Spring 2019: Advanced Topics in Numerical Analysis: High Performance Computing Assignment 2

Kaizhe Wang (kw2223)

1. **Finding Memory bugs.**
   Comments in the codes.

2. **Optimizing matrix-matrix multiplication.**
   The processor I used for computing is: `Intel(R) Core(TM) i5-7287U CPU @ 3.30GHz, 2 Cores, 4 Threads, Max Turbo Frequency 3.7GHz`. The block size I used is 32, and I found this block size has better performance than block size 16.
   The time for various matrix size obtained with the blocked version:

Dimension	Time	Gflop/s	GB/s
32	0.122775	16.290187	260.642984
512	0.106976	20.074443	321.191093
992	0.189864	20.566121	329.057932
1472	0.324522	19.656640	314.506239
1952	0.823768	18.057787	288.924587

   The time for various matrix size obtained with the blocked OpenMP version (number of thread is 4):

Dimension	Time	Gflop/s	GB/s
32	1.576696	1.268493	20.295886
512	0.054245	39.588601	633.417612
992	0.099871	39.098096	625.569537
1472	0.159663	39.952977	639.247625
1952	0.384915	38.646004	618.336072

   For size 32, OpenMP needs to initialize multi-threads, so it took longer than the unparalleled version.

3. **Finding OpenMP bugs.**
   Comments in the codes.

4. **OpenMP version of 2D Jacobi/Gauss-Seidel smoothing.**
   The processor I used for computing is: `Intel(R) Xeon(R) CPU E5630 @ 2.53GHz` (CIMS Compute Servers). The iteration number I used is 1000.
   For the Jacobi method, the time for different matrix size and different numbers of threads are shown in the table below:

Matrix Size	1 Thread	2 Threads	4 Threads	8 Threads	16 Threads
100	0.535856	0.274605	0.144738	0.082691	0.073158
200	2.139034	1.077112	0.548631	0.294177	0.239523
500	13.344843	6.686164	3.381802	1.773384	1.391263
1000	53.563724	26.618653	13.676684	7.086718	5.523454

We can see that if the number of threads is less than 8, the parallel version speed up the program significantly, but for 16 threads, there isn't much improvement from 8 threads.
For the Gauss-Seidel method, the time for different matrix size and different numbers of threads are shown in the table below:

Matrix Size	1 Thread	2 Threads	4 Threads	8 Threads	16 Threads
100	0.545459	0.279640	0.152417	0.087271	0.082813
200	2.173655	1.095316	0.561747	0.302996	0.251964
500	13.555463	6.777217	3.466832	1.802307	1.424178
1000	54.663060	27.137006	13.756388	7.184554	5.576750