Due Thursday January 20th at 11:59pm

Problem 1

In the coding lectures we did some examples of using loops to fill in matrices that we did not necessarily have to use loops for. Now, let's fill in a matrix where a loop is necessary.

Create the following 20×21 matrix in MATLAB/python *without typing every entry in directly*:

$$A = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{21} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{6} & \cdots & \frac{1}{42} \\ \frac{1}{3} & \frac{1}{6} & \frac{1}{9} & \cdots & \frac{1}{63} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{20} & \frac{1}{40} & \frac{1}{60} & \cdots & \frac{1}{420} \end{pmatrix}.$$

Note that a_{ij} (i.e., the entry in the ith row, jth column, is equal to $1/(ij)$.

(1) Save a copy of the matrix A in the variable `A1`. (**Note: In MATLAB, the word "copy" doesn't matter here; you can just say `A1 = A`, but in python it is very important that you use `A1 = A.copy()`. Otherwise, any changes you make to A will also change A1.**)

(2) Make a new matrix B which is a copy of A, except the 15th row and 16th column are all zeros. Save a copy of the matrix B in the variable `A2`. (**Note: Just like in the last part, the word "copy" doesn't matter in MATLAB, but in python it is important not to write `B = A` or `A2 = B` and then modify the entries of B, because that will also change A.**

(3) Make a third matrix which contains the last three rows and the last five columns of the matrix B. Save a copy of the resulting matrix in the variable `A3`.

(4) Make a *column vector* containing the tenth column of the matrix B and save it in the variable `A4`.

Problem 2

The harmonic series is the sum

$$\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots$$

The nth partial sum of the harmonic series is

$$S_n = \sum_{k=1}^{n} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n-1} + \frac{1}{n}.$$

(1) Calculate S_{20} using a for loop. Save your answer in the variable `A5`.

(2) Calculate S_{200} using a for loop. Save your answer in the variable `A6`.

(3) Calculate the smallest number n such that $S_n > 10$. Save the corresponding value of n in a variable named `A7` and the corresponding value of S_n in a variable named `A8`.

(4) Calculate the smallest number n such that $S_n > 20$. Save the corresponding value of n in a variable named `A9` and the corresponding value of S_n in a variable named `A10`. **Warning: If you are not careful, this code will be extremely slow. You should be able to write a solution that finishes in at most a few seconds, but it is also possible to write a solution that takes several minutes. Gradescope has a time limit, and so you will only get credit if your solution finishes in a reasonable amount of time.**

Problem 3

Recall from the previous coding project that the logistic map is often used in population ecology to model population growth. Ecologists think in terms of elapsed time, however in math we often think in terms of timescales and timesteps in order to generalize the behavior for many other systems. Lets write it in a more mathematical format

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

and try to spot some interesting trajectories. Here each n represents a timestep, but will work the same way as the previous coding project. This is called a *discrete dynamical system*, and it can become *chaotic*. We'll talk more about chaos theory later on in the quarter, but for now we can think of chaos as a system having seemingly random long-time behavior.

In the last assignment, you found $P(3)$ and $P(4)$ for several different values of $P(0)$, r and K. You probably noticed that it is somewhat annoying to solve this problem without using a loop. In this problem we want to analyze long-time behavior, so we will need to use loops.

It will also be very useful to write a function (either a local function in MATLAB or a normal function with the `def` keyword in python) to calculate x_n for an arbitrary n. In the solution template I will include guidance on where to include your function and what goes in it.

(1) Suppose that $r = 2.75$ and $x_0 = 0.2$. Calculate the first hundred iterates, $x_{n+1} = rx_n(1 - x_n)$, and save it as a row vector named `A11`.

Now here comes the hard part: in a separate file or command window (for MATLAB)/console (for Python) plot the timeseries (iteration number $[1, 2, \ldots, 100]$ vs iterates $[x_0, x_1, \ldots, x_{99}]$). We can visually analyze the behavior of the map from the timeseries plot: it is either going to converge to a fixed point (called a "sink"), bounce between the same two (or several) points (called "periodic"), or behave in a seemingly random manner (called "chaotic").

After you observe the plot, come back to your solution file and save the variable `behavior` as `behavior = 1` if there appears to be a sink, `behavior = 2` if there appears to be a periodic orbit, or `behavior = 3` if it appears to be chaotic. We can also analyze this more quantitatively by looking at the distribution of iterates: the more disordered the iterates become the more its topological entropy rises. Calculate the standard deviation of the iterates using the function `x_std = std(x)` on MATLAB or `x_std = np.std(x)` on Python. Save the variable `A12` as a row vector $[x_{std}, \text{behavior}]$.

(2) Repeat the same process with $r = 3.25$ and $x_0 = 0.2$ and save the values of the first hundred iterates and save it as a row vector named `A13`. Just as above, save the variable `A14` as a row vector $[x_{std}, \text{behavior}]$.

(3) Repeat the same process with $r = 3.75$ and $x_0 = 0.2$ and save the values of the first hundred iterates and save it as a row vector named `A15`. Save the variable `A16` as a row vector $[x_{std}, \text{behavior}]$.