Compact-Reconstruction WENO Methods for Compressible Flows

Kilian Cooley
kcooley2@math.umd.edu
Applied Mathematics, Statistics, and Scientific Computing
University of Maryland, College Park

Advisor
Dr. James Baeder
baeder@umd.edu
Department of Aerospace Engineering
University of Maryland, College Park

October 1, 2014

Abstract
This project develops a program that solves the Euler equations of compressible
flow in two dimensions using a Compact-Reconstruction Weighted Essentially Non-
Oscillatory (CRWENO) scheme, modified to apply to grids with unequal spacing. The
code will operate in parallel to reconstruct quantities along cell lines of a fixed Cartesian
grid, and will be validated against several standard compressible flow problems.

1 Background
1.1 The Euler Equations
We seek solutions of the Euler equations which can be written as
\[
\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0
\]
\[
\frac{\partial \rho u_j}{\partial t} + \frac{\partial \rho u_i u_j}{\partial x_i} = -\frac{\partial p}{\partial x_j}
\]
\[
\frac{\partial e}{\partial t} + \frac{\partial (e + p) u_i}{\partial x_i} = 0
\]
(1)

Where \( \rho \) is the density, \( u_i \) the Cartesian components of velocity, \( p \) the pressure, and \( e \) the internal energy. We adopt the convention that repeated indices in a single additive term
indicate summation over the values of that index.
To close this system requires an equation of state that relates $e$ to the other flow variables. For a perfect gas, that equation is

$$e = \frac{p}{\gamma - 1} + \frac{1}{2} \rho u_i u_i$$  

($2$)

$\gamma$ is the gas-dependent ratio of specific heats whose value is 1.4 for air. These equations represent the conservation of mass, momentum, and energy of the fluid and can be cast as a system of hyperbolic conservation laws:

$$\frac{\partial q}{\partial t} + \frac{\partial F_i}{\partial x_i} = 0$$  

($3$)

Where

$$q = \begin{bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ e \end{bmatrix}, \quad F_1 = \begin{bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ (e + p) u_1 \end{bmatrix}, \quad F_2 = \begin{bmatrix} \rho u_2 \\ \rho u_1 u_2 \\ \rho u_2^2 + p \\ (e + p) u_2 \end{bmatrix}$$

### 1.2 The Finite Volume Framework

To make the solution of (3) tractable by computer, we discretize the spatial domain into disjoint cells. Averaging (3) over one such cell $\Omega_k$ and applying the divergence theorem:

$$0 = \frac{1}{|\Omega_k|} \int_{\Omega_k} \frac{dq}{dt} dx + \frac{1}{|\Omega_k|} \int_{\Omega_k} \frac{\partial F_i}{\partial x_i} dx$$

$$\frac{d}{dt} \bar{q}_k = -\frac{1}{|\Omega_k|} \int_{\partial \Omega_k} F_i n_i ds$$  

($4$)

Where $n_i$ are the components of the unit normal vector pointing out of $\Omega_k$ and we have defined

$$\bar{q}_k = \frac{1}{|\Omega_k|} \int_{\Omega_k} q dx$$

The quantities $\bar{q}_k$ are the unknowns to be found. This formulation requires solving the fundamental problem of reconstructing $F$ (or $q$ from which to calculate $F$) on the boundary of the cell to evaluate the line integral in (4) from knowledge only of the averages of $q$ over the cell interior. With this accomplished (4) becomes an ODE for the $q_k$ that can be solved by any time-marching method. The CRWENO method to be described is one of many approaches to this reconstruction problem.

### 1.3 The CRWENO scheme

Consider a one-dimensional domain discretized into cells of equal length $h$. Let $u_j$ be the average of some quantity $u$ over cell $j$. The reconstruction problem is to find $\hat{u}_{j+1/2}$, approximations to the point values of $u$ at the cell interfaces. Three schemes to accomplish this task are shown below:

$$\frac{2}{3} \hat{u}_{j-1/2} + \frac{1}{3} \hat{u}_{j+1/2} = \frac{1}{6} u_{j-1} + \frac{5}{6} u_j + \mathcal{O}(h^3)$$  

($5$)
Each of these schemes is compact, meaning that the unknowns on the left-hand sides are coupled and a linear system must be solved to obtain the interface values. A particular convex combination of (5)-(7) gives a single fifth-order scheme:

\[
\frac{1}{5} \text{Eq. (5)} + \frac{1}{2} \text{Eq. (6)} + \frac{3}{10} \text{Eq. (7)} = \frac{3}{10} \hat{u}_{j-1/2} + \frac{3}{5} \hat{u}_{j+1/2} + \frac{1}{10} \hat{u}_{j+3/2} = \frac{1}{30} u_{j-1} + \frac{19}{30} u_j + \frac{1}{3} u_{j+1} + O(h^5)
\]

The coefficients in this combination are referred to as optimal weights. The CRWENO scheme replaces these optimal weights with solution-dependent weights \(\omega_i, i = 1, 2, 3\). They are chosen so that they become essentially zero when the corresponding stencil contains a discontinuity yet revert to the optimal weights when the solution is smooth. This behavior is achieved by specifying

\[
\omega_i = \frac{\alpha_i}{\sum_{i=1}^{3} \alpha_i}
\]

\[
\alpha_i = \frac{\omega_{i,\text{opt}}}{(\epsilon + \beta_i)^p}
\]

\[
\omega_{1,\text{opt}} = \frac{1}{5}, \quad \omega_{2,\text{opt}} = \frac{1}{2}, \quad \omega_{3,\text{opt}} = \frac{3}{10}
\]

Where \(\omega_{i,\text{opt}}\) are the optimal weights, \(\epsilon\) is a small number (typically \(10^{-6}\) or less) to prevent division by zero, \(p\) is chosen to promote large denominators and is usually taken to be 2, and the \(\beta_i\) are smoothness indicators that reflect the smoothness of the solution. Several variations on the smoothness indicators have been proposed; here we use the formulation in [5] which defines the smoothness indicators in terms of the polynomial \(p_i(x)\) whose averages over the cells in stencil \(i\) equal the cell-averaged solutions in that stencil (and has minimal degree):

\[
\beta_i = \sum_{l=1}^{r-1} \int_{x_{j-l/2}}^{x_{j+1/2}} h^{2l-1} \left( p_i^{(l)}(x) \right)^2 dx
\]

This is a sum of normalized \(L^2\) norms of derivatives of \(p_i(x)\). For a uniform grid it can be shown that this definition gives rise to:

\[
\beta_1 = \frac{13}{12} (u_{j-2} - 2u_{j-1} + u_j)^2 + \frac{1}{4} (u_{j-2} - 4u_{j-2} + 3u_j)^2
\]

\[
\beta_2 = \frac{13}{12} (u_{j-1} - 2u_j + u_{j+1})^2 + \frac{1}{4} (u_{j-1} - u_{j+1})^2
\]

\[
\beta_3 = \frac{13}{12} (u_j - 2u_{j+1} + u_{j+2})^2 + \frac{1}{4} (3u_j - 4u_{j+1} + u_{j+2})^2
\]

The solution-dependence of the weights (9) arises via the solution-dependence of these indicators. The CRWENO scheme is described in detail in [2] and [1].
It can be shown that the above CRWENO method can be modified to apply to non-uniform grids (in yet-unpublished work by me). The modified scheme involves the three third-order schemes:

\[
\frac{L+1}{2L+1} \hat{u}_{j-\frac{1}{2}} + \frac{L}{2L+1} \hat{u}_{j+\frac{1}{2}} = \frac{1}{(2L+1)(L+1)} u_{j-1} + \frac{L(2L+3)}{(2L+1)(L+1)} u_j
\]  

(16)

\[
\frac{R}{2R+1} \hat{u}_{j-\frac{1}{2}} + \frac{R+1}{2R+1} \hat{u}_{j+\frac{1}{2}} = \frac{R(2R+3)}{(2R+1)(R+1)} u_j + \frac{1}{(2R+1)(R+1)} u_{j+1}
\]  

(17)

\[
\frac{R+1}{R+2} \hat{u}_{j+\frac{3}{2}} + \frac{1}{R+2} \hat{u}_{j+\frac{1}{2}} = \frac{R^2}{(R+2)(R+1)} u_j + \frac{3R+2}{(R+2)(R+1)} u_{j+1}
\]  

(18)

With the corresponding optimal weights:

\[
\omega_{i,\text{opt}} = \frac{p_i}{p_1 + p_2 + p_3}, \quad i = 1, 2, 3
\]  

(19)

\[p_1 = R^2(2L+1)(R+1)^2\]

\[p_2 = LR(2R+1)(L+1)(L+2R+2)\]

\[p_3 = L(L+1)(R+2)(L+R+1)\]

The quantities \(L\) and \(R\) are the ratios of the lengths of cells \(j-1\) and \(j+1\) to that of cell \(j\), respectively.

Continuing to use (12) as the definition of the smoothness indicators, they can be expressed in the form:

\[
\beta_i = 39 \left( A_i \right)^2 + (3A_i - B_i)^2
\]  

(20)

Where the parameters \(A_i, B_i\) are given below:

<table>
<thead>
<tr>
<th>(i)</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_i)</td>
<td>(\frac{u_{j-2} - u_{j-1}}{K+L} \cdot \frac{u_{j-1} - u_j}{1+L})</td>
<td>(\frac{u_{j-1} - u_{j}}{1+L} \cdot \frac{u_{j} - u_{j+1}}{1+R})</td>
<td>(\frac{u_{j} - u_{j+1}}{1+R} \cdot \frac{u_{j+1} - u_{j+2}}{R+S})</td>
</tr>
<tr>
<td>(B_i)</td>
<td>(2A_1(3+2L+K) - 2\frac{(u_{j-2} - u_{j-1})}{K+L})</td>
<td>(2A_2(L+2) - 2\frac{(u_{j-1} - u_j)}{1+L})</td>
<td>(2A_3(1-R) - 2\frac{(u_{j} - u_{j+1})}{1+R})</td>
</tr>
</tbody>
</table>

Table 1: Intermediate parameters for the smoothness indicators

The quantities \(K\) and \(S\) are the ratios of the lengths of cells \(j-2\) and \(j+2\) to that of cell \(j\), respectively. **Note:** Both versions of the CRWENO scheme are biased in the sense that the fifth-order scheme involves more points to the left of face \(j\) than to the right. This allows the physics of advection to be respected which is necessary for stability.

With these modifications, the rest of the scheme operates in exactly the same way as before. This extension of the CRWENO scheme will be validated as part of this project.
1.4 Characteristic Decompositions

Physically, the solutions of hyperbolic systems are comprised of waves traveling at various speeds in various directions. For problems where discontinuities are present, the numerical method must respect this physical behavior in order to avoid spurious oscillations. However, the quantities being advected by these waves are not necessarily the conserved variables so a change of variables must be performed. This change of variables is referred to as the characteristic decomposition.

For simplicity consider a one-dimensional hyperbolic system:

\[
\frac{\partial q}{\partial t} + \frac{\partial F(q)}{\partial x} = 0 \tag{21}
\]

We may write this equation in quasilinear form, involving the flux Jacobian matrix \(J(q) = \frac{\partial F}{\partial q}\) as

\[
\frac{\partial q}{\partial t} + J(q) \frac{\partial q}{\partial x} = 0 \tag{22}
\]

Because the system is hyperbolic, by definition the Jacobian matrix is diagonalizable. We write \(J\) in terms of its eigen-decomposition as \(J = X\Lambda X^{-1}\), so (22) becomes:

\[
\frac{\partial q}{\partial t} + X\Lambda X^{-1} \frac{\partial q}{\partial x} = 0 \tag{23}
\]

Multiplying through by \(X^{-1}\) and assuming \(X^{-1}\) is locally constant, we have

\[
X^{-1} \frac{\partial q}{\partial t} + \Lambda X^{-1} \frac{\partial q}{\partial x} = 0
\]

\[
\frac{\partial \alpha}{\partial t} + \Lambda \frac{\partial \alpha}{\partial x} = 0 \tag{24}
\]

Where \(\alpha = X^{-1}q\) is the vector of characteristic variables. Since \(\Lambda\) is a diagonal matrix (24) is now a system of decoupled advection equations involving components of \(\alpha\), which can be solved separately and then premultiplied by \(X\) to recover \(q\), the vector of conserved variables.

The characteristic decomposition depends on the choice of \(q\) at which the flux Jacobian is calculated. In practice, for reconstruction at a particular face a density-weighted Roe average of states in adjacent cells is used as an estimate of the state at that face; that average state is then used to calculate the Jacobian. Some computations can be avoided by using a closed form for the matrices \(X, X^{-1}\), and \(\Lambda\) that allow them to be calculated directly from the state vector. This way no eigenproblems need be solved.

2 Project Goal

This project aims to develop and validate parallel software to solve the Euler equations of compressible flow for an ideal gas in two space dimensions. The domain is discretized into a fixed Cartesian mesh with possibly unequal spacing. If time allows, add the capability to use general curvilinear meshes.
3 Approach

The overall structure of the program that achieves the project goal is as follows:

1. Populate the grid data structure based on a grid specification file
2. Initialize cell values based on initial conditions
3. For each Runge-Kutta stage:
   (a) For each cell line (in both coordinate directions):
      i. For each face:
         A. Calculate the Roe-averaged state based on the cells to either side of the face
         B. Calculate the transformation matrix for the characteristic decomposition
         C. Transform conserved variables in the stencil to characteristic variables
         D. Reconstruct each characteristic variable via the non-uniform CRWENO method (getting left- and right-biased states)
         E. Transform the biased states to conserved variables and compute fluxes
         F. Upwind the computed fluxes using the Roe scheme
   (b) For each cell, compute the right-hand side of (4) (i.e., add the flux contributions from each face)
4. Update the stored solution at each cell.
5. If the specified end time is not reached, perform another time step. Otherwise:
6. Export cell values in a suitable data format.

Note: Although it is not expected, there is a possibility that some validation problems may be stiff. In that case a different time-stepping scheme would be used other than the explicit Runge-Kutta scheme implied above (and specified below). A backward-difference formula would be appropriate but would be significantly more difficult to implement. It is therefore assumed for now that explicit time advancement will not require impractically small time steps.

Note: The reconstruction processes along different cell lines can be performed independently of each other, creating an opportunity to improve performance by parallelization. The time derivative evaluations can also be parallelized. This is addressed in a later section.

Note: The reconstruction process must account for boundary conditions, which are implemented by placing ghost points outside the domain and setting their values to enforce the correct condition. I am investigating how best to set those values for some boundary condition types.

Note: Because the time derivative calculation requires traversal of all faces, there is a possibility for inefficient cache usage. This problem can be alleviated by suitable design of the data structures that store the solution data.
4 Scientific Computing Algorithms

4.1 3rd-order TVD Runge-Kutta scheme

The solution is advanced in time using the following TVD Runge-Kutta method, which is 3rd order:

\[
\begin{align*}
    u^{(1)} &= u^n + \Delta t L(u^n) \\
    u^{(2)} &= \frac{3}{4} u^n + \frac{1}{4} u^{(1)} + \frac{1}{4} \Delta t L(u^{(1)}) \\
    u^{n+1} &= \frac{1}{3} u^n + \frac{2}{3} u^{(2)} + \frac{2}{3} \Delta t L(u^{(2)})
\end{align*}
\]

Where \( \Delta t \) is the time step and \( L(u) \) is the operator that computes the right-hand side of (4).

4.2 Roe’s Approximate Riemann Solver

Applying the CRWENO reconstruction along the same cell line but in different directions gives two different states for each face. The correct choice depends on the local flow direction. The actual flux should be computed via the scheme biased in the upwind direction so that numerical information moves in the same direction that physical signals (e.g. pressure waves) propagate. This process is termed upwinding and is accomplished in this project by use of Roe’s solver ([6]) for an approximate Riemann problem.

Let \( X \) be the matrix of left eigenvectors of a flux Jacobian calculated at an interface, and \( \Lambda \) the corresponding diagonal matrix of eigenvalues. The result of applying the Roe scheme is to compute the flux \( F_{j+1/2} \) at the face as

\[
F_{j+1/2} = \frac{1}{2} (F_{j+1/2}^L + F_{j+1/2}^R) - \frac{1}{2} X \Lambda |X|^{-1} (u_{j+1/2}^L + u_{j+1/2}^R)
\]

Where superscripts \( L, R \) indicate left- and right-biased estimates respectively. By taking the absolute value of the eigenvalue matrix the directions of information propagation are respected. Fuller details can be found in [6] or [7].

As presented, the Roe scheme allows the computation of non-physical expansion shocks that may occur if the flow and sonic speeds are nearly equal. This problem can be corrected by including an entropy fix to adjust the eigenvalues when they are near zero. This project will use the fix described by Harten in [4], where each eigenvalue \( \lambda \) is replaced by:

\[
\lambda \rightarrow \phi(\lambda) = \begin{cases} 
|\lambda| & \text{when } |\lambda| \geq \delta \\
\frac{\lambda^2 + \delta^2}{2\delta} & \text{otherwise.}
\end{cases}
\]

\( \delta \) is a small parameter taken to be 0.05 for this project.

4.3 Solution of block-tridiagonal linear systems

Because a characteristic decomposition is performed and the reconstruction process is applied to the resulting transformed variables, the system to be solved for the reconstructed face
values is block-tridiagonal. Each block is a $4 \times 4$ matrix representing the transformation from conserved to characteristic variables. This project will use the Eigen [3] package for linear algebra computations because it already uses OpenMP (which will be used for parallelization) in some algorithms and interfaces well with OpenMP.

5 Hardware and Software

The project code will be written in C++ to exploit its speed as a compiled language, the existence of available libraries for parallel capability, and the developer's familiarity with the language. All code is intended to run on a personal computer. The software will be validated on my personal computer which uses an Intel Core T6500 dual-core processor (rated at 2.10 GHz) and 4 GB RAM and runs the 64-bit Windows Vista operating system. For the specific validation problems to be tested, these computing resources should be adequate. OpenMP will be used to provide parallel capability.

6 Complexity

Assuming a grid with $N$ cells in the horizontal and $M$ cells in the vertical direction, for each dimension of the system (4, for 2-D Euler equations) each stage of the time-advancement algorithm would require at most (depending on boundary conditions):

- For the characteristic decomposition:
  - $F = (M + 1)(N + 1)$ calculations of the flux Jacobian eigenvectors. Here $F$ is the total number of faces.
  - $5F$ matrix-vector products to apply the characteristic decomposition to the five points involved in the stencil, of which there is one for each face.
  - $F$ matrix-vector products to convert reconstructed characteristic quantities to conserved quantities.

- For spatial reconstruction:
  - $F$ evaluations of the CRWENO weights.
  - $2M$ solutions of $(N + 1) \times (N + 1)$ block tridiagonal linear systems for left- and right-biased states at vertical faces
  - $2N$ solutions of $(M + 1) \times (M + 1)$ block tridiagonal linear systems for left- and right-biased states at horizontal faces
  - $F_I = M(N - 1) + N(M - 1)$ approximate Riemann solutions (one per interior face, returning the fluxes). $F_I$ is the number of interior faces.

- For time derivative calculation:
  - $F$ additions and $3F$ multiplications (i.e., 1 scalar product of 2D vectors per cell face, plus multiplication by face length) to evaluate the normal components of fluxes.
- $3C$ additions (3 per cell, $C = MN$ is the total number of cells) to calculate the sum of face-normal fluxes over all cells.

The overall cost is $O(MN)$, albeit with a large constant and some dependence on the efficiency of the solution algorithm for the linear systems. For the chosen validation problems the expected largest values of $M$ and $N$ are 100 and 400 respectively.

7 Validation Methods

The output of the project code will be a vector field of the conserved quantities at cell centers, so the ideal test of the software would be to compare that vector field with a reference solution of the same problem. Numerical data may or not be available but standard test cases for compressible flow can be used, if only for visual comparison. In the case of visual agreement between the computed and reference solutions, the computed solution will be compared to one obtained on a much finer grid that will be taken as an “exact” solution.

8 Test Problems for Verification

8.1 Exact functions

This project requires validating the accuracy and convergence of the non-uniform CRWENO scheme. This entails applying it to functions whose values are known and verifying the order of accuracy. The particular functions will have various numbers of continuous derivatives (possibly none) and will be defined on a randomly perturbed uniform grid.

8.2 1-D shock

The domain is a unit square with periodic boundary conditions on the top and bottom edges, supersonic inflow at the left edge and subsonic outflow at the right edge (flow goes from left to right). The initial condition is a standing shock wave at the halfway point and the goal is for the output to be the shock remaining stationary, with limited smearing. This problem tests the ability of the code to handle a discontinuity and correctly balance fluxes to either side of it.

8.3 Isentropic vortex convection

The isentropic vortex is an exact analytical solution to the 2D Euler equations where a vortex convects with a fixed velocity without any change in its strength or shape. This is the only test case for which an analytic solution exists for comparison, so exact error calculations will only be available for this problem. Following the version of this problem in [1], the domain is a square with side length 10 and periodic boundary conditions on all sides. The initial conditions are a uniform flow with a superimposed vortex. The freestream properties are:

$$\rho_\infty = 1, \ u_\infty = 0.5, \ v_\infty = 0, \ P_\infty = 1$$
A vortex of strength $b$ is initially centered at $x = x_c, y = y_c$ with density, velocity, and pressure:

$$\rho = \left[1 - \frac{(\gamma - 1)b^2}{8\gamma \pi^2} e^{1-r^2}\right]^\frac{1}{\gamma - 1}$$

$$u = u_\infty - \frac{b}{2\pi} e^{1-r^2}(y - y_c)$$

$$v = v_\infty + \frac{b}{2\pi} e^{1-r^2}(x - x_c)$$

$$P = \rho^\gamma$$

Where $r^2 = (x - x_c)^2 + (y - y_c)^2$ and $\gamma$ is the ratio of specific heats (equal to 1.4 for air). The vortex is allowed to convect, and the result should be that the vortex does not change shape or speed as it moves. This problem tests the code’s ability to preserve the shapes of vortex structures while also not dissipating their strength.

8.4 Mach 10 shock reflection

The domain is a rectangle of length 4 and height 1, and the initial condition is a Mach 10 shock intersecting the lower edge at 1/6 units from the left edge and at a 60° angle to that edge. Boundary conditions correspond to the exact motion of a Mach 10 shock. This problem involves reflected shocks and sharp discontinuities that are not aligned with the grid, which the project code must be able to handle. Details can be found in [1].

8.5 Mach 3 flow into a forward-facing step

The domain is a rectangle of length 3 and height 1, with a step of height 0.2 placed 0.6 units from the left edge from which the flow comes. The flow is unsteady and includes a rarefaction fan at the step corner. Unlike the previous two test cases, this case tests the code’s ability to handle protrusions into the domain, which in those cases is simply a rectangle. Details can be found in [8].

9 Results

At this time, there is no reason to expect the results of the project code to differ substantially from the test case results in the absence of bugs. However, because some test cases were computed with methods other than the CRWENO scheme used here it is possible that some flow features will be resolved to different quality compared to the test cases. Such discrepancies would not be surprising, and as long as the same flow features appear in both sets of results they should not be viewed as defects of the project code.

10 Concluding Remarks

This project develops a parallel program that solves the two-dimensional Euler equations of compressible flow. It employs a modified CRWENO scheme that was shown to be valid
for grids with unequal spacing. To simplify the development effort, a fixed Cartesian grid is used where some regions may correspond to solid objects. The CRWENO reconstruction is applied along lines of adjacent cells and the resulting fluxes are upwinded using the Roe scheme. The calculated fluxes are used to evaluate the rates of change of cell-averaged quantities which are fed to a 3rd-order Runge-Kutta scheme to advance the solution in time.

The results of the project code are compared to established benchmark problems for compressible flows, including a stationary shock, convecting vortex, shock reflection, and flow over a step. These problems verify the code’s ability to handle smooth and discontinuous solutions, resolve flow features not aligned with the grid, and properly handle obstacles. These tests are performed on a laptop computer with moderately fine grids.

11 Timeline and Milestones

This project has four phases which correspond to planning, development, validation, and extension. The time spans for these phases and the associated milestones are as follows:

<table>
<thead>
<tr>
<th>Dates</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/2</td>
<td>Proposal Presentation</td>
</tr>
<tr>
<td>Phase 1</td>
<td>10/3 - 10/31 Validate the non-uniform CRWENO scheme.</td>
</tr>
<tr>
<td></td>
<td>Select data structures, data formats, and boundary condition treatments.</td>
</tr>
<tr>
<td>Phase 2</td>
<td>11/1 - 11/21 Complete code to populate the grid data structure from grid specifications and initial conditions.</td>
</tr>
<tr>
<td>Phase 2</td>
<td>11/22 - 12/19 Complete code to perform the characteristic decompositions and spatial reconstructions.</td>
</tr>
<tr>
<td>Phase 2</td>
<td>12/20 - 1/16 Complete code for time advancement.</td>
</tr>
<tr>
<td>Phase 3</td>
<td>1/17 - 1/30 Complete validation for the 1-D shock case.</td>
</tr>
<tr>
<td>Phase 3</td>
<td>1/31 - 2/13 Complete validation for isentropic vortex convection.</td>
</tr>
<tr>
<td>Phase 3</td>
<td>2/14 - 2/27 Complete validation for the Mach 10 shock reflection case.</td>
</tr>
<tr>
<td>Phase 3</td>
<td>2/28 - 3/13 Complete validation for the Mach 3 flow with step case.</td>
</tr>
<tr>
<td>Phase 4</td>
<td>3/14 - 4/1 Buffer time to either complete development and validation or possibly extend the code to handle curvilinear meshes.</td>
</tr>
<tr>
<td></td>
<td>After 4/1 Finish code documentation and otherwise wrap up the project</td>
</tr>
</tbody>
</table>

12 Deliverables

The deliverable products of this project are the following

1. Validation of the non-uniform CRWENO reconstruction against analytic functions

2. Documented code for the flow solver itself

3. Validation of the results from the code applied to four test cases (1-D shock, Mach 3 flow with step, Mach 10 shock with reflections, and isentropic vortex convection) against previous results, fine-grid solutions or, for the last case only, an analytic solution.
4. Mid-year and final written reports

5. Mid-year and final oral presentations

References


