**Proceedings of the 8th International and 47th National Conference on Fluid Mechanics and Fluid Power (FMFP)**
**December 9-11, 2020, IIT Guwahati, Assam, India**

**FMFP2020-196**

# Performance of various shock capturing schemes on CPU's and GPU's

**Kalyan Deepak. G[1], Sujan B Thapa[1], Raja Mangalagiri[1], and Satya P Jammy[1]**

[1]Department of Mechanical Engineering, SRM University AP, Guntur 522502, India

**ABSTRACT**

In this work, the performance of schemes that capture the discontinuities in the flow field are analysed for the computing performance on GPU's and CPU's. Vanleer, AUSM, AUSM $\delta$, AUSM$^+$, Roe and HLLE schemes are implemented into the automatic code generation framework, OpenSBLIFVM. This framework is capable of generating codes for CPU's and GPU's from a high-level script written in Python. The computational run time analysis revealed that the Nvidia Tesla K80 GPU for the flux vector type splitting schemes is $40\times$ faster than a single CPU and $8\times$ faster than a full CPU node with 6 cores. For the the flux diffrence splitting schemes it is $60\times$ and $10\times$ compared to sequential and MPI. The Vanleer, AUSM and the delta variant of AUSM scheme are approximately 30 % faster than the other schemes considered. AUSM $\delta$ seems to be the best choice in terms of runtime and its ability to capture the shocks.

**Keywords:** Shock capturing, GPU, Code generation

## I. INTRODUCTION

Supersonic and hypersonic flows generate discontinuities in the flow field, these are referred to as shocks in the literature. A shock is a standing wave which causes a sudden increase or decrease of flow properties where the solution is discontinuous across these jumps. Shocks are found in flows over re-entry vehicles, missiles and supersonic aircraft to name a few. Numerical modelling of shocks is a challenge as the standard discretisation techniques like finite difference methods (FDM), finite volume methods (FVM) etc, assume the flow to be continuous. In order to handle these discontinuities special discretisation methods are needed. These techniques are known as shock capturing methods. There are numerous choices for shock capturing, some of the popular schemes in FDM are, Total Variation Diminishing (TVD) schemes, Weighted Essentially Non Oscillatory (WENO) [16] family of schemes and Targeted Essentially Non Oscillatory (TENO) [4] schemes. In FVM there are two main methods, flux vector splitting (FVS) and flux difference splitting (FDS). The flux vector splitting methods of Vanleer [18], Advection Upstream Splitting Method (AUSM) family of schemes, such as AUSM $\delta$, AUSM$^+$, AUSMPW are popular [7], [8]. These methods are based on an upwind scheme in the supersonic regions and retorts to central schemes in the subsonic regime. Contrary to FVS schemes, FDS schemes employ an approximate Reimann solvers, Roe [14], HLLE [3] are popular choices of FDS methods.

There have been numerous studies in the literature to demonstrate the abilities of these methods to capture shocks. These methods are able to select the correct upwind stencil based on the interface Mach number direction or evaluation of the flow properties at the interface using complex functions like Roe averaging, characteristic decomposition to name a few. The computational complexity of these methods are varied and there exists very limited literature on the runtime of these methods.

With the advent of Graphical Processing Units (GPU's) that are capable of delivering very high floating point operations (FLOPS) on a single chip, they have the capability to reduce the runtime of the solvers. However, the complexity of implementation takes a lot of re-write of the code and it is a tedious job on the developers. There are recently proposed automatic code generation approaches for future proofing and developing solvers for various architectures from a single source file. In these approaches the numerical developer implements the methods in symbolic form and a low-level C code is generated automatically [5], [9], [10]. The low-level C code written in domain specific languages like OPSC [12], [13], can automatically parallelise the code to various architectures.

OPSC is a domain specific language in the standard C++ code format which has calls to the OPS library. It can run the code/simulations on CPU (sequential, MPI) and GPU.

In this work, the ability of FVM based shock capturing schemes are used. The performance of these schemes to capture shocks, and the computational runtime on various architectures is presented. We have opted for FVM in this paper as it is a widely used discretisation method for a variety of fluid flow simulations. This method was deemed fit as it enforces conservation and also does not require any coordinate transformation for complex geometries, unlike finite difference methods. The brief methodology of the schemes and implementation are reported in section II, validation cases are reported in section III and in the final section the computational performance of these methods on Central Processing Units (CPU's) using Sequential and Message Passing Interface (MPI), using Compute Unified Device Architecture (CUDA) on the GPU's is presented.

## II. METHODOLOGY

The governing equations are the two dimensional compressible Euler equations, in the integral form. They are

written as,

$$\frac{\partial}{\partial t}\int_{\Omega}\vec{W}d\Omega + \oint_{\partial\Omega}\frac{\partial\vec{F_c}}{\partial x}\,d\vec{S} = 0 \tag{1}$$

where $\vec{W}$ is the conservative vector, and $\vec{F_c}$ is the convective flux, given by,

$$\vec{W} = \begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho E\end{bmatrix}, \vec{F_c} = \begin{bmatrix}\rho V\\ \rho u V + n_x p\\ \rho v V + n_y p\\ \rho H V\end{bmatrix} \tag{2}$$

where, $\vec{v}\,[u, v]$ is the velocity vector, $\rho$ is density, $p$ is the pressure, $E$ is the internal energy, $H$ is the enthalpy, $\bar{n}\ [n_x, n_y]$ is the unit normal and $V$ is the contravariant velocity, defined as the dot product of velocity and unit normal vectors $(\vec{v}.\vec{n})$.

This hyperbolic system is then solved numerically using FVM with Euler time integration (see [2] for details of the method) and can be written as,

$$\vec{W}^{n+1} = \vec{W}^{n} - \frac{\Delta t}{\Omega}\sum_{i=1}^{\text{nfaces}} F_{ci} \tag{3}$$

where, $\Delta t$ is the time step, $F_{ci}$ is the convective flux through the face $i$ and the superscript $n$ is the time level.

As outlined in section I, six different convective schemes are used in the present work. These differ in the way the convective flux $F_{ci}$ in Eq. (3) is evaluated. Four of these are the FVS type namely, Vanleer scheme, three AUSM family of schemes (AUSM, AUSM $\delta$ and AUSM$^+$), and the remaining two are Roe scheme with entropy fix and HLLE scheme which are FDS type schemes. The discretisation of these methods is explained briefly in the following paragraphs, for detailed discretisation see the references of the scheme.

The FVS scheme Advection Upstream Splitting Method (AUSM) developed by Liou and Steffen [8], this differentiates the convective and pressure fluxes. The scheme determines an interface Mach number based on the left and right cell values of a face. The discretisation can be written as,

$$F_{ci} = M_{ni} * \left[\vec{F_C} * a\right]_{L/R} + \text{pressure flux}_i \tag{4}$$

where $M_{ni}$ is the normal Mach number at the face $i$, it is evaluated with a split form as given in [8], $\vec{F_C}$ is the convective terms in eq. 2 without the pressure terms, $a$ is the speed of sound and pressure flux is the contribution of pressure terms in eq. 2 at the face $i$.

AUSM $\delta$ differs from the AUSM scheme in the evaluation of advection Mach number to resolve the subsonic flows better and is given by,

$$|M_{ni}] = \begin{cases} |M_{ni}| & \textbf{if} \quad |M_{ni}| > \delta \\ \frac{(M_{ni})^2+\delta^2}{2\delta} & \textbf{if} \quad |M_{ni}| \le \delta \end{cases} \tag{5}$$

where $\delta$ is a value $(0 < \delta < 0.5)$. AUSM$^+$ differs in the evaluation of the interface Mach number and pressure for details see, [7].

The other FVS scheme used in this work is the Van Leer's scheme based on decomposition of the convective fluxes into a positive and a negative part based on the interface Mach number.

The FDS schemes solves the approximate Riemann solver at the interface based on the eigenvalues of the flux jacobian. The entropy fix formulation of the Roe's scheme [2] and the entropy as well as positivity preserving formulation of the HLLE scheme are implemented.

### A. OpenSBLIFVM implementation

These schemes are implemented in the OpenSBLIFVM framework, which uses the OpenSBLI [10] framework and SymPy [11] as the building blocks. The auto generated code from the framework coupled with the OPS [13] library can run on various architectures. On the CPU's it can run in sequential, Message Passing Interface (MPI), OpenMP, Hybrid MPI + OpenMP versions, where as on GPU it can run in CUDA, OpenCL and the MPI versions of these. In the current work the sequential, MPI and CUDA versions are used.

## III. SIMULATIONS

To validate the implementation of different shock capturing schemes various standard test cases are simulated. They are (a) Shu-Osher test case, (b) Sod Shock tube and (c) Bow shock. The first two cases are standard test cases where as the third one is taken from the 5th higher order CFD workshop validation cases, which provides the grids and the criterion for the test case to be considered as validated [1].

### A. Sod Shock tube

A ubiquitous test for testing the ability of various schemes in the numerical methods is the sod shock tube, and was initially proposed by [17]. It is a Riemann problem, used as a standard test case for the validation of the solver. The domain is taken as $1\times0.125$ non dimensional units. Different grids are generated from $100\times5$, $200\times5$, $300\times5$, $400\times5$, $500\times5$ and $2000\times5$. These grids are chosen inorder to differentiate the capabilities of various solvers. The ratio of specific heats $(\gamma)$ is taken as 1.4. As the problem is 1D in nature, the length and grid points in $y$ are choosen to be small.

A shock wave at x = 0.5 is initialized in the domain at t = 0. The left and right values of the primitive variables are given by,

$$(\rho, p, u, v) = \begin{cases} 1, 1, 0, 0 & \textbf{for} \quad x \le 0.5 \\ 0.125, 0.1, 0, 0 & \textbf{for} \quad x > 0.5 \end{cases} \tag{6}$$

Inviscid wall boundary condition is used on all the boundaries of the domain and the simulations are run with a constant time step of $0.00002$, until a non dimensional time of 0.2.

Figure 1 shows the density profile for the grids considered using AUSM $\delta$ scheme along with the theoretical caluclation. It can be inferred from the figure that the solution with grid size $2000\times5$ points matches with the

theory and the solution is bounded within the limits. The dissipation is high for other grid sizes and the shock is smeared.

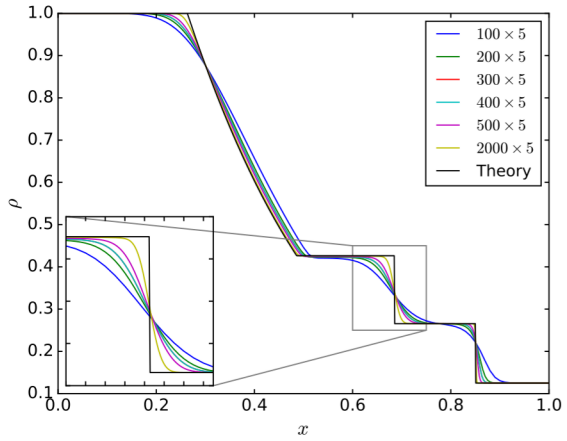


**Figure** 1: **Grid Independence for Sod Shock Tube case using AUSM $\delta$ scheme.**

The capability of the various schemes to solve the test case is shown in fig.2 for the grid $100 \times 5$. It can be inferred from the figure that Vanleer shows maximum resolvance, AUSM family of schemes predict the properties similar to Vanleer and HLLE schemes are the least accurate.

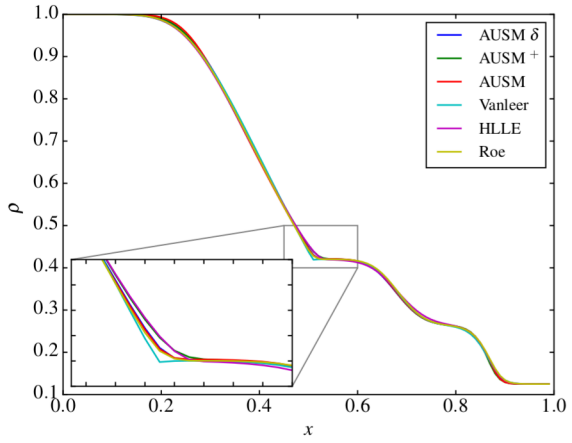


**Figure** 2: **Sod Shock tube problem using various schemes for a grid of** $100 \times 5$**.**

## B. Shu and Osher

The Shu-Osher problem simulates the movement of the normal shock front in a one dimensional inviscid flow with density fluctuations. This test case is performed to validate the capability of the solver to accurately capture the rich structure behind the shock. The simulations are run with Vanleer, AUSM, Roe and HLLE schemes.

A two dimensional rectangular non dimensional domain of $1.0 \times 0.2$ with different grids sizes of $200 \times 4, 2500 \times 4, 3000 \times 4$ and $4000 \times 4$ are taken and the simulation results are compared with the results from [15].

The inflow and outflow boundary conditions are applied to the left and right faces of the domain where as the inviscid wall is applied on the other faces as the problem is one dimensional.

The domain is initialized by varying the density with respect to length. The left and the right states are given by,

$$(\rho, p, u, v) = \begin{cases} 3.86, 10.33, 2.63, 0 & \textbf{for} \quad x \leq 0.1 \\ 1 + 0.2sin(50x), 1, 0, 0 & \textbf{for} \quad x > 0.1 \end{cases} \tag{7}$$

The Mach number at the inflow is taken as 3, ratio of the specific heats $\gamma = 1.4$. The simulations are run till a non dimensional time of t = 0.18 with a time step of $1 \times 10^{-7}$. The smaller timestep is chosen in order to eliminate temporal discretisation errors.

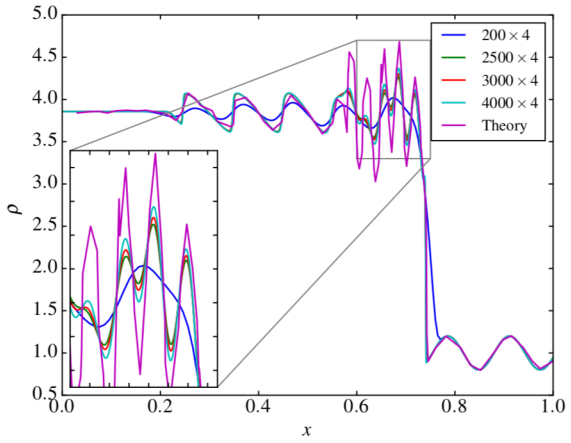


**Figure** 3: **The grid independence plot of the density variations for the Shu-Osher test case using AUSM $\delta$.**

The grid independence results are shown in Fig. 3. It can be inferred from the figure that the $4000 \times 4$ grid was able to capture the shock location and the upstream disturbances close to the reference solution. There is some deviation in the upstream disturbances, this is due to the lower order of accuracy of the solver used in this work.

The best grid from the grid independence i.e $4000 \times 4$ is taken and the simulations are performed with different schemes. The density profile along the length of the domain is shown in Fig. 4. Similar to the Sod Shock tube case the HLLE schemes are found to be more diffusive compared to AUSM, Vanleer and Roe schemes.

## C. Bow Shock

Apart from the shock tube and Shu-Osher cases, a bow shock is a challenging case due to the presence of stagnation point and curvature in the shock. It becomes more

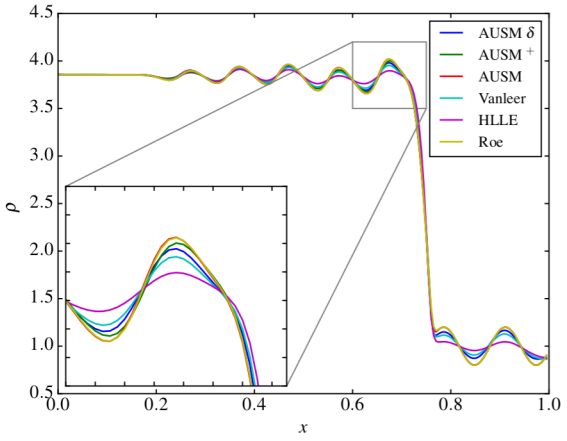


**Figure** 4: **Shu-Osher problem using various schemes for a grid of** $200 \times 4$**.**

challenging when the grid is aligned with the shock and stagnation line.

Given the complexities, this case is given as one of the validation cases in the 5th HOWCFD workshop [1]. The grids are provided by the team at NASA where the test case is subjected to a Mach 4 inflow. The ratio of specific heats $\gamma$ is taken as 1.4.

The sample grid and the boundary conditions applied are shown in fig.5. As seen from the figure the grid is aligned with the stagnation line and the density of the grid is high at the expected bow shock location.

Five different grids provided by [1] are taken, the grid sizes are $71 \times 30$, $148 \times 154$, $296 \times 98$, $590 \times 186$, and $1180 \times 362$.

The simulations were run till $L_\infty$ norm of the density residual is below $5 \times 10^{-9}$ for all the grid sizes. The convergence of the AUSM $\delta$ scheme is shown in Fig. 6.

The performance of the AUSM $\delta$ scheme to capture the stagnation line shock for the grids considered is shown in Fig. 7. As seen, the smallest grid generated oscillations around the shock and also at the stagnation point.

The contours of pressure and density are reported in Fig. 8(a) and Fig. 8(b) for the $1180 \times 362$ grid. As seen, the bow shock is very well captured and upon close observation it can be seen that the bow shock was within the stretched grid region of Fig. 5.

Figure 9 shows the comparison between AUSM family of schemes for the $590 \times 186$ grid size, as seen all three schemes capture the discontinuity.

The FDS schemes were unable to converge except for the grid $71 \times 30$. This might be due to the carbuncle phenomenon as well as grid aligned shock structure and stagnation line. It is similar to the findings of [6] in which the Roe's scheme didn't converge for the double ellipse problem. This might be due to the implementation of the inviscid wall boundary condition.

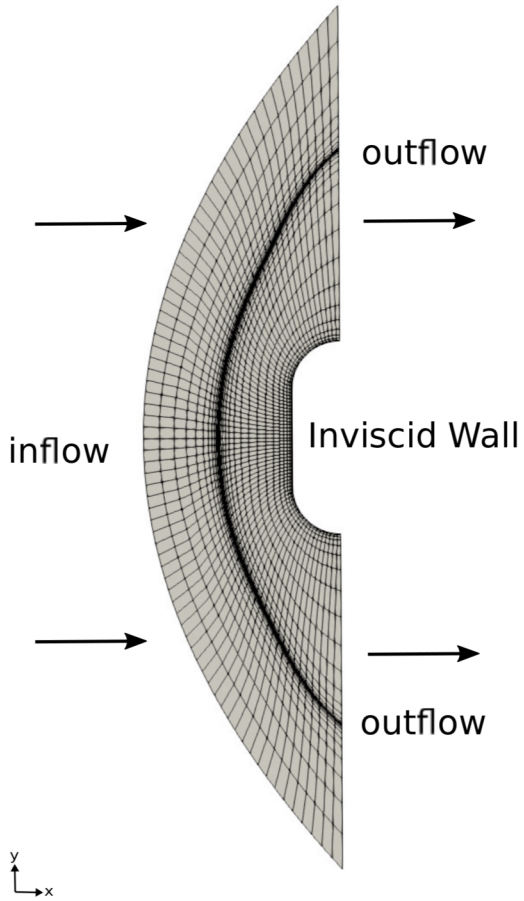


**Figure** 5: **Boundary conditions for bow shock depicted on a mesh of** $71 \times 30$

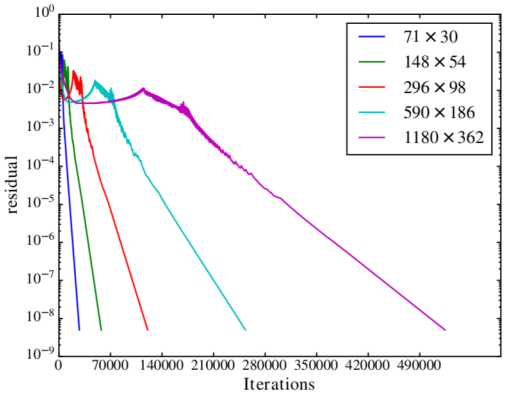


**Figure** 6: **Iterations taken to converge for varying grid sizes in bow shock.**

## IV. Computational cost comparison on GPU's and CPU's

After validating the solver, the runtime on various architectures is analyzed. Two test systems have been used,

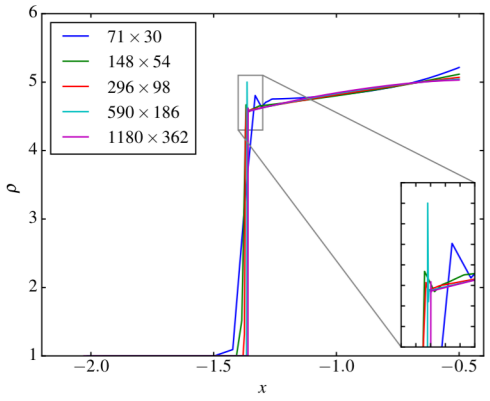


**Figure** 7: **Grid independence for Bow Shock case using AUSM $\delta$ scheme.**

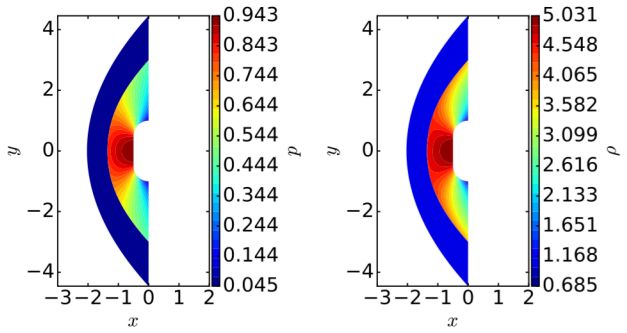


**Figure** 8: **Pressure and $\rho$ contours of bow shock for a grid of** $1180 \times 362$**.**

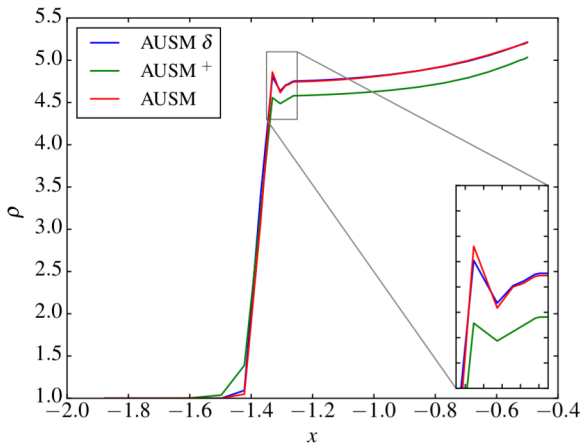


**Figure** 9: **Bow Shock problem using various schemes for a grid of** $590 \times 186$**.**

one is a workstation with AMD Ryzen 7, 2700 CPU having 8 cores and the other is a virtual machine with NVIDIA Tesla K80 GPU on google cloud. On the CPU,

**Table** 1: **The MPI runtime for 1000 iterations of the Shu-Osher test case on** $4000 \times 400$ **grid size.**

| | Sequential | 4core | 6core |
|---|---|---|---|
| **AUSM** | 396.87 | 107.64 | 82.59 |
| **AUSM$^+$** | 440.75 | 134.34 | 97.27 |
| **AUSM $\delta$** | 467.18 | 118.63 | 87.95 |
| **HLLE** | 604.24 | 156.09 | 110.98 |
| **Roe** | 725.53 | 191.84 | 135.37 |
| **Vanleer** | 390.93 | 109.62 | 89.19 |

**Table** 2: **The cumulative runtime for 1000 iterations of the Shu-Osher test case on** $4000 \times 400$ **grid size.**

| | Sequential | MPI | GPU |
|---|---|---|---|
| **AUSM** | 396.87 | 82.59 | 9.58 |
| **AUSM$^+$** | 440.75 | 97.27 | 10.94 |
| **AUSM $\delta$** | 467.18 | 87.95 | 13.67 |
| **HLLE** | 604.24 | 110.98 | 10.04 |
| **Roe** | 725.53 | 135.37 | 11.31 |
| **Vanleer** | 390.93 | 89.19 | 13.31 |

the GNU compliler `g++ version 9.3.0, with -O3 vector optimization` and on the GPU, `CUDA 10 with -O3 optimization` is used for compilation, the MPI library (`mpich`) is utilized on the CPU.

The sequential and MPI codes are run on CPU and the CUDA version was run on the GPU machine. Note, only 6 cores out of the available 16 on the workstation are used in the MPI runs. The test case chosen is the Shu-Osher case with $4000 \times 400$ grid points in x and y respectively. The cumulative runtime of the time stepping loop is recorded for 1000 iterations. The recorded runtime for the seq, MPI, and CUDA are shown in Table 2.

The runtimes are normalized with the sequential runtime and the result is inversed to evaluate the speedup of the schemes on various architectures. These are presented in Table 3. As seen from the table, the AUSM MPI code with 6 processors is $5\times$ times faster than the sequential version. The AUSM GPU is $41.5\times$ faster than single CPU and $8.5\times$ times faster than 6 core CPU. Similar speedups are found for other schemes on CPU's, on GPU's the speedup for FDS schemes is around $60\times$.

On a GPU, the AUSM scheme is the fastest of all. Compared to AUSM, AUSM $\delta$ takes 10% more time, AUSM$^+$ takes 30% more time, Roe's scheme takes 60% and HLLE

**Table** 3: **Speed-up compared to sequential**

| | Sequential | MPI | GPU |
|---|---|---|---|
| **AUSM** | 1 | 4.76 | 41.43 |
| **AUSM$^+$** | 1 | 4.54 | 40.29 |
| **AUSM $\delta$** | 1 | 5.33 | 34.18 |
| **HLLE** | 1 | 5.37 | 60.18 |
| **Roe** | 1 | 5.34 | 64.15 |
| **Vanleer** | 1 | 4.38 | 29.37 |

schemes take $40\%$ time. Vanleer is very close to AUSM scheme

## V. CONCLUSIONS

In this work, the ability of the schemes to capture discontinuities in the flow are presented for canonical test cases such as Shu-Osher and Sod's shock tube. The schemes that are implemented are Vanleer, AUSM, AUSM $\delta$, AUSM$^+$, Roe and HLLE. They are implemented into the automatic code generation framework, which is capable of generating codes that can run simulations using CPU's and GPU's with a wide variety of programming paradigms. As available in the literature, these schemes work well with the test cases.

The computational performance of various schemes listed in this work are investigated by running them on CPU and GPU with MPI and CUDA respectively. The MPI code gets near linear scaling with increasing processors as shown in Table 1. However, the performance of the solvers on GPU is very impressive achieving a speedup of $40\times$ for all flux vector splitting schemes compared to sequential version and around $8\times$ compared to a 6 core CPU. For the flux difference splitting schemes (Roe and HLLE), the speed up on GPU is $60\times$ compared to sequential and $10\times$ compared with MPI respectively. These flux difference splitting schemes offer better ability to capture shocks and runtime compared to other schemes used in this work.

## NOMENCLATURE

| | |
|---|---|
| $\vec{W}$ | Conservative vector |
| $\vec{F_c}$ | Convective flux vector |
| $M$ | Mach number |
| HPC | High Performance Computing architectures |
| CUDA | Compute Unified Device Architecture |
| OPS | Oxford Parallel Library for Structured grids |
| GPU | Graphical Processing Units |
| MPI | Message Passing Interface |
| Seq | sequential |

## REFERENCES


[1] *5th international workshop on high order cfd methods*, https://how5.cenaero.be/.

[2] Jiri Blazek, *Computational Fluid Dynamics: Principles and Applications: Third Edition*, 2015.

[3] Bernd Einfeldt, *On godunov-type methods for gas dynamics*, SIAM Journal on Numerical Analysis **25** (1988), no. 2, 294–318.

[4] Lin Fu, Xiangyu Y Hu, and Nikolaus A Adams, *Targeted eno schemes with tailored resolution property for hyperbolic conservation laws*, Journal of Computational Physics **349** (2017), 97–121.

[5] Christian T. Jacobs, Satya P. Jammy, and Neil D. Sandham, *OpenSBLI: A framework for the automated derivation and parallel execution of finite difference solvers on a range of computer architectures*, Journal of Computational Science **18** (2017), 12–23.

[6] Bibin John, G Sarath, Vinayak Kulkarni, and Ganesh Natarajan, *Performance comparison of flux schemes for numerical simulation of high-speed inviscid flows*, Progress in Computational Fluid Dynamics, an International Journal **14** (2014), no. 2, 83–96.

[7] Meng Sing Liou, *A sequel to AUSM: AUSM+*, Journal of Computational Physics **129** (1996), no. 2, 364–382.

[8] Meng-Sing Liou and Christopher J. Steffen, *A new flux splitting scheme*, Journal of Computational Physics **107** (1993), no. 1, 23 – 39.

[9] David J. Lusher, Satya P. Jammy, and Neil D. Sandham, *Shock-wave/boundary-layer interactions in the automatic source-code generation framework opensbli*, Computers & Fluids (2018), 1–5.

[10] David J. Lusher, Satya P. Jammy, and Neil D. Sandham, *OpenSBLI: Automated code-generation for heterogeneous computing architectures applied to compressible fluid dynamics on structured grids*, **173** (2020), no. 2018, 17–21.

[11] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz, *Sympy: symbolic computing in python*, PeerJ Computer Science **3** (2017), e103.

[12] G.R. Mudalige, I.Z. Reguly, S.P. Jammy, C.T. Jacobs, M.B. Giles, and N.D. Sandham, *Large-scale performance of a dsl-based multi-block structured-mesh application for direct numerical simulation*, Journal of Parallel and Distributed Computing **131** (2019), 130 – 146.

[13] István Z. Reguly, Gihan R. Mudalige, Michael B. Giles, Dan Curran, and Simon McIntosh-Smith, *The OPS domain specific abstraction for multi-block structured grid computations*, Proceedings of WOLFHPC 2014: 4th International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing - Held in Conjunction with SC 2014: The International Conference for High Performance Computing, Networking, Storage and Analysis (2014), 58–67.

[14] P. L. Roe, *Approximate Riemann solvers, parameter vectors, and difference schemes*, Journal of Computational Physics **43** (1981), no. 2, 357–372.

[15] Shu sheng Chen, Chao Yan, Bo xi Lin, and Yan su Li, *A New Robust Carbuncle-Free Roe Scheme for Strong Shock*, Journal of Scientific Computing **77** (2018), no. 2, 1250–1277.

[16] Chi-Wang Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, Advanced numerical approximation of nonlinear hyperbolic equations, Springer, 1998, pp. 325–432.

[17] Gary A Sod, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, Journal of Computational Physics **27** (1978), no. 1, 1 – 31.

[18] Bram van Leer, *Flux-vector splitting for the euler equations*, Eighth International Conference on Numerical Methods in Fluid Dynamics (Berlin, Heidelberg) (E. Krause, ed.), Springer Berlin Heidelberg, 1982, pp. 507–512.