MACM 316 — Computing Assignment #2

David Pham / dhpham@sfu.ca / 301318482

Values of N and N_{ex}

I knew that I was eventually going to take $log_{10}(N)$ and log_{10}(avg time to solve mtx), so I chose to define $N = \lceil 10^k \rceil$, where $k = 1, 1.52, 2.53$, to correspond nicely with log_{10} values.

I started by defining `dense_Nex`, `tri_Nex`, and `perm_Nex` to be the number of solves for each type of matrix $[M_d], [M_t]$, and $[M_p]$, respectively. After some fiddling around, I managed to find values for each `*_Nex` that would automatically scale well as N increased each loop.

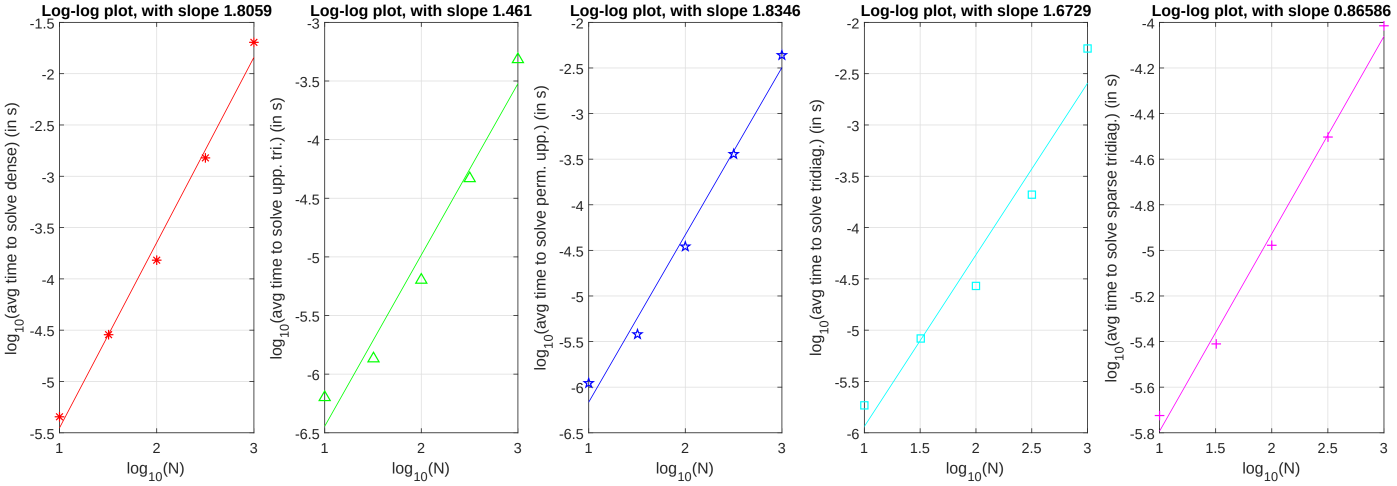

Figure 1: log-log plots for dense, upper triangular, permuted upper triangular, tridiagonal, and sparse tridiagonal matrices.

Conclusions

It seems like using *backslash* in MATLAB somehow cuts down on the computational cost as N increases. Theoretically, the number of flops in solving $[M_d]$ should increase by $O(N^3)$, giving the log-log plot a slope of 3. However, the computations have shown the slope to be ≈ 1.8 (Figure 1), which is jarring. Perhaps, since we don't generate a new random matrix for each solve, MATLAB caches the LU factorization of the Matrix somehow?

Interestingly, the slope in the log-log plot of the permuted upper triangular matrix $[M_p]$ is greater than that of the dense matrix $[M_d]$. MATLAB seems to treat them both similarly, but solving $[M_p]$ takes longer for some reason. I take this to mean that MATLAB doesn't recognize the matrix as being a permuted upper triangular matrix, and tries to solve it as a dense matrix: as MATLAB goes about row reduction, it often encounters situations where row interchanges are necessary, unlike in the dense matrix case.

$[M_3]$ still requires some row reduction, whereas $[M_t]$ requires only back-subsitution. This bit of reduction is still enough to put tridiagonal matrices behind upper triangular matrices, and the time it takes to solve $[M_3]$ is somewhere between the time it takes to solve a dense matrix and an upper triangular matrix. Out of these matrices, $[M_{3s}]$ is the fastest type of matrix to solve: its slope is ≈ 0.87, making it even faster than $O(n)$.