Accelerated composite distribution function methods for computational fluid dynamics using GPU

Matthew R. Smith[1]
National Cheng-Kung University
No. 1, University Road, Tainan, Taiwan.
E-mail: `msmith@mail.ncku.edu.tw`

Yen-Chih Chen
National Cheng-Kung University
No. 1, University Road, Tainan, Taiwan.
E-mail: `johnchen83@yahoo.com.tw`

The Kinetic Theory of Gases has long been established as a useful tool for the solution of modern Computational Fluid Dynamics (CFD) problems. Together with the Finite Volume Method, such approaches have been popular in CFD for over 30 years, with techniques such as the Equilibrium Flux Method (EFM) or Kinetic Flux Vector Splitting (KFVS), Equilibrium Interface Method (EIM) together with more recent developments. One of the disadvantages to using such an approach are the expensive exponential ($exp(-x^2)$) and error function ($erf(x)$)evaluations often associated with the moments taken around the distribution functions for the computation of interface fluxes. One common approach for avoiding such expenses is to employ discrete velocities in the flux calculation, taking moments around these rather than a continuous distribution function. In this talk we will discuss how we can approximate the governing particle velocity distribution function with a series of Composite Distribution Functions (CDF's) - made of more than one distribution function – to simplify the moment equations. The resulting expressions are then applied to multi-dimensional computation using Graphics Processing Units (GPU's), to which the application is well suited due to the simplicity of the flux expressions and locality of the schemes. Very high levels of speedup are demonstrated using C2075 (Fermi) and newer Kepler GPU architectures when compared to modern Xeon E5 processing cores.

2013 International Workshop on Computational Science and Engineering (IWCSE 2013)
October 14-17, 2013
Taipei, Taiwan

[1] Speaker

1. Introduction

The Finite Volume Method (FVM) has risen in popularity to become one of the cornerstone methods in Computational Fluid Dynamics (CFD). The goal of the Finite Volume Method is the numerical solution of a set of governing Partial Differential Equations (PDE) in integral form (without source terms):

$$\frac{d}{dt}\int UdV + \int FdS = 0 \tag{1}$$

where U is a vector of conserved quantities and F is a vector of the fluxes of conserved quantities. Using the divergence theorem, it is possible to use differencing in time to obtain an expression for the update of the average value of the conserved quantity after a discrete time:

$$\bar{U}^{k+1} = \bar{U}^{k} - \frac{\Delta t}{V}\sum_{i=1}^{N} F_i\,\hat{n}\cdot A \tag{2}$$

where F_i is the flux across surface i of the cell volume being examined. Hence, the problem reduces to the evaluation of the flux F across cell surfaces. There are many ways of computing this – however, the approach can generally be divided into two methods: (i) Flux Difference splitting, and (ii) Flux Vector Splitting. This work will focus on the latter approach, with states that the net flux across any surface seperating two states can be broken down into some linear contribution from either side:

$$F_i = F_L^+ + F_R^- \tag{3}$$

where F_L^+ is the contribition from the left hand side of the interface (moving right) and F_R^- is the contribution from the right hand side (moving left). Such fluxes are often overly dissipative and cannot capture many of the waves resulting from the interaction of the left and right hand states. However, these fluxes possess a unique advantage over flux difference methods because each contribution from the left and right hand side can be computed independently of each other, resulting in a highly local scheme.

Many modern engineering applications are multi-scaled and require high resolution (i.e. a large number of finite volumes) in order to achieve a suitable solution. Hence, in many instances, parallel computing is required to obtain a solution in a reasonable turnaround time. While conventional computing systems are quite expensive, in both terms of capital and power related expenses, the use of Graphics Processing Units (GPU's) for general scientific computing has provided a cheaper alternative. Through the use of GPU devices, we can achieve a compute density and power efficiency which is not easily obtained using conventional architectures. However, since the GPU device continues to focus on rendering and drawing related operations, which are SIMD (Single Instructions on Multiple Data elements) centered, care must be taken when implementing any scientific algorithm on the device. For this reason, algorithms which are highly local (i.e. the flux splitting formulations presented here) perform very well on GPU devices.

2. Kinetic Vector Splitting

2.1 Background

The approaches used to determine the formulation of the split fluxes described above are varied – several mathematical models, integral-balanced approaches [1] and kinetic-theory based [2] formulations have been investigated. This work will focus on the latter approach – which is to obtain the fluxes of conserved quantities by taking moments around a governing molecular velocity probability distribution function. In one dimensional form, these fluxes can be described by the integral expressions:

$$F^{+} = \int_{0}^{\infty} f(v_n) v_n U(v_n)\, dv_n \qquad F^{-} = \int_{-\infty}^{0} f(v_n) v_n U(v_n)\, dv_n \tag{4}$$

where v_n is the particle velocity normal to the interface and f is our governing molecular velocity probability distribution function. In the case where our gas is assumed to be in thermal equilibrium, our velocity distribution function is the Maxwell-Boltzman distribution, given in one dimensional form as:

$$f(v_n) = \frac{1}{\sqrt{2RT\pi}} \exp\left(\frac{-(v_n - \overline{v_n})^2}{2RT}\right) \tag{5}$$

This is the core of both Pullin's EFM [2] and Mandal et al.'s KFVS [3]. The resulting evaulation of the integrals in Equations 4 results in a series of expressions involving the error function (erf) and the exponential function (exp). The evaluation of these terms is non trivial, computationally intensive and is often performed using advanced series expansions. There are several works which have been performed examining non-equilibrium distribution functions – as required for real gas flows governed by the Navier-Stokes equations – however, this research will focus on the limit of thermal equilibrium.

2.2 Composite Distribution Functions for Flux Splitting

The concept behind the proposed schemes is that any distribution function can be approximated through the sum of a large number of simpler distribution functions:

$$f(v) \approx \sum_{i=1}^{N} w_i f^*(a_i) \tag{6}$$

where *w* is a weighting function and a is a property of the new distribution function and is related to the particle velocity v. Our "simpler" distribution function f^* can be chosen almost arbitrarily – however, there are several restrictions which follow directly from the conservation of macroscopic properties. The first requirement is that our sum of distribution functions equates to a mathematically correct statement – that is, the chance that a particle has a velocity between negative infinity and positive infinity must be unity:

$$\int_{-\infty}^{\infty} \sum_{i=1}^{N} w_i f^*(a_i)\, dv = 1 \tag{7}$$

This statement is equivilent to a conservation of mass statement for our approximations. The second restriction is equivilent to the conservation of momentum – the average particle velocity for both the original distribution function and the sum of weighted simple distribution functions should be identical:

$$\int_{-\infty}^{\infty} \sum_{i=1}^{N} v \cdot w_i f^*(a_i)\, dv = \bar{v} \tag{8}$$

It is quite trivial to show that Equation 8 leads to the same requirements as Equation 7, as will be discussed in up-coming sections. The final restriction is that the total energy of particles is the same for both the approximated sum of distribution functions and our original. According to the theorem of equipitartion of energy, the total thermal energy (not accounting bulk kinetic energy) can be calculated as:

$$\int_{-\infty}^{\infty} \sum_{i=1}^{N} (v-\bar{v})^2 \cdot w_i f^*(a_i)\, dv = RT \tag{9}$$

Regardless of our choice of *N* or f^*, we must control our choices for the values of *w* and *a* in order to ensure that the restrictions listed in Equations 7-9 are enforced.

2.2.1 Multiple Uniform Distribution Functions

The simplest place to begin when searching for an ideal simple particle velocity probability distribution function is the uniform distribution as given by the equation:

$$f^*(v) = \begin{cases} \frac{1}{2a}, & v > \bar{v} - a \; and \; v < \bar{v} + a \\ 0, & otherwise \end{cases} \tag{10}$$

where *a* is effectivly a thermal velocity and acts as the bounds for our uniform distribution function. In this instance, the restrictions for the choice of *a* and *w* are easily shown to be:

$$\int_{-\infty}^{\infty} \sum_{i=1}^{N} w_i f^*(a_i)\, dv = \sum_{i=1}^{N} w_i = 1 \tag{11}$$

$$\int_{-\infty}^{\infty} \sum_{i=1}^{N} (v-\bar{v})^2 \cdot w_i f^*(a_i)\, dv = \sum_{i=1}^{N} \int_{\bar{v}-a}^{\bar{v}+a} (v-\bar{v})^2 w_i f^*(a)\, dv = RT \tag{12}$$

The expression in Equation 12 can be evaluated to show that:

$$\sum_{i=1}^{N} \frac{a_i^2 w_i}{3} = RT \tag{13}$$

These rules must be satisfied when choosing a and w if the schemes are to be equivilent to the result obtained by EFM or KFVS. There are an infinite number of choices which satisfy these criteria, however – hence error analysis can be used to determine an optimal choice. These values depend heavily on the number of approximating distribution functions to use, which is also a variable which can be changed by the user. In the case where 2 approximating uniform distribution functions are used, sample values of $w = [1/6, 5/6]$ and $a = [3/2(3)^{1/2}, 3/2]s^{1/2}$ can be used. The forward split fluxes – written here for reference only – are:

$$F_i^+ = w_i \begin{bmatrix} \rho\left(\frac{a_i}{4} + \frac{\bar{v}}{2} + \frac{\bar{v}^2}{a_i}\right) \\ \rho\left(\frac{(a_i+\bar{v})^3}{6a_i}\right) \\ \rho\left(\frac{(a_i+\bar{v})^2(4E_{in}+(a_i+\bar{v})^2)}{16a_i}\right) \end{bmatrix} \tag{14}$$

This is (in essence) a surface split equivilent of the volumetric fluxes computed by Ferguson et al. [4] disregarding volumetric effects in the flux computation. These fluxes are first order accurate in both time – higher order implementation may be performed by computing gradients of split fluxes and performing reconstructions at cell interfaces.

2.2.2 Multiple Linear (Triangular) Distribution Functions

The rate at which multiple uniform distribution functions approach the Maxwell-Boltzmann distribution is limited – this can be improved by increasing the order of the approximating probability distribution function. The logical progression is a symmetrical form which is a linear function of velocity in the form:

$$f(v) = \begin{cases} -\frac{b_i}{a_i}(v-\bar{v}) + b_i,\ \bar{v} \le v \le \bar{v} + a_i \\ \frac{b_i}{a_i}(v-\bar{v}) + b_i,\ \bar{v} - a_i \le v \le \bar{v} \\ 0, elsewhere \end{cases} \tag{15}$$

The same restrictions described in Equations 7-9 are still required to make the approximation valid. The weighting restriction described in Equation 11 holds regardless of the choice of approximating distribution function so long as it is a mathematically valid one. Evaluation of the energy conservation restriction integral provided in Equation 9 leads to the restriction:

$$\sum_{i=1}^{N} \frac{a_i^2 w_i}{6} = RT \tag{16}$$

The forward fluxes based on the linear distribution shown above in Equation 15 is written for reference below:

$$F_i^+ = w_i \begin{bmatrix} \rho\left(-\frac{1}{6a_i^2}(\bar{v}-a_i)^3 + \bar{v}\right) \\ \rho\left(-\frac{1}{12a_i^2}(\bar{v}-a_i)^4 + \bar{v}^2 + s^2\right) \\ \rho e_{in}\left(-\frac{1}{6a_i^2}(\bar{v}-a_i)^3 + \bar{v}\right) + \frac{1}{2}\rho\left(-\frac{1}{20a_i^2}(\bar{v}-a_i)^5 + \bar{v}^3 + 3\bar{v}s^2\right) \end{bmatrix} \tag{17}$$

In the case where 2 approximating uniform distribution functions are used, again a large number of possible options are available for selecting the weighting and thermal bound values. For example, sample values of $w = [1/6,\ 5/6]$ and $a = [3/2(3)^{1/2},\ 3/2]s^{1/2}$ can be used for simulation. For the sake of reference, we shall refer to the use of a linearly varying distribution function as the basis for flux computation as the Triangular Equilibrium Distribution Method (TEFM).

2.3 Error Analysis

Predicting the error in the use of a sum of approximating distribution functions can be performed in a large number of ways. As a starting point, we can make several assumptions: (i) that our composite distribution employs two triangular distributions f_1^* and f_2^* with weights w_1 and w_2 respectively, and (ii) we compute the error over the entire distribution based on the integral the difference squared. This equation can be represented by the expressions:

$$E = \int_0^{a_1}\left(f(c) - w_1 f_1^*(c) - w_2 f_2^*(c)\right)^2 dc + \int_{a_1}^{a_2}\left(f(c) - w_2 f_2^*(c)\right)^2 dc + \int_{a_2}^{\infty} f^2(c)dc \tag{18}$$

$$E = \frac{1}{4\sqrt{\pi}s} + \frac{w_1^2}{3a_1} + \frac{w_2^2}{3a_2} - \frac{2w_1 s}{\sqrt{2\pi}a_1^2}\left(exp\left(-\frac{a_1^2}{2s^2}\right) - 1\right) - \frac{w_1}{a_1} erf\left(\frac{a_1}{\sqrt{2}s}\right) \tag{19}$$
$$- \frac{2w_2 s}{\sqrt{2\pi}a_2^2}\left(exp\left(-\frac{a_2^2}{2s^2}\right) - 1\right) - \frac{w_2}{a_2} erf\left(\frac{a_2}{\sqrt{2}s}\right) + \frac{w_1 w_2}{a_2}\left(1 - \frac{a_1}{3a_2}\right)$$

These expressions may be used to evaulate the likeness of our composite distribution function to our original. For the two composite functions shown in Figure 1, the error values are described in Table 1. For the weights and thermal bounds shown in the previous section, the error integral evaluates to approximately 4.3e-4 s/m. Through inspection of the error function above, it can be seen a local minimum for integral error occurs near the region $a_1 = 2s$ – evaluating the integral error associated with this value (and its corresponding value for a_2) leads to an error of 5.9e-5 s/m, which is a significant improvement. Hence, the error is very sensistive to selection of thermal bounds for any given weighting values. The local minima containing this value (for given

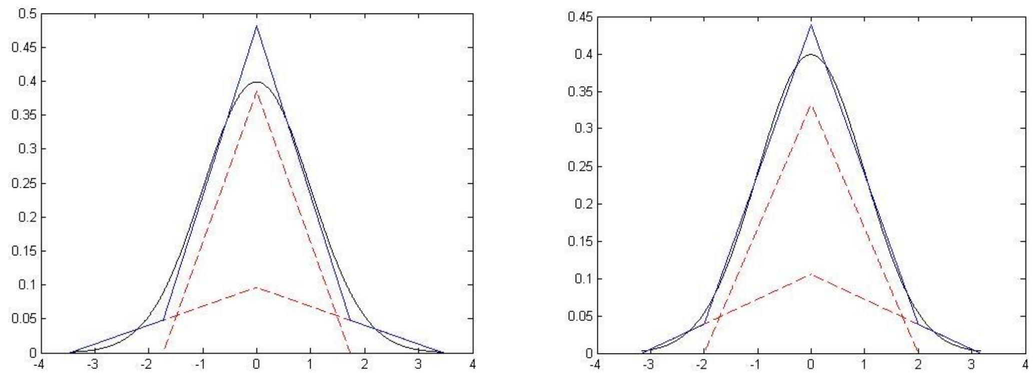

Figure 1: The original (dimensionless) one dimensional Maxwell-Boltzmann distribution function and several sample composite triangular distribution functions using [Left] w = [2/3, 1/3], a = $[3^{1/2}, 2(3)^{1/2}]$, and [Right] w = [2/3, 1/3], a = $[2, 10^{1/2}]$.

Case	Weights (w)	Thermal Bounds (as^{-1})	Error
1	[2/3, 1/3]	$[3^{1/2}, 2(3)^{1/2}]$	0.000437
2	[2/3, 1/3]	$[2, 10^{1/2}]$	0.000059

Table 1: Sample weights, thermal bounds and error for several selections of thermal bounds.

values of *w*) can also be located through the use of a Newton-Raphson scheme with an expression of the form:

$$a_{1,new} = a_1 - \frac{E\prime(a_1)}{E\prime\prime(a_1)} \tag{20}$$

However, care must be taken when using this approach. The global minimum (i.e. root) of the expression shown in Equation 19 is located outside of the bounds of possible values determined by the restrictions placed on our thermal bounds for the conservation equations listed in Equations 7-9. Another means of determining the error associated with the use of a composite distribution function is through comparison of the Finite Volume results for simulated test cases, which shall be covered in following sections.

2.4 Implementation for GPU using CUDA

The Compute Unified Device Architecture, or CUDA, is a programming model developed by NVIDIA for the purpose of allowing general computing tasks to be performed on NVIDIA graphics processing units. Several authors have successfully applied CUDA to performing kinetic-theory based simulation tasks using GPU devices [5,6] with significant performace gains due to the high locality of such schemes. The parallelization of the proposed vector split FVM can be performed in two steps:

1. By the parallelization across cells – i.e. allocate a thread to each cell (or multiple cells to each thread) - for embarassingly parallel computation of split fluxes F+ and F-.
2. Following computation across all cells and synchronization, parallization across cells for the computation of the change in conserved quantities U through contributions of fluxes across the cell surfaces.

3. A third (optional) parallelization across cells for computation of gradients for higher order implementation.

The parallelization using CUDA is obtained through the creation of multiple CUDA kernels. These kernels may either by __global__ (may be called from the host) or __device__ kernels, i.e. able to be called only from within a kernel running on the GPU. The total work to be performed in parallel is broken down into blocks of threads, which is then further broken down into warps of threads which are sent to Streaming Multi Processors (SMP's) on the GPU for computation. Hence, to ensure that no threads are idle, the resolution of the grid should be evenly divisible by both the number of threads per warp (32) and the number of threads per block. In the current studies, simulations are limited to resolutions which obey this restriction.

Any implementation of a general computing algorithm using CUDA and GPU devices requires the use of global memory on the GPU device. This memory is physically and conceptually seperated from the host (CPU's) memory and thus (i) memory must be allocated on both seperately, (ii) information required for computation must be copied between the GPU and the CPU using the PCI-E slot, and (iii) memory must be freed seperately. Due to the limited bandwidth of the PCI-E slot, this work focuses on use of the GPU as the sole computing entity. Hence, the model shown in the code fragment in Figure 2 is used for the implementation. A while loop – used for transient analysis of the unsteady equations – is used within the main() function executed on the host. Within the while loop, three kernels (for first order analysis) are launched sequentially – the parallel computation of fluxes in the x direction (GPU_Calc_X_Fluxes), y direction (GPU_Calc_Y_Fluxes) and finally computation of the state after time t + dt (GPU_Calc_U_from_G).

```
// setup execution parameters
dim3 threads(Ntx, Nty);
dim3 grid(Nbx, Nby);
dim3 grid_flux(Nbx+1, Nby+1);

//GPU calculation
for (int t = 1; t <= NO_STEPS; t++){
        GPU_Calc_X_Fluxes<<<grid_flux,threads>>>(d_P, d_F, d_Fp, d_Fm);
        GPU_Calc_Y_Fluxes<<<grid_flux,threads>>>(d_P, d_G, d_Gp, d_Gm);
        GPU_Calc_U_from_F_and_G<<<grid, threads>>>(d_U, d_P, d_F, d_G);
}
-----------------------------------------------------------------------------------------------------------
__global__ void GPU_Calc_X_Fluxes(float *d_P, float *d_F, float *d_Fp, float *d_Fm){

        int i = blockDim.x * blockIdx.x + threadIdx.x;
        int j = blockDim.y * blockIdx.y + threadIdx.y;

        if((i <= Nx) && (j < Ny)){
                GPU_Calc_F_from_P(d_P, d_Fp, d_Fm, i, j);
        }
}
```

Figure 2: Sample codes for [Top] the transient computation, and [Bottom] the GPU kernel for computing fluxes. A device function (GPU_Calc_F_from_P) is used for flux computation.

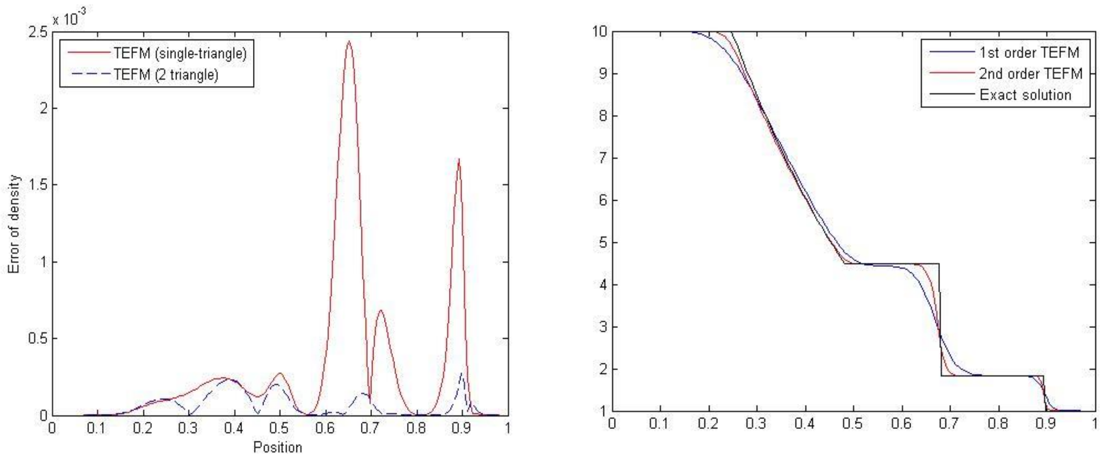

Figure 3: [Left] Sum of error squared for density comparing single and dual composite linear (triangular) distribution functions computed against the EFM solution. [Right] Comparison of the first and second order (spatial accuracy) extension of the TEFM scheme using the MINMOD limiter for the one dimensional shock tube problem.

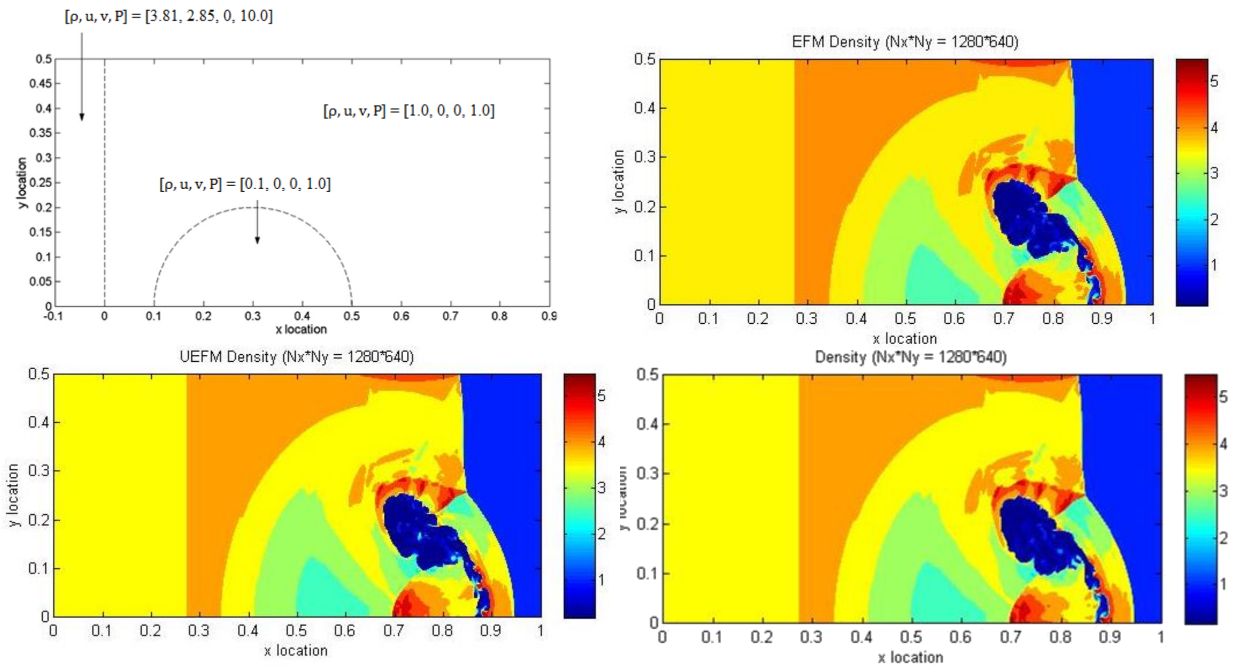

Figure 4: [Top Left] Initial conditions for the two dimensional shock interaction problem. Coloured contours of density for [Top, Right] EFM, and composite distribution functions using two [Bottom, Left] uniform distribution functions (UEFM) and [Bottom, Right] triangular distribution functions (TEFM).

Processor / GPU	Test 1	Test 2	Test 3	Average (s)
Intel Xeon E5-2670 (1 core)	2727.91	2703.16	2719.25	2716.773
Xeon E5-2670 + 1x Tesla C2075	7.82	7.83	7.83	7.827
Intel i3 + 1x GTX-670	9.33	9.28	9.19	9.267

Table 2: Performance comparison between a single core of an Intel Xeon (E5-2670) CPU against several NVIDIA GPU devices. The test is for a three dimensional blast wave problem of resolution 80x80x80 (~0.5 million cells) for 1000 time steps using the TEFM solver.

3. Results

3.1 Sod’s Shock Tube

A commonly used test for the compressible Euler equations for ideal gas flows is Sod’s one dimensional shock tube problem [7]. In this problem, a tube of dimensionless length unity is seperated into two parts at $x = 0.5$ by an infinitely thin diaphram. The conditions on each side of the interface are $[\rho, U, P]$=[10,0,10] and [1,0,1] on the left and right hand sides respectively. The diaphragm is removed at $t = 0$ and the resulting gas motion computed. Results for TEFM using two distribution functions is compared to the analytical solution and are shown in Figure 3. The sum of difference squared computed against the target solution (that using EFM) is also shown in Figure 3 for both a single triangular distribution function and the sum of two weighted triangular distributions. Regardless of the number of approximating distribution functions, the greatest errors occur in regions of the flow where gradients are high, with the greatest error occuring in the contact surface for a single distribution function and the shock wave for two distributions. The error is also seen to decrease when using two approximating distributions as compared to one, with a decrease in error of between 5 to 10 times in the shock and contact surface region. The maximum error for either a single or composite approximating distribution function does not exceed 0.2 percent when compared against the EFM solution.

3.2 Two Dimensional Shock Bubble Interaction Problem

The two dimensional simulation of a shock wave interaction with a region of different density has been previously used to test various qualitities of a solver [8] and is used here to compare the differences between the benchmark solution (EFM) and sevearl composite distribution functions using both uniform distribution (UEFM) and linear (triangular) distribution (TEFM) models. The initial conditions for the two dimensional problem are in Figure 4. The top and bottom boundaries are reflective, full slip surfaces with left and right hand boundaries treated using diriclet conditions. The MINMOD limiter is used for all flux reconstructions at cell interfaces. All simulations are performed using 1280x640 cells in the x and y directions respectively. With regard to the CUDA implementation, all kernels employed two dimensional thread blocks of 16x16 threads per block, resulting in a total of 3200 blocks of threads to be computed per time step. The GNU compiler was employed with –O3 optimization on all computations.

The simulation results are also shown in Figure 4. As expected, a travelling shock wave (from left to right) impacts with the low density / high temperature bubble. The difference in temperature results in a difference of wave speeds, resulting in a complex series of shocks, expansions and contact surfaces. The locations of most of the key features of the flow is identical for EFM, UEFM and TEFM. The dissipative nature of the solution is less apparent due to the second order extension, with all three solutions being very similar in nature. From these results, we conclude that either UEFM or TEFM are acceptable approximations to the original Equilibrium Flux Method.

3.3 Parallel Performance using GPU

The computational time required for the single-core CPU and single-device GPU simulations were recorded for a three dimensional blast wave problem using 80x80x80 (approximately 0.5 million) cells. The resulting times are presented in Table 2. For the tests, we employed several different CPUs – the Intel Xeon E5-2670, and the Intel i3-3220, all using a single core – and several GPU devices – the Nvidia C2075 Tesla, and the Nvidia GeForce GTX-670. The C2075 is a Fermi architecture device containing 448 CUDA computing cores operating at 1.15GHz with a memory bandwidth of 144 GB/s. The GTX-670 is a Kepler architecture based device, containing 1344 CUDA computing cores operating at 0.98 GHz with a memory bandwidth of approximately 192 GB/s. The comparison between the high-end system performance (Xeon+Tesla GPU) and the low-end system performance (i3 + GTX GPU) reveals a speedup of approximately 340 times and 293 times respectively when compared to the computational time required for a single Xeon core. These reported times are for the TEFM solver – the comparitive (i.e. GPU-CPU) performance for the UEFM solver is very similar to TEFM, meaning both methods are readily applied to GPU computation. The high performance is a result of the large degree to which vectorization is applied to the flux and state computation. In addition, careful and extensive use of registers are used to ensure reduced communications with global memory, providing a very high degree of performance.

4. Conclusions

Presented is the investigation into the use of Composite Distribution Functions (CDF's) for the approximation of the governing Maxwell-Boltzmann equilibrium particle velocity distribution function for use in Computational Fluid Dynamics (CFD) simulations. The original continuous distribution is replaced with a sum of weighted distribution functions which are simpler than the original distribution functions, with the goal of providing a computationally more efficient scheme. The resulting schemes, the TEFM and UEFM schemes, are shown to be approximately 10% faster than EFM alone. These schemes are then extended to parallel computation using Graphics Processing Units (GPUs'), reporting speedups of approximately 340 times using a C2075 Tesla GPU and 290 times using a GTX-670 when solving a three dimensional benchmark problem. Results are shown for a two dimensional shock bubble interaction problem, demonstrating that the GPU implementation is effective and that the proposed approximations to the governing distribution provide almost equivilent results. Future research will involve extension of the schemes to both multiple GPU and multiple CPU with explicit AVX intrinsic parallelization.

Acknowledgements

The corresponding author would like to acknowledge financial assistance made by Taiwan's National Science Council (NCS) through grants NSC 99-2221-E-492-005-MY3 and NSC 102-2221-E-006-115. We would also like to thank Acer for their valuable support through the loaning of workstations and GPU equipment for testing.

References

[1] F.-A. Kuo, M.R. Smith, C.-W. Hsieh, C.-Y. Chou and J.-S. Wu, GPU acceleration for general conservation equations and its application to several engineering problems, Computers and Fluids, 45[1]: pp. 147-154, 2011.

[2] D.I. Pullin, Direct simulation methods for compressible inviscid ideal-gas flow. Journal of Computational Physics, 34, pp. 231-44, 1980.

[3] J.C. Mandal and S.M. Deshpande, Kinetic flux vector splitting for Euler equations, Computers and Fluids, 23[2], pp. 447-478, 1994.

[4] A. Ferguson, M.R. Smith and J.-S. Wu, Accurate True Direction Solutions to the Euler Equations Using a Uniform Distribution Equilibrium Method, CMES, 63[1]: pp. 79-100, 2010.

[5] M.R. Smith F.-A. Kuo, C.-Y. Chou, J.-S. Wu and H.M. Cave, Application of a Kinetic Theory based solver of the Euler Equations using GPU, in Parallel Computational Fluid Dynamics: Recent Advances and Future Directions, Edited by R. Biswas, Pg. 440-445, 2010.

[6] L.-S. Lin, H.-W. Chang, C.-A. Lin, Multi relaxation time lattice Boltzmann simulations of transition in deep 2D lid driven cavity using GPU, Computers and Fluids, 80, pp. 381-387, 2013.

[7] G.A. Sod, A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws". Journal of Computational Physics, 27: pp. 1–31, 1978.

[8] M. Čada, M. Torrilhon, Compact third-order limiter functions for finite volume methods, Journal of Computational Physics, 228: pp. 4118–4145, 2009.