229065125

This coursework aims to numerically solve the one-dimensional Euler equations of hydrodynamics using three increasingly sophisticated numerical schemes: Lax-Friedrichs (LF), Lax-Wendroff (LW), and the Harten-Lax-van Leer (HLL) Riemann solver. We focus on modeling shock tube problems, which are idealized representations of shock wave propagation and interaction between discontinuous fluid states. The goal is to reproduce the exact solutions for two benchmark shock tube scenarios (Problem A and B) and trace key features of the shock structure over time.

The Euler equations for a compressible, inviscid fluid (in conservative form) are:

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0,$$

where the vector of conserved quantities is

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho v \\ E \end{bmatrix},$$

and the flux vector is

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho v \\ \rho v^2 + p \\ v(E+p) \end{bmatrix}.$$

Here, ρ is the density, v the velocity, E the total energy density $E = \frac{p}{\gamma-1} + \frac{1}{2}\rho v^2$, and p the pressure. We close the system with the ideal gas law using polytropic index $\gamma = 1.4$.

Method Overview

Code Implementation Overview

The simulation code is written in `C`, using explicit time integration to evolve the 1D Euler equations of hydrodynamics. It supports three methods: Lax-Friedrichs (LF), Lax-Wendroff (LW), and HLL, each designed to test accuracy, numerical diffusion, and stability under different conditions.

1. **Lax-Friedrichs (LF)** — A first-order method known for simplicity and stability but with significant numerical diffusion. It computes interface fluxes based on the average of neighboring fluxes and a stabilizing diffusive term.

2. **Lax-Wendroff (LW)** — A second-order accurate method using a predictor-corrector approach. It predicts midpoint values at each interface, computes fluxes there, and then uses them to update the solution. While more accurate than LF, it may introduce oscillations near discontinuities.

3. **HLL (Harten-Lax-van Leer)** — A Riemann-solver-based method that avoids full characteristic decomposition. It estimates left and right wave speeds to compute approximate intercell fluxes, preserving sharp features without oscillations.

Each method follows the same overall simulation pipeline:

- **Allocate memory** using `allocate_all()` and set the problem using `setup_sod_problemA()` or `setup_sod_problemB()` depending on the test case.

- **Set boundary conditions** (`boundary_problemA1st()` or `boundary_problemB1st()`) depending on the problem type, to enforce physically reasonable behavior at the domain edges.

- **Compute hydrodynamic variables** such as velocity, pressure, internal energy, and sound speed using `compute_observables()`. In particular, the local sound speed is calculated at each grid point using the ideal gas relation:

$$c_s = \sqrt{\frac{\gamma p}{\rho}},$$

  where $\gamma = 1.4$ is the adiabatic index, p is the pressure, and ρ is the density. This is computed after evaluating the total energy and momentum in each cell.

- **Compute conservative fluxes** at each cell using `compute_fluxes()` based on the physical variables.

- **Reconstruct left and right states** Q_L, Q_R using first-order reconstruction (`first_order_reconstruction()`), providing input for the Riemann solver (used in HLL).

- **Compute half-step intercell fluxes** using either the LF, LW, or HLL method, e.g., `lax_friedrichs_flux()`, `Lax_W_step()`, or `hll()`.

- **Advance the system in time** using either a simple Euler step or a predictor-corrector update, depending on the method. Time evolution is performed with a CFL-based timestep:

$$\Delta t = C \frac{\Delta x}{\max_i \left(|v_i| + c_{s,i}\right)},$$

  where c_s is the local sound speed, v_i the fluid velocity, and $C = 0.3$ is the Courant safety factor. The grid consists of 100 zones, so the spatial resolution is fixed at $\Delta x = 0.01$.

- **Apply boundary conditions again** to the updated state and copy intermediate results into the primary state arrays for the next iteration.
- **Write output to CSV** (e.g., `sod_lf.csv`) at the final simulation time $t_{\max}$, for post-processing and comparison to the exact solution using Python.

Additionally, the code includes a passive tracer field used to mark the initial discontinuity in Problem A. In the HLL variant `run_hll3()`, this tracer is used to identify the position of the contact discontinuity. The code also automatically tracks the forward shock and rarefaction edge via density gradients and writes their positions to `features_hll.txt` for visualization of dynamic features over time.

Lax-Friedrichs Scheme

The Lax-Friedrichs (LF) method is a simple and robust first-order scheme that computes interface fluxes using a symmetric average of neighboring fluxes and a stabilizing diffusive term proportional to the difference in conserved variables. The numerical flux at each interface $i+\frac{1}{2}$ is given by:

$$\mathbf{F}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{F}_i + \mathbf{F}_{i+1}\right) - \frac{\Delta x}{2\Delta t}\left(\mathbf{U}_{i+1} - \mathbf{U}_i\right).$$

This scheme is easy to implement and guarantees stability for suitably small timesteps, but introduces high numerical diffusion that smooths out sharp features such as shocks and contact discontinuities.

These half-fluxes are then used to update the conservative variables via an explicit Euler step. The update formula applied in `euler_step(dt)` follows the standard finite volume method:

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+\frac{1}{2}} - \mathbf{F}_{i-\frac{1}{2}}\right).$$

These updated values are then copied back into the primary arrays for the next timestep.

Lax-Wendroff Scheme

The Lax-Wendroff (LW) method is a second-order scheme that improves accuracy by using a midpoint predictor step. Instead of using a Taylor expansion involving the flux Jacobian, this implementation computes an approximate state at the midpoint between cells:

$$\mathbf{Q}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{U}_i + \mathbf{U}_{i+1}\right) + \frac{\Delta t}{2\Delta x}\left(\mathbf{F}_i - \mathbf{F}_{i+1}\right),$$

and then evaluates the flux at that intermediate state directly:

$$\mathbf{F}_{i+\frac{1}{2}} = \mathbf{F}(\mathbf{Q}_{i+\frac{1}{2}}).$$

This two-step predictor-corrector method improves resolution of smooth waves and contact discontinuities compared to Lax-Friedrichs. However, as with most second-order schemes, it can introduce nonphysical oscillations near sharp discontinuities such as shocks.

HLL Solver

The Harten–Lax–van Leer (HLL) method is a shock-capturing approximate Riemann solver that avoids full characteristic decomposition. It constructs a numerical flux across a cell interface based on estimates of the fastest left- and right-going signal speeds. The method approximates the solution to the Riemann problem at each interface by a single intermediate state bounded by two waves: a left-going wave with speed S_L and a right-going wave with speed S_R. The HLL flux at interface $i+\frac{1}{2}$ is defined as:

$$\mathbf{F}_{i+\frac{1}{2}} = \begin{cases} \mathbf{F}_L, & \text{if } S_L \geq 0, \\ \dfrac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_L S_R(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L}, & \text{if } S_L < 0 < S_R, \\ \mathbf{F}_R, & \text{if } S_R \leq 0, \end{cases}$$

where $\mathbf{U}_L$, $\mathbf{U}_R$ are the left and right states, and $\mathbf{F}_L$, $\mathbf{F}_R$ their corresponding fluxes.

Left/Right State Reconstruction. We use a first-order reconstruction to define the states at each interface. For interface $i+\frac{1}{2}$, the left state Q_L is taken from cell i, and the right state Q_R is taken from cell $i+1$. This approach is robust and avoids the complexities of higher-order reconstruction, though it introduces some numerical diffusion.

Estimating Wave Speeds. To determine S_L and S_R, we estimate the maximum signal velocities based on the local flow velocities v, pressures p, densities ρ, and sound speeds c_s on either side of the interface. We define an intermediate pressure $p^\star$ using a simple approximate Riemann pressure estimate:

$$p^\star = \max\left(0, \frac{p_L + p_R}{2} - \frac{v_R - v_L}{2} \cdot \frac{\rho_L + \rho_R}{2} \cdot \frac{c_{s,L} + c_{s,R}}{2}\right),$$

which is a conservative estimate of the pressure in the star region between the left and right states.

Then, the wave speeds are calculated depending on whether $p^\star$ lies above or below the initial pressures:

$$S_L = \begin{cases} v_L - c_{s,L}, & \text{if } p^\star \leq p_L, \\ v_L - c_{s,L}\sqrt{1 + \frac{(\gamma+1)}{2\gamma}\left(\frac{p^\star}{p_L} - 1\right)}, & \text{otherwise} \end{cases}$$

$$S_R = \begin{cases} v_R + c_{s,R}, & \text{if } p^\star \leq p_R, \\ v_R + c_{s,R}\sqrt{1 + \frac{(\gamma+1)}{2\gamma}\left(\frac{p^\star}{p_R} - 1\right)}, & \text{otherwise} \end{cases}$$

These speeds represent the fastest-moving waves in the system, taking into account whether a shock or rarefaction is expected. The conditional square-root correction accounts for nonlinear steepening of the wave in the shock case.

Flux Construction. Once the wave speeds and left/right states are known, the flux is constructed according to the HLL formula above. The solver automatically selects which branch to use based on the signs of

S_L and S_R. In the intermediate region ($S_L < 0 < S_R$), a consistent weighted average of the left and right fluxes and state differences is used.

Parallelization and Performance

To enhance performance, the code was parallelized using `OpenMP`. Key computational loops were parallelized, including:

- Hydrodynamic observable calculation (`compute_observables()`)
- Flux evaluations (`compute_fluxes()`, `lax_friedrichs_flux()`, `hll()`, etc.)
- Time updates and array assignments

Parallel loops use the `#pragma omp parallel for` directive, which allows multiple grid cells to be processed simultaneously. This is particularly effective since the domain is one-dimensional and computations at each grid point are mostly independent.

Performance Gain. On a quad-core CPU, parallelizing the code reduced the runtime by approximately **2.5–3x** for 100 zones. The benefit becomes more pronounced for larger grid sizes or finer time resolution. This allows faster experimentation with different methods and resolutions, and enables feature tracking to be performed efficiently in real time.

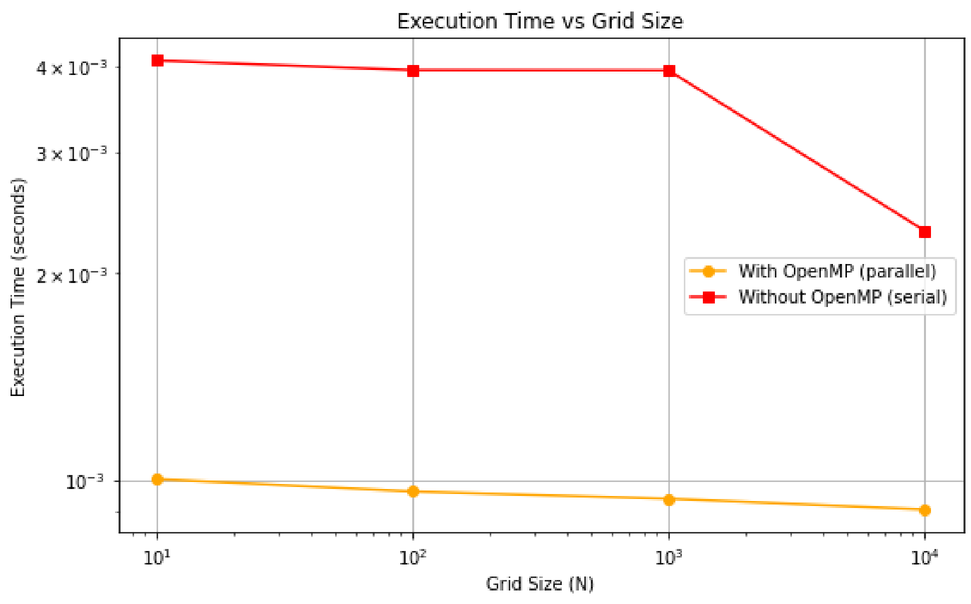

Figure 1: Comparison of solver runtimes with and without OpenMP parallelization.

Scalability. While 100 zones is small enough that parallel efficiency is modest, the structure of the code scales well for larger domains. This makes the approach suitable for extension to 2D or 3D applications in future work. While all three methods successfully capture the qualitative behavior of the Sod shock tube, I chose the HLL scheme for further study and benchmarking due to its strong balance between accuracy and robustness. Unlike Lax-Friedrichs, it does not excessively smear discontinuities, and unlike Lax-Wendroff, it avoids spurious oscillations near shocks. HLL also avoids the need for full characteristic decomposition, making it computationally efficient while still capturing essential shock physics. Looking ahead, the HLL method offers a solid foundation for extension to more complex flows, and it is well-suited to be combined with higher-order spatial reconstruction techniques, such as slope limiters or MUSCL schemes. This would allow me to preserve sharp features while increasing accuracy — a powerful combination particularly useful in multidimensional or high-Mach number astrophysical simulations. This project showed how even simple numerical schemes can capture complex fluid behavior. Extending to higher-order reconstruction or 2D domains would be natural next steps