Crowdmark Due Date: Monday 28 January, 2019 at 11:55

- Please follow the *Guidelines for Computing Assignments* found on Canvas (under FAQ).
- Submit a one-page report via Crowdmark, grading scheme is also found on the FAQ.
- Acknowledge any and all collaborations and assistance from classmates/TAs/instructor.

GE Timing Test

From the lecture on operation counts, the *flop counts* involved in the GE solution for an $N \times N$ linear system, $A\vec{x} = \vec{b}$, are

- $O(n^3)$, if A is dense and we use GE; and
- $O(n^2)$, if A is upper triangular and only back substitution is needed.

In this assignment, you will measure the actual computing performance of Matlab on a particular computer, and compare and contrast the data with the theoretical flop counts. We will also be discovering that Matlab's 'backslash' algorithm is more sophisticated than the textbook GE, and offers built-in efficiencies for special matrix classes.

Before you begin, note that your performance measurements will depend on the computing platform used. Be sure to gather all the data on the same machine! Include in your report relevant identifying information, including computer model and year, processor and clockspeed, operating system, and Matlab version (type *version* at the Matlab prompt). Furthermore, it is advantageous that you stop all other activities on your computer while testing.

To begin, start by following the link in the Canvas assignment to the *math-apps* demo. Run this demo once, and check that the ouput gives estimated solve times for 3 types of matricies (see below). **Then, copy the entire code (including portions outside the edit window), and paste it into your own Matlab script. You will build upon this code to complete the assignment.** Once you have succesfully run the demo and extracted the code, you **do not** need to submit anything else to the *math-apps* online portal for this assignment.

This script that you have just run and copied performs the set-up for three $N \times N$ linear solve tests with matrices $[M]$ having certain patterns of zero entries

- $[M_d]$, a dense matrix with random entries using *randn* (no zeros);
- $[M_t]$, an upper triangular matrix extracted from $[M_d]$ (using *triu*);
- $[M_p]$, a randomly row-permuted form of $[M_t]$.

The test uses righthand side vectors $\vec{b}$ that give solutions $\vec{x}$ that are vectors of all 1's.

Matlab has a handy set of tools, called *tic* and *toc*, that can be used to time portions of your script. Calling *tic* in Matlab will start a timer, and *toc* will output the current time elapsed. Wrapping a portion of your script between *tic* and *toc* will allow you to time how long, in seconds, that section of code took to run. However, Matlab documentation does NOT recommend to use *tic* and *toc* for processes that take less than 0.1 seconds to run, as the accuracy for short times is unreliable. Since Matlab can solve even large matrix problems incredibly quickly, this presents a problem.

The way that the demo works around this is that it repeats each matrix solve N_{ex} number of times, and it records the **total** time for all the solves. This total time can be designed to be well in excess of the 0.1 second minimum, despite the fact that the average time per solve may be much less.

For this computing assignment, incorporate the following into your experiment and report:

- Modify the script to collect average solve times over a range of matrix sizes N_{ex}, for all three of the above described matrix classes. However, we expect that solve times for a smaller matrix should be faster, therefore a larger N_{ex} is needed to get the measured time above *tic* and *toc*'s range of accuracy. Likewise, different types of matrices may require more or less N_{ex} than others. The takeaway: using the same N_{ex} for all sizes and types of matrix is poor experimental design.
- The theory for GE operation counts suggests that for large N, a power-law relationship between the flop count and matrix size. This suggests that a log plot of solve times versus matrix size should be considered. On such a log-log plot, the power for the relationship can be obtained — see the plot example that accompanies this assignment (*power_law_plot.m*).
- Discuss the comparison of these results, to each other, as well as to the theoretical flop counts. What might this reveal for what happens with *backslash* in Matlab? Particularly, what might one make of the observations for the permuted upper-triangular case, $[M_p]$? When adressing this point, it may help to recall the flowchart shown in lecture, detailing Matlab's 'backslash' implementation.
- Having benchmarked the timing against some known results, now do additional testing on two other matrices
  - $[M_3]$, a tri-diagonal matrix extracted from $[M_d]$ (using *diag*); and
  - $[M_{3s}]$, a sparse-version of $[M_3]$ (using *sparse*).

  The code to build these matricies has already been included in the demo script, and simply needs to be uncommented.