MAT 341 | N. Battista | Computational Math
HW 2 | | Fall 2019

HW Goals:

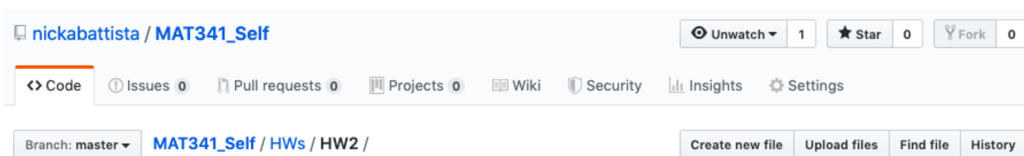
The purpose of this homework is to familiarize yourself with `for-loops` *and* `while-loops` *for mathematical purposes, using series approximations.*

HW DUE:

Friday, Sept. 20 by 11:59pm

WHERE:

Put all of the codes for this homework into your personal GitHub: `MAT341_Self→HWs→HW2`. That is, you will have to `add, commit` and `push` these scripts to your GitHub for me to pull, see below for example:



1. Each of the following sequences converges to π:

$$a_n = \frac{6}{\sqrt{3}} \sum_{k=0}^{n} \frac{(-1)^k}{3^k\ (2k+1)}$$

$$b_n = 16 \sum_{k=0}^{n} \frac{(-1)^k}{5^{2k+1}\ (2k+1)} - 4 \sum_{k=0}^{n} \frac{(-1)^k}{239^{2k+1}\ (2k+1)}$$

Write a script called `calculate_Pi_Sums.m`, which takes no input arguments, nor returns anything, e.g.,

```
function calculate_Pi_Sums()
```

In this script, compute the above sums to find N_A and N_B (and have it print to screen $a_0, a_1, \ldots, a_{N_A}$ and $b_0, b_1, \ldots, b_{N_B}$) where N_A and N_B are the smallest integers such that

$$|a_{N_A} - \pi| < tol \quad \text{and} \quad |b_{N_B} - \pi| < tol,$$

where $tol = 10^{-6}$. Write this script using either `for-loops` or `while-loops`, or a combination.

2. Write a script called `calculate_Nested_For_Loop_Time.m` that takes a single input argument of an integer **N** and passes back **time**, e.g.,

   function time = calculate_Nested_For_Loop_Time(N)

   We can use the `tic/toc` commands in MATLAB to test how long it takes a snippet of code to run. Explore how long it takes to execute the following series of *Nested For Loops*:

```
tic
k=0;
for i1 = 1:N
    for i2 = 1:N
        for i3 = 1:N
            for i4 = 1:N

                k = k+1;

            end
        end
    end
end
time = toc
```

   Write another script called `plot_Nested_Times.m` that takes no inputs (nor returns anything), e.g.,

   function plot_Nested_Times()

   that uses the previous script `calculate_Nested_For_Loop_Time` to save the *time* it takes to run for a variety of different **N** values and then makes a plot of *Time vs. N*. In particular, have it test over the following vector of N-values:

$$N = [1{:}1{:}10 \;\; 20{:}10{:}100 \;\; 125\ 150\ 175\ 200\ 225\ 250];$$

   Use a **log-log** plot to plot the data (instead of `plot` use `loglog`). Make sure to use the *Times* you calculate as the dependent variable (vertical axis) and the number N as the independent variable (horizontal axis).

   Make sure to:

   (a) Label the axes
   (b) Change the color of the plot to *magenta*
   (c) Change the thickness of the line (e.g., line width)

3. Let m be a positive integer and consider the sequence $\{t_n\}_{n=1}^{\infty}$:

$$\begin{aligned}
t_1 &= \sqrt{m}\\
t_2 &= \sqrt{m-\sqrt{m}}\\
t_3 &= \sqrt{m-\sqrt{m+\sqrt{m}}}\\
t_4 &= \sqrt{m-\sqrt{m+\sqrt{m-\sqrt{m}}}}\\
t_5 &= \sqrt{m-\sqrt{m+\sqrt{m-\sqrt{m+\sqrt{m}}}}}\\
t_6 &= \sqrt{m-\sqrt{m+\sqrt{m-\sqrt{m+\sqrt{m-\sqrt{m}}}}}}\\
&\vdots
\end{aligned}$$

Notice how the sign changes underneath the successive square roots. Study the pattern and write a script called `square_Root_Sequence.m` to determine the limit of t_n as n gets large. Have the script take as input: m (value to test in sequence) and n (to determine n^{th} value in sequence, e.g., t_n), and have it return the value of the sequence, call it *val*, e.g.,

```
function val = square_Root_Sequence(m,n)
```

What does the limit appear to be for $m = 13, 31$ and 43? Write your answers as comments at the bottom of the script.

Hints:

(a) Do **<u>not</u>** calculate t_n from t_{n-1}, e.g., knowing t_4 won't help you get t_5. Compute each t_n independently.

(b) For $m = 7$, the sequence converges to 2.

Note that we have not proved any of these sequences for particular m converge to these values, but rather have gained insight into what it looks like they do.