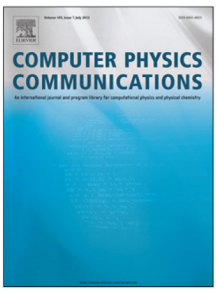
Accepted Manuscript



Methods for compressible fluid simulation on GPUs using high-order finite differences

Johannes Pekkilä, Miikka Väisälä, Maarit Käpylä, Petri Käpylä, Omer Anjum

PII: S0010-4655(17)30098-X
DOI: http://dx.doi.org/10.1016/j.cpc.2017.03.011
Reference: COMPHY 6184

To appear in: *Computer Physics Communications*

Received date : 10 March 2016
Revised date : 6 October 2016
Accepted date : 29 March 2017

Please cite this article as: J. Pekkilä, et al., Methods for compressible fluid simulation on GPUs using high-order finite differences, *Computer Physics Communications* (2017), http://dx.doi.org/10.1016/j.cpc.2017.03.011

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

*Manuscript

ACCEPTED MANUSCRIPT

Methods for compressible fluid simulation on GPUs using high-order finite differences

Johannes Pekkilä[a], Miikka Väisälä[b], Maarit Käpylä[c,a], Petri Käpylä[d,a,c], Omer Anjum[e,a]

[a] *ReSoLVE Centre of Excellence, Department of Computer Science, Aalto University, PO Box 15400, FI-00076 Aalto, Finland*
[b] *Department of Physics, Gustaf Hällströmin katu 2a, PO Box 64, FI-00014 University of Helsinki, Finland*
[c] *Max-Planck-Institut für Sonnensystemforschung, Justus-von-Liebig-Weg 3, D-37077 Göttingen, Germany*
[d] *Leibniz-Institut für Astrophysik Potsdam, An der Sternwarte 16, D-11482 Potsdam, Germany*
[e] *Nokia Solutions and Networks, Finland*

Abstract

We focus on implementing and optimizing a sixth-order finite-difference solver for simulating compressible fluids on a GPU using third-order Runge-Kutta integration. Since *graphics processing units* perform well in data-parallel tasks, this makes them an attractive platform for fluid simulation. However, high-order stencil computation is memory-intensive with respect to both main memory and the caches of the GPU. We present two approaches for simulating compressible fluids using 55-point and 19-point stencils. We seek to reduce the requirements for memory bandwidth and cache size in our methods by using *cache blocking* and decomposing a latency-bound kernel into several bandwidth-bound kernels. Our fastest implementation is bandwidth-bound and **integrates** 343 **million grid points per second** on a Tesla K40t GPU, **achieving a** 3.6× **speedup** over a comparable hydrodynamics solver benchmarked on two Intel Xeon E5-2690v3 processors. Our alternative GPU implementation is latency-bound and **achieves the rate of** 128 **million updates per second**.

Keywords: Computational techniques: fluid dynamics, Finite difference methods in fluid dynamics, Hydrodynamics: astrophysical applications, Computer science and technology
PACS: 47.11.-j, 47.11.Bc, 95.30.Lz, 89.20.Ff

1. Introduction

The number of transistors in a microprocessor has been doubling approximately every two years and as a result, the performance of supercomputers measured in *floating-point operations per second* (FLOPS) has been following a similar increase. However, since increasing the clock frequencies of microprocessors to gain better performance is no longer feasible because of power constraints, this has lead to a change in their architectures from single-core to multi-core.

While modern *central processing units* (CPUs) utilize more cores and wider SIMD units, they are designed to perform well in general tasks where low memory access latency is important. On the other hand, *graphics processing units* (GPUs) are specialized in solving data-parallel problems found in real-time computer graphics and as a result, house more parallel thread processors and use higher-bandwidth memory than CPUs. With the introduction of general-purpose programming frameworks, such as *OpenCL* and *CUDA*, GPUs can now also be programmed to do general purpose tasks using a C-like language instead of using a graphics *application-programming interface* (API), such as *OpenGL*. In addition, APIs such as *OpenACC* can be used to convert existing CPU programs to work on a GPU. For these reasons, GPUs offer an attractive platform for physical simulations which can be solved in a data-parallel fashion.

In this work we concentrate on investigating sixth-order central finite-difference scheme implementations on GPUs, suitable for multiphysics applications. The justification for the use of central differences with explicit time stepping, a configuration which is not ideal concerning its stability properties, comes from the fact that, even though some amount of diffusion is required for stability, they provide very good accuracy and are easy to implement (see, e.g. [1]). In addition, the various types of boundary conditions and grid geometries needed in multiphysics codes such as the Pencil Code[1] are easy to implement with central schemes. Moreover, the problem has the potential to exhibit strong scaling with the number of parallel cores in the optimal case.

There are astrophysical hydro- and magnetohydrodynamic solvers already modified to take advantage of accelerator platforms (i.e. [2], [3], [4]), that **most often** use low-order discretization. **As an example of a higher-order scheme for cosmological hydrodynamics, we refer to [5]. We also note that more theoretical than application-driven work on investigating higher-order stencils on GPU architecture exists in the literature, see e.g. [6].** There are many scientific problems, such as modeling hydromagnetic dynamos, where long integration times are required, either to reach a saturated state (see e.g. [7]), or to exhibit non-stationary phenomena and secular trends (see e.g. [8]). Therefore, it is highly desirable to find efficient

Email addresses: johannes.pekkila@aalto.fi (Johannes Pekkilä), miikka.vaisala@helsinki.fi (Miikka Väisälä)

[1] http://github.com/pencil-code

Download English Version:

https://daneshyari.com/en/article/4964391

Download Persian Version:

https://daneshyari.com/article/4964391

Daneshyari.com