HOMEWORK 3

**Due: Tuesday, April 24, 2018, 11:55 pm**

1. **The Three Species Problem.**
In class, the predator-prey problem of paramecium and yeast was explored. Now, consider the interaction between three groups of imaginary creatures X, Y, and Z competing for the same food. The populations of these three species are governed by the following Lotka-Volterra equations:

$$\begin{aligned}
\frac{\mathrm{d}x}{\mathrm{d}t} &= 0.75x\left(1-\frac{x}{20}\right) \quad -1.5xy \quad -0.5xz \\
\frac{\mathrm{d}y}{\mathrm{d}t} &= \quad y\left(1-\frac{y}{25}\right) \quad -0.75xy \quad -1.25yz \\
\frac{\mathrm{d}z}{\mathrm{d}t} &= 1.5z\left(1-\frac{z}{30}\right) \quad -xz \quad -yz
\end{aligned}$$

where $x$, $y$, and $z$ are the population per unit area of species X, Y, and Z. Note that there are additional quadratic terms, differing signs, and carrying capacities unique to each species when compared with the predator-prey example presented in class.

   (a) Before you start programming, write down the discretized governing equations for this problem using the forward Euler method. Use subscripts $k$ and $k+1$ to denote the known values at timestep $k$ and the values to be determined at timestep $k+1$, respectively.

   (b) The initial populations of $x$, $y$, and $z$ are 2, 2.49, and 1.5 at $t = 0$. Write a program using the forward Euler method to calculate the population of these three species up to $t_{final} = 12$. Use timestep size $\Delta t = 0.001$. Print the time-varying populations of these three species to the command window using the format shown below:

```
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
  ⋮
```

   Where the rows in *your* table will be printed up to $t = t_{final}$. Note that the entries in the table are printed only at $t = 0.0, 0.5, 1.0, ..., 12.0$, an interval chosen *for visualization purposes only* and *not* related to the underlying physics of the problem (i.e., the timestep size, $\Delta t$, can still be 0.001, you'll just print lines in the table less frequently than every step to reduce table size).

   (c) Experiment with different initial population values (perhaps around three different sets) and observe the population-vs-time table for each. Using this guess and check approach, can you find a balancing point at which all these species coexist peacefully, or will one always crowd out the other two?

   (d) Use the `tic` and `toc` commands to time your code (`help tic`). What happens to the timing results when you use values of $\Delta t$ both larger and smaller than 0.001? Be sure to include the timing results in your report.

(e) Using the initial conditions listed in Part (b) and $\Delta t = 0.001$, which species survives? Which species survives when the initial conditions are kept the same but the timestep is increased to $\Delta t = 0.01$? Which of these two outcomes reflects the more accurate interaction between these competing populations and why?

**Note:** Although you're encouraged to change the initial conditions and $\Delta t$ to generate results for your report, please turn in the version of your code with the initial conditions listed in Part (b) and $\Delta t = 0.001$.

2. **The Pocket Change Problem.**
Write a script using iteration to answer the following question: *what is the average number of coins you can expect to receive in your change after a cash transaction?* To solve this problem, your script must calculate the number of coins required to form all 100 cent amounts in the range $[00, 01, 02, ..., 99]$ and then compute the *average*. We'll assume that all quantities of change are equally likely (sales tax is probably a good randomizer) and that the denominations available are:

$$\begin{aligned} \text{Quarter} &= 25 \\ \text{Dime} &= 10 \\ \text{Nickel} &= 5 \\ \text{Penny} &= 1 \end{aligned}$$

Additionally, we'll assume that all the individual amounts are formed using the minimum number of coins. For example, forming 58 cents requires 2 quarters, 1 nickel, and 3 pennies for a total of 6 coins.

(a) Report the average number of coins using the exact formatting shown below:

```
Average Number of Coins = 0.00
```

Where `0.00` will be replaced with your answer formatted using `'%.2f'`.

(b) How does the average number of coins change if we imagine eliminating pennies from circulation? This would require rounding all prices to the nearest nickel; that is, we'll now only consider forming cent amounts in the range $[00, 05, 10, ..., 95]$. Can you come up with a simple mathematical argument or intuitive explanation to check your answer?

(c) Assuming we decide to leave the pennies alone, can you reduce the average number of coins by modifying any/all of the other three denominations? For example, if a dime was worth 11 cents and a quarter was worth 27 cents, would the average number of coins be decreased when compared with your answer to Part (a)? We'll assume that the value of the penny remains fixed at 1, and that we return to forming amounts in the familiar range $[00, 01, 02, ..., 99]$. As you experiment, keep in mind that the conventional logic for counting back change requires that a given coin does not exceed half the denomination above it; that is, $0.5 * \text{Quarter} \geq \text{Dime}$, $0.5 * \text{Dime} \geq \text{Nickel}$, etc. Try different values and report your optimal monetary system. Are there any disadvantages to your new system?

**Note:** Although you're encouraged to change the value of coins to generate results for your report, please turn in the version of your code using the denominations listed above forming the amounts in the range $[00, 01, 02, ..., 99]$.

3. **Nested Radicals.**
*You do not need to include a write up for Problem 3 in your report!* In this problem, we'll attempt to calculate the terms in the following series for a user-input value of $m$.

$$t_1 = \sqrt{m}$$
$$t_2 = \sqrt{m - \sqrt{m}}$$
$$t_3 = \sqrt{m - \sqrt{m + \sqrt{m}}}$$
$$t_4 = \sqrt{m - \sqrt{m + \sqrt{m - \sqrt{m}}}}$$
$$t_5 = \sqrt{m - \sqrt{m + \sqrt{m - \sqrt{m + \sqrt{m}}}}}$$

where $m > 1$ and an integer. Your script should prompt the user to enter a value for $m$, test that the value is valid, and *repeatedly* prompt for input as long as $m$ is invalid (this will mean displaying your "error" message using `fprintf` as opposed to `error`, which would immediately halt the script). Display your results using exactly the following format:

```
m = 31
t1 = 5.567764362830
t2 = 5.043038333899
t3 = 4.995284916320
t4 = 4.999641441490
   ⋮
```

Continue displaying this pattern until the difference between successive terms is less than $1 \times 10^{-12}$ in magnitude, i.e., $|t_n - t_{n-1}| < 1 \times 10^{-12}$. You should find that the limit is an integer for $m = 7,\ 13,\ 21,\ \text{or}\ 31$.

**Note:** Remember, `ctrl + c` to break an infinite loop, or better yet, avoid this situation by putting an upper limit on the number of iterations! *Remember, you do not need to include a write up for Problem 3 in your report.* Turn in the code only.

Using the naming convention presented in the syllabus, submit **two** separate files to the CCLE course website: (1) a .pdf of your written report and (2) a .zip file containing all of the MATLAB files written for the assignment. Remember to use good coding practices by keeping your code organized, choosing suitable variable names, and commenting where applicable. Any of your MATLAB .m files should contain a few comment lines at the top to provide the name of the script, a brief description of the function of the script, and your name and UID.