APCOM & ISCM
11-14th December, 2013, Singapore

An implicit multigrid solver for high-order compressible flow simulations on GPUs

*V. Esfahanian[1,2], M. Hedayat[2], B. Baghapour[2], M. Torabzadeh[2] and S.J. Hosseini[2]
[1]Vehicle, Fuel and Environment Research Institute, University of Tehran, Iran
[2]School of Mechanical Engineering, University of Tehran, Iran

*Corresponding author: e.vahid@ut.ac.ir

Abstract

The multigrid method has proved to be effective for a large class of numerical methods. In this study, a strategy based on Full Approximation Storage (FAS) scheme is implemented together with Full Multigrid Algorithm (FMG) to accelerate convergence of steady state solutions of the two-dimensional compressible Euler equations on Graphics Processing Unit (GPU). The Beam and Warming linearization scheme in curvilinear coordinates is used to discretize the governing equation. The second-order central and the fourth-order compact finite-difference schemes are applied for spatial discretization. A high-performance GPU-implemented block-tridiagonal solver based on Block Cyclic Reduction (BCR) algorithm is utilized. The proposed BCR solver is applied to finite-difference discretization on structured grids via Alternating Direction Implicit (ADI) scheme. Attention is directed towards the computational performance of the V-cycle and W-cycle multigrid strategies in two and three grid levels using the NVIDIA GTX480 graphics card. Speedups between 2x–6.2x are achieved in comparison to the Intel Core i7-920 2.67GHz CPU for different grid sizes.

Keywords: Euler Equations, high-order method, Multigrid Acceleration, GPU computing, Block-tridiagonal solver

Introduction

Among the different schemes that are proposed to accelerate convergence rate such as local time stepping, residual smoothing, multigrid method and most recently local preconditioning, the multigrid method has received great attention. In spite of time stepping schemes that efficiently damp high-frequency error components of numerical solutions, the multigrid method can accelerate the convergence rate by damping the low-frequency error components. This method was first introduced by Fedorenko (1962) and Bakhvalov (1966) and then developed by Brandt (1977). Although the multigrid theory was first developed for elliptic problems, it has been demonstrated in later works by (Ni, 1982; Jameson, 1983) that multigrid method can also greatly affect the convergence rate of numerical schemes applied to Euler equations. Recently, there have been some studies to implement the multigrid methods on high-order numerical discretization (Gupta, Kouatchou, & Zhang 1997; Sakurai, Aoki, Lee, & Kato 2002).

The multigrid method has also been developed for the implicit schemes, which enabled the utilization of the ADI methods (Jameson and Yoon, 1986). In this matter, the computational cost of block-structure matrix inversions is expensive. Therefore, accelerating the linear system solution is an effective approach towards the computational performance improvement.

Recent developments on Graphics Processing Units (GPU) have led to a high computing power device with a GPU-oriented programming language (CUDA), which enables the applicability of GPU devices for accelerating a variety of CFD problems.

To accelerate the solution of block tridiagonal matrices, appeared in the ADI discretization of the multigrid method, on GPUs, the parallel Cyclic Reduction (CR) algorithm is at the center of attention. This algorithm was developed for block tridiagonal systems by Heller (1976). Performance of high-rank block cyclic reduction solver on distributed memory CPU cluster was reported in work by (Hirshmman, Perumalla, Lynch, & Sánchez, 2010). A recent implementation of BCR algorithm on GPUs was done by (Stone, Duque, Zhang, Car, Owens, & Davis, 2011; Baghapour, Esfahanian, Torabzadeh, Mahmoodi Darian, 2013) for different CFD applications.

In the current work, a multigrid method based on FAS scheme is implemented beside a FMG scheme (Brandt, 1981) to accelerate convergence of steady state solutions of the two-dimensional compressible Euler equations. Alternate Directional Implicit (ADI) method is used for time advancement and the second-order central and fourth-order compact finite-difference schemes are applied to spatial discretization. The computational performance of the V-cycle and W-cycle multigrid strategies in two and three grid levels is investigated by using the NVIDIA GTX480 graphics card.

Governing Equations

The two-dimensional Euler equations for a Cartesian coordinate system are given by:

$$\frac{\partial Q}{\partial t}+\frac{\partial F}{\partial \xi}+\frac{\partial G}{\partial \eta}=0 \tag{1}$$

where,

$$Q=J^{-1}\begin{Bmatrix}\rho\\ \rho u\\ \rho v\\ E\end{Bmatrix},\ F=J^{-1}\begin{Bmatrix}\rho U\\ \rho uU+\xi_x p\\ \rho vU+\xi_y p\\ (E+p)U\end{Bmatrix},\qquad G=J^{-1}\begin{Bmatrix}\rho V\\ \rho uV+\eta_x p\\ \rho uV+\eta_y p\\ (E+P)V\end{Bmatrix} \tag{2}$$

and

$$U=\xi_x u+\xi_y v\ ,\qquad V=\eta_x u+\eta_y v\ ,\qquad J^{-1}=\left(x_{\xi}y_{\eta}-x_{\eta}y_{\xi}\right) \tag{3}$$

The variables p,ρ,T,u,v, and E are pressure, density, temperature, velocity components and total energy, respectively. Assuming air as an ideal gas, the equation of state is used to calculate the pressure and temperature:

$$p = (\gamma - 1)\rho\left[E - \frac{u^2 + v^2}{2}\right], \qquad T = \frac{p}{\rho R} \tag{4}$$

where γ is the ratio of specific heats and R is the gas constant.

Numerical Method

Spatial Discretization

For spatial discretization, the second-order central and the fourth-order compact finite difference schemes are applied. A general centered compact scheme (Lele, 1992) for the first derivative is the following:

$$\beta f'_{i-2} + \alpha f'_{i-1} + f'_i + \alpha f'_{i+1} + \beta f'_{i+2} = c\frac{f_{i+3} - f_{i-3}}{6\Delta x} + b\frac{f_{i+2} - f_{i-2}}{4\Delta x} + a\frac{f_{i+1} - f_{i-1}}{2\Delta x} \tag{5}$$

Herein, f and f' can be any flow variable and its derivative, respectively. Δx is the grid spacing in x-direction. Depending on the coefficients a and b, a tridiagonal or a pentadiagonal system will be produced. The compact method used in this study is as following:

$$f'_{i+1} + 4f' + f'_{i-1} = \frac{3}{\Delta x}(f_{i+1} - f_{i-1}) \tag{6}$$

Multigird Method

The FAS scheme for two grid levels with multigrid V-cycle is described as follows, where the fine grid is denoted by the h subscript.

1) Calculate the solution correction (ΔQ_h) on the fine grid and update the solution on the fine grid: $Q_h^{n+1} = Q_h^n + \Delta Q_h$
2) Calculate the fine grid residual: R_h
3) Transfer value of conservative variables to the coarse grid: $q_{2h} = I_h^{2h} q_h$
   where $q = JQ$ and I_h^{2h} is the restriction operator (Wesseling, 1995).
4) Collect the residuals on the fine grid for the coarse grid: $R_{2h}^t = I_h^{2h} R_h$
5) Calculate the residuals on the coarse grid using restricted conservative variables from the fine grid to the coarse grid: $R_{2h}\left(Q_{2h}^{(0)}\right)$. To restrict flow variable from the fine grid to the coarse grid, a full-weighted restriction is used.
6) The forcing function is defined by: $P_{2h} = R_{2h}^t - R_{2h}\left(Q_{2h}^{(0)}\right)$
   where, R_{2h}^t is the residual that is restricted to the coarse grid ($2h$) and $R_{2h}\left(Q_{2h}^{(0)}\right)$ is the residual evaluated by restricting $Q_{2h}^{(0)}$ to the coarse grid from the fine grid.
7) The residual on the coarse grid is defined as: $R_{2h} = P_{2h} + R_{2h}\left(Q_{2h}^{(0)}\right)$

8) After one or several iterations, the solution on the coarse grid is updated. Then, the solution correction on the coarse grid is calculated as : $Q_{2h}^{n+1} = Q_{2h}^{n} + \Delta Q_{2h}$
9) In order to update the solution on the fine grid (h), the solution corrections are prolonged to the fine grid as follows: $Q_{h}^{n+1} = Q_{h}^{n} + I_{2h}^{h}\left(Q_{2h}^{n+1} - Q_{2h}^{(0)}\right)$

   where, Q_{h}^{n} is the solution before the restriction to the coarse grid and I_{2h}^{h} is the interpolation operator.

Accuracy of Transfer Operator

Orders of prolongation and restriction operators should satisfy the following equation in order to damp the frequency error (Hemker, 1990):

$$\mathrm{m}_{\mathrm{Res}} + \mathrm{m}_{\mathrm{Pro}} > \mathrm{m}_{\mathrm{Eqn}} \tag{7}$$

where, $\mathrm{m}_{\mathrm{Res}}$, $\mathrm{m}_{\mathrm{Pro}}$ and $\mathrm{m}_{\mathrm{Eqn}}$ denote the order of restriction, prolongation operator and the order of spatial numerical discretization for the system of equations, respectively. A full-weighting restriction and fourth-order compact interpolation (Lele, 1992) that are used in this work will satisfy Eq. (7).

BCR algorithm and GPU implementation

CUDA architecture and GPU memory hierarchy

In CUDA architecture, a kernel execution is divided among a batch of concurrent threads, which are partitioned in blocks of threads. The blocks are mapped to the stream multiprocessors (SM) of the GPU device for execution. The concurrent threads access data from different memory resources when executing on the GPU. Each thread within a group of blocks has access to the large global memory (Equal to the DRAM of the GPU) with high transaction latency. All threads have low latency access to 48KB of shared memory for thread communication within a block (CUDA, 2012). The NVIDIA GTX 480, used in this research, has 1536 MB of frame buffer global memory runs through a 384-bit bus and delivers 177.4 GB/s of memory bandwidth.

The parallel Block Cyclic Reduction (BCR) algorithm

The BCR algorithm is based on divide and conquer strategy. In the first level, the primary $N \times N$ system of equations is divided into two decoupled $N/2 \times N/2$ sub matrices. The two smaller sub matrices are consecutively divided again in the same way until the total number of $N/2$ sub matrices of size 2×2 are achieved in the last level, which can be solved in parallel. In each level of the BCR algorithm, the diagonals (L, M, U) and the right-hand-side (D) are calculated by Eq. (8). Accordding to this above calculations consist of many small matrix multiplication and inversion operations. The basic approach to implement the above matrix operations on the GPU is to use different streams for parallel kernel execution (CUDA, 2012). However, this approach is bounded by 16 maximum concurrently running streams in CUDA architecture and cannot lead to performance improvements in large matrices. In order to benefit from the parallel nature of BCR algorithm together with the high computational throughput of the

GPU device, single GPU kernels are developed to perform all matrix multiplication and inversion operations in parallel.

In both matrix kernels, the sub matrices are first loaded from the global memory to the shared memory, which is provided for each block of threads. Then the calculations on the sub matrices are obtained and the results are written back to the global memory. The number of offloading sub matrices to the shared memory of each block is optimized to avoid extra memory allocation than the 48KB limit and achieve the maximum performance on GPU.

$$\begin{aligned}
T_1^{level} &= L_i^{level}\left(M_{i-1}^{level}\right)^{-1}, T_2^{level} = U_i^{level}\left(M_{i+1}^{level}\right)^{-1}\\
R_i^{level+1} &= R_i^{level} - T_1^{level} R_{i-1}^{level} - T_2^{level} R_{i+1}^{level}\\
M_i^{level+1} &= M_i^{level} - T_1^{level} U_{i-1}^{level} - T_2^{level} L_{i+1}^{level}\\
U_i^{level+1} &= -T_2^{level} U_{i+1}^{level}\\
L_i^{level+1} &= -T_1^{level} L_{i-1}^{level}
\end{aligned} \tag{8}$$

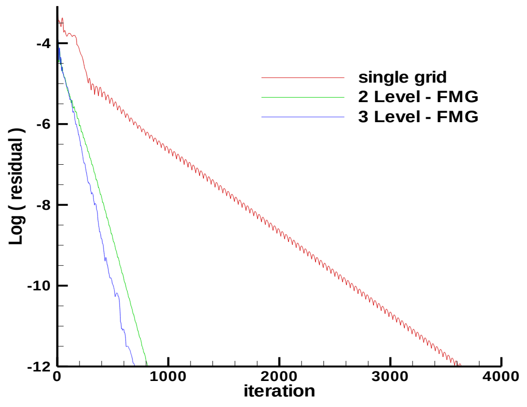
Results

The double precision numerical results presented in this section demonstrate the accuracy and computational efficiency of the multigrid method for inviscid flows. The performance of the GPU-based multigrid solver with the implicit scheme is studied for two-dimensional circular arc bump. The thickness-to-chord ratio is 10% based on the standard test cases of the GAMM conference (Rizzi and Viviand, 1981). In the case of computations on two grid levels and three grid levels, the FMG method is applied to provide an initial solution for the fine grid. The CFL number of 0.6 is used on all grids so that larger time steps can be used on the coarse grid. Convergence is monitored using L_2 norm of density residual. The performance of computations on GTX480 GPU (Core Clock rate: 700 MHz) is compared with a single core of Intel Core i7-920 2.67GHz CPU.

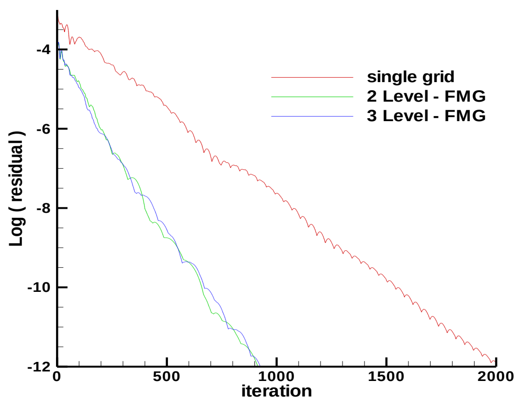
Multigrid performance

In first step, the performance of multigrid method for computational grid including 32×16 cells at free stream Mach number for subsonic and transonic flows using second-order finite difference method is presented. Figure 3(a) and Fig. 3(b) compares the convergence rates between a single grid level, two grid levels and three grid levels multigrid with FMG for subsonic and transonic flow. Computations are performed using V-cycle procedure.

In the second step, the performance of multigrid method for the fourth-order compact method on the same geometery for both subsonic and transonic flows is investigated. Figure 4(a) and Fig. 4(b) demonstrate the convergence rates between a single grid level, two grid levels and three grid levels multigrid with FMG for subsonic and transonic flow. Computations are performed using V-cycle procedure. It is worth mentioning that no significant difference between convergence rates of V-cycle and W-cycle computations is observed.

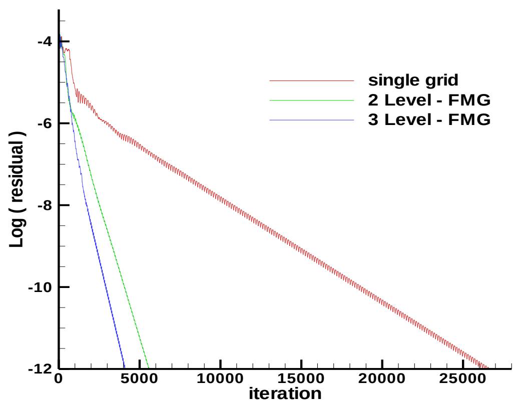


(a) Subsonic flow M_∞=0.5

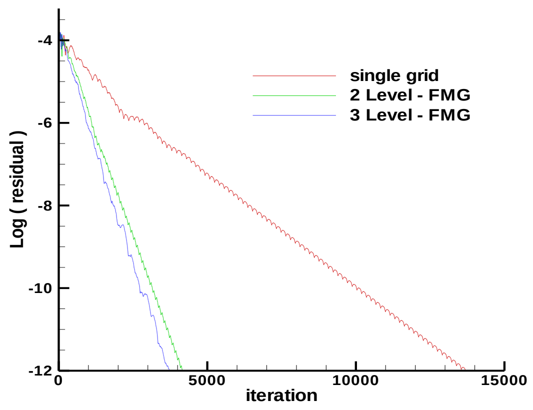


(b) Transonic flow, M_∞=0.675

Figure 3. Convergence acceleration for the second-order scheme



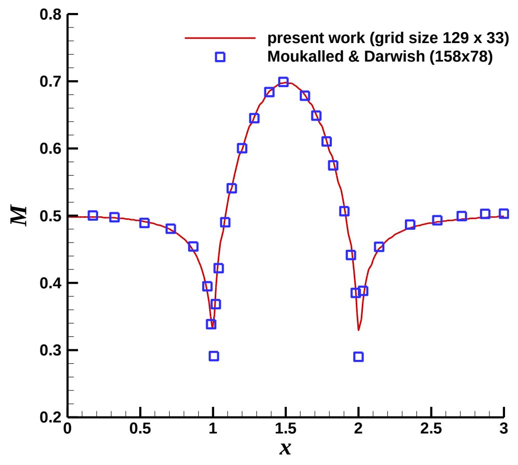
(a) Subsonic flow, M_∞=0.5



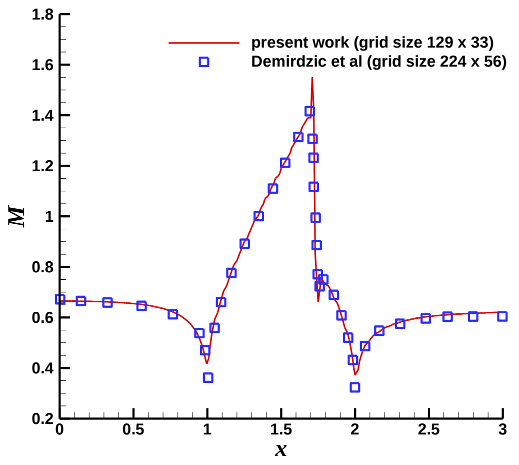
(b) Transonic flow, M_∞=0.675

Figure 4. Convergence Acceleration for the fourth-order scheme

In Fig. 5(a) and Fig. 5(b) Mach number distribution of the present study compared to (Demirdžić, Lilek, & Perić, 1993). Figure 6(a) and Fig. 6(b) demonstrate the pressure contour for subsonic and transonic flow, respectively.

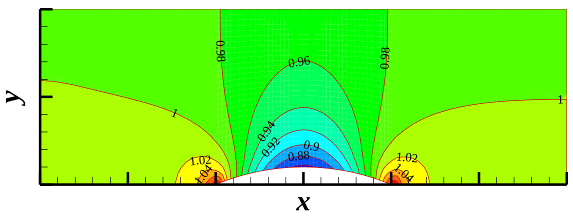


(a) Subsonic flow, M_∞=0.5

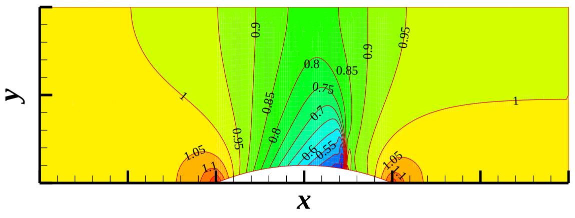


(b) Transonic flow, M_∞=0.675

Figure 5. The Mach number distribution for lower wall of the circular arc bump



(a) Subsonic flow, M_∞=0.5

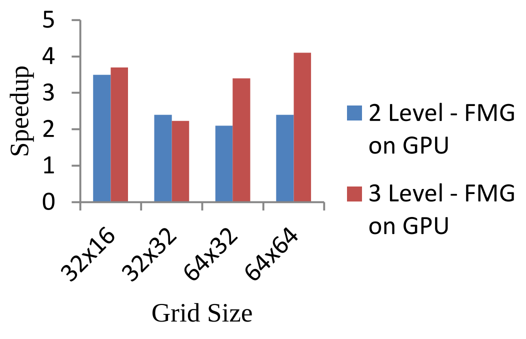


(b) Transonic flow, M_∞=0.675

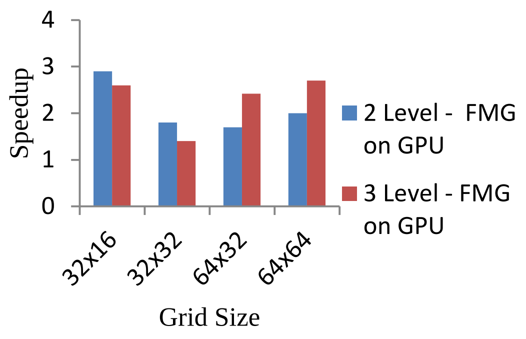
Figure 6. The pressure contours over the circular arc bump

Total performance

The total performance of the CFD code on the GPU, including the multigrid kernels and the BCR linear solver, is studied. Figure 7(a) and Fig. 7(b) show the overall GPU speedup of the two-dimensional subsonic and transonic flow test cases compared to the CPU platform for the second-order finite difference method**.** Figure 8(a) and Fig. 8(b) compare the solver speedup for the fourth-order compact finite difference method for $M_\infty = 0.5$ and $M_\infty = 0.675$, respectively. Speedup term here is defined as the run time ratio obtained by single grid CPU- solver over the multigrid GPU-solver.

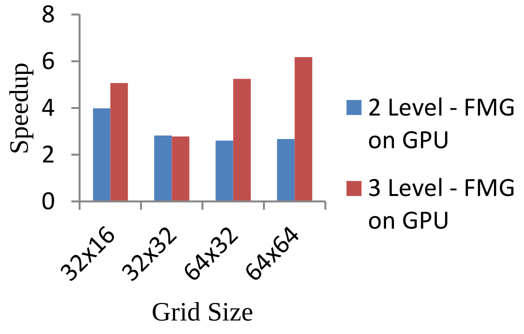


(a) Subsonic flow, M_∞=0.5

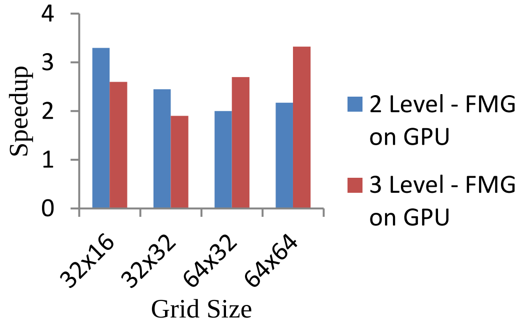


(b) Transonic flow, M_∞=0.675

Figure 7. Total speed up for the second-order scheme on GPU



(a) Subsonic flow, M_∞=0.5



(b) Transonic flow, M_∞=0.675

Figure 8. Total speed up for the fourth-order scheme on GPU

CONCLUSIONS

A multigrid scheme is implemented on a two-dimensional Euler solver in order to accelerate convergence to steady state. Subsonic and transonic inviscid flows in a channel with circular arc bump for both second-order central and fourth-order compact finite-difference methods are investigated as the test cases. The convergence acceleration obtained for Fourth-order method is more than convergence acceleration for the second-order method. Moreover, the higher convergence rate for subsonic flow is observed in comparison to transonic flow. Total speedups between 2x–6.2x are achieved for different mesh sizes by implementing the GPU-based BCR solver in comparison to computations on the CPU platform.

ACKNOWLEDGEMENTS

The authors would like to thank the Vehicle, Fuel and Environment Research Institute (VFERI) of University of Tehran for general support during this study.

References

Baghapour B., Esfahanian V., Torabzadeh M., Mahmoodi Darian H., (2013), A discontinuous Galerkin method with block cyclic reduction solver for simulating compressible flows on GPUs. *Submitted to Mathematics and Computers in Simulation*, MATCOM-S-13-00553.

Bakhvalov, N. S. (1966), On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR* Journal of *Computational Mathematics and Mathematical Physics*, 6(5), 101-135.

Brandt, A. (1977), Multi-level adaptive solutions to boundary-value. *problems. Mathematics of computation*, 31(138), 333-390.

Brandt, A. (1981), Guide to Multigrid Development, Multigrid Methods I. *Lecture Notes in Mathematics,* 960, Springer Verlag.

CUDA, C. (2012). Programming guide. NVIDIA Corporation.

Demirdžić, I., Lilek, Ž., & Perić, M. (1993), A collocated finite volume method for predicting flows at all speeds. *International Journal for Numerical Methods in Fluids*, 16(12), 1029-1050.

Fedorenko, R. P. (1962), A relaxation method for solving elliptic difference equations. *USSR Journal of Computational Mathematics and Mathematical Physics,* 1(4), 1092-1096.

Gupta, M. M., Kouatchou, J. & Zhang, J. (1997), Comparison of second-and fourth-order discretizations for multigrid Poisson solvers. *Journal of Computational Physics*, 132(2), 226-232.

Heller, D. (1976), Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems. *SIAM Journal on Numerical Analysis*, 13(4), 484-496.

Hemker, P.W. (1990), On the Order of Prolongations and Restrictions in Multigrid Procedures. *Journal of Computational and Applied Mathematics*, 32, pp. 423-429.

Hirshmman, S. P., Perumalla, K. S., Lynch, V. E., & Sánchez, R. (2010), BCYCLIC: A parallel block tridiagonal matrix cyclic solver. *Computational Physics*, 229(18), 6392-6404.

Jameson, A. and Yoon, S. (1986), Multigrid Solutions of the Euler Equations using Implicit Schemes*," AIAA J., 24, pp. 1737-1743.*

Jameson, A.(1983), Solution of the Euler Equations for Two-dimensional, Transonic Flow by a Multigrid Method. *Journal of Applied Mathematics and Computation*, 13, pp.327-356.

Lele S. K. (1992), Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics* ;103:16–42.

Ni, R.H. (1982), A multiple grid scheme for solving the Euler equations, *A1AA J. 20,* 1565-1571.

Rizzi A. and Viviand H. (eds) (1981), Numerical methods for the computation of inviscid transonic flows with shock waves. *A GAMM workshop, in Notes on Numerical Fluid Mechanics Vieweg,* Braunschweig;

Sakurai, K., Aoki, T., Lee, W. H., & Kato, K. (2002), Poisson equation solver with fourth-order accuracy by using interpolated differential operator scheme. *Computers & Mathematics with Applications*, 43(6), 621-630.

Stone, C. P., Duque, E. P., Zhang, Y., Car, D., Owens, J. D., & Davis, R. L. (2011). GPGPU parallel algorithms for structured-grid CFD codes. *InProceedings of the 20th AIAA Computational Fluid Dynamics Conference* (Vol. 3221).

Wesseling, P. (1995), Introduction To Multigrid Methods (No. ICASE-95-11). *Institute for Computer Application in Science and Engineering Hampton VA*.