Numerical Mathematics: Homework 1

Problem 1

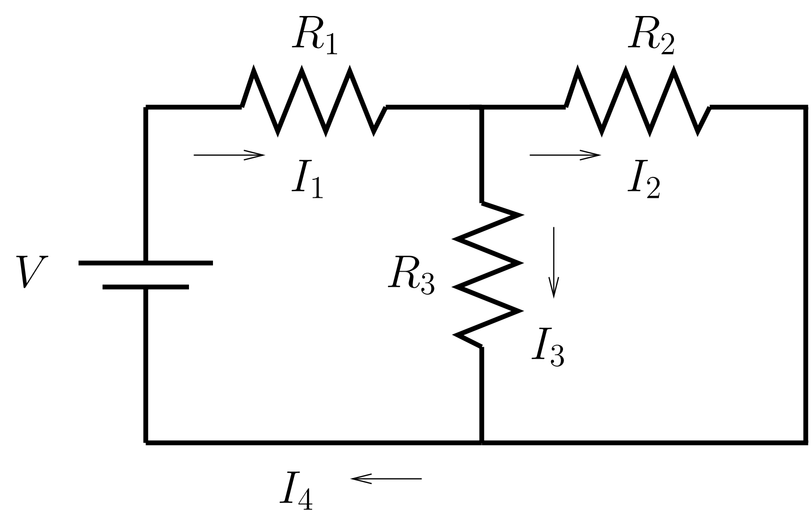
Prove that for any $\epsilon > 0$, $n^{1+\epsilon} \notin \mathcal{O}(n \log n)$ (but $n \log n \in \mathcal{O}(n^{1+\epsilon})$). Hence, $n^{1+\epsilon}$ grows asymptotically faster than $n \log n$.

Problem 2

For the DC circuit shown here, Kirchhoff's Rules and Ohm's Law tell us the following relationships between the currents I_1, I_2, I_3, I_4, the resistances R_1, R_2, R_3, and the voltage V provided by the battery:

$$\begin{aligned} I_2 + I_3 &= I_1 \\ I_2 + I_3 &= I_4 \\ I_1 R_1 + I_3 R_3 &= V \\ I_1 R_1 + I_2 R_2 &= V \end{aligned}$$

Submit a function `find_current.m` which takes as inputs the values for V, R_1, R_2, and R_3 (in that order), and returns as output a column vector containing the values of the currents I_1, I_2, I_3, I_4 (again, in that same order). You are allowed to use the backslash operator here if you want!



Problem 3

In this problem, you are going to get practice in plotting functions. Consider the signal:

$$x = \sin(50\pi t) + \frac{1}{4}\sin(226\pi t) + (1/100)t^2$$

(a)
Build a function called `PlotSignal.m` which plots this curve on the interval $0 \leq t \leq 0.1$. Use an increment of $\Delta t = 0.001$. Create this signal by first allocating a vector of zeros for x and then use a for loop to fill in the vector. Plot the function ($x(t)$ in the vertical axis, t in the horizontal axis). Give the figure a title called 'signal', add labels in the x-axis ('t') and y-axis ('x(t)'). Note that `PlotSignal.m` doesn't have any inputs

and outputs.

(b)
Do the same as in part IIa, but now vectorize the code. That is, use the element-by-element operation wherever it is necessary so that *you no longer need to use a for loop*. Call this new function `PlotSignal_vectorized.m`.

Problem 4

Make a function calld `evaluate_pi.m` which plots the evaluates π using the formula:

$$\frac{\pi}{2} = 1 + \frac{1}{3} + \frac{1 \cdot 2}{3 \cdot 5} + \frac{1 \cdot 2 \cdot 3}{3 \cdot 5 \cdot 7} + \frac{1 \cdot 2 \cdot 3 \cdot 4}{3 \cdot 5 \cdot 7 \cdot 9} + \dots$$

The function must take as argument n, the number of terms in the sum. Code it efficiently!

(extra) Problem 5

Apply Horner's method to evaluate a polynomial of the form:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

In other words, by rewriting the polynomial into the form:

$$p(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + x(\cdots(a_{n-1} + a_n x)\cdots))))$$

we can introduce the variables:

$$\begin{aligned} b_n &:= a_n \\ b_{n-1} &:= a_{n-1} + b_n x \\ &\vdots \\ b_0 &= a_0 + b_1 x \end{aligned}$$

to evaluate the polynomial efficiently. Write a MATLAB function called `poly_eval_horner.m` which evaluates a polynomial of degree n using Horner's iteration scheme. Use the following structure: `function [pofx] = poly_eval_horner(x,a)` where `x` is a scalar input, `a` is a column (or row) vector of size $n+1$ with the coefficients $a_0, a_1, \dots, a_n$ in it, and `pofx` is the function value $p(x)$.