed once and not repeatedly

2.4 Calculations and Results

Using this code, the average number of coins you can expect to receive in your change is:

```
Average Number of Coins = 4.70
```

When pennies are eliminated from circulation, the average number of coins you can expect to receive in your change is:

```
Average Number of Coins = 2.70
```

When a quarter is set to 27 cents, a dime to 11 cents, a nickel to 5 cents, and a penny to 1 cent, the average number of coins you can expect to receive in your change is:

```
Average Number of Coins = 4.44
```

When a quarter is set to 30 cents, a dime to 11 cents, a nickel to 5 cents, and a penny to 1 cent, the average number of coins you can expect to receive in your change is:

```
Average Number of Coins = 4.36
```

When a quarter is set to 37 cents, a dime to 11 cents, a nickel to 3 cents, and a penny to 1 cent, the average number of coins you can expect to receive in your change is:

```
Average Number of Coins = 4.10
```

When a quarter is set to 37 cents, a dime to 11 cents, a nickel to 2 cents, and a penny to 1 cent, the average number of coins you can expect to receive in your change is:

```
Average Number of Coins = 3.83
```

2.5 Discussions and Conclusions

With the regular coin denominations for quarters, dimes, nickels, and pennies, the average number of coins expected to receive in change is 4.70. When pennies are eliminated from

circulation, the average number of coins changes to 2.70. This occurs because any amount of change that ends with 1-4 or 6-9, which are made with 4 pennies, is not considered. When we keep the value of a penny as 1 but adjust the rest of the coins' values, we can see the average generally decreases as the quarter value increases and the nickel value decreased. The most optimal monetary system that could be found has an average number of coins of 3.83.