## **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 Nex number of times, yielding a total time for Nex 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 Nex**

To calculate the average solve time accurately using Matlab’s *tic* and *toc* functions, the timed process must take longer than 0.1 seconds. Nex 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 Nex decrease at a linear rate as N increases will ensure that the total compute time remains linear. Five matrices were solved:

Md = Dense matrix,  
Mt = Upper triangular matrix,  
Mp = Row-permuted form of Mt,  
M3 = Tri-diagonal matrix extracted from Md,  
M3s = Sparse version of M3.

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

Let  
 Nex (Md­) = -0.025\*n+100  
 Nex (Mt) = -0.025\*n+1000  
 Nex (Mp) = -0.025\*n+300  
 Nex (M3) = -0.025\*n+200  
 Nex (M3s) = -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** |
| Md | 2.8291 | O(n2.8291) |
| Mt | 2.1176 | O(n2.1176) |
| Mp | 2.1405 | O(n2.1405) |
| M3 | 2.2977 | O(n2.2977) |
| M3s | 1.0364 | O(n1.0364) |

*Table 1: Power Relationship for Square Matrices Size N*

Out of the results shown in Table 1, Md has the largest flop count and M3s has the least. This result is expected, as a dense matrix requires row reduction with back substitution and a sparse matrix has many zeros. Mp was created by row-permuting Mt so the flop counts for Mp­ should be slightly larger than Mt. M3 was created by extracting the tri-diagonal matrix from Md, so we expect M3 to have a lesser flop count than Md.

**Evaluating Matlabs Backslash**

Comparing the theoretical and actual flop counts of Md 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 Md,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 Mp is slightly larger than Mt. These logic checks could also be why the actual flop count of Mt is larger than the theoretical.