HOMEWORK 3

JASON MEDCOFF

1. Parallel Algorithm

For the parallel algorithm, I wanted to begin with a simple implementation using tiles of shared memory, then "upgrade" the code to take advantage of asynchronous transfer and multiplication. The serial implementation was finished easily, and is error-free as far as I can tell. Upon beginning work on the parallel version, I ran into numerical error in the result of the multiplication. After reimplementing the multiplication kernel, the same error was appearing, so I resorted to testing my code with cuBLAS's floating point matrix multiplication subroutine `cublasSgemm_v2`.

The use of this function initially demonstrated a different numerical error. Soon after, I read the documentation more closely and noticed that column major matrices are assumed. So, I used the cuBLAS operator `cublas_op_t` to transpose the input matrices appropriately, making them column major. Doing this initially resulted in error of zero, but after changing the dimensions of the input matrices, the error appeared again.

I continued to experiment and tweak my code, looking for the source of the numerical error. I was printing the input matrices to console, and observed that all of them were correctly constructed. Despite this, error still appeared in the product. I temporarily modified my verification function to look at the individual submatrices in the result of multiplication, and noticed that the error was accumulating only in the upper right block matrix. This initially led me to believe that the error was appearing due to negation of the identity and X somehow not carrying through. Further investigation showed that the values in the top right block represented exactly $4X$; meaning instead of the expected result

$$\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}$$

I was obtaining the result

$$\begin{bmatrix} I & 4X \\ 0 & -I \end{bmatrix}.$$

After careful scrutiny of my functions for input matrix construction, I could not find any errors. My suspicion at this point is that the matrices are being properly multiplied, but somehow there is some logic error that I cannot see. So despite the numerical error, I will still try to report on the running time and execution details.

2. Compilation and Running

Some executions were run on my home cuda-enabled machine, but most work was done on yoko.secs.oakland.edu. The file of interest is `parallel2.cu`, which contains the most recent changes and experimentation on the algorithm. The macro `SUBMATRIX_SIZE` denotes the size of X and the other block matrices. The compiler arguments follow:

```
nvcc -lcublas parallel2.cu
```

3. Experiment

Running the code on some different sizes, we have

Table 1. Running times

Size (N)	Time (real)	Time (user)	Time (sys)	GPU Time (ms)	Error
500	3.003	0.055	1.297	6.264	126238.46
1000	2.833	0.154	1.270	20.543	502254.844
2000	3.552	0.661	1.497	83.725	2004453.375
5000	8.213	4.066	2.881	507.214	12510072.0
10000	27.446	15.616	9.089	2104.926	16797216.0

4. Code Listing

Due to size considerations, the serial and parallel code are attached as separate files.