CSIS
Computer Simulation in Science

Lab Course II, WS 22/23
Prof. Dr. F. Knechtli
Dr. Roman Höllwieser

Homework 8
due Dec 14, 2022

Exercise 8.1 : The Trapezoidal Rule with OpenMP

Write an OpenMP program having individual threads compute the areas of individual trapezoids and add them to a shared variable, *e.g.*: `global_result`, avoiding the race condition or critical section using

a) the `critical` directive

```
# pragma omp critical
  global_result += my_result;
```

b) the `atomic` directive

```
# pragma omp atomic
  global_result += my_result;
```

c) a `reduction` clause

```
# pragma omp parallel num_threads(thread_count) \
    reduction(+: global_result)
  global_result += trap(n);
```

and provide scaling results (10 points)

Exercise 8.2 : Estimating π

One way to get a numerical approximation to π is to use many terms in the formula

$$\pi = 4\left[1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots\right] = 4\sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1}. \tag{1}$$

Implement this formula using the OpenMP `parallel for` directive and provide an estimate of π with the maximal number of threads available on stromboli. (10 points)

Exercise 8.3 : Estimating π via unit circle/square area ratio

Why does the program not scale properly with the number of threads? Find the problem and submit a correct version with scaling results!

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#define MAXTHREADS 4
int main(int argc, char * argv[]){
    int nthd_req = atoi(argv[1]);
    int nsubdiv  = atoi(argv[2]);
    omp_set_num_threads(nthd_req);
    double sum[4];
    for(int i=0; i<MAXTHREADS; i++){
        sum[i] = 0.0;
    }
    #pragma omp parallel{
        int tid  = omp_get_thread_num();
        int nthd = omp_get_num_threads();
        double dx=2./((double)(nsubdiv));
        double dx2=dx*dx;
        for(int i= tid*(nsubdiv/nthd); i<(tid+1)*\
            (nsubdiv/nthd); i++){
            double x = -1.0 + dx/2 + dx*i;
            for(int j=0; j<nsubdiv; j++){
                double y = -1.0 + dx/2 + dx*j;
                if(x*x+y*y<1){
                    sum[tid] += dx2;
    }}}}
    double fullsum=0;
    for(int i=0; i<MAXTHREADS; i++){
        fullsum += sum[i];
    }
    printf(”Our estimate of pi is %f\n”, fullsum);
}
```

(10 points)

Remark: From what you hand in, how to verify the correctness of your program should be clear and simple.