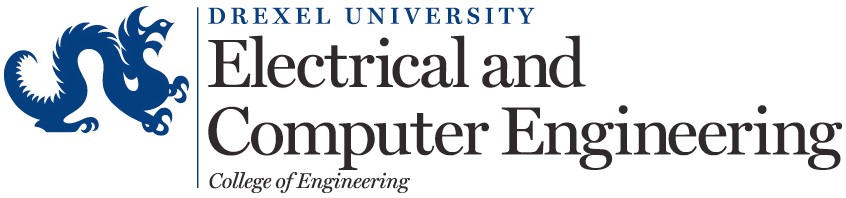
****

Drexel University

Electrical and Computer Engineering Dept.

ECEC-413

**Jacobi Iterative Method with CUDA**

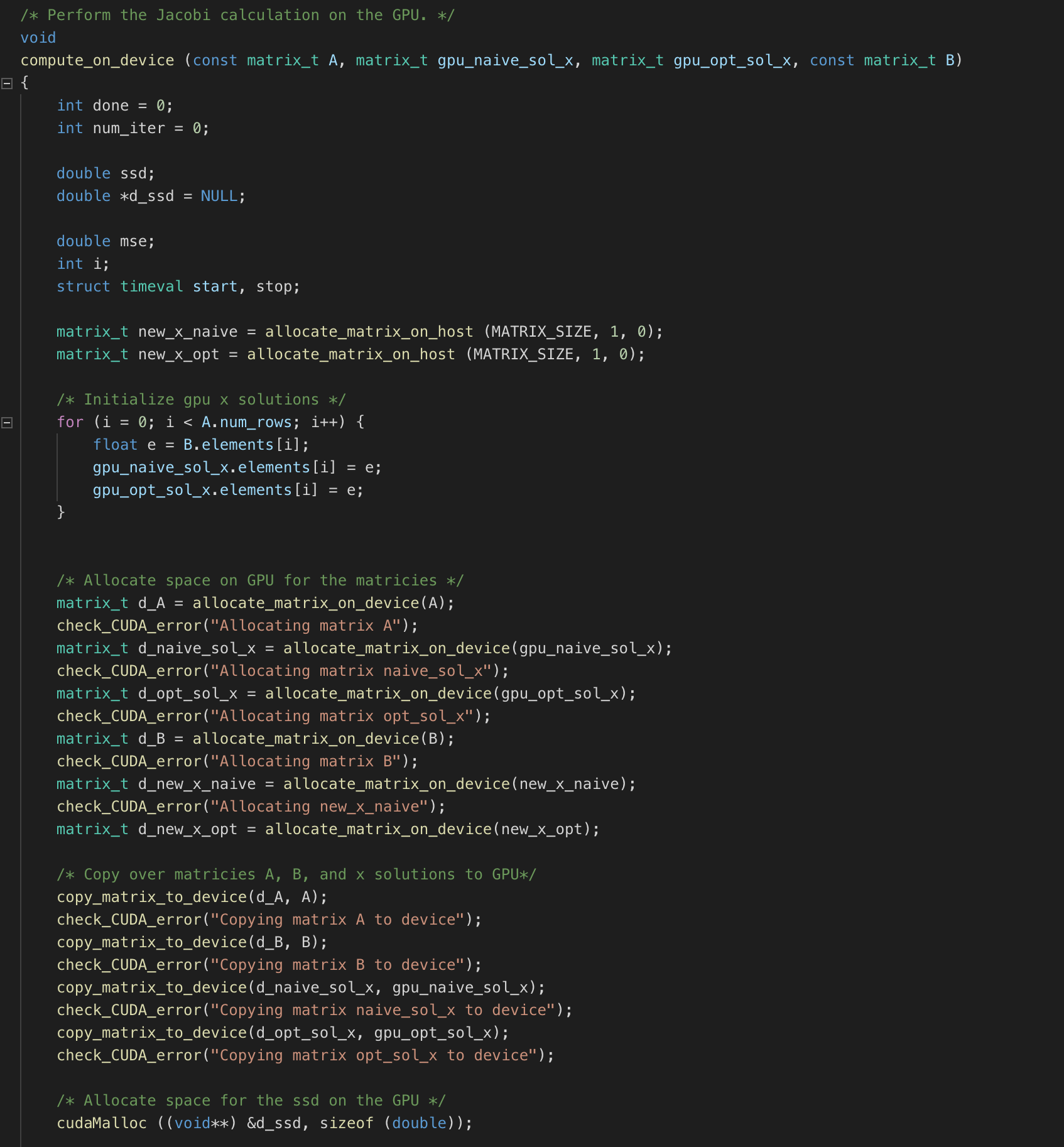
**Chris Kasper**

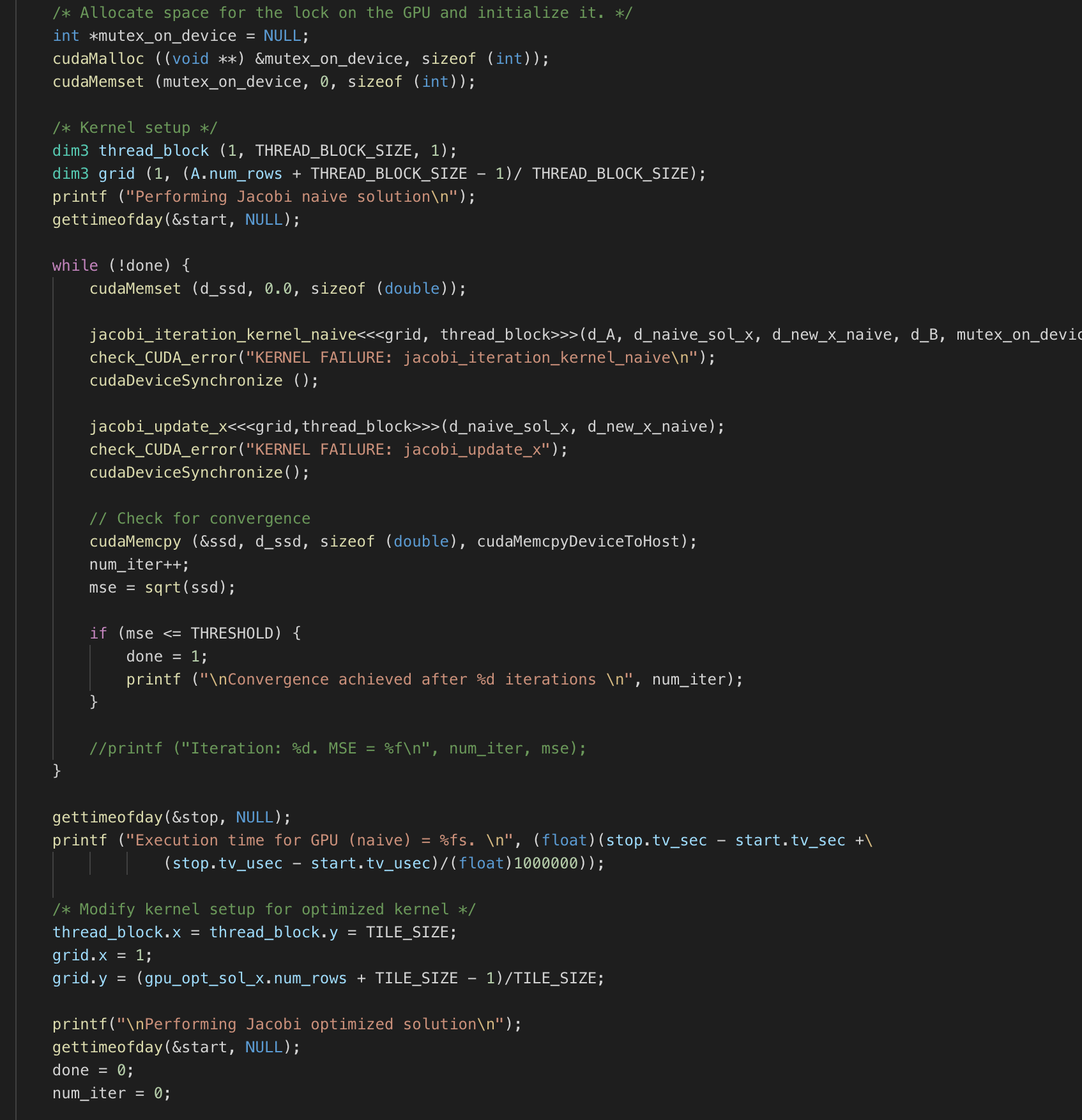
**Prof. Naga Kandasamy**

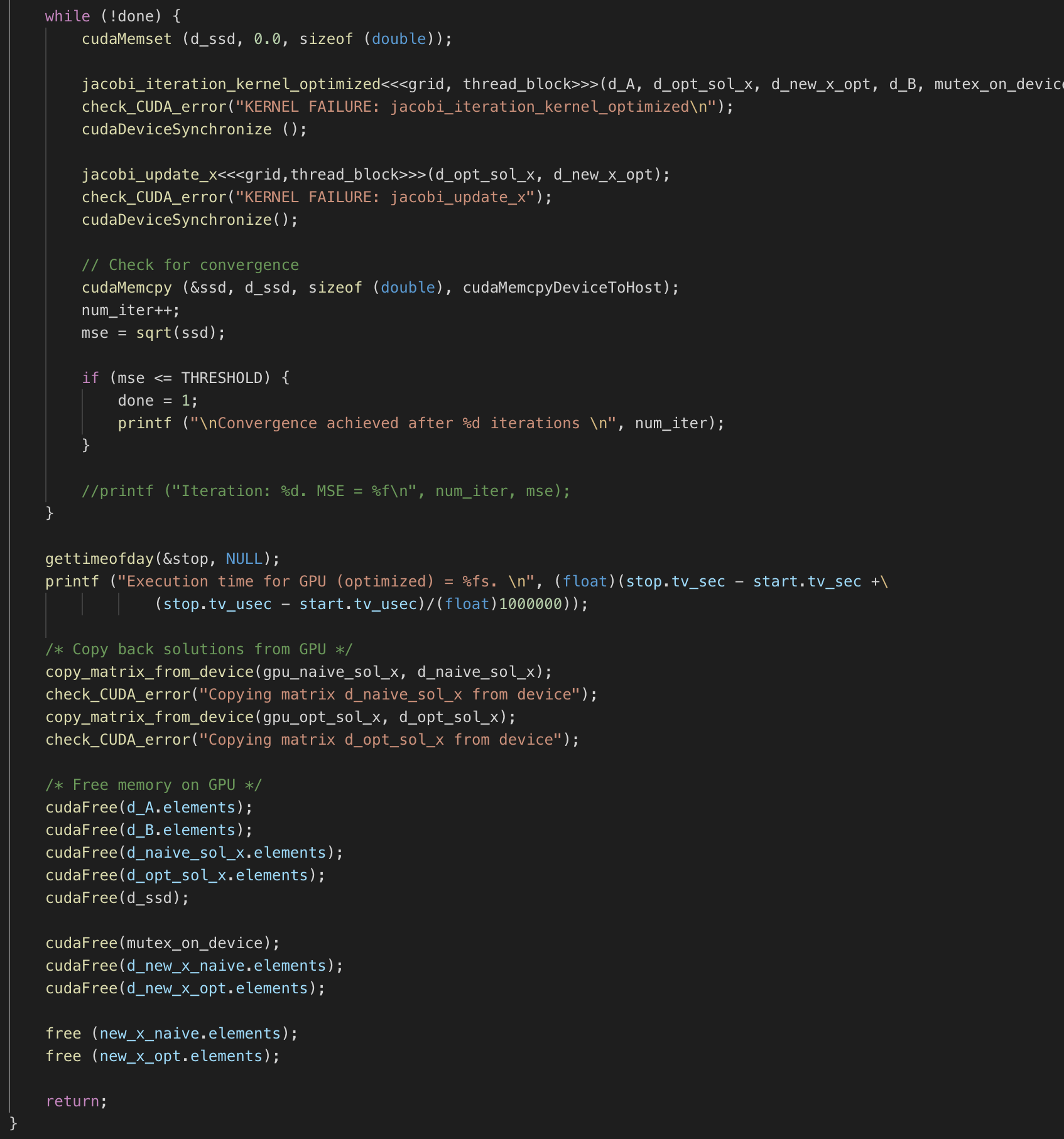
**DATE: 06/5/19**

**GPU Implementation (using CUDA)**

Initialize/free code is as follows:





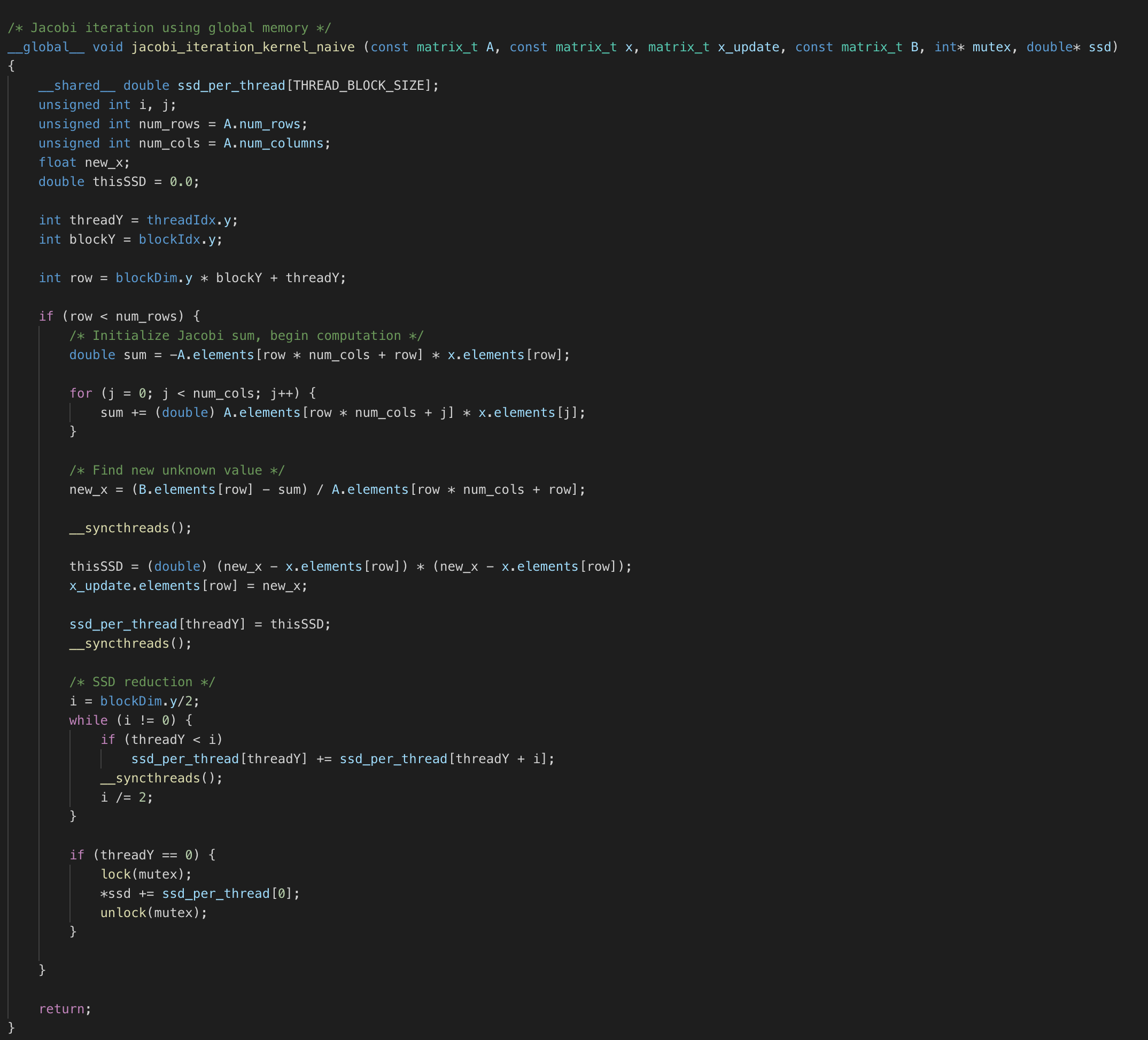


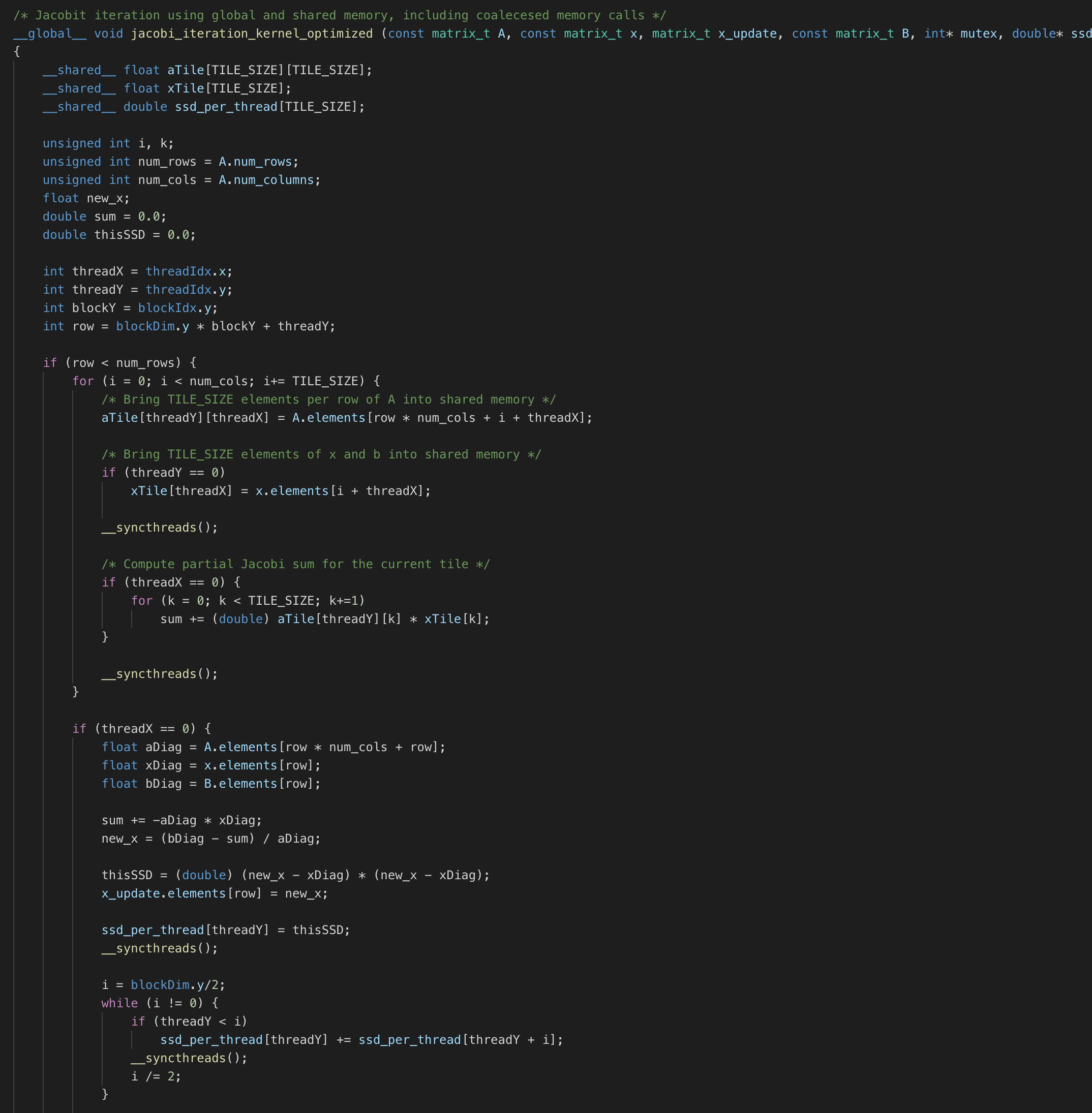
One key difference between the CUDA kernels and the CPU compute\_gold(), is that there is an additional kernel function to update the x solution values. This is mostly due to the fact that grid-level thread syncing doesn’t exist.

For the naïve kernel, the thread block and grid is based off the #define constant THREAD\_BLOCK\_SIZE and the number of rows of matrix A. For the optimized approach, which uses tiling, the configuration is based off the #define constant TILE\_SIZE, and the number of rows in matrix A.

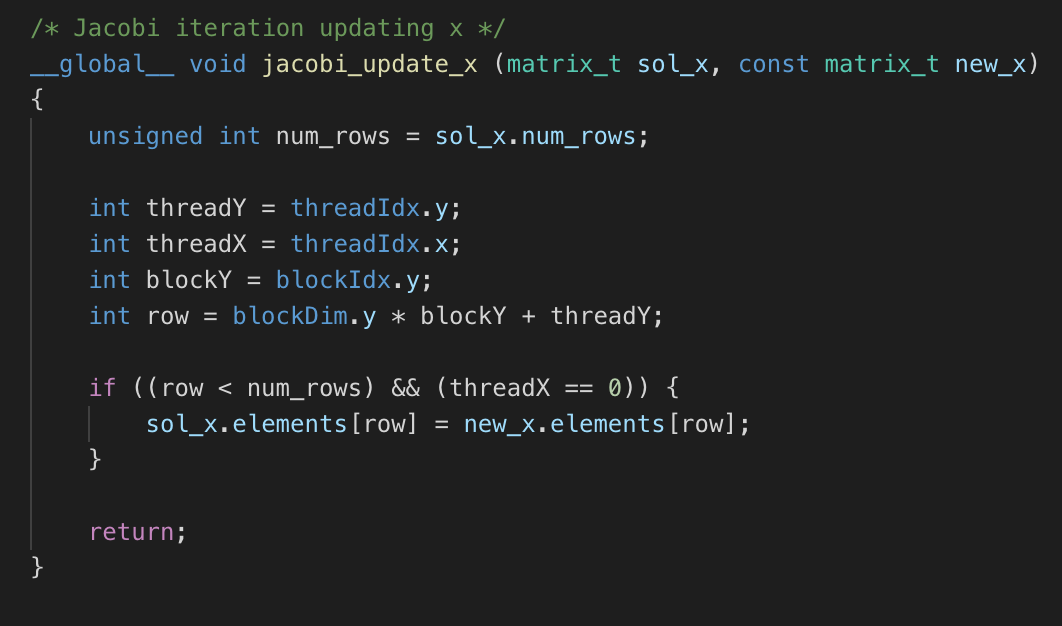
Checking for convergence is also done on the host-side, as the SSD value is transmitted back to the host from the device.

Here is the kernel code:









For the naïve approach, each thread is responsible for a row in the matrix, which are also in a thread block with other rows. To compute the SSD value, reduction is used, and then by using a mutex lock, is added to the SSD pointer.

For the optimized approach, more shared memory is being used, as tiling is implemented. Once the tiles are populated, only half the threads in the block do the summation calculation. This is repeated as the tiles go across their respective rows. Once this part is finished, the new x values are found and the SSD computation occurs, just like the naïve function.

The last kernel function is used to update the x solution values from the other kernel calls.

The speedup table below shows the times of the kernel functions, with some different kernel configurations to show sensitivity. I’m not sure why my naïve function is faster than the optimized function for larger matrix sizes. It could be due to the amount of synchronizing that is occurring with those large matrices.

**Speedup**

**Table 1: Speedup with Jacobi Iterative Method on xunil-05**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Matrix Size** | CPU | GPU naïve (Thread Block Size = 128) | GPU optimized (Tile Size = 16) | GPU naïve (Thread Block Size = 256) | GPU optimized (Tile Size = 8) |
| 512 x 512 | 3.66 s | 1.10 s | 1.00 s | 1.38 s | 1.16 s |
| 1024 x 1024 | 28.38 s | 2.82 s | 7.74 s | 3.89 s | 4.91 s |
| 2048 x 2048 | 239.30 s | 9.85 s | 51.40 s | 12.09 s | 27.27 s |

*Note: xunil-05 has a Nvidia 1080 GTX GPU*