Financial Computing

1807730

March 2023

1 Task 1: Word Count

1. Listed as "mywc.c". Command: `gcc mywc.c -o mywc; ./mywc [filename]`.
2. Command: `gcc mywc.c -o mywc; ./mywc mywc.c`. Output: `Number of words: 169`
3. Written work is brief and accurate.
4. Code is well commented. Validation:
   (a) `argc == 2`.
   (b) `fopen(argv[1], ‘‘r") != NULL`.

2 Task 2: Quadrature

1. Listed as "quadrature_1.c". Command: "gcc -fopenmp -O9 -lm quadrature_1.c -o quadrature_1; ./quadrature_1".
2. Included as function `u(double x)`.
3. `f(1, 2) = 4.875603`. Listed as "quadrature_3.c". Command: "gcc -fopenmp -O9 -lm quadrature_3.c -o quadrature_3; ./quadrature_3". Output for $N \in \{8, 16, \ldots, 512, 1024\}$:

```
N = 8; f(1., 2.)  = 4.487628; error = 0.000068; Duration = 0.000021s;
N = 16; f(1., 2.)  = 4.487578; error = 0.000018; Duration = 0.000002s;
N = 32; f(1., 2.)  = 4.487565; error = 0.000005; Duration = 0.000004s;
N = 64; f(1., 2.)  = 4.487561; error = 0.000001; Duration = 0.000011s;
N = 128; f(1., 2.)  = 4.487561; error = 0.000001; Duration = 0.000004s;
N = 256; f(1., 2.)  = 4.487560; error = 0.000000; Duration = 0.000025s;
N = 512; f(1., 2.)  = 4.487560; error = 0.000000; Duration = 0.000016s;
N = 1024; f(1., 2.)  = 4.487560; error = 0.000000; Duration= 0.000032s;
```

4. It depends on the type of precision required but a meaningful value for `N` would be `256`. The error at this value is `0`, with regards to an error measured as a `double`.

5. Listed as "quadrature_5.c". Command: "gcc -fopenmp -O9 -lm quadrature_5.c -o quadrature_5; ./quadrature_5".

6. T = Threads:
```
T = 1; N = 1e6; f(1, 2) = 4.487560; error = 0; Duration = 0.041437s;
T = 1; N = 2e6; f(1, 2) = 4.487560; error = 0; Duration = 0.057819s;
T = 1; N = 4e6; f(1, 2) = 4.487560; error = 0; Duration = 0.115049s;
T = 1; N = 8e6; f(1, 2) = 4.487560; error = 0; Duration = 0.233571s;
T = 1; N = 16e6; f(1, 2) = 4.487560; error = 0; Duration = 0.463174s;
T = 1; N = 32e6; f(1, 2) = 4.487560; error = 0; Duration = 0.955284s;
T = 1; N = 64e6; f(1, 2) = 4.487560; error = 0; Duration = 2.046854s;
T = 1; N = 128e6; f(1, 2) = 4.487560; error = 0; Duration = 4.043025s;
T = 1; N = 256e6; f(1, 2) = 4.487560; error = 0; Duration = 7.561919s;
T = 1; N = 512e6; f(1, 2) = 4.487560; error = 0; Duration = 15.204530s;
T = 1; N = 1024e6; f(1, 2) = 4.487560; error = 0; Duration = 30.547538s;
T = 2; N = 1e6; f(1, 2) = 4.487560; error = 0; Duration = 0.033733s;
T = 2; N = 2e6; f(1, 2) = 4.487560; error = 0; Duration = 0.039853s;
T = 2; N = 4e6; f(1, 2) = 4.487560; error = 0; Duration = 0.058274s;
T = 2; N = 8e6; f(1, 2) = 4.487560; error = 0; Duration = 0.119180s;
T = 2; N = 16e6; f(1, 2) = 4.487560; error = 0; Duration = 0.230960s;
T = 2; N = 32e6; f(1, 2) = 4.487560; error = 0; Duration = 0.464811s;
T = 2; N = 64e6; f(1, 2) = 4.487560; error = 0; Duration = 0.933162s;
T = 2; N = 128e6; f(1, 2) = 4.487560; error = 0; Duration = 1.911113s;
T = 2; N = 256e6; f(1, 2) = 4.487560; error = 0; Duration = 3.757755s;
T = 2; N = 512e6; f(1, 2) = 4.487560; error = 0; Duration = 7.915173s;
T = 2; N = 1024e6; f(1, 2) = 4.487560; error = 0; Duration = 15.274461s;
```

7. Written work is brief and accurate.

8. Code is well commented. "quadrature_1.c" validation:

   (a) `scanf(``%lf", x) == 1`.

   (b) `scanf(``%lf", y) == 1`.

   (c) `scanf(``%d", Nx) == 1`.

   (d) `N > 0`.

3 Task 3: Matrix Algebra

1. Listed as "matrix_1.c". Command: `gcc -fopenmp -O9 -lm matrix_1.c -o matrix_1; ./matrix_1`.

2. Listed as "matrix_2.c". Command: `gcc -fopenmp -O9 -lm matrix_2.c -o matrix_2; ./matrix_2`.
```
N = 256; Iterations = 32733; Lambda = 3.062312; Duration = 0.005061
N = 512; Iterations = 32740; Lambda = 3.062318; Duration = 0.020196
N = 1024; Iterations = 32747; Lambda = 3.062319; Duration = 0.084548
```

```
N = 2048; Iterations = 32754; Lambda = 3.062319; Duration = 0.312998
N = 4096; Iterations = 32761; Lambda = 3.062319; Duration = 1.277784
```

3. Based on the various values of N and λ, I suspect the largest eigenvalue for every N is the same and therefore does not change with increasing N. The time taken increases as N increases because the number of iterations increases. This is because there are more numbers being multiplied at each stage of algorithm. I believe the big-O notation is $O(N^2)$ due to the matrix multiplication.

4. It would depend on the value of N. If N is very large, it may be worth parallelizing every for loop, even those of $O(N)$ as the communication time would be negligible compared to the computation time. for all values of N, it is certainly worth parallelizing the double loop for the matrix multiplication since this dominates the execution time.

5. Written work is brief and accurate.

6. Code is well commented. Validation for "matrix_1.c":

   (a) `scanf(``%d", &N) == 1`.

   (b) `N > 0`.