Application of a Kinetic Theory based solver of the Euler Equations using GPU

MATTHEW R. SMITH* (National Centre for High Performance Computing, Hsinchu, Taiwan.
msmith@nchc.org.tw)

FANG-AN KUO (National Centre for High Performance Computing, Hsinchu, Taiwan.
mathppp@nchc.org.tw)

CHAU-YI CHOU (National Centre for High Performance Computing, Hsinchu, Taiwan.
b00cyc00@nchc.org.tw)

JONG-SHINN WU (National Chiao Tung University, Hsinchu, Taiwan.
chongsin@faculty.nctu.edu.tw)

HADLEY M. CAVE (University of Canterbury, New Zealand
hmcave@canterbury.edu.nz)

Keywords: GPU, Euler solver, Kinetic Theory, Finite Volume Method, CFD.

Overview

Presented is a modified form of the Quiet Direct Simulation (QDS) method [1] adapted for application of Graphics Processing Units (GPU) for flux calculation. Fluxes between source and destination cells calculated by QDS are flux-vector split and (on a regular Cartesian grid) a function of the source cell alone. The resulting advantage is the rapid calculation of fluxes between cells without the prior exchange of information between them, allowing highly efficient calculation using GPU. Various flow problems have been solved and consistent speed-ups of over 35 times (when compared to an equivalent single CPU code) are reported.

Introduction

The use of Graphics Processing Units (GPU's) to assist in the solution of various engineering problems is hardly recent [2,3]. The solution to the Euler Equations is an example of a problem which still maintains relevance in modern engineering problems. This hyperbolic set of partial differential equations possesses analytical solutions in the rarest (and often least useful) conditions and so a significant amount of effort has been spent on their numerical solution. The increase in application of Computational Fluid Dynamics (CFD) over the recent decades has lead to a large number of mathematical and physically based numerical solutions to the Euler Equations [2,3].

Numerous numerical methods have been used in conjunction with GPU technology to solve the Euler equations and have all demonstrated, to varying extent, considerable speed up when compared to existing single CPU codes. Elsen *et.al.* [3] applied a vertex-based finite difference method (with a multi-grid scheme) to the solution of hypersonic flows. However, possibly the most common of these numerical methods is the Finite Volume Method (FVM) [3,4]. In a majority of cases the fluxes are calculated using flux-difference splitting which requires knowledge of conditions on both sides of a cell interface, used together simultaneously, to calculate a net flux across a surface [5]. Such solvers work by calculating the conditions normal to cell interfaces and calculating a series of one- dimensional fluxes across each. This concept is known as direction decoupling and has been shown to result in errors when the flow is not aligned with the computational grid [5].

An alternative are flux-vector split methods, where fluxes are linearly separated into components from both sides of a cell interface and can be calculated separately. These methods are infamous for their typical excessive numerical dissipation. While there are many mathematically split methods available, various kinetic theory based schemes have emerged. Such schemes base fluxes on the mathematical interpretation of phenomenological models. Possibly the most well-known kinetic-theory based finite volume solution method is Pullin's Equilibrium Flux Method [6]. The fluxes of EFM were derived by taking moments of the Maxwell-Boltzmann equilibrium distribution function at the cell interfaces. The resulting flux required the evaluation of two moments – one from each side of the interface – with the net flux across the surface the difference between each. The validity of the method lies in the assumption of flow being divided into a collision phase and a free molecular flight phase. During free m molecular flight, no forces are placed upon fluxing particles as they move from their source region to their destination. These fluxes are calculated based on the conditions normal to the interface; extensions to two dimensional flows were still performed using a series of one- dimensional fluxes.

A more general form of EFM was developed by Smith *et al.* [5] taking into account the true direction nature of the equilibrium fluxes. This method, called TDEFM (True Direction Equilibrium Flux Method) provided the analytical solution to gaseous motion from one cell to an arbitrary destination cell, regardless of whether or not these cells share an adjacent interface. The primary disadvantage to this method was the large computational expense associated with the evaluation of multiple error and exponential functions. As an alternative, the Quiet Direct Simulation (QDS) method has been developed in its current form by Smith *et al.* [1] and is based on the particle-based QDSMC method of Albright *et al.* as a method for simulating plasmas and for Eulerian flow [7,8]. The fluxes obtained by QDS are approximations of the TDEFM fluxes which avoid the evaluation of any expensive mathematical functions while retaining the true direction quality of TDEFM.

Here, the QDS algorithm is slightly modified and applied to employ the Graphics Processing Units (GPU's) of a typical video card using the CUDA library. The nature of QDS makes it ideal for such calculation and results demonstrate significant speedup when compared to a similar (non-GPU) code. Presented here is a description of the QDS algorithm, simulation and code validation using several standard benchmark problems. Finally, details of the computational expense associated with each code are presented.

Quiet Direct Simulation Algorithm

The underlying fundamentals behind QDS flux calculation remain constant regardless of whether or not the solver employs the local Graphics Processing Units. The QDS algorithm consists of three basic steps:

1. Generation of a small number of representative particles N (typically 3-4 per coordinate direction) which are used to carry fluxes of mass, momentum and energy between cells. The amount of mass and velocity of these representative particles are drawn from the Maxwellian distribution of velocities by approximating this distribution by the weights and abscissas of a Gauss-Hermite quadrature. For a spatially first order accurate simulation, each particles masses, velocities and internal energies are [1]:

$$m_{ij} = \frac{\rho_i \Delta x w_j}{\sqrt{\pi}}$$

$$v_{ij} = u_i + \sqrt{2\sigma_{vi}^2} q_j$$

$$\varepsilon_{ij} = \frac{(\xi - \Omega)\sigma_{vi}^2}{2}$$

   where Δx is the uniform grid size, ζ is the total number of degrees of freedom and Ω is the number of simulated translation degrees of freedom (for example, one dimensional simulations use $\Omega = 1$). The values of w_j and q_j are the weights and abscissas of the Guass-Hermite quadrature [9]. For $N = 3$, these are approximately $w_j = \{1.18163, 0.2954, 0.2954\}$ with corresponding abscissas of $q_j = \{0, 1.2247, -1.2247\}$. The quantities of density (ρ_i), velocity (u_i), velocity variance (σ^2_{vi}) and energy (E_i) are cell properties.
2. Calculation of fluxes between cells over the computational time step Δt. During this time, fluxes are calculating assuming free molecular flight in the same way as conventional kinetic theory based schemes. When a regular, Cartesian grid is employed, no knowledge is required of the neighbouring cell locations or properties – these fluxes are a function of the source cell alone.
3. Exchange fluxes of mass, momentum and energy between cells. Calculate each cells equilibrium macroscopic properties (i.e. density, temperature, bulk velocity) using the updated values.

A flowchart describing the calculation procedure is provided in Figure 1. The main strengths of the QDS algorithm are (i) the entire simulation procedure (minus initialisation and post-processing) are performed on the GPU device, and (ii) the calculation of fluxes from each source cell can be easily performed on separate threads. During the flux calculation procedure, no communication is required between cells. This communication is required only when the fluxes are exchanged between cells. The current implementation employs a second order spatial accuracy, meaning that neighbouring cell information is required for the calculation of the local gradients in each cell used for higher order flux calculation, the details of which can be found in [1].

Validation and Performance

1D Shock Tube Test

A standard test for the compressible Euler Equations is Sod's 1D shock tube. The Riemann analytical continuum solution for a shock tube allows the properties of the flow structure, including the shock propagation velocity W, the contact surface velocity u_P along with the pressure, temperature and density, to be determined at any given time [10]. The simulations were conducted using an ideal monatomic gas. The end walls were simulated as reflective walls. The initial conditions in the high pressure and low pressure ends of the shock tube

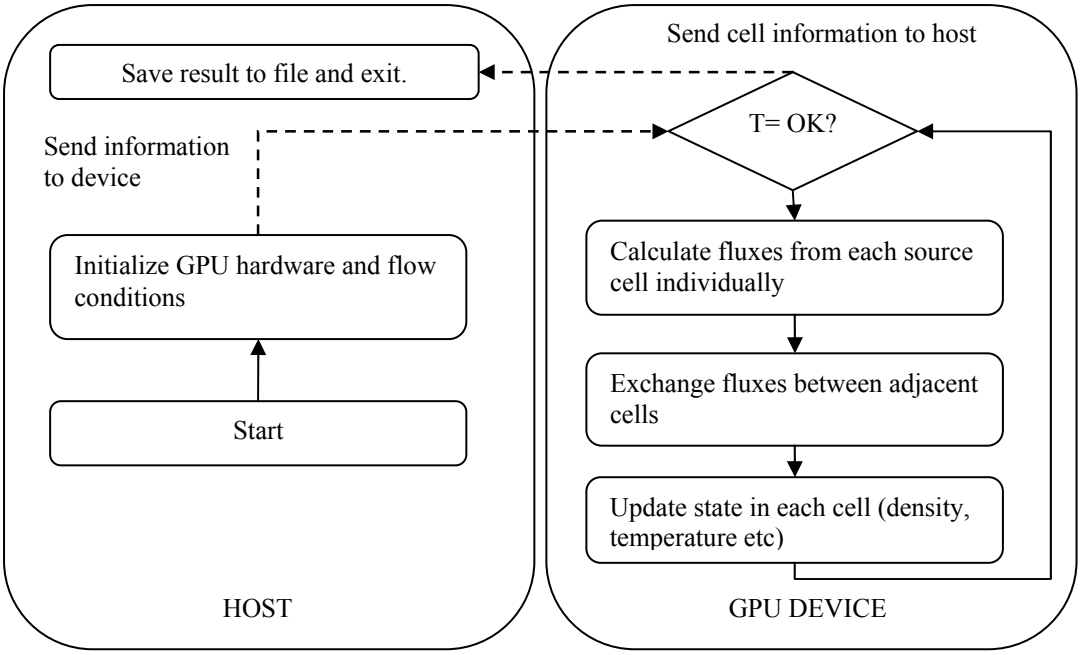

Figure 1: Basic flow diagram showing application of QDS using GPU. The entire calculation procedure occurs on the GPU device with communication between the host and device only occurring prior to and after successful completion of the simulation.

are $p_H = 10p_L$ and the temperatures at both ends of the tube are the same. At t = 0 the infinitely thin diaphragm separating the two gases at $x = 0.5L$ is removed. The resulting propagating shock wave ($M_S = 1.55$) travels from the high pressure region into the low pressure region. The simulation is performed on both single CPU and employing the GPU hardware specified in Table 1. The results obtained are identical in each case and both are in sound agreement with the analytical solution, as shown in Figure 2. The CPU time required by each code is presented in Table 2. As the number of cells increases, disadvantages associated with device initialisation diminish and the GPU capable code quickly demonstrates speed-ups of over 30 times when compared to the single CPU code.

2D/3D Blast Wave Simulation in Urban Environment

Following the previous application of TDEFM to the simulation of blast waves in urban environments [11] we apply the QDS solver to two dimensional (and later, three dimensional) simulations to the simulation of shocked gas flow in urban environments. Some sample results are shown in Figure 3 with computational times and speed-ups demonstrated in Table 3. As expected, speed-ups continue to increase with problem-size. The times shown are for two dimensional simulations due to the difficulty of performing large 3D simulations on single CPU systems. Two dimensional results (not shown) are in agreement with existing TDEFM and direction-coupled EFM simulations.

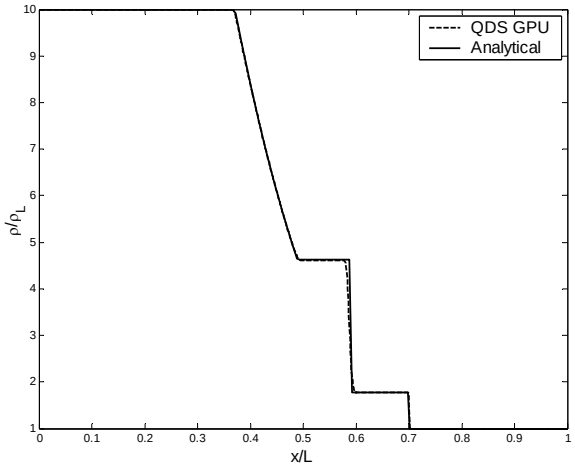

Figure 2: Normalised density from both the analytical solution [ref] and 2nd order accurate QDS at flow-time $T(RT)^{0.5}/L = 0.2$ using 1000 cells. The initial pressure ratio across the diaphragm placed at $x = 0.5L$ was $P_H/P_L = 10.0$ with a uniform temperature throughout. The gas is assumed ideal and monatomic.

Hardware	Details
CPU	Intel Xeon quad-core X5472, 3.0 GHz clock, L2 cache = 12MB.
GPU Hardware	NVidia Tesla S1070 GPU Computing Server (4 x Nvidia Tesla T10 GPU's each with 1.44 GHz clock, 4GB DDR3 Ram, 240 cores) Capable of single or double precision.

Table 1: Details of computer hardware used to compile and run QDS simulations. The operating system used was openSUSE 10.2.

Code	**CPU time (milliseconds) vs. Number of Cells**						
	256	**512**	**1024**	**10000**	**20000**	**40000**	**50000**
CPU	12.8	51.2	203.2	24167	110519	467783	742792
CPU+T10 (GPU)	14.2	29.2	60.1	1197	3715	13063	20125
Speedup	**0.9x**	**~1.7x**	**~3.4x**	**~20x**	**~29x**	**~35x**	**~37x**

Table 2: Details of CPU times required by both single CPU and GPU capable QDS codes using the computer hardware specified in Table 1. The specified CPU times are the total run times and include the GPU hardware initialisation times and time required to write results to local hard drive.

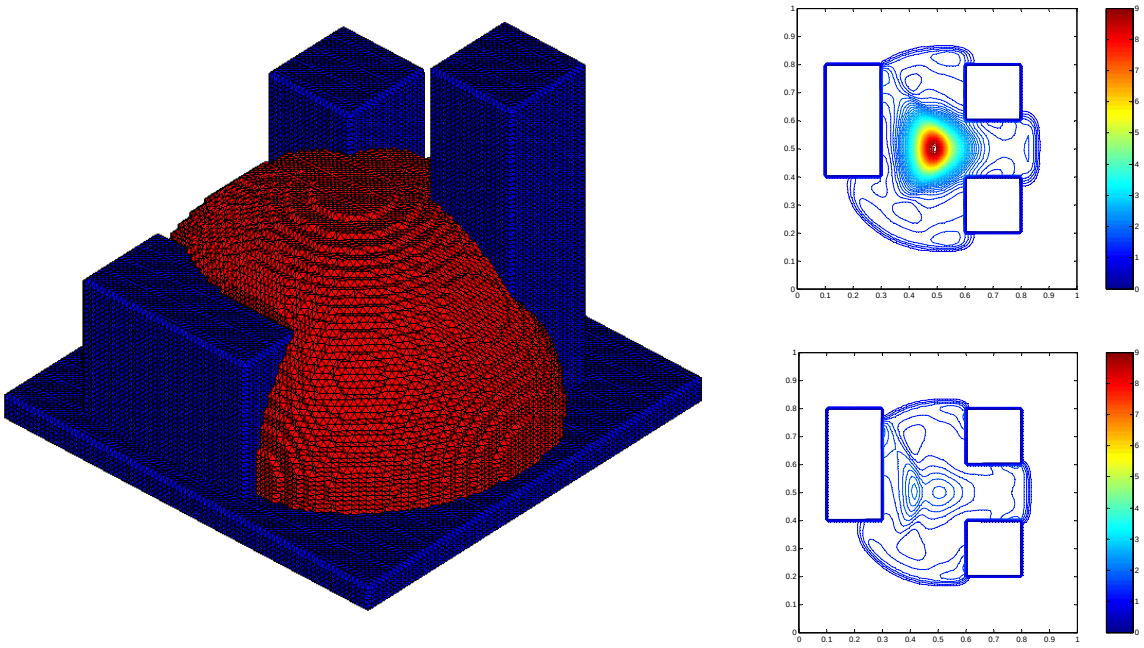

Figure 3: 3D Blast wave simulation in an urban environment using 2nd order accurate QDS at flow-time $T(RT)^{0.5}/L = 0.1$ using 1000000 cells. The "bomb" is modelled using a high temperature region with $T_B/To = 100$ in the region bounded by ($0.45L < x < 0.55L$), ($0.45H<x<55H$). (Left) 3D rendered image of the buildings (in blue) and the location of the propagating shock wave front (in red), (Right, Top) Contours of temperature at ground level, and (Right, Bottom) Contours of temperatures at 0.3H above ground level.

Code	CPU time (seconds) vs. Number of Cells			
	128x128	**256x256**	**512x512**	**1024x1024**
CPU	10.3	88.5	718.3	5970
CPU+T10 (GPU)	0.3	2.2	16.3	124
Speedup	**~34x**	**~40x**	**~45x**	**~48x**

Table 3: Details of CPU times required by both single CPU and GPU capable QDS codes using the computer hardware specified in Table 1 for 2D simulation of blast waves in urban environments. The specified CPU times are the total run times and include the GPU hardware initialisation times and time required to write results to local hard drive.

Conclusion

Presented is the Quiet Direct Simulation (QDS) method slightly modified for and applied to calculation using Graphics Processing Units (GPU's). The simplicity of the QDS algorithm, which requires no evaluation of complex functions (only addition, subtraction, multiplication and division are used), makes the code not only very fast but also suitable for calculation on GPUs. Presented are results for a simple one dimensional benchmark test calculated with codes using both single CPU and multiple CPU (GPU) implementations. The presented results demonstrate that QDS, when combined with GPU computation technology, offers a significant speed up (> 30 times) when compared to conventional, single CPU simulations. This allows the future possibility of very large scale simulations to be run on relatively small (and cheap) equipment without the requirement for traditional supercomputing clusters.

Acknowledgements

The computing facilities and financial support provided by the National Centre for High Performance Computing (NCHC) in HsinChu, Taiwan, is greatly appreciated. In addition, the financial and academic support provided by National Chiao Tung University (NCTU, Taiwan) is also appreciated.

References

1. M.R. Smith, H.M. Cave, Y.-S. Chen, M.C. Jermy and J.-S. Wu, An Improved Quiet Direct Simulation Method for Eulerian Fluids Using a Second-Order Scheme, *J. Comput. Phys.* 228, 2213-2224 (2009).
2. T. Brandvik and G. Pullan, Acceleration of a 3D Euler Solver using commodity graphics hardware, in *46th AIAA Aerospace Sciences Meeting and Exhibit* (2008).
3. E. Elsen, P. LeGresley and E. Darve, Large calculation of the flow over a hypersonic vehicle using a GPU, *J. Comput. Phys.* 10148-10161 (2008).
4. T.R. Hagen, J.M. and J.R. Natvig, Solving the equation equations on Graphics Processing Units, in *International Conference on Computing Science,* (4), 220-227, (2006).
5. M.R. Smith, M.N. Macrossan and M.M. Abdel-Jawad, Effects of direction decoupling in flux calculation in finite volume solvers, *J. Comput. Phys.* 227(8), 4142-4161 (2008).
6. D.I. Pullin, Direct simulation methods for compressible ideal gas flow, *J. Comput. Phys.* 34, 231-144 (1980).
7. B.J. Albright, W. Daughton, D.S. Lemons, D. Winske, and M.E. Jones, Quiet direct simulation of plasmas, *Phys. Plasma* 9(5), 1898-1904 (2002).
8. B. J. Albright, D.S. Lemons, M.E. Jones, and D. Winske, Quiet direct simulation of Eulerian fluids, *Physical Review E* 65, 1-4 (2002).
9. W.H. Beyer, CRC Standard Mathematical Tables, (CRC Press, Boca Raton (FL), 1987).
10. E. F. Toro, Riemann solvers and numerical methods for fluid dynamics: A practical introduction, Springer-Verlag Berlin (1999).
11. M.R. Smith, H.M. Cave, J.-S. Wu and A. Ferguson, Simulation of debris formation and movement resulting from a blast wave in an urban environment, in *15th National Taiwan CFD Conference (Best Paper Award), Kaohsiung, Taiwan,* (2008).