Steger-Warming Flux Vector Splitting Scheme

prof. Ing. Jaroslav Fořt, CSc.
Eng.Suliman Badour,MSc.
ČVUT FS

January 4, 2024

1 Numerical Steger-Warming Scheme

The Riemann problem for the one-dimensional Euler equations is fundamental in computational fluid dynamics. It involves solving the conservation laws for mass, momentum, and energy, given an initial discontinuity in the fluid properties. The Euler equations in the conservative form are given by:

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho u\\ \rho e\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\rho u\\ \rho u^2+p\\ (\rho e+p)u\end{bmatrix}=0 \tag{1}$$

where ρ is the density, u is the velocity, e is the specific total energy, and p is the pressure. The Steger-Warming Flux Vector Splitting scheme is a method used to solve these equations numerically.

1.1 Define Physical and Numerical Parameters

Specific heat ratio (γ), usually 1.4 for diatomic gases like air.
Number of cells (M) to discretize the spatial domain.
The initial position of discontinuity (x_0), which divides the computational domain into left (W_L) and right (W_R) states.

1.2 Initialize the Spatial Domain

Discretize the spatial domain into M cells, and calculate the cell width Δ_x

1.3 Set Initial Conditions

Specify the left and right states for the primitive variables. For example, for Test 1, we have:

$$W_L=\begin{bmatrix}1.0\\ 0.75\\ 1.0\end{bmatrix},\quad W_R=\begin{bmatrix}0.125\\ 0.0\\ 0.1\end{bmatrix} \tag{2}$$

1.4 Convert Pressure to Total Energy

The pressure is converted to total energy using the ideal gas law:

$$\rho e = \frac{p}{\gamma - 1} + \frac{1}{2}\rho u^2 \tag{3}$$

1.5 Implement the Steger-Warming Scheme

The scheme computes fluxes for each variable and splits them based on the sign of the eigenvalues of the flux Jacobian. The fluxes F are defined as follows:

$$F = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ (\rho e + p)u \end{bmatrix} \tag{4}$$

1.6 Split the Fluxes

1. The eigenvalues associated with the flux Jacobian are computed as:

$$\lambda_{1,3} = u \pm a, \quad a = \sqrt{\frac{\gamma p}{\rho}} \tag{5}$$

2. The fluxes are then split into positive and negative parts:

$$F^+ = \frac{1}{2}(F + |\lambda_1|U), \quad F^- = \frac{1}{2}(F - |\lambda_3|U) \tag{6}$$

1.7 Compute Numerical Flux at Interfaces

The numerical flux across each cell interface is calculated by summing the positive flux from the left cell and the negative flux from the right cell.

1.8 Update Conservative Variables

The conservative variables (U) are updated over time using the numerical flux and the finite volume method:

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}(\text{Numerical Flux}_{i+1} - \text{Numerical Flux}_i) \tag{7}$$

1.9 Apply Transmissive Boundary Conditions

These boundary conditions allow waves to exit the computational domain without reflecting into it.

1.10 Time Integration

The time integration is performed using a loop that advances the solution in time until the final time is reached. The time step (Δt) is constrained by the CFL condition for stability.

1.11 Extract Physical Quantities

After the time integration, the conservative variables are converted back to primitive variables (density, velocity, pressure) for analysis and visualization.

2 Links to the code

This is the link to the GitHub repo, visit Suliman Github repo.

Google Colab file to run the tests