CSC630/730, Fall 2025
Assignment #3 CUDA Implementation of the Midpoint Rule for Parallel Numerical Integration

Objectives:

(i) Design a parallel algorithm for numerical integration using the midpoint rule.
(ii) Implement the parallel algorithm using C and CUDA.
(iii) Learn how to compile and run your parallel program on the USM HPC cluster.
(iv) Report and analyze your results.

This assignment includes two parts.

Part 1: Serial Implementation

You will write a serial program using C programming language to calculate the value of the following integral using midpoint rule:

$$\int_0^1 \frac{4}{1+x^2}\,dx$$

The formula of the midpoint rule for a function *f(x)* is given as follows

$$\int_a^b f(x)dx = h[f(\frac{x_0+x_1}{2})+f(\frac{x_1+x_2}{2})+\ldots+f(\frac{x_{n-2}+x_{n-1}}{2})+f(\frac{x_{n-1}+x_n}{2})]$$

where $h=\frac{b-a}{n}$

a is the lower limit of integration,
b is the upper limit of integration,
n is the total number of intervals,
h is the length of each interval.

Part 2: Parallel Implementation

Describe the design of your parallel algorithm and implement a parallel version using **C and CUDA** to solve the same problem. Your submission should include both the serial and parallel code within a single program. A parallel solution for numerical integration using the trapezoidal rule has been demonstrated in class, and example programs are available on Canvas.

In your CUDA implementation, threads should be organized into one-dimensional blocks. The number of blocks is represented by *block_ct*, and the number of threads per block is represented by *threads_per_bk*. The total number of threads is given by: *threads_total = block_ct * threads_per_bk.*

When executing the parallel program, the user will input *n* and *threads_per_bk*. The program should then calculate *block_ct*. For different values of *n* and *threads_per_bk*, display the inputs, integration results and performance metrics on the screen.

Report the integration results and performance metrics, including serial runtime T_s, parallel runtime T_p and speedup S in Table 1. Results should be accurate to six decimal places.

Table 1: Estimated integral and performance metrics

Number of intervals, n	*block_ct*	*thread_per_bk*	Serial integral	Parallel integral	T_s	T_p	S
n = 100		128					
n= 1,000		256					
n = 10,000		256					

Report:

Write a report that includes the following sections:

1. Problem Description and Method
2. Serial Implementation
3. Design and Implementation of the Parallel Algorithm, including thread organization and mapping
4. Results of the Estimated Integral, as presented in Table 1
5. Screenshots of the Compilation and Execution Process for your CUDA program, corresponding to the cases in Table 1
6. Discussion and Conclusions, comparing the integral results and performance between the serial and parallel implementations

Submission:

Upload the source code (properly formatted) and a report (PDF file) to Canvas.

Student Name: _____

Assignment #2: Grading Policy

Grade components	Points	Points earned	Comments
Serial Program	5		
C/CUDA Program	30		
Compilation and Execution	5		
Correct method and results	10		
Report	20		
Total	70		