**Tenth International Conference on Computational Fluid Dynamics (ICCFD10), Barcelona,Spain, July 9-13, 2018**

**ICCFD10-2018-0304**

# Development of a Pressure Based, Unstructured, GPU Accelerated CFD Solver for Compressible Reacting Flows at all MACH Numbers

B. Ozkan*,**, S. Uslu*
Corresponding author: bertanozkan@etu.edu.tr

* TOBB University of Economics and Technology, Turkey
** ROKETSAN A.S. Turkey

CFD, Computational Fluid Dynamics, tools have become an integral part of product design and analysis of gas turbines, rocket engines and thrusters. In order to model the chemical reacting flow in the combustion chambers, a CFD solver dealing with combustion of multi species reactions is developed. The solver must also be able to cope with the three flow regimes, subsonic, transonic and supersonic flow that occur in many flow problems. This is namely called a solver for all flow speeds. This solver also utilizes all resources of computers by being CPU parallel and GPU accelerated. A third party Open Source matrix solver ViennaCL is integrated to the Solver to solve linear system of equations.

Reactive Computational Fluid Dynamics RCFD, Numerical Combustion, GPU acceleration

## Nomenclature

| | |
|---|---|
| c | Speed of Sound |
| $c_p$ | Specific heat at constant pressure |
| $\mathbf{u}$ | Velocity Vector |
| $\phi$ | Transported scalar |
| $\Gamma$ | Coefficient of the Diffusion term |
| T | Temperature |
| $\mu$ | Viscosity |
| $P$ | Pressure |
| $\rho$ | Density |
| $Y_n$ | Mass fraction of the n'th molecule |
| t | Time |
| $\phi_P$ | Transported scalar of the main cell |
| $\phi_A$ | Transported scalar of the neighbor cell |
| $\Delta A$ | Surface area of the cell face |
| $\Delta V$ | Volume of the cell |
| $\Delta\zeta$ | Distance between cell centers |
| $\mathbf{n}$ | Normal vector of the cell face |
| $\Delta t$ | Time step |
| $\sigma_h$ | Prandlt Number |
| CPU | Central Processing Unit |
| GPU | Graphics Processing Unit |
| CFL | Courant-Friedrichs-Lewy condition |

# 1 Introduction

In this manuscript, development of a Reactive Computational Fluid Dynamics solver will be briefly described and validation efforts for this solver will be presented. The solver is a Pressure Based RCFD, Reacting CFD, solver that uses Finite Volume Methodology. The solver that can operate on 3D unstructured grids containing arbitrary polyhedral computational meshes. This solver can be used for modeling flows with multi-species and chemical reactions. It uses a Pressure Based approach named "A collocated finite volume method for predicting flows at all speeds" Demirdzic [1] for Pressure-Velocity coupling. The solver uses 7'th order polynomials for ideal gas calculations and Sutherland viscosity model. Arrhenius type equation is used in Finite Rate Chemistry model for simulating chemical reactions. First order Implicit Euler time integration with various matrix solvers for sparse matrices are used in the methodology. It can solve Unsteady Navier-Stokes Equations or Euler equations for modeling transient flows or uses local time-stepping for modeling steady flows. This solver is a CUDA GPU accelerated code to utilize the numerical capabilities of Graphical Processing Units. The solver also uses multiple CPU's in a parallel fashion. Validation of the solver has been carried out on "Subsonic, Transonic and Supersonic Flow over a circular arc bump" [1] , laminar flow over a flat plate and Sandia Flame D Combustion test case. Parallel computing and GPU acceleration performance of this solver also studied in this manuscript. A third party, Open source, GPU-CPU parallel matrix solver named ViennaCL [2] is integrated to the Solver in order to be an alternative to the built-in Successive over-relaxation method liner equation solver.

# 2 Methodology

In this section, analytical and numerical methodologies of the solver is briefly described.

Finite volume discretization is applied to the multi-species, chemically reacting flow equations. Integral version of the general transport equation in 3D space can be given as [3]:

$$\int_V \frac{\partial(\rho\phi)}{\partial t} dV + \int_V \nabla \cdot (\rho\phi \mathbf{u}) dV = \int_V \nabla \cdot (\Gamma \nabla\phi) dV + \int_V S_\phi dV \tag{1}$$

A set of transport equations are being used in the solver as shown in equation 2. The equations can be described as continuity, momentum, enthalpy and species transport. In enthalpy equation unity Lewis number has been chosen for the sake of simplicity.

$$\phi = \begin{pmatrix} 1 \\ u_x \\ u_y \\ u_z \\ h \\ Y_1 \\ \vdots \\ Y_n \end{pmatrix} \quad ; \quad \Gamma = \begin{pmatrix} 0 \\ \mu \\ \mu \\ \mu \\ \frac{\mu}{\sigma_h} \\ \rho D_1 \\ \vdots \\ \rho D_n \end{pmatrix} \quad ; \quad S_\phi = \begin{pmatrix} 0 \\ (-\nabla P)_x \\ (-\nabla P)_y \\ (-\nabla P)_z \\ (S_h + \frac{\partial P}{\partial t}) \\ \dot{\omega}_1 \\ \vdots \\ \dot{\omega}_n \end{pmatrix} \tag{2}$$

After applying finite volume method, general transport equation becomes like this.

$$\rho \left( \frac{\phi_P^n - \phi_P^{n-1}}{\Delta t} \right) \Delta V + \sum \mathbf{n}_i \cdot (\rho \mathbf{u}) \Delta A_i \phi_i^n = \sum \Gamma \left( \frac{\phi_A^n - \phi_P^n}{\Delta \zeta} \right) \Delta A_i + \bar{S} \Delta V \tag{3}$$

A visualization of the variables in this equation can be seen in figure 1.

First order implicit Euler method is applied to the time integral, first order upwind scheme is applied to the convective flux term and central discretization is applied to the diffusive flux term. In convective flux, first upwind cell is identified and flux is calculated in the upwind direction.

$$\phi_i = \begin{Bmatrix} F_i > 0 & \phi_P \\ F_i < 0 & \phi_A \end{Bmatrix} \tag{4}$$

After jacobians of cells has calculated. General transport equation becomes linear.

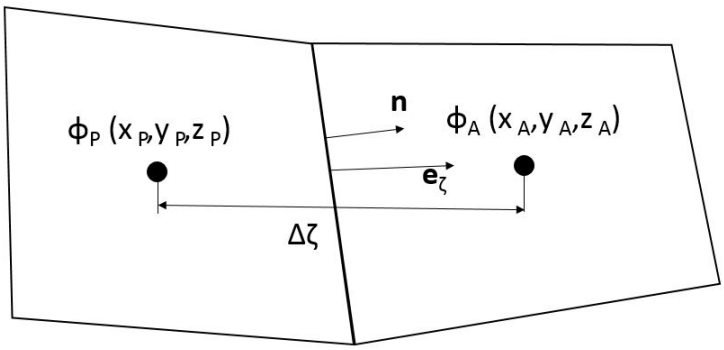



Figure 1: Visualization of the Finite Volume Method variables.

$$a_p\phi_P^n + \sum a_A\phi_A^n - \left(\frac{\rho\Delta V}{\Delta t}\right)\phi_P^{n-1} + \bar{S}\Delta V = 0 \tag{5}$$

General transport equation for all cells creates a system of linear equations and this system of linear equations is solved with a matrix solver. Solver has a built in serial matrix solver which uses Successive over-relaxation method but the main matrix solver of this code is the third party matrix solver named ViennaCL [2]. ViennaCL is a state of the art matrix solver which can be used CPU parallel with OpenMP [4] or GPU parallel with CUDA [5]. ViennaCl has a variety of direct solvers, iterative solvers and smoothers In this code mainly Conjugate Gradient [6] solver with Algebraic Multigrid Smoother [7] is used.

In Steady simulations, local timesteps are calculated with CFL number input using this equation:

$$\Delta t = N_{CFL}\left(\frac{\Delta V}{\sum\left(|\mathbf{u}_i\cdot\mathbf{n}_i| + c_i\right)\Delta A_i}\right) \tag{6}$$

The Solver being pressure based, a pressure correction equation is need to be solved in order to couple velocity with pressure. The Solver being compressible, aforementioned pressure correction equation should be compressible and should be able to correct densities of the cells as well as pressures and velocities. Pressure-Density-Velocity correction method from Demirdzic [1] is chosen to be the solution algorithm. In this algorithm, first velocity equations is solved an then pressure correction equation is solved. Detailed description of this method is given in the book from Ferziger and Peric [8]. Summary of the pressure correction equations is given below.

$$\frac{\rho'\Delta V}{\Delta t} + \sum_{f=1}^{n_f} -(\rho^{m-1}\Delta A_f d)(\nabla P)'_f + \frac{\dot{m}_f^*}{\rho^{m-1}}\left(\frac{\partial\rho}{\partial P}\right)_T P' + Q_m = 0 \tag{7}$$

The $Q_m$ term in this equation is given below.

$$\frac{\rho'\Delta V}{\Delta t} + \sum \dot{m}'_f + Q_m = 0 \tag{8}$$

In these equations $\dot{m}_f^*$ is uncorrected total mass flow rate entering and exiting the cell. $\rho^{m-1}$ is density of the cell in the latest time step. $P'$ is pressure correction variable of the cell, which is the purpose of this equation. $d$ variable of the cell is calculated like this.

$$d = \frac{\Delta V}{a_p} \tag{9}$$

$a_p$ term in this equation stands for Jacobian of the velocity equation.

$(\nabla P)'_f$ term in equation 7 is the gradient of the pressure correction term at the face and $\rho'$ term at this equation is density correction term. Density correction term is calculated like this.

$$\frac{\rho'}{P'} \approx \left(\frac{\partial \rho}{\partial P}\right)_T \tag{10}$$

Which uses the ideal gas equation:

$$P = \rho R T \tag{11}$$

It can bee seen that every term in equation 7 is known at this point except pressure correction term, so pressure correction equation can be solved implicitly to determine pressure correction variables of the cells. After Pressure correction terms of cells have calculated, pressures of cells is correcting with this equation.

$$P = P^* + \omega_p P' \tag{12}$$

$\omega_p$ term in this equation is called pressure under relaxation factor which is obtained from the user. This value is generally inputted as 0.3. After correcting pressures, velocities is corrected with this equation:

$$\mathbf{u} = \mathbf{u}^* + \omega_u \mathbf{u}' \tag{13}$$

$\omega_u$ term is also an under relaxation factor and it is the under relaxation factor of velocities, and this value is generally inputted as 0.9. After correcting pressures and velocities, densities are calculated with the ideal gas equation.

After correction step, algorithm solves enthalpy and species equations. Number of species is obtained from the user as well as number of chemical reactions.

In order to calculate the reaction rates for simulation of chemically reacting flows, Arrhenius equation is being used. In literature this model is named Finite Rate Chemistry.

$$RR_m = A_p e^{\left(\frac{-E_A}{RT}\right)} \prod^{n_n} [C_n]^{\psi_n} \tag{14}$$

Source terms of the species equations is calculating using reaction rate values.

$$\dot{\omega}_n = M_n \sum_{m=1}^{n_m} \dot{q}_{m,n} \tag{15}$$

$$\dot{q}_{m,n} = (v''_{n,m} - v'_{n,m}) R R_m \tag{16}$$

Temperature of the cells are calculated using this formula [3] .

$$T = \frac{h - \sum_{n=1}^{n_n} Y_n h^0_{f,n} - \left(\frac{|\mathbf{u}|^2}{2}\right)}{c_{p,mix}} \tag{17}$$

Mixture specific heat values ($c_{p,mix}$) is calculating using this formula.

$$c_{p,mix} = \sum_{n=1}^{n_n} Y_n c_{p,n} \tag{18}$$

Specific heat values of species is obtained from user as well as conductivity data. Generally these variables is inputted as polynomials and can be calculated using this formula.

$$c_p = c_1 + c_2 T + c_3 T^2 + c_4 T^3 + c_5 T^4 + c_6 T^5 + c_7 T^6 \tag{19}$$

Constants of specific heat and conductivity can be obtained from NASA material databases [9].

Viscosity of the cell is also calculated empirically with the coefficients obtained from user. Formula for calculating viscosity is given below.

$$\mu = \frac{C_1 T^{3/2}}{T + C_2} \tag{20}$$

Rhie and Chow face velocity interpolation method is being used to calculate face velocities and Gauss-Green method is used for gradient reconstruction.

# 3 Algorithms

In this section, computational algorithms and parallel computation methods of the Solver and ViennaCL is briefly described.

This solver needs 7 input files to work. First of all a configuration file is needed to obtain initial conditions, numerical methods to use , variables of these numerical methods , number of species , number of reactions, filenames of other input files and simulation control variables. Second input file needed is the mesh file. The Solver uses ASCII mesh files with ".neu" extensions. The name of this file is obtained in the configuration file. Third input file is boundary conditions file which obtains boundary condition info from user. Fourth input file is molecules file which contains information about molecules which will be solved. Fifth input file is material polynomial data file which is needed to calculate material specifications like specific heat and Enthalpy of formation. Sixth input file is reactions file which contains information about reactions. Main inputs of the reaction file is the stoichiometric coefficients of the reactions and Arrhenius coefficients of the reactions. Seventh input file is the restart file which is needed if a simulation is needed to be restarted from a previous state. Restart operation is controlled with a flag in the configuration file.

This solver generates three output files. First file generated is the solution file which includes solution fields of the selected variables. Second file generated from the code is surface file which contains information about flagged surface like forces and temperatures. Third file generated with the Solver is aforementioned restart file.

The Solver can work with three different cell types. These cell types are tetrahedron, hexahedron and wedge. Two dimensional simulations is performed with generating a 3D mesh with one cell thickness and designating the boundaries in third dimension as "empty" boundary condition.

Performance wise, a CFD iteration can be splitted into two parts. Preparation of the matrices and solution of the matrices. In this solver, matrix solving time is calculated to be aproximately %40 of the total iteration time. In light of this information, GPU acceletation is only applied to the matrix solution part, considering writing CUDA code for matrix preparation parts of the Solver will be complex and not as efficient as GPU accelerating the matrix solver.

For the Matrix preparation part, loop base parallelism is developed with OpenMP [4] libraries. The reason loop base parallelism is selected instead of partition based parallelism is simply avoiding partitioning operations. Performance evaluation of this method can be seen in performance section.

For the matrix solving part there are two options in the ViennaCL. First option is computing AMG smoother operations and iterative solver operations with CPU using OpenMP. Second option is computing AMG smoother operations with CPU using OpenMP and then transferring matrix data to GPU and computing iterative solver operations using CUDA and transferring results back to host device [2]. Performance of these options are evaluated in the performance section.

# 4 Validation Cases

Laminar flow over a flat plate is being studied to validate the solution of Navier-Stokes equation. In this problem, boundary layer on 1 feet of wall is studied. As inlet, a boundary with total pressure of 1000000 $N/m^2$ is applied. As output, a boundary with static pressure of 97250 $N/m^2$ is applied. These boundary conditions results a flow with mach number of 0.2. Solution of this study can be seen on Figure 2. Comparison of the computational solution with Blasius analytical solution can be seen on Figure 3.

"Subsonic, Transonic and Supersonic Flow over a circular arc bump" [1] test cases have been extensively studied to validate convective flux and pressure-velocity coupling schemes of the solver. Solutions of subsonic (Mach = 0.5), transonic (Mach = 0.675) and supersonic (Mach = 1.65) flow are presented in Figures 4, 6, 8 respectively. Comparisons of the Solver's solutions and Demirdzic's solutions [1] on the bottom walls of the simulations are presented in the plots 5 , 7 , 9 respectively. Subsonic and Transonic cases have bump with %10 thickness and supersonic case has %2.

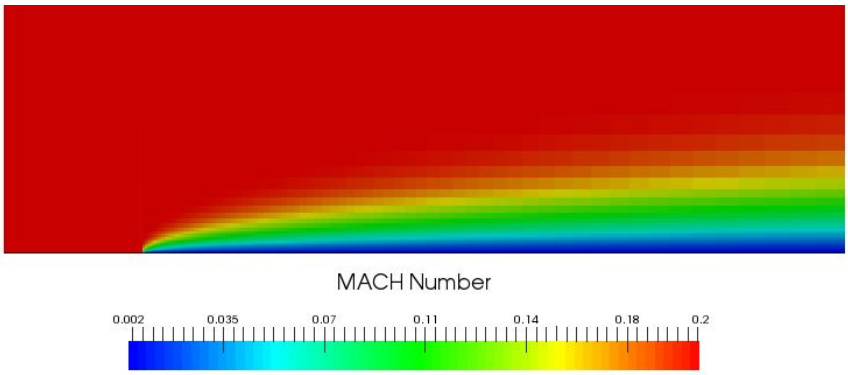



Figure 2: Scaled picture of the solution of laminar flat plate case

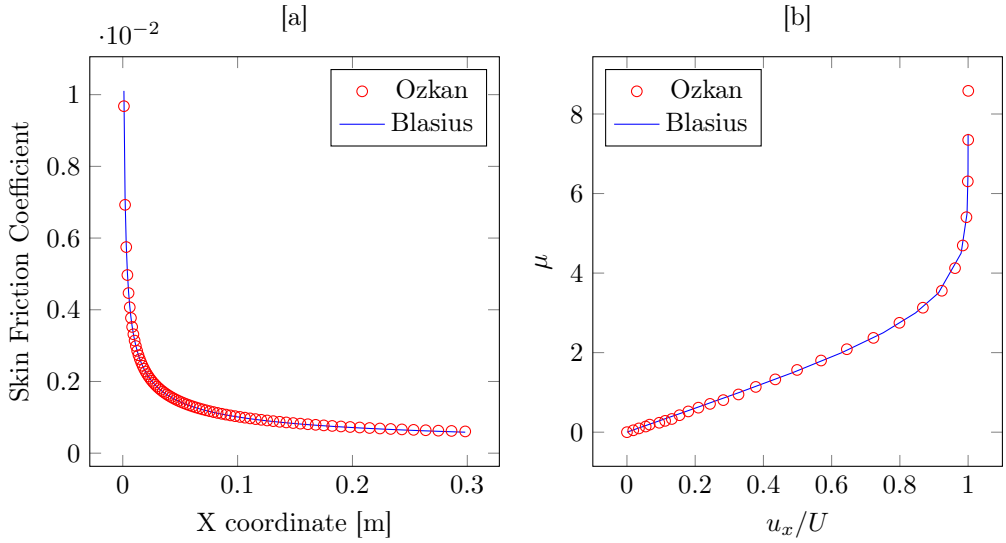



Figure 3: Skin Friction Coefficients (a) and Velocity on boundary layer (b) in laminar flat plate case

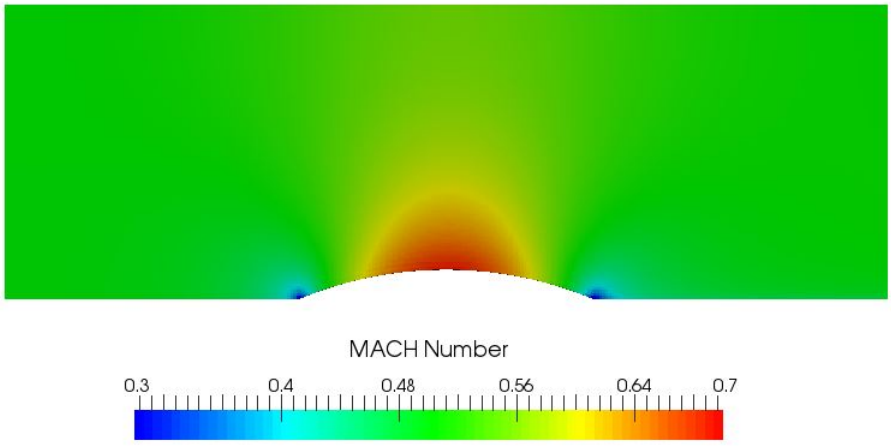



Figure 4: MACH number field of subsonic flow over %10 thickness circular arc bump

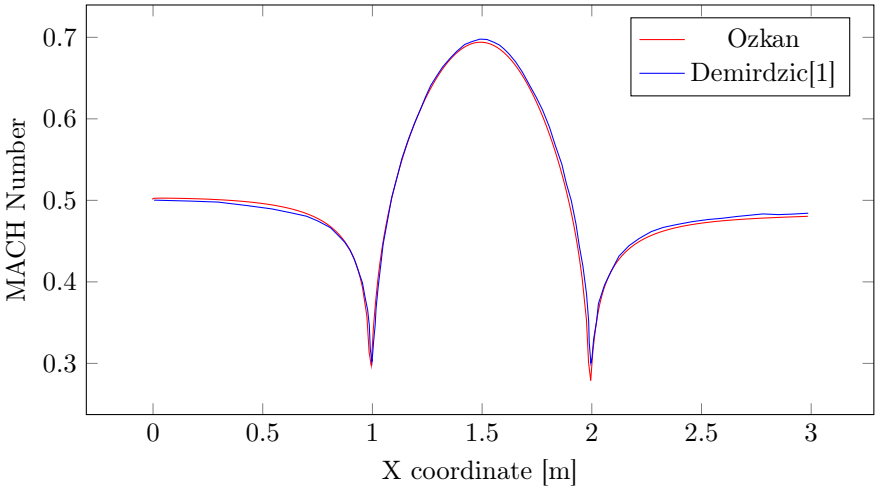



Figure 5: MACH number plot of subsonic flow over %10 thickness circular arc bump

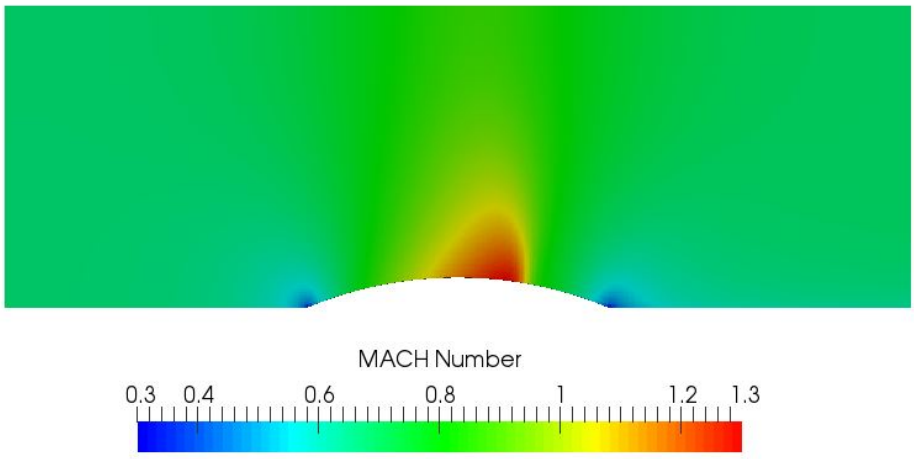



Figure 6: MACH number field of transonic flow over %10 thickness circular arc bump

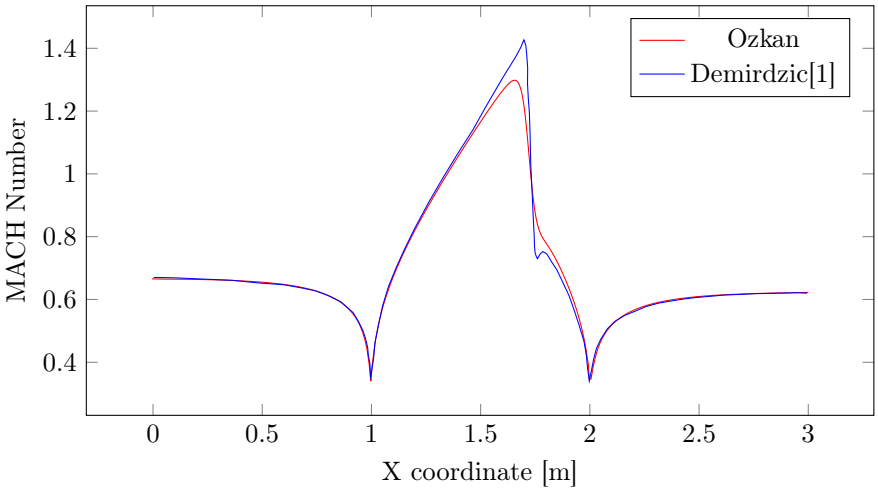



Figure 7: MACH number plot of transonic flow over %10 thickness circular arc bump

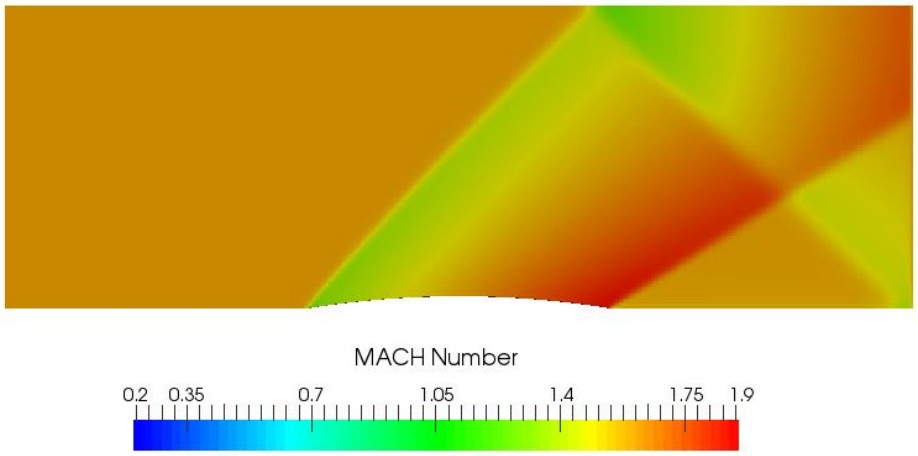



Figure 8: MACH number field of supersonic flow over %4 thickness circular arc bump

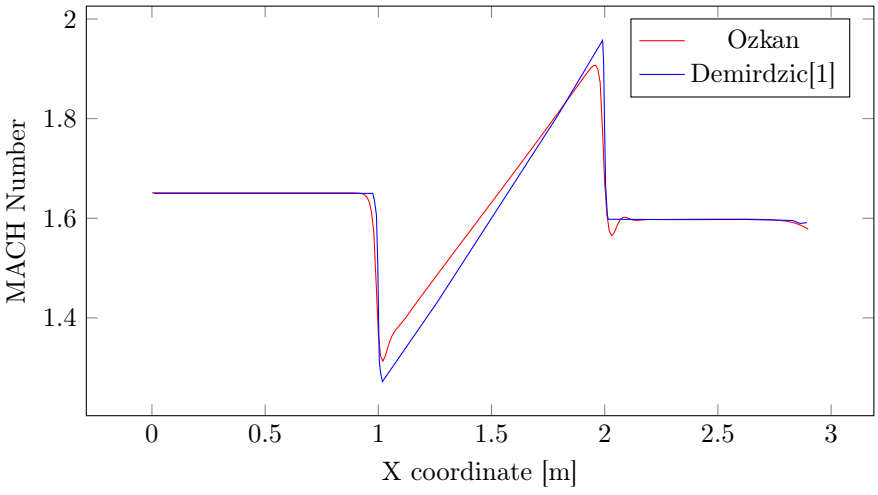



Figure 9: MACH number plot of supersonic flow over %4 thickness circular arc bump

Sod's Shock Tube problem [10] is studied to validate time integration schemes in the Solver. In Shock Tube problem a tube with size of 1 ft x 0.1 ft has two different sides with ten times different pressures and these sides are separated with a membrane. At t equals zero the membrane brakes and gas from high pressure zone starts to flow to low pressure zone. Comparison between Solver and Analytical solution in Sod's Shock Tube problem can be seen in Figure 10. Every boundary of this problem is applied as slip wall.

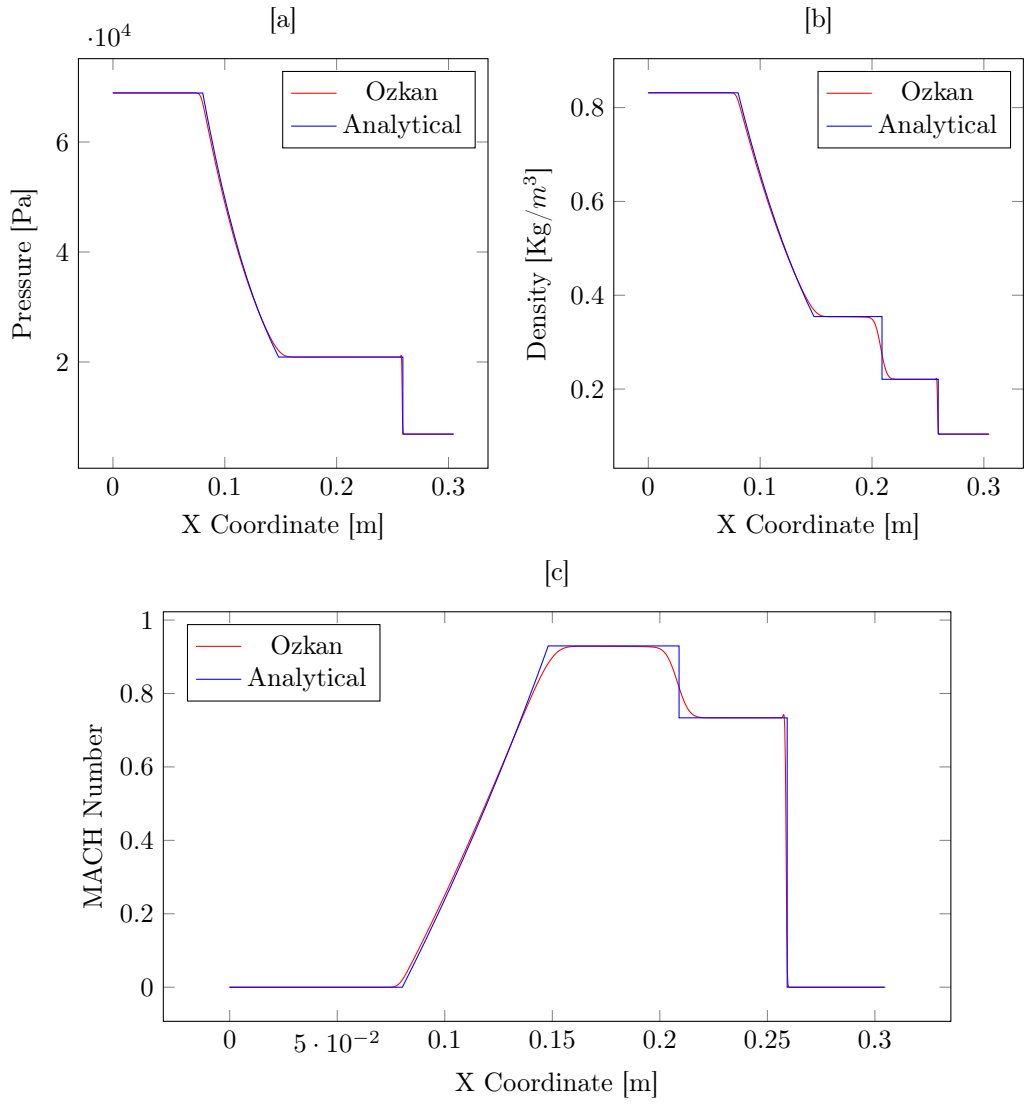
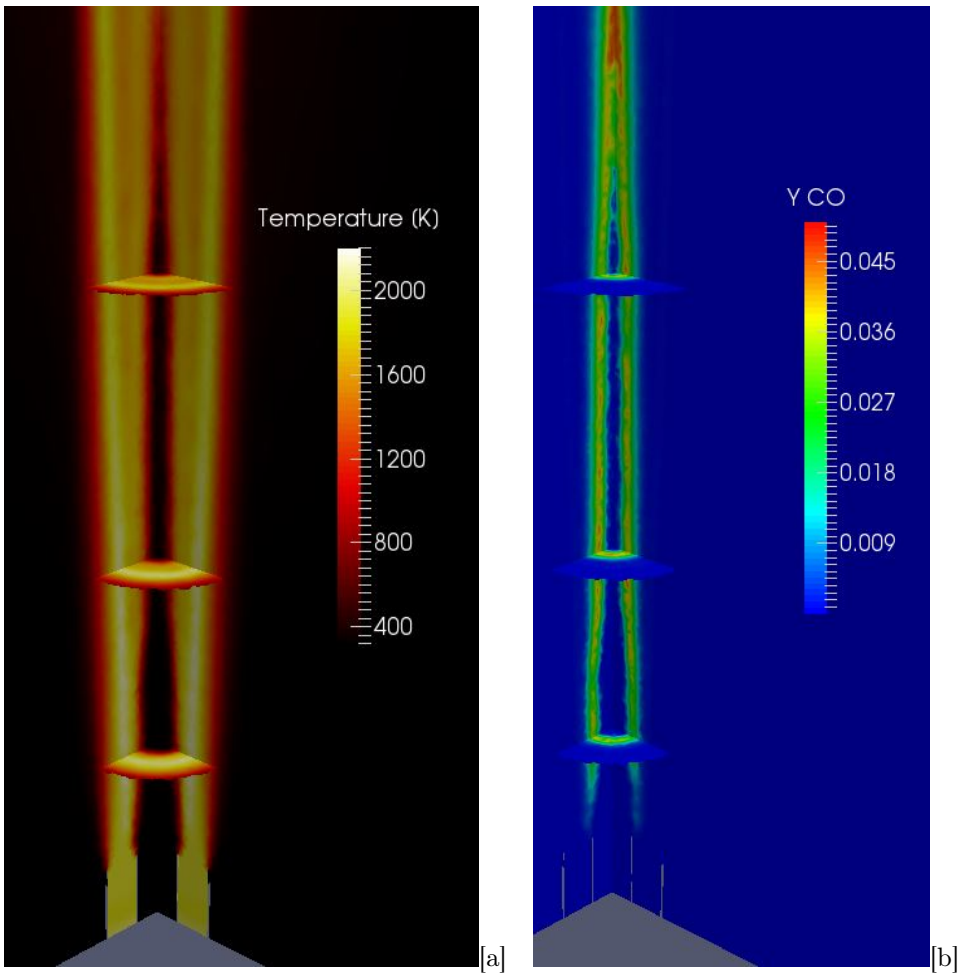



Figure 10: Comparison of Pressure (a), Density (b) and Mach Number (b) between solver and analytical solution at last timestep in Sod's Shock Tube Problem.

Flame D test case from Sandia [11] is studied to validate chemistry and multi species flow capabilities of the Solver. Flame D case is a piloted flame case. A %75 air and %25 Methane mixture entering the test chamber with velocity of 49.6 m/s velocity and 294 kelvin. This main gas mixture is burned with 1880 kelvin pilot with phi value of 0.77. This test case is run in three dimensions with 6 species and 2 reactions.

Reduced reaction mechanism used in this simulation is taken from the paper from Franzelli [12]. Temperature and carbon monoxide mass fraction fields of this simulation can be seen in Figure 11. Comparison of the Solver results and experimental data is in Figure 12.




[a] [b]

Figure 11: Temperature (a) and CO mass fraction (b) fields of Sandia flame D

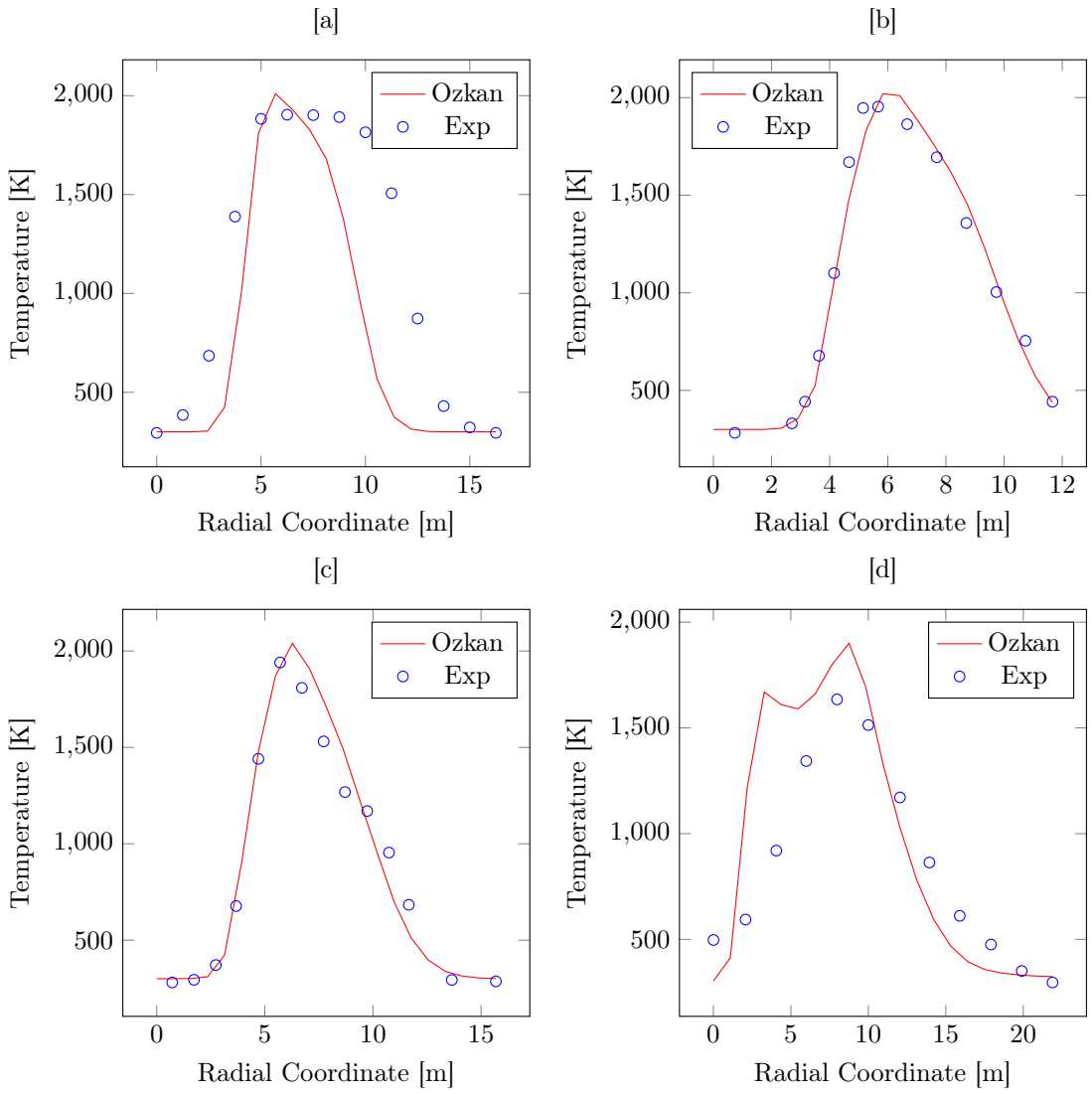



Figure 12: Temperatures of x = 7.2 mm (a) , x = 14.4 mm (b) , x = 21.6 mm (c) , x = 108 mm (d)

# 5 Performance

In this section, computing performance of the solver is studied. Performance of the loop parallelism with CPU using OpenMP is performed using cloud computing. EC2 service from Amazon Web Services [13] used to perform this study. Two 3 GHz, Intel Xeon Platinum 8124M hardware used in ubuntu server operating system for this study. Results of this study can be seen in Figure 13. It can be seen that performance is degrading after code starting to use hyperthreaded cores simultaneously.

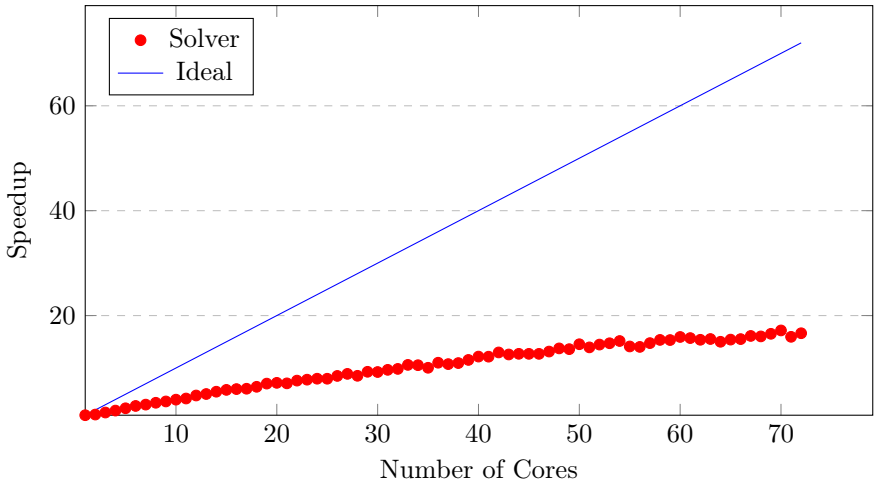



Figure 13: Speedup values for iteration time without matrix solving on only CPU parallel simulation

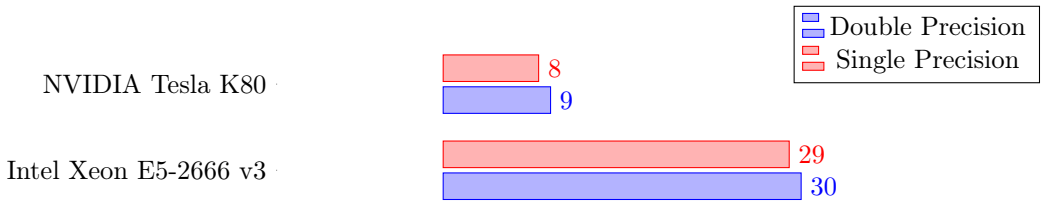



Figure 14: Matrix solving times on CPU and GPU hardwares [Seconds].

Performance of the GPU acceleration is performed also on cloud computing. EC2 service from Amazon Web Services [13] used to perform this study. Matrix solving performance of 2.90GHz Intel Xeon E5-2666 v3 CPU is compared with the performance of NVIDIA Tesla K80 GPU. Although two hardware is similarly priced, performance of the GPU was 3.5 times greater than the CPU in both single precision and double precision including data transfer times. Results of the GPU performance case can be seen in Figure 14.

# 6 Conclusions

A solver with multi species and multi reactions capabilities is introduced. This solver is also pressure based and can perform at both compressible and incompressible flows. Numerical schemes and CFD methodologies used in this solver is explained. Computational algorithms used in the solver is detailed.A state of the art open source matrix solver is implemented to the solver.

Laminar Flat plate problem is studied to validate viscous flux schemes. Subsonic, Transonic and Supersonic over Bump in a Channel problems have been studied to validate Pressure-Velocity Coupling and convective flux schemes. Sod's shock tube study is performed to validate time integration performance of the solver. Flame D experimental case from Sandia Laboratories have been studied to validate solvers capabilities on reacting and turbulent flows with multiple species.

Performance evaluation of both solver and ViennaCL is performed. It is concluded that between similar priced hardwares, being GPU accelerated gave the solver 3.5 times performance increase.

For future work, a second order version of the solver is planning to develop and more test case studies is planned to estimate solver capabilities.

# References


[1] I. Demirdzic, Z. Lilek, and M. Peric. A collocated finite volume method for predicting flows at all speeds. *International Journal for Numerical Methods in Fluids*, 2013.

[2] J. Weinbub. K. Rupp, F. Rudolf. ViennaCL - A High Level Linear Algebra Library for GPUs and Multi-Core CPUs. *Conference: Proceedings of the International Workshop on GPUs and Scientific Applications (GPUScA)*, 2010.

[3] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics The Finite Volume Method Second Edition.* Pearson, 2007.

[4] L. Dagum D. Kohr D. Maydan J. McDonald R. Chandra, R. Menon. *Parallel Programming in OpenMP 1st Edition.* Morgan Kaufmann, 2000.

[5] NVIDIA Corp. Cuda Toolkit Documantation v9.1.85. [https://docs.nvidia.com/cuda/], 2018.

[6] E. Stiefel M. R. Hestenes. Methods of Conjugate Gradients for Solving Linear Systems. *Journal of Research of the National Bureau of Standards Vol. 49, No.6, Research Paper 2379*, 1952.

[7] J. Weinbub. M. Wagner, K. Rupp. A Comparison of Algebraic Multigrid Preconditioners using Graphics Processing Units and Multi-Core Central Processing Units. *HPC '12 Proceedings of the 2012 Symposium on High Performance Computing*, 2012.

[8] J. H. Ferziger and M. Peric. *Computational Methods For Fluid Dynamics.* Springer, 1997.

[9] S. Gordon B. J. McBride and M. A. Reno. Coefficients for Calculating Thermodynamic and Transport Properties of Individual Species. *NASA Technical Memotandum 4513*, 1993.

[10] G. A. Sod. A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws. *J. Comput. Phys 27*, 1978.

[11] J. Frank R. Barlow. Piloted CH4/Air Flames C, D, E, and F âĂŞ Release 2.1. *Sandia National Laboratories*, 2007.

[12] L.Y.M. Giquel T. Poinsot B. Franzelli, E. Riber. Large-Eddy Simulation of combustion instabilities in a lean partially premixed swirled flame. *Elsevier*, 2011.

[13] Amazon. Amazon AWS EC2. [https://aws.amazon.com/ec2/], 2018.