FUN WITH PRIME NUMBERS

Math 3343 · Introduction to Mathematical Software · Homework 3

Due on Wednesday 02/07/2018 in class

Consider the pseudocode for computing an array of prime numbers less than or equal to n.

INPUT n - positive integer
OUTPUT p is an array of prime numbers less than or equal to n. ($\pi(n)$ is the number of prime numbers less than or equal to n.)

```
$i = 0$
for $k = 2, \dots n$
  for $j = 2, 3, \dots k-1$
    if remainder$(k/j) = 0$
      then $k$ is composite, stop loop and go to next $k$ value
    end if statement
  end for loop
  If inner loop cycled all the way through
    then $i = i + 1$
    then $p_i = k$
  end if statement
end for loop
```

The algorithm can be summarized in this way. Consider whether each integer from 2 to n is prime by dividing it by all integers greater than 1 and less than the number being considered. If one of those lesser numbers divides evenly into our candidate prime, then it's composite. Otherwise it is prime and we add that candidate number to the list.

This algorithm is implemented in the code `prime_list_slow.m`. As the code name suggests, we can accelerate the speed of the code. That is the subject of this homework.

Theorem 1 *If n is not divisible by any prime number less than n, then n is a prime number.*
Theorem 2 *If n is not divisible by any integer between two and $\sqrt{n}$, then n is prime.*

1. (20 points) Explain how the the two theorems above can be used to accelerate the code `prime_list_slow.m` created in class (and available via Blackboard).

2. (20 points) Write a pseudocode incorporating these changes with input n and output $p \in \mathbb{N}^{\pi(n)}$, which is an array of all prime numbers less than or equal to n.

3. (20 points) Create a Matlab function, titled `prime_list` that implements your pseudocode above. Your function should have the following input and output:

   Input n - positive integer
   Output p - an array of prime number less than or equal to n

   The first line of your code should therefore read: `function p = prime_list(n)`.

Consider the following alternate method for computing a list or prime numbers, (known as the Sieve of Eratosthenes)

```
INPUT $n$ - positive integer
OUTPUT $p$ is an array of prime numbers less than or equal to $n$. ($\pi(n)$ is the number of prime numbers less than or equal to $n$.)
$p =$ array of integers 1 through $n$
$p_1 = 0$.
for $k = 2, 3, \ldots, \sqrt{n}$
  If $p_k \neq 0$
    then set $p_j = 0$ for $j = k^2,\ k^2 + k,\ k^2 + 2k,\ \ldots, n$
  end If statement
end for loop
Extract all nonzero elements of the array $p$.
```

---

4. (20 points) Create a Matlab function, titled `prime_sieve` that implements the pseudocode above. Your function should have the following input and output:

   Input n - positive integer
   Output p - an array of prime number less than or equal to n

   The first line of your code should therefore read: `function p = prime_sieve(n)`.

5. (20 points) Write a script titled `HW3.m` that uses the `tic` and `toc` functions to plot in the same figure the amount of time each of the three codes requires to produce the prime number list as a function of n for values of $n = 10^1,\ 10^2,\ \ldots, 10^5$. Use a `loglog` scaling in your plot.

Bonus (20 points) Create a Matlab function titled `prime_count.m` that takes as an input n, and returns the array $\psi \in \mathbb{N}^n$, where $\psi_k = \pi(k)$ is the number of prime numbers less than or equal to k (for example, $\psi_2 = \pi(2) = 1$, $\psi_5 = \pi(5) = 3$, $\psi_{10} = \pi(10) = 4$) for $k = 1, 2, \ldots n$. Use that function to create a plot of $\pi(k)$ vs. k for $k = 2,\ 3,\ 4, \ldots, 10^6$. In order to receive credit, your code must call a prime generating function **exactly once!** to compute $\pi(k)$ for $k = 1, 2, \ldots, n$.