# Math 343 Lab 3: Spatial and Temporal Complexity

## Objective

This lab examines both the spatial and temporal complexity of numerical algorithms.

## The Timing of Algorithms

In this section, we determine the time it takes to multiply a matrix and a vector together. Type the following into a file called mytimer.m. Be sure the path is set to the correct directory:

```
n = 1000;
A = randn(n);
b = randn(n,1);

tic
A*b;
toc
```

Now execute the script by typing `mytimer` at the command line. Recall that semi-colons suppress the output, which is particularly important for this script because these are big matrices!

The `tic` command starts the clock and the `toc` command outputs the time that has elapsed since the clock started. Note that we didn't call `tic` at the beginning of the script because we wanted to measure the time it took to multiply $A$ and $b$, and not include the time it took to create them.

## Spatial Complexity

Re-run the script `mytimer` and increase $n$ by increments of 500. At what point is there a considerable jump in the execution time? How many megabytes[1]

[1] A floating-point number requires 8 bytes of storage. Hence to compute amount of memory needed to store the matrix $A$, multiply 8 by the number of entries in the matrix. Because $A$ is an $n \times n$ matrix, the space needed is $8n^2$ bytes.

is the matrix A when the big jump occurs? The reason the program jumped in execution time is that the matrix $A$ was so big that it couldn't fit in the computer's RAM. As a result the operating system had to write part of the matrix to the hard disk in what's called virtual memory. The access time for virtual memory is at least an order of magnitude longer than that of RAM, and so once your program spills into virtual memory, your execution time jumps considerably.

As $n$ increases, the size of the matrix increases like $8n^2$. We denote this rate of growth as "quadratic" or $O(n^2)$, which means that if $n$ doubles, then the memory approximately quadruples. This is the spatial complexity of matrix-vector multiplication.

# Temporal Complexity

In this section we explore the temporal complexity. Modify the above script to plot the run-time of the matrix-vector multiplication for several values of $n$ ranging from small values to near the value in the previous section where the big jump occurred. For example, if the big jump occurred slightly after $n = 5000$, do something like the following:

```
k = 1;
inc = 1000:250:5000;
for n = inc  %starts at 1000, increments by 250, stops at 5000.
A = randn(n);
B = randn(n,1);
tic
A*B;
out(k) = toc;
k = k + 1;
end

plot(log(inc),log(out))
```

The last line takes the log of both the $x$ and $y$ axis and plots the logarithm of the run-time against the logarithm of $n$. Below is an example of the output.

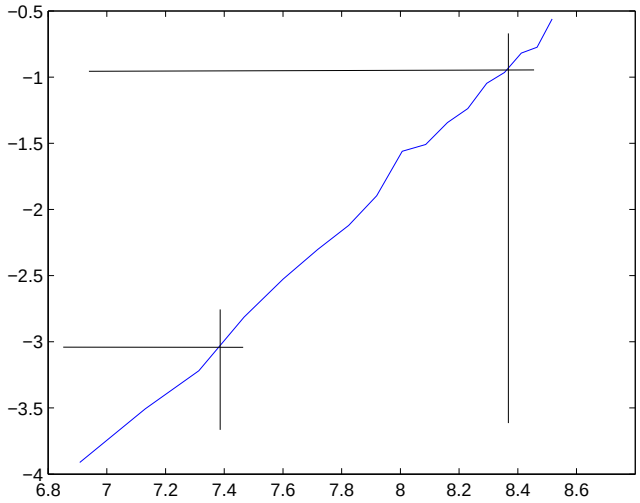


Figure 1: By drawing lines, it's easy to see that the slope $m$ in the logarithmic scale is close to $m = 2$. This implies the the temporal complexity of matrix-vector multiplication is $O(n^2)$. In other words, if $n$ doubles, then the run-time approximately quadruples.

# Assignment

**Problem 1.** *Repeat the above for matrix-matrix multiplication. In other words, find the spatial and temporal complexity of matrix-matrix multiplication, i.e., multiplying two $n \times n$ random matrices together. Graph the temporal complexity on a logarithmic scale and justify your answers with a short discussion.*