MACM 316 – D100
January 28, 2019

Kai Sackville-Hii (301310336)
ksackvil@sfu.ca

Computing Assignment 2

GE Timing Test

The goal of this assignment is to compare the actual computing performance of Matlab's backslash to the theoretical flop count, with a variety of matrices and matrix sizes. Each matrix is solved N_{ex} number of times, yielding a total time for N_{ex} solves. The average solve time, which is computed using this total time, is used to obtain the power-law relationship between the flop count and matrix size. The machine used for this experiment has the following specs: 2014 M5A78L ASUS motherboard with AMD FX-4350 CPU, clock speed of 4218.57 MHz, running Windows 10 and Matlab R2018b.

Determining Values of N_{ex}

To calculate the average solve time accurately using Matlab's *tic* and *toc* functions, the timed process must take longer than 0.1 seconds. N_{ex} was chosen so that the total solve time is in excess of 0.1 seconds. N was set to range from 1000 to 3000, ensuring accuracy while remaining efficient. As N increases we expect solve times to also increase, Having N_{ex} decrease at a linear rate as N increases will ensure that the total compute time remains linear. Five matrices were solved:

M_d = Dense matrix,
M_t = Upper triangular matrix,
M_p = Row-permuted form of M_t,
M_3 = Tri-diagonal matrix extracted from M_d,
M_{3s} = Sparse version of M_3.

Different compute times are expected for each matrix. This means that N_{ex} must be a different function for each matrix to ensure the total compute time always exceeds 0.1 seconds.

Let

$N_{ex}(M_d) = -0.025*n+100$
$N_{ex}(M_t) = -0.025*n+1000$
$N_{ex}(M_p) = -0.025*n+300$
$N_{ex}(M_3) = -0.025*n+200$
$N_{ex}(M_{3s}) = -0.025*n+10000$

Obtaining the Power Relationship

The power relationship between the flop count and matrix size can be obtained by plotting the log of average solve time verses log of matrix size. The results for each matrix are shown in Table 1.

Matrix	Power	Flop Count
M_d	2.8291	$O(n^{2.8291})$
M_t	2.1176	$O(n^{2.1176})$
M_p	2.1405	$O(n^{2.1405})$
M_3	2.2977	$O(n^{2.2977})$
M_{3s}	1.0364	$O(n^{1.0364})$

Table 1: Power Relationship for Square Matrices Size N

Out of the results shown in Table 1, M_d has the largest flop count and M_{3s} has the least. This result is expected, as a dense matrix requires row reduction with back substitution and a sparse matrix has many zeros. M_p was created by row-permuting M_t so the flop counts for M_p should be slightly larger than M_t. M_3 was created by extracting the tri-diagonal matrix from M_d, so we expect M_3 to have a lesser flop count than M_d.

Evaluating Matlabs Backslash

Comparing the theoretical and actual flop counts of M_d we see that the actual flop count is less. This means Matlab's backslash must be more robust then a straight GE. Referencing MathWorks published algorithm confirms this. Matlab's backslash implementation changes its approach based on the input matrix. For the matrix M_d, LU factorization is used, which explains our efficient resulting flop count. Matlab's backslash checks if the input matrix is triangular. If not it checks if it is permuted triangular. It takes one extra logic check to evaluate a permuted triangular matrix verses a triangular, which could explain why the flop count of M_p is slightly larger than M_t. These logic checks could also be why the actual flop count of M_t is larger than the theoretical.