P. Hadjidoukas, E. Dermatas, E. Gallopoulos

Set 2 - SIMD and CUDA

Issued: December 2, 2024

Question 1: SIMD and WENO5

The fifth-order Weighted Essentially Non-Oscillatory (WENO) scheme is a numerical method for solving hyperbolic partial differential equations, particularly in problems involving shocks and discontinuities. It is an extension of high-order finite difference schemes designed to provide high accuracy in smooth regions of the solution and stability near discontinuities.

The fifth-order WENO scheme is widely used in computational fluid dynamics (CFD) and other fields requiring accurate and robust simulation of problems with shocks, rarefactions, and other sharp features, such as compressible flows and astrophysical simulations. This scheme is a popular choice due to its balance of accuracy and robustness, especially in problems with mixed smooth and discontinuous regions.

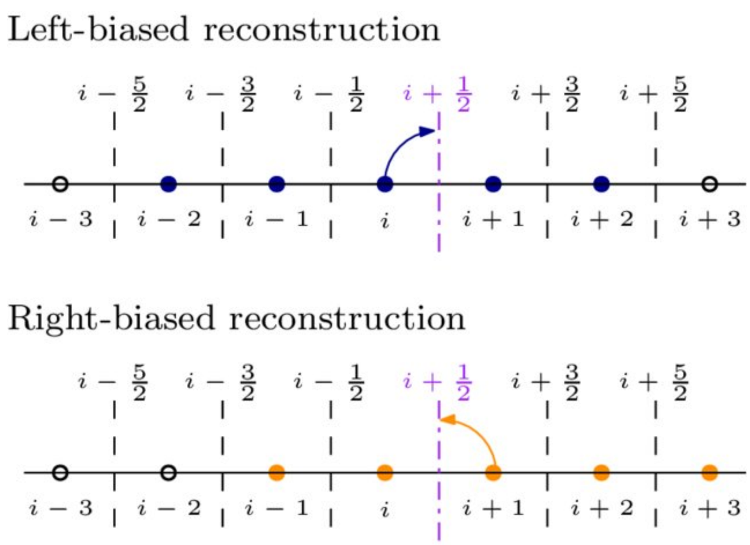

Figure 1: WENO left-biased and right-biased reconstruction stencils

An implementation of the WENO (WENO5) scheme in C is provided in the `task1` folder. The `bench.c` file offers a framework for benchmarking while `weno.c` gives a reference implementation of the WENO5 scheme using its left-biased reconstruction stencil (thus the suffix 'minus' in the function).

a) In this exercise, you should extend the code (Makefile and source code files) to implement the following:

- Enable automatic vectorization for the reference implementation (study the compilers' vectorization report).
- Provide a vectorized implementation of the WENO5 computation using OpenMP.
- Provide a vectorized implementation of the WENO5 computation using SSE and/or AVX intrinsics.

b) Extend the benchmarking procedure and adjust the problem size to perform an experimental evaluation and compare the various implementations. Comment on your results.

Optionally, you can apply any further optimizations to the code.

Question 2: CUDA and Complex Matrix Multiplication

2D complex matrix multiplication involves the multiplication of two matrices where the elements are complex numbers. The standard rules of matrix multiplication apply, in addition to handling complex number arithmetic.

To compute the product of two NxN matrices of the form $(A + Bi)$ and $(C + Di)$, where A, B, C, and D are real NxN matrices, we use the rules of complex number multiplication and matrix operations. The computation follows the following formula:

$$(A + Bi)(C + Di) = (AC - BD) + (AD + BC)i = E + Fi \tag{1}$$

where E and F are the real and imaginary parts of the result.

a) Develop a CUDA application that implements the above-described complex matrix multiplication. The four matrices, A, B, C, and D, are allocated and initialized with random values on the host. The results will be stored in the two matrices, E and F.

b) In your report, describe the main design and optimization decisions you applied to your implementation.

c) Measure and report the performance of your implementation for different values of N. Optionally, compare the performance of your CUDA implementation with an equivalent sequential CPU-based implementation.

Make sure that all your implementations give correct results.