EC527
Lab7 report
Bin Xu
Seyed Reza Sajjadinasab

Part1

```
Length of the array = 50000

Initializing the arrays ...     ... done


GPU time: 0.141216 (msec)

TEST PASSED: All results matched
```

Part2

For part2, we created three versions of SOR, including SOR_block, SOR_strip, SOR_not_interleave. Each patch we set for SOR_block is 16 * 16 block size.
We also add the same CPU function for the host. The CPU version is slower than the GPU versions.
For 2a, we implement SOR_block. In optional, we choose SOR_1D_strip and SOR_not_interleave. But we cannot get the correct answer on SOR_not_interleave. We commented the SOR_not_interleave method out, because it was too slow.
From the below data, we find the CPU time is , GPU time is . GPU is obviously faster than CPU. It takes so long to run for 2k*2k. However for the first block it works and here is the result.

```
calculating results on host: 415543.984221 (msec)

GPU time block: 39496.343750 (msec)

Compare: 0
```

To show the result for the strip we make the array size smaller 256*256.

```
calculating results on host: 5263.735364 (msec)

GPU time block: 2106.545166 (msec)

Compare: 0



GPU time strip: 1918.132324 (msec)

Compare: 0
```

We couldn't print the result for the not interleaved version. It took so long. The 0 value for compare corresponds to the number of errors that existed when comparing the cpu with gpu output. 0 means everything is alright.

Part3

For this part, we used 2 different functions. One using interleaving, and the other without. Here, by interleaving we mean that in each block, the next element of the matrix is assigned to the next thread of the same block. Also, non-interleaving means that the next element of the matrix is calculated by the same thread index of another block. For example, assume that we have 4 threads per block then we have: (Numbers in the same color belongs to the same block)

Interleaved:
1 2 | 1 2 | 1 2 | 1 2 | …
3 4 | 3 4 | 3 4 | 3 4 | …
…

Non-interleaved:
1 1 | 2 2 | 3 3 | 4 4 | …
1 1 | 2 2 | 3 3 | 4 4 | …
…

Here the definition of interleaved is slightly different.

We can do the same thing with strips. So we have 4 different versions. For a row length of 2048 Here is the result of different timing: (The 0 value for compare corresponds to the number of errors that existed when comparing the cpu with gpu output. 0 means everything is alright)

As we can see, the GPU is faster in all scenarios.

```
calculating results on host: 820481.122041 (msec)

GPU time block (interleaved): 1450.714355 (msec)

Compare: 0



GPU time strip (interleaved): 1027.407959 (msec)

Compare: 0



GPU time block (not interleaved): 9365.088867 (msec)

Compare: 0



GPU time strip (not interleaved): 3567.682617 (msec)

Compare: 0
```

In the GPU versions, the interleaved version is a better one, since it is benefitting from the locality (Every threads of a block are calculating their neighbors). In the interleaved versions the strip version is better because it has less co-to-co ratio.

Note: In this experiment we set the TOL to be 5%. Also results are reported on the local machine with a GPU of GeForce GTX 3050 Ti. We also use the iteration in the cpu side to help the convergence.
Each block has 256 threads. In block version it's a square of 16*16 and in strips it's a line of 256 threads.