

计算流体力学期末大作业

朱林-2200011028

2025 年 6 月 20 日

1 数理算法原理

1.1 问题描述

1.1.1 物理情形

Sod 激波管问题是一个一维理想气体流动问题：无限长管道中，初始时刻 ($t = 0$) 在 $x = 0$ 处有一薄膜分隔两侧气体：

- 左侧 ($x < 0$): 高压区，状态为 (ρ_L, u_L, p_L)
- 右侧 ($x > 0$): 低压区，状态为 (ρ_R, u_R, p_R)

薄膜在 $t = 0^+$ 时刻瞬时破裂，两侧气体开始相互作用，产生复杂的波系结构。

1.1.2 标准初始条件

采用以下无量纲初始条件：

左侧: $\rho_L = 1.0, u_L = 0.0, p_L = 1.0$

右侧: $\rho_R = 0.125, u_R = 0.0, p_R = 0.1$

1.2 控制方程

流动由一维欧拉方程描述：

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial f(\mathbf{U})}{\partial x} = 0 \quad (1)$$

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix}, \quad f(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E + p) \end{bmatrix} \quad (2)$$

其中总能密度 $E = \rho e = \rho(C_v T + \frac{1}{2}u^2)$ 。

1.3 Riemann 问题精确解

1.3.1 波系结构

根据空气动力学知识 [1]，该 Sod 激波管中可能出现三种波：

- 激波：流体密度、速度、压力均发生突变，满足 Rankine-Hugoniot (R-H) 关系式。

- 接触间断：流体仅密度发生突变，速度与压力不变。
- 膨胀波（稀疏波）：一种等熵波，其内部物理量连续、光滑，头、尾物理量连续但导数不连续（弱间断），且 Riemann 不变量不变。

对于一维 Sod 激波管问题，薄膜破裂后将形成向左传播的膨胀波、向右传播的接触间断和激波，如图1。这些波将流场划分为五个特征区域（如图2所示¹）：

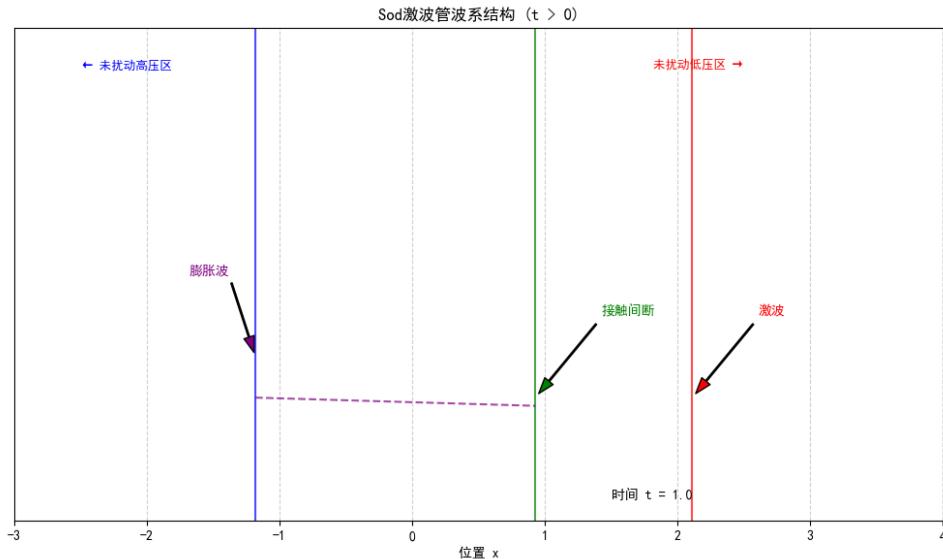


图 1: Sod 激波管典型波系结构 ($t > 0$)

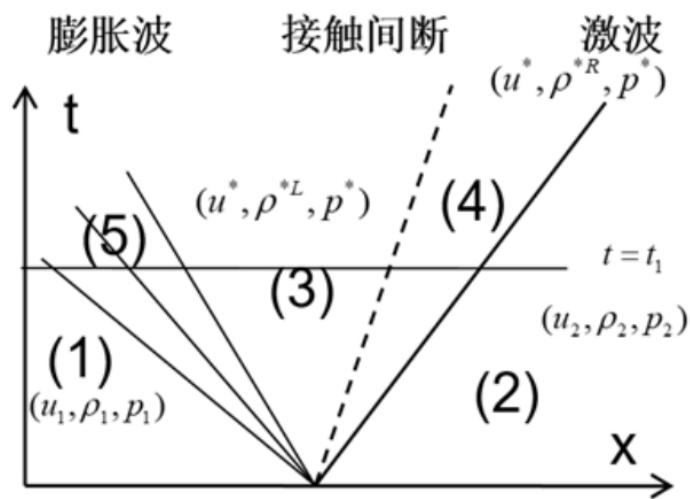


图 2: Sod 激波管典型波系结构

- 区域 1 未扰动的左侧高压区，保持初始状态 (ρ_L, u_L, p_L)
- 区域 2 未扰动的右侧低压区，保持初始状态 (ρ_R, u_R, p_R)

¹ url:<https://blog.csdn.net/Nidebear/article/details/109300513>

- 区域 3 膨胀波后, 状态为 (ρ_2, u^*, p^*)
- 区域 4 接触间断与激波间均匀区, 状态为 (ρ_3, u^*, p^*)
- 区域 5 膨胀波内部

其中 u^* 和 p^* 为接触间断处的速度和压力, 各波位置随时间线性变化:

$$x_{\text{left}} = -c_L t, \quad x_{\text{contact}} = u^* t, \quad x_{\text{shock}} = W_s t$$

W_s 为激波传播速度, $c_L = \sqrt{\gamma p_L / \rho_L}$ 为左侧声速。

1.3.2 解析解表达式

解析解通过求解以下方程组获得:

1-3 两区, 等熵关系式 [2]

$$\frac{p^*}{(\rho^{*L})^\gamma} = \frac{p_1}{(\rho_1)^\gamma} \quad (3)$$

$$u_1 + \frac{2c_1}{\gamma - 1} = u^* + \frac{2c^L}{\gamma - 1} \quad (4)$$

其中, $c^L = \sqrt{\gamma p^* / \rho^{*L}}$ 。

2-4 两区, 激波 R-H 关系式

$$\begin{cases} \rho_2(u_2 - Z_2) = \rho^{*R}(u^* - Z_2) \\ \rho_2 u_2(u_2 - Z_2) + p_2 = \rho^{*R} u^*(u^* - Z_2) + p^* \\ E_2(u_2 - Z_2) + u_2 p_2 = E^{*R}(u^* - Z_2) + p^* u^* \end{cases} \quad (5)$$

以上变量说明从略。综上 5 个方程、5 个未知数, 故方程组可解, 求解方法为联立以上两个方程组, 解出 3、4 区内速度对压力的依赖关系, 有

$$u^* = u_1 - f(p^*, p_1, \rho_1)$$

其中, 满足

$$f(p^*, p_i, \rho_i) = \frac{2c_i}{\gamma - 1} \left[\left(\frac{p^*}{p_i} \right)^{\frac{\gamma-1}{2\gamma}} - 1 \right] \quad (6)$$

注意到, 激波、膨胀波前后速度-压力的依赖关系可写成统一的形式:

左波 (激波或膨胀波)

$$u^* = u_1 - f(p^*, p_1, \rho_1)$$

右波 (激波或膨胀波)

$$u^* = u_2 + f(p^*, p_2, \rho_2)$$

以上 u^*, p^* 表示 3、4 区内的速度与压力, 其中

$$f(p^*, p_i, \rho_i) = \begin{cases} \frac{p^* - p_i}{\rho_i c_i \left[\frac{\gamma+1}{2\gamma} \left(\frac{p^*}{p_i} \right) + \frac{\gamma-1}{2\gamma} \right]^{1/2}}, & p^* > p_i \\ \frac{2c_i}{\gamma-1} \left[\left(\frac{p^*}{p_i} \right)^{\frac{\gamma-1}{2\gamma}} - 1 \right], & p^* < p_i \end{cases} \quad (7)$$

求解上式可得到 3、4 区内的压力, 然后可以解得速度和密度。

膨胀波内部 对于膨胀波内部物理量的计算，首先由波头传播速度 $u_1 - c_1$ 与波尾传播速度 $u^* - c^{*L}$ 可计算膨胀波的范围。在膨胀波区内，利用特征相容关系和等熵关系计算物理量，可利用简单波的特性来简化计算。以下直接给出各个物理量的计算表达式：

$$\begin{aligned}c(t, x) &= \frac{\gamma - 1}{\gamma + 1} \left(u_1 - \frac{x}{t} \right) + \frac{2}{\gamma + 1} c_1 \\u(x, t) &= c + x/t \\p &= p_1 (c/c_1)^{2\gamma/\gamma-1} \\\rho &= \gamma p/c^2\end{aligned}$$

综上所述，一维 Riemann 问题的精确解的求解思路与方程介绍完毕。本文 Sod 激波管参考精确解程序来自于 GitLab 上的项目 simple shock tube calculator [3]。

1.4 数值计算方法

1.4.1 计算域与网格

计算域默认设置为 $x \in [-5, 5]$ ，时间计算域为 $t \in [0, 2.0]$ ，该范围足以捕捉 Sod 问题中激波、接触间断和膨胀波的完整演化过程。空间离散采用均匀网格划分，网格间距 Δx 由计算域长度和网格数动态确定。时间步长 Δt 根据 CFL 条件自适应调整：

$$\Delta t = \text{CFL} \cdot \frac{\Delta x}{\max(|u| + c)} \quad (8)$$

边界条件采用无反射处理：

$$\begin{aligned}U_0 &= U_1 \\U_{N+1} &= U_N\end{aligned}$$

此边界处理可有效抑制数值反射，确保波系在计算域内自由传播。网格收敛性研究表明，接触间断分辨率对网格依赖性显著，需足够细密的网格才能准确捕捉密度突变特征。

1.4.2 激波捕捉格式

TVD 格式（Minmod 限制器） [4] 总变差定义：

$$TV(U^n) = \sum_{i=-\infty}^{\infty} |U_{i+1}^n - U_i^n| \quad (9)$$

TVD 条件：

$$TV(U^{n+1}) \leq TV(U^n) \quad (10)$$

通量限制器格式：

$$F_{i+\frac{1}{2}} = F_i + \frac{1}{2} \phi(r_i) (F_{i+1} - F_i) \quad (11)$$

梯度比：

$$r_i = \frac{U_i - U_{i-1}}{U_{i+1} - U_i} \quad (12)$$

Minmod 限制器函数：

$$\phi(r) = \text{minmod}(1, r) = \begin{cases} 0 & r \leq 0 \\ r & 0 < r < 1 \\ 1 & r \geq 1 \end{cases} \quad (13)$$

性质：

- 满足二阶精度条件： $\phi(1) = 1$
- 满足 TVD 条件： $0 \leq \phi(r) \leq \min(2, 2r)$
- 在间断处退化为单调的一阶格式

群速度控制（GVC）格式 [5] 通量修正方程：

$$\mathbf{F}^{GVC} = \mathbf{F}^{base} + \epsilon \Delta x^3 \frac{\partial^3 \mathbf{U}}{\partial x^3} \quad (14)$$

三阶导数离散：

$$\left. \frac{\partial^3 U}{\partial x^3} \right|_i \approx \frac{-U_{i-2} + 2U_{i-1} - 2U_{i+1} + U_{i+2}}{2\Delta x^3} \quad (15)$$

自适应控制系数：

$$\epsilon = C \cdot |u \pm c|, \quad C \in [0.1, 0.5] \quad (16)$$

修正后群速度：

$$v_g^{num} = v_g^{exact} - 4\epsilon k^2 \Delta x^2 \quad (17)$$

WENO 格式 基本框架

1. 选择模板： $S_0 = \{I_{i-2}, I_{i-1}, I_i\}$, $S_1 = \{I_{i-1}, I_i, I_{i+1}\}$, $S_2 = \{I_i, I_{i+1}, I_{i+2}\}$
2. 重构多项式：

$$p_0(x) = \frac{1}{3}U_{i-2} - \frac{7}{6}U_{i-1} + \frac{11}{6}U_i \quad (18)$$

$$p_1(x) = -\frac{1}{6}U_{i-1} + \frac{5}{6}U_i + \frac{1}{3}U_{i+1} \quad (19)$$

$$p_2(x) = \frac{1}{3}U_i + \frac{5}{6}U_{i+1} - \frac{1}{6}U_{i+2} \quad (20)$$

WENO-JS [6] 光滑指示器：

$$\beta_k = \sum_{l=1}^2 \Delta x^{2l-1} \int_{x_{i-1/2}}^{x_{i+1/2}} \left(\frac{d^l p_k(x)}{dx^l} \right)^2 dx \quad (21)$$

$$\beta_0 = \frac{13}{12}(U_{i-2} - 2U_{i-1} + U_i)^2 + \frac{1}{4}(U_{i-2} - 4U_{i-1} + 3U_i)^2 \quad (22)$$

$$\beta_1 = \frac{13}{12}(U_{i-1} - 2U_i + U_{i+1})^2 + \frac{1}{4}(U_{i-1} - U_{i+1})^2 \quad (23)$$

$$\beta_2 = \frac{13}{12}(U_i - 2U_{i+1} + U_{i+2})^2 + \frac{1}{4}(3U_i - 4U_{i+1} + U_{i+2})^2 \quad (24)$$

非线性权重：

$$\alpha_k = \frac{d_k}{(\beta_k + \varepsilon)^p}, \quad \omega_k = \frac{\alpha_k}{\sum_{m=0}^2 \alpha_m} \quad (25)$$

$$d_0 = 0.3, \quad d_1 = 0.6, \quad d_2 = 0.1 \quad (26)$$

$$\varepsilon = 10^{-6}, \quad p = 2 \quad (27)$$

WENO-Z

改进权重:

$$\tau_5 = |\beta_0 - \beta_2| \quad (28)$$

$$\alpha_k^Z = d_k \left(1 + \frac{\tau_5}{\beta_k + \varepsilon} \right) \quad (29)$$

$$\omega_k^Z = \frac{\alpha_k^Z}{\sum_{m=0}^2 \alpha_m^Z} \quad (30)$$

优势: 光滑区 $\omega_k \rightarrow d_k$, 收敛阶 $\mathcal{O}(\Delta x^5)$

1.4.3 通量处理方法

通量矢量分裂 (FVS) Steger-Warming 分裂

$$\mathbf{F} = \mathbf{F}^+ + \mathbf{F}^- = \mathbf{R}^+ \mathbf{R}^{-1} \mathbf{U} + \mathbf{R}^- \mathbf{R}^{-1} \mathbf{U} \quad (31)$$

$$\lambda^\pm = \frac{\lambda \pm |\lambda|}{2} \quad (32)$$

一维欧拉方程形式:

$$\mathbf{f}^\pm = \frac{\rho}{2\gamma} \begin{bmatrix} (\gamma-1)\lambda_1^\pm + \lambda_3^\pm + \lambda_5^\pm \\ 2(\gamma-1)\lambda_1^\pm u + \lambda_3^\pm(u+c) + \lambda_5^\pm(u-c) \\ (\gamma-1)\lambda_1^\pm u^2 + \frac{1}{2}\lambda_3^\pm(u+c)^2 + \frac{1}{2}\lambda_5^\pm(u-c)^2 + \frac{3-\gamma}{2(\gamma-1)}(\lambda_3^\pm + \lambda_5^\pm)c^2 \end{bmatrix} \quad (33)$$

Van Leer 分裂 [7] 质量通量分裂:

$$(\rho u)^\pm = \pm \frac{\rho c}{4} (M \pm 1)^2, \quad |M| \leq 1 \quad (34)$$

通量函数:

$$\mathbf{F}^\pm = \begin{bmatrix} (\rho u)^\pm \\ (\rho u)^\pm \left[\frac{(\gamma-1)u \pm 2c}{\gamma} \right] \\ (\rho u)^\pm \left[\frac{((\gamma-1)u \pm 2c)^2}{2(\gamma^2-1)} \right] \end{bmatrix} \quad (35)$$

AUSM 分裂 [8] 通量分解:

$$\mathbf{F} = \dot{m} + \mathbf{P}, \quad = [1, u, H]^T \quad (36)$$

界面通量:

$$\mathbf{f}_{1/2} = \dot{m}_{1/2 L/R} + \mathbf{P}_{1/2} \quad (37)$$

质量流量:

$$\dot{m}_{1/2} = c_{1/2} \begin{cases} M_{1/2} \rho_L & M_{1/2} > 0 \\ M_{1/2} \rho_R & \text{其他} \end{cases}, \quad M_{1/2} = M_L^+ + M_R^- \quad (38)$$

Lax-Friedrichs 分裂

$$\mathbf{F}^\pm = \frac{1}{2} (\mathbf{F} \pm \alpha \mathbf{U}), \quad \alpha = \max |\lambda_i| \quad (39)$$

通量差分分裂 (FDS) Roe 方法

Roe 平均:

$$\hat{\rho} = \sqrt{\rho_L \rho_R} \quad (40)$$

$$\hat{u} = \frac{\sqrt{\rho_L} u_L + \sqrt{\rho_R} u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad (41)$$

$$\hat{H} = \frac{\sqrt{\rho_L} H_L + \sqrt{\rho_R} H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \quad (42)$$

通量计算:

$$\mathbf{f}_{1/2} = \frac{1}{2} (\mathbf{F}_L + \mathbf{F}_R) - \frac{1}{2} \sum_{k=1}^3 |\hat{\lambda}_k| \hat{\alpha}_k \hat{\mathbf{r}}_k \quad (43)$$

HLL 方法

通量计算:

$$\mathbf{f}_{\text{HLL}} = \begin{cases} \mathbf{f}_L & S_L \geq 0 \\ \frac{S_R \mathbf{f}_L - S_L \mathbf{f}_R + S_L S_R (\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L} & S_L < 0 < S_R \\ \mathbf{f}_R & S_R \leq 0 \end{cases} \quad (44)$$

波速估计:

$$S_L = \min(u_L - c_L, \tilde{u} - \tilde{c}), \quad S_R = \max(u_R + c_R, \tilde{u} + \tilde{c}) \quad (45)$$

Lax-Wendroff 方法

二阶中心格式:

$$\mathbf{f}_{i+1/2} = \mathbf{f}(\mathbf{u}_{i+1/2}^{\text{LW}}) \quad (46)$$

中间状态:

$$\mathbf{u}_{i+1/2}^{\text{LW}} = \frac{1}{2} (\mathbf{u}_i + \mathbf{u}_{i+1}) - \frac{\Delta t}{2\Delta x} (\mathbf{f}_{i+1} - \mathbf{f}_i) \quad (47)$$

1.4.4 时间推进格式

三阶 Runge-Kutta (SSP-RK3)[9]:

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \Delta t \mathcal{L}(\mathbf{U}^n) \quad (48)$$

$$\mathbf{U}^{(2)} = \frac{3}{4} \mathbf{U}^n + \frac{1}{4} [\mathbf{U}^{(1)} + \Delta t \mathcal{L}(\mathbf{U}^{(1)})] \quad (49)$$

$$\mathbf{U}^{n+1} = \frac{1}{3} \mathbf{U}^n + \frac{2}{3} [\mathbf{U}^{(2)} + \Delta t \mathcal{L}(\mathbf{U}^{(2)})] \quad (50)$$

稳定性条件:

$$\text{CFL} = \max(|u| + c) \frac{\Delta t}{\Delta x} \leq 1.5 \quad (51)$$

1.4.5 附加题: 应用特征重构于 FVS 框架

基本思想 特征重构方法 (Characteristic Reconstruction) 是将物理问题的特征结构融入数值格式的技术。在 FVS(Flux Vector Splitting) 框架中应用特征重构的核心思想是:

1. **特征空间投影:** 将守恒变量投影到特征空间 (由系统特征向量张成)
2. **特征变量重构:** 在特征空间应用高精度重构方法
3. **守恒空间转换:** 将重构后的特征变量转换回守恒空间

该方法旨在：

- 减少接触间断处的数值耗散
- 提高密度台阶的分辨率
- 保持激波捕捉能力

数学原理 对于一维欧拉方程，特征分解基于 Jacobian 矩阵 $A = \frac{\partial f}{\partial U}$ ：

$$A = R\Lambda L$$

其中：

- Λ 是特征值对角矩阵
- R 是右特征向量矩阵
- $L = R^{-1}$ 是左特征向量矩阵

具体形式为：

$$\lambda_1 = u - c, \quad \lambda_2 = u, \quad \lambda_3 = u + c$$

$$R = \begin{bmatrix} 1 & 1 & 1 \\ u - c & u & u + c \\ H - uc & \frac{1}{2}u^2 & H + uc \end{bmatrix}$$

特征重构过程 特征重构在 FVS 框架中的实现步骤：

1. 局部特征分解：在界面处计算 Roe 平均状态

$$\hat{\rho} = \sqrt{\rho_L \rho_R}$$

$$\hat{u} = \frac{\sqrt{\rho_L} u_L + \sqrt{\rho_R} u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

$$\hat{H} = \frac{\sqrt{\rho_L} H_L + \sqrt{\rho_R} H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

2. 计算特征矩阵：使用 Roe 平均状态计算 R 和 L
3. 特征空间投影： $W = LU$ (W 为特征变量)
4. 特征变量重构：在特征空间应用 WENO 重构

$$W_{\text{recon}} = \text{WENO}(W)$$

5. 守恒空间转换： $U_{\text{recon}} = RW_{\text{recon}}$
6. 通量计算：使用重构状态计算 FVS 通量

$$F_{i+1/2} = F^{\text{FVS}}(U_{\text{recon},L}, U_{\text{recon},R})$$

优势：

- 显著减少接触间断处的数值耗散
- 提高密度分布的分辨率
- 保持激波捕捉能力
- 与 FVS 框架兼容

挑战:

- 计算复杂度增加
- 数值稳定性问题 (特别是在低密度区域)
- 需要额外的边界处理
- 对时间步长更敏感

符号说明

符号	物理意义
ρ	密度
u	速度
p	压强
E	总能
c	声速 $c = \sqrt{\gamma p / \rho}$
H	总焓 $H = (E + p) / \rho$
γ	比热比
\mathbf{R}, \mathbf{L}	右/左特征向量矩阵
λ	特征值
Δx	空间步长
Δt	时间步长

2 代码生成与调试

由于本次的代码量大，且要求复杂，所以编写代码的过程中严格遵循模块化思路，在编写之初就按照功能划分了多个模块，分别为：

- **main.py**: 主程序入口，负责整体流程控制
- **config.py**: 配置模块，存储参数设置和初始条件
- **time_integration.py**: 时间积分模块，包含三阶 Runge–Kutta 方法实现
- **Characteristic.py**: 附加题，单独实现特征重构方法在 FVS 框架中的应用
- **utils**: 工具函数库，包含数据处理、可视化、调用精确解、边界条件处理等功能
- **initialization**: 初始条件模块，负责设置初始状态和网格划分
- **flux**: 通量计算模块，包含 FVS、FDS 等格式实现
- **schemes**: 数值格式模块，包含 TVD、WENO 等格式实现

对于每个模块，在编写完成后都进行了单元测试，确保各个模块功能正确。主程序通过调用这些模块实现整体流程控制。测试所用的程序在 **test** 目录下²，包含了对各个模块的单元测试脚本。接下来我将按照编写的顺序对各个模块进行介绍。

2.1 initialization

在初始化模块下，我一共编写两个文件：一个是 **domain_setup.py**，用于设置计算域和网格划分；另一个是 **sod_initial.py**，用于设置初始条件。

在 **test** 目录中的 **initialization_test.py** 中，我对这两个模块进行了单元测试，确保它们能够正确设置计算域和初始条件。

2.2 utils

在工具函数模块下，我编写了 **boundary.py**、**exact_solution.py**、**visualization.py** 三个文件，同时还将来自 GitLab^[3] 的代码 **gitlab_sod_analytical.py** 也放置于此模块下。其中 **boundary.py** 实现了三种边界条件处理，**exact_solution.py** 在调用 GitLab 上的代码的基础上实现了 Sod 激波管问题的精确解计算，**visualization.py** 实现了结果可视化功能。

在 **test** 目录中的 **boundary_test.py**、**exact_solution_test.py** 和 **visualization_test.py** 中，我对这些工具函数进行了单元测试，确保它们能够正确处理边界条件、计算精确解和可视化结果。

2.3 flux

在通量计算模块下，我编写了 **flux_fvs.py** 和 **flux_fds.py** 两个文件，分别实现了 FVS 和 FDS 格式的通量计算。其中 **flux_fvs.py** 实现了 Steger-Warming、Van Leer、AUSM 和 Lax-Friedrichs 四种通量分裂方法，**flux_fds.py** 实现了 HLL、Roe 和 Lax-Wendroff 三种通量差分分裂方法。但是其中的 Roe 方法存在问题，我在尝试多次后仍然未能解决 bug，所以在择时请勿使用 Roe 方法。

²如果希望运行测试程序，请将其放置到 **src** 目录下

2.4 schemes

在数值格式模块下，我编写了 `tvd.py`、`weno.py` 和 `gvc.py` 三个文件，分别实现了 TVD 格式、WENO 格式和群速度控制格式。其中 `tvd.py` 实现了 Minmod 限制器的 TVD 格式，`weno.py` 实现了 WENO-JS 和 WENO-Z 格式，`gvc.py` 实现了群速度控制格式。

在 test 目录中的 `tvd_fvs_rk3_test.py` 中，我对于 TVD、FVS 和 RK3 格式进行了单元测试，确保在这一组合下能够正确计算 Sod 激波管问题的数值解。

在 test 目录中的 `weno_test.py` 中，我对 WENO 格式进行了单元测试，确保其能够正确处理 Sod 激波管问题的数值解。

在 test 目录中的 `gvc_test.py` 中，我对群速度控制格式进行了单元测试，确保其能够正确处理 Sod 激波管问题的数值解。

在 test 目录中的 `fds_test.py` 中，我在使用 TVD 格式的基础上，测试了 FDS 格式的通量计算，确保其能够正确处理 Sod 激波管问题的数值解。

2.5 time_integration.py

主目录下，我编写了 `time_integration.py`，用于实现三阶 Runge-Kutta 时间积分方法和计算 Δt 。

2.6 config.py

在主目录下，我编写了 `config.py`，用于存储参数设置和初始条件。该文件包含了所有需要的参数，如计算域、网格划分、初始条件等。此外，在这个文件中，用户可以自主选择使用的数值格式和通量计算方法。这样可以方便地进行参数调整和格式切换，而无需修改其他代码。

2.7 main.py

作为项目的主程序入口，`main.py` 负责整体流程控制。它首先导入配置文件中的参数设置，然后调用各个模块实现 Sod 激波管问题的数值计算和可视化。

2.8 characteristic.py

独立于主程序的附加题模块，`characteristic.py` 实现了特征重构方法在 FVS 框架中的应用。该模块包含特征分解、特征变量重构和守恒空间转换等功能，采用 weno 作为重构方法，并在 FVS 框架中实现了特征重构的通量计算。

2.9 使用方法

在项目的 src 目录下，用户可以通过以下命令运行主程序：

```
python main.py
```

如果希望运行附加题的特征重构方法，可以使用以下命令：

```
python characteristic.py
```

运行后，程序将自动执行以下步骤：

1. 导入配置文件中的参数设置

```

# 计算域参数
self.nx = 200
self.x_min = -5.0
self.x_max = 5.0
self.t_end = 1.0

# 物理参数
self.gamma = 1.4
self.cv = 1.0

# 数值方法选择
self.scheme = 'tvd'          # 激波捕捉格式: 'tvd', 'gvc', 'weno'
self.flux_method = 'fvs'      # 通量处理方法: 'fvs', 'fds'
self.flux_type = 'van_leer'
                                # 具体通量类型:
                                # FVS: 'steger_warming', 'van_leer', 'ausm', 'lax_friedrichs'
                                # FDS: 'hll', 'lax_wendroff', 'roe (存在bug)'
self.limiter = 'minmod'       # TVD限制器类型
self.weno_variant = 'z'        # WENO变体: 'js', 'z'

# 边界条件
self.bc_type = 'non-reflective' # 'non-reflective', 'periodic', 'fixed'
self.num_ghost = 3             # 虚单元层数

```

图 3: config.py 参数设置示例

2. 调用初始化模块设置计算域和初始条件
3. 调用工具函数模块处理边界条件和计算精确解
4. 调用通量计算模块计算通量
5. 调用数值格式模块进行数值计算
6. 调用时间积分模块进行时间推进
7. 可视化结果并保存图像文件到 **result** 目录下

运行完成后，用户可以在 **result** 目录下找到生成的图像文件，这些图像展示了 Sod 激波管问题的数值解和精确解的对比。如果需要修改参数设置或选择不同的数值格式和通量计算方法，只需编辑 **config.py** 文件即可。

2.10 代码提交信息

在项目开发过程中，我使用了 Git 进行版本控制。以下是一些关键的提交信息。如果需要查看详细的提交记录，可以查看 https://github.com/ZeroLevelKing/CFD_final.git

The screenshot shows a GitHub repository page for 'ZeroLevelKing / CRD_final'. The 'Commits' tab is selected, showing a list of commits. The commits are grouped by date: June 18, 2025, and June 19, 2025. Each commit includes the author, message, timestamp, and a copy icon.

Date	Commit Message	Author	Timestamp
June 18, 2025	characteristic	ZeroLevelKing	committed 1 minute ago
	correct spell mistake	ZeroLevelKing	committed 1 hour ago
	site update	ZeroLevelKing	committed 1 hour ago
	doc part 3	ZeroLevelKing	committed 1 hour ago
	format	ZeroLevelKing	committed 3 hours ago
	AI usage table	ZeroLevelKing	committed 3 hours ago
	update readme	ZeroLevelKing	committed 4 hours ago
	doc part 2	ZeroLevelKing	committed 6 hours ago
	main.py and config.py	ZeroLevelKing	committed 7 hours ago
	weno temp	ZeroLevelKing	committed 8 hours ago
	gvc finished	ZeroLevelKing	committed 20 hours ago
	fds finished	ZeroLevelKing	committed 20 hours ago
June 19, 2025	fds finished	ZeroLevelKing	committed yesterday
	tvd+fvs+r3 finished	ZeroLevelKing	committed yesterday
	exact solution part finished	ZeroLevelKing	committed yesterday
	boundary finished	ZeroLevelKing	committed yesterday
	test for visual	ZeroLevelKing	committed yesterday
	write visualization.py	ZeroLevelKing	committed yesterday
	finish initialization part	ZeroLevelKing	committed yesterday
	restructure the whole program	ZeroLevelKing	committed yesterday
	add r3	ZeroLevelKing	committed yesterday
	add result dir	ZeroLevelKing	committed 2 days ago
	update chinese support	ZeroLevelKing	committed 2 days ago
	add visualization function	ZeroLevelKing	committed 2 days ago
	add initialize function	ZeroLevelKing	committed 2 days ago
	remove temp file	ZeroLevelKing	committed 2 days ago
update .gitignore	ZeroLevelKing	committed 2 days ago	

图 4: Git 提交信息示例 1

This screenshot shows the same GitHub repository and commit history as Figure 4, but with more commits added. The commits are grouped by date: June 18, 2025, and June 19, 2025. The commit messages and authors are identical to those in Figure 4.

Date	Commit Message	Author	Timestamp
June 18, 2025	fds finished	ZeroLevelKing	committed yesterday
	tvd+fvs+r3 finished	ZeroLevelKing	committed yesterday
	exact solution part finished	ZeroLevelKing	committed yesterday
	boundary finished	ZeroLevelKing	committed yesterday
	test for visual	ZeroLevelKing	committed yesterday
	write visualization.py	ZeroLevelKing	committed yesterday
	finish initialization part	ZeroLevelKing	committed yesterday
	restructure the whole program	ZeroLevelKing	committed yesterday
	add r3	ZeroLevelKing	committed yesterday
	add result dir	ZeroLevelKing	committed 2 days ago
	update chinese support	ZeroLevelKing	committed 2 days ago
	add visualization function	ZeroLevelKing	committed 2 days ago
	add initialize function	ZeroLevelKing	committed 2 days ago
	remove temp file	ZeroLevelKing	committed 2 days ago
update .gitignore	ZeroLevelKing	committed 2 days ago	
June 19, 2025	fds finished	ZeroLevelKing	committed yesterday
	tvd+fvs+r3 finished	ZeroLevelKing	committed yesterday
	exact solution part finished	ZeroLevelKing	committed yesterday
	boundary finished	ZeroLevelKing	committed yesterday
	test for visual	ZeroLevelKing	committed yesterday
	write visualization.py	ZeroLevelKing	committed yesterday
	finish initialization part	ZeroLevelKing	committed yesterday
	restructure the whole program	ZeroLevelKing	committed yesterday
	add r3	ZeroLevelKing	committed yesterday
	add result dir	ZeroLevelKing	committed 2 days ago
	update chinese support	ZeroLevelKing	committed 2 days ago
	add visualization function	ZeroLevelKing	committed 2 days ago
	add initialize function	ZeroLevelKing	committed 2 days ago
	remove temp file	ZeroLevelKing	committed 2 days ago
update .gitignore	ZeroLevelKing	committed 2 days ago	

图 5: Git 提交信息示例 2

This screenshot shows the GitHub repository and commit history for 'ZeroLevelKing / CRD_final' with commits from June 14, 2025, to June 11, 2025. The commits are grouped by date: June 14, 2025; June 15, 2025; June 16, 2025; June 17, 2025; and June 11, 2025. The commit messages and authors are identical to those in Figures 4 and 5.

Date	Commit Message	Author	Timestamp
June 14, 2025	update chinese support	ZeroLevelKing	committed 2 days ago
	add visualization function	ZeroLevelKing	committed 2 days ago
	add initialize function	ZeroLevelKing	committed 2 days ago
	remove temp file	ZeroLevelKing	committed 2 days ago
	update .gitignore	ZeroLevelKing	committed 2 days ago
	overwrite the digital part	ZeroLevelKing	committed 2 days ago
	add reference and rewrite riemann part	ZeroLevelKing	committed 2 days ago
	add the description of compute method	ZeroLevelKing	committed 3 days ago
	add Riemann solution part to doc	ZeroLevelKing	committed 5 days ago
	update doc.description of the question	ZeroLevelKing	committed 5 days ago
	update .gitignore	ZeroLevelKing	committed last week
	Initial commit	ZeroLevelKing	authored last week
June 11, 2025	update .gitignore	ZeroLevelKing	committed last week
	Initial commit	ZeroLevelKing	authored last week

图 6: Git 提交信息示例 3

3 结果讨论和物理解释

3.1 计算域对于结果的影响

以 WENO + FVS(van_leer) + RK3 格式为例，在网格密度为 200 时，计算域的选择对结果有显著影响。较小的计算域可能导致激波和膨胀波未能完全发展，而过大的计算域则可能增加计算时间和资源消耗。通过调整计算域，可以更好地捕捉激波、接触间断和膨胀波的演化过程。观察图7，可以看到不同计算域下的激波、接触间断和膨胀波位置和形状的变化。较小的计算域可能导致激波未能完全发展，而较大的计算域则能够更好地捕捉到激波和膨胀波的完整演化过程。根据多次计算实验判断，计算域在取值为 $x_{range}/t_{range} \approx 5$ 时为佳，在接下来，我们取 $x \in [-5, 5]$ ， $t \in [0, 2.0]$ ，能够较好地反映 Sod 激波管问题的物理特性。

3.2 网格密度对结果的影响

同样以 WENO + FVS(van_leer) + RK3 格式为例，网格密度的选择对结果也有显著影响。较粗的网格可能导致激波和膨胀波的分辨率不足，而较细的网格则能够更好地捕捉到激波和膨胀波的细节。通过调整网格密度，可以更好地平衡计算精度和计算效率。考虑到计算成本和结果精度的平衡，网格密度设置为 200 时能够较好地捕捉激波、接触间断和膨胀波的演化过程。图8展示了不同网格密度下的结果对比，可以看到随着网格密度的增加，激波和膨胀波的形状和位置逐渐趋于稳定。

3.3 激波捕捉格式的影响

激波捕捉格式的选择对结果有显著影响。不同的格式在处理激波和膨胀波时具有不同的数值耗散特性和精度。在通量处理方法采用 FVS(van_leer) 的情况下，TVD、WENO 和群速度控制格式在捕捉激波和膨胀波方面表现出不同的特性。观察图9，可以看到不同激波捕捉格式下的激波和膨胀波形状和位置的变化。结合我对于这几种格式的多次计算实验，得出以下结论：

- TVD 格式（Minmod 限制器）能够捕捉激波和膨胀波，但在接触间断处计算的结果会呈现平滑化现象，可能导致激波和膨胀波的细节损失。
- WENO 格式在处理激波和膨胀波时具有更高的精度，能够较好地捕捉激波和膨胀波的细节，但计算成本较高，尤其在网格密度较大时。
- 群速度控制格式能够有效抑制数值耗散，保持激波和膨胀波的细节，尤其在接触间断处表现出较好的稳定性，但需要适当选择控制系数以平衡计算精度和效率。

3.4 通量处理方法的影响

通量处理方法的选择对结果也有显著影响。在激波捕捉格式采用 WENO 的情况下，FVS、FDS 和 Lax-Wendroff 等通量处理方法在捕捉激波和膨胀波方面表现出不同的特性。

对于 FVS 方法，Steger-Warming、Van Leer、AUSM 和 Lax-Friedrichs 等分裂方法在处理激波和膨胀波时的表现如图10所示。可以看到，不同分裂方法在激波和膨胀波的形状和位置上存在差异。结合多次计算实验，得出以下结论：

- Steger-Warming 分裂方法是经典的激波捕捉方法，但是与其他分裂方法相比，其在处理激波和膨胀波时可能会引入较大的数值耗散，导致激波和膨胀波的形状出现平滑化现象。
- Van Leer 分裂方法在处理激波时表现出较好的精度，但在接触间断处可能出现数值振荡现象。

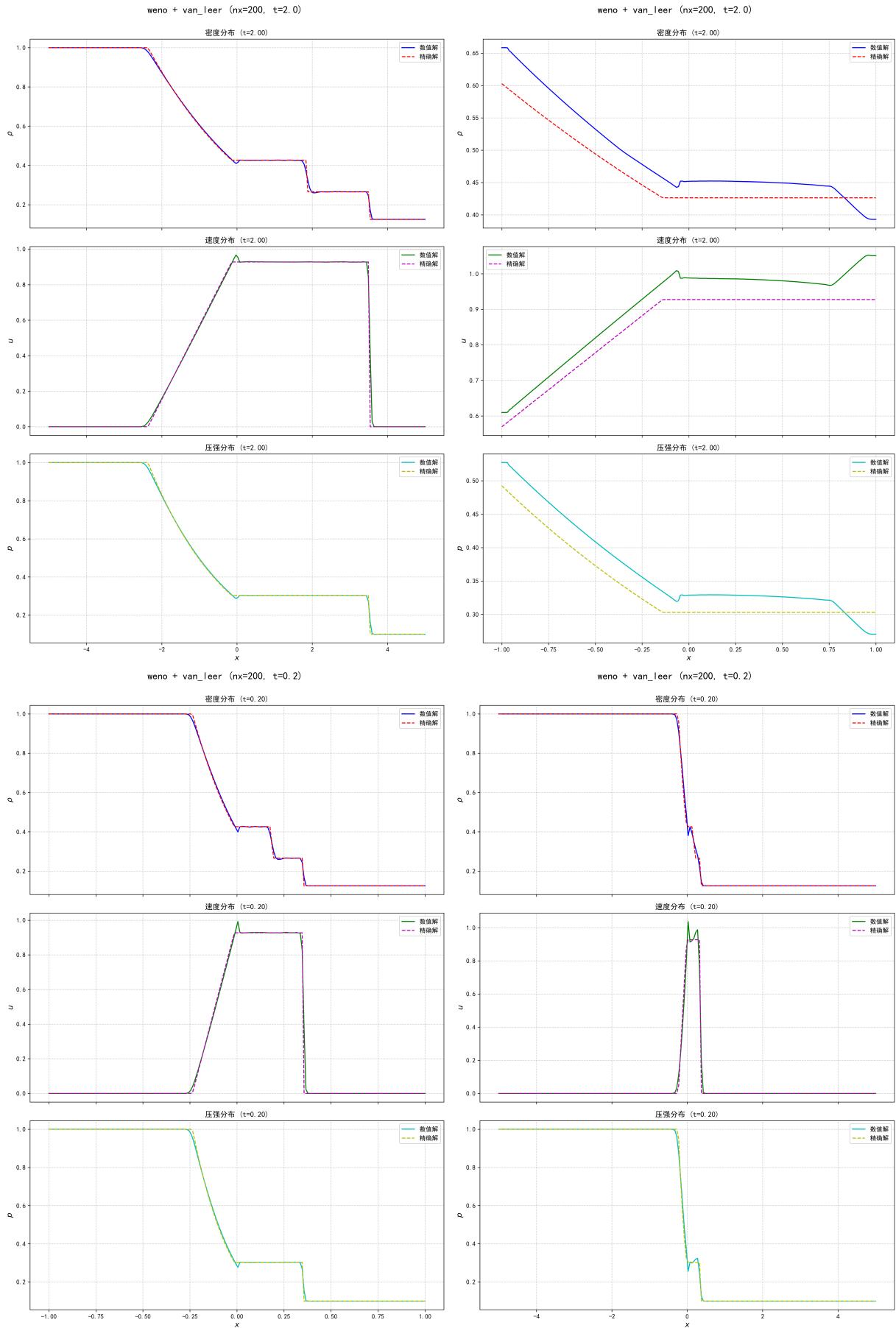


图 7: 计算域对结果的影响示例

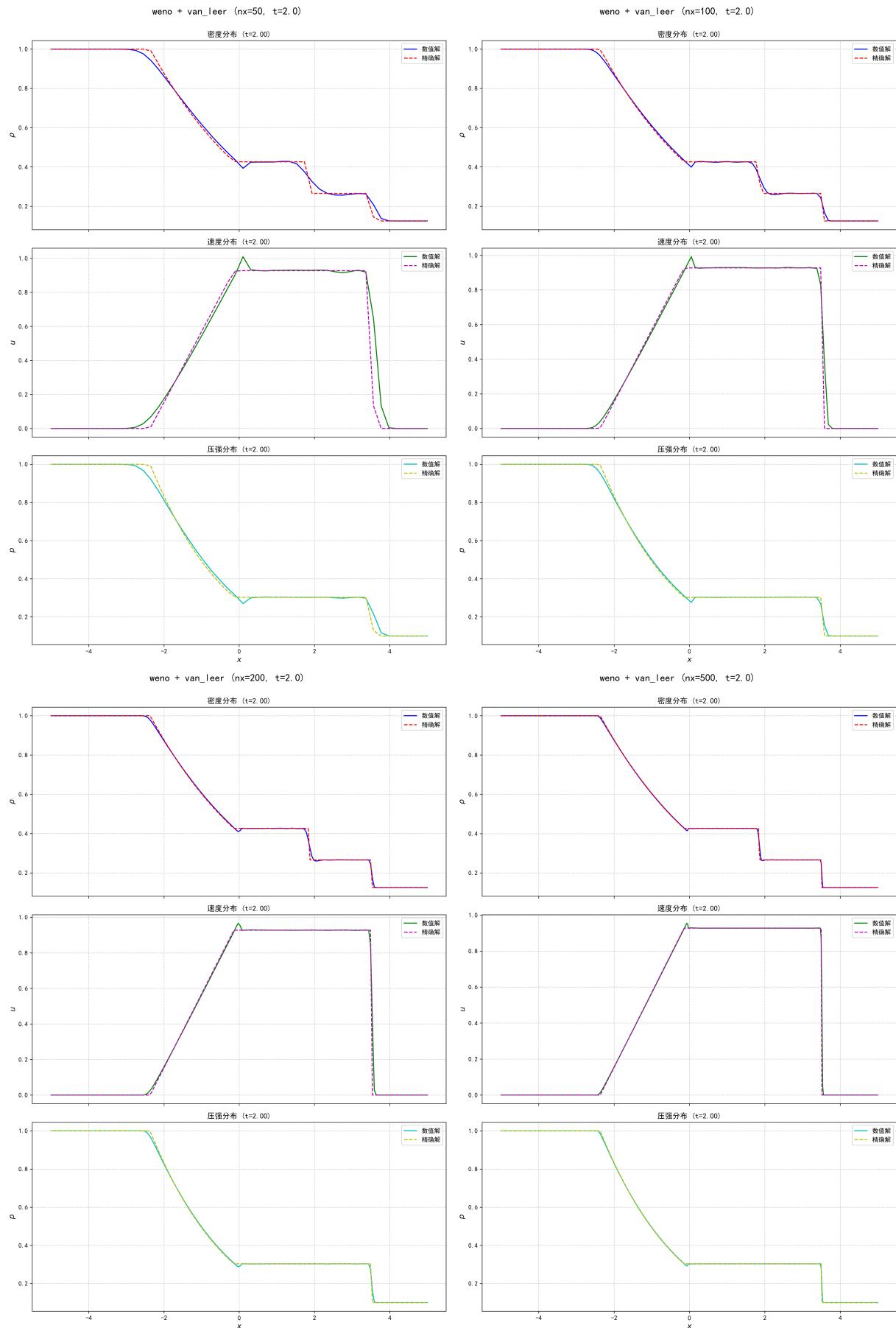


图 8: 网格密度对结果的影响示例

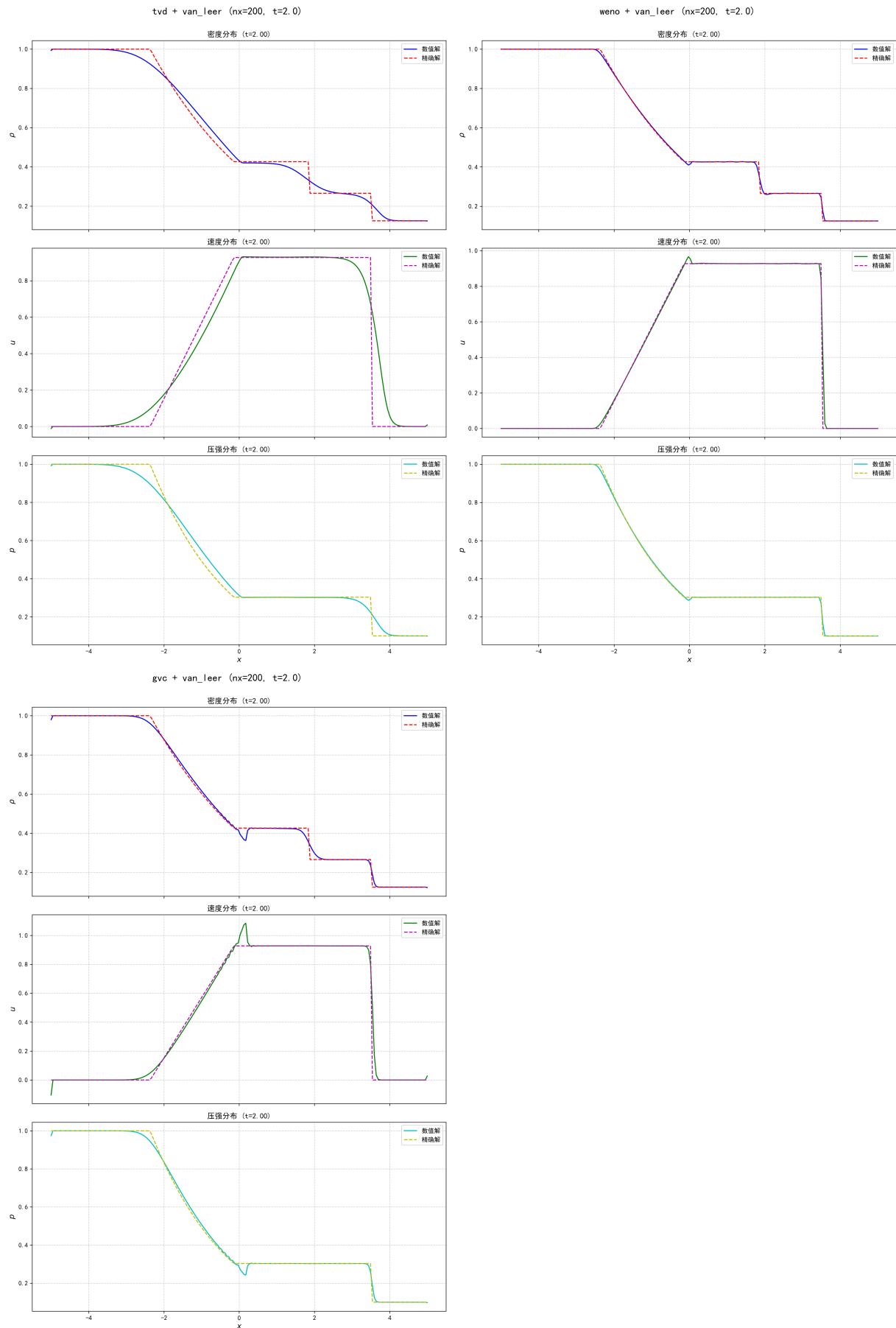


图 9: 激波捕捉格式对结果的影响示例

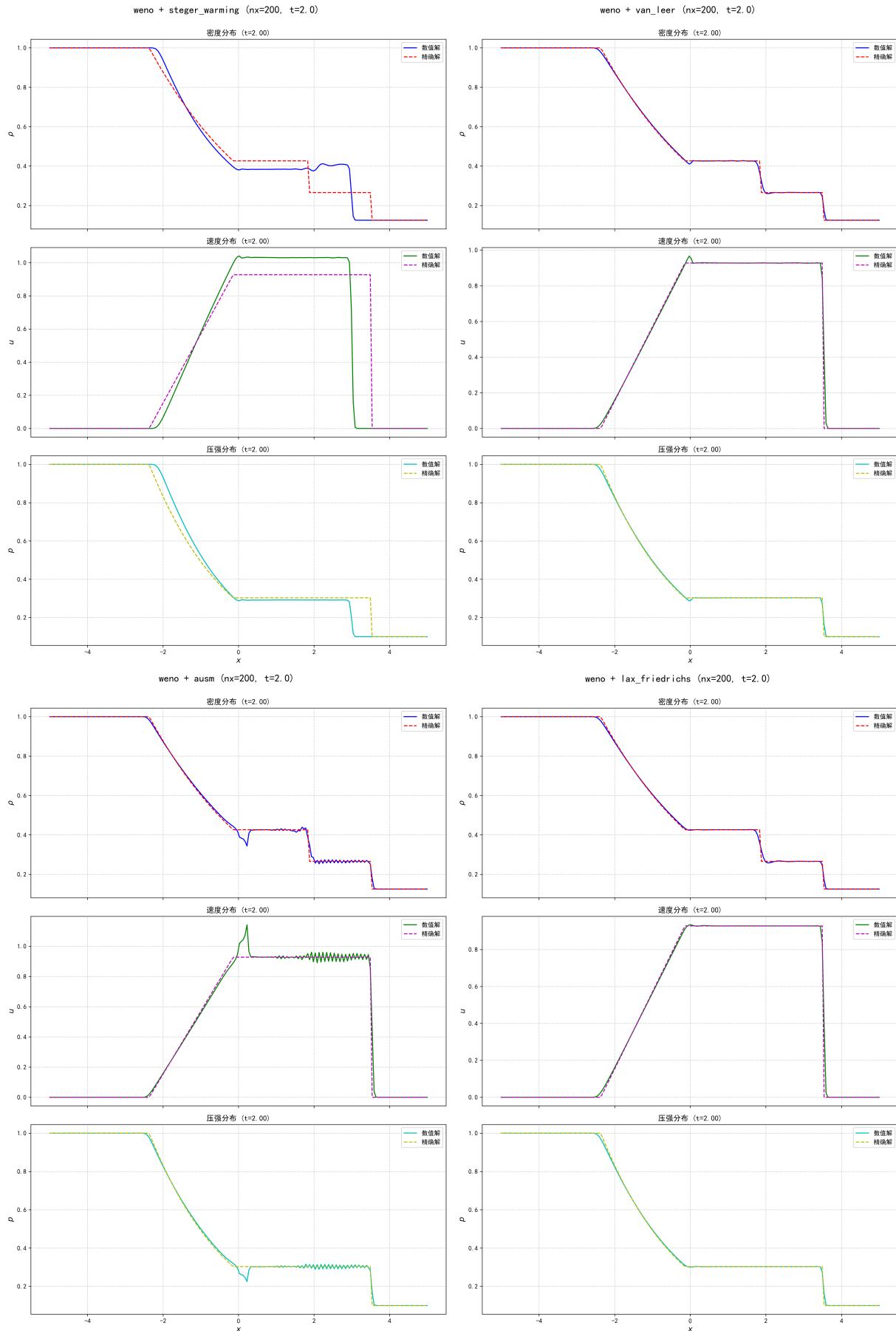


图 10: 通量处理方法对结果的影响示例

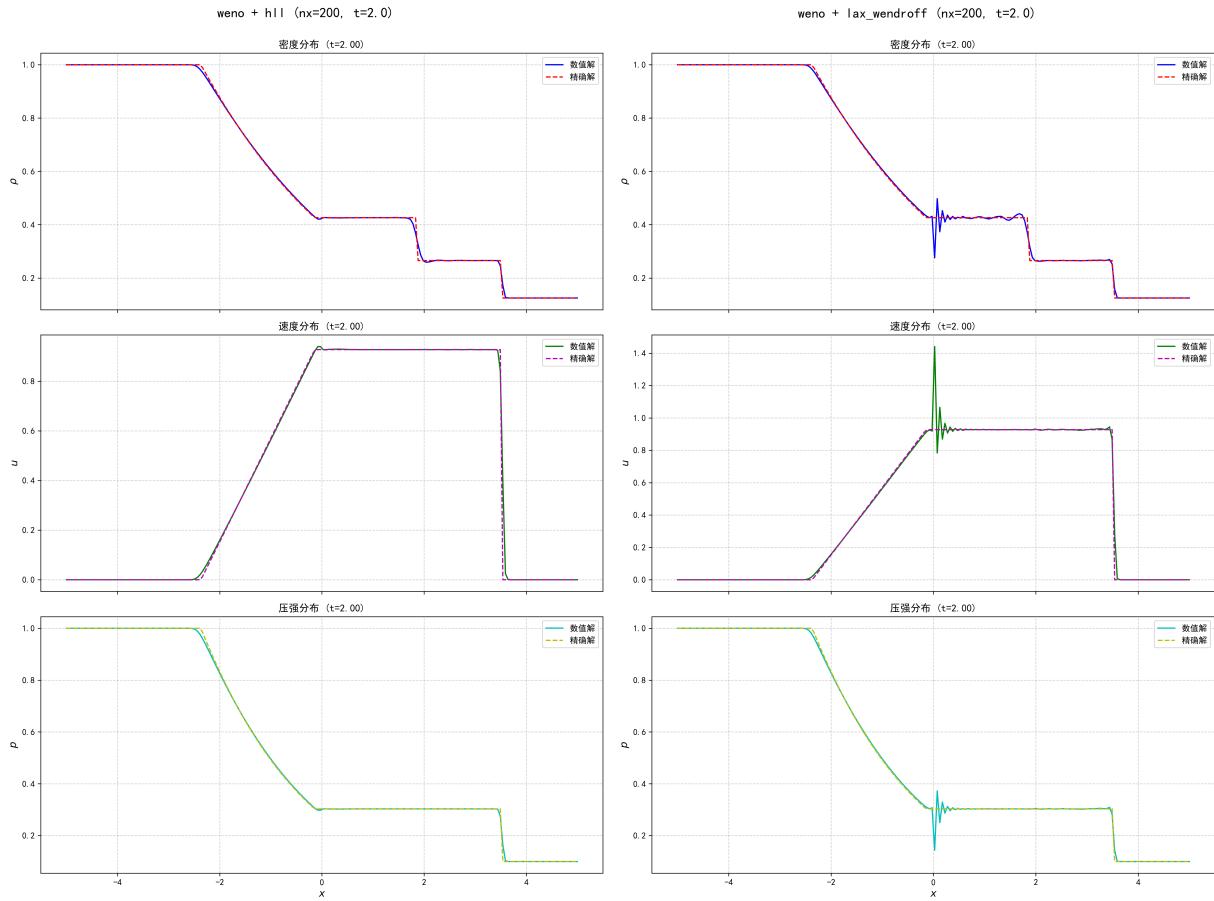


图 11: 通量差分分裂方法对结果的影响示例

- AUSM 分裂方法在处理激波和膨胀波时具有较低的数值耗散，能够保持激波和膨胀波的细节，但是在均匀区域可能出现数值振荡现象。
- Lax-Friedrichs 分裂方法在处理激波时表现出较低的数值耗散，在激波捕捉格式采用 WENO 的情况下，能够较好地捕捉激波和膨胀波的细节，在接触间断处仅有轻微的平滑化现象。

对于 FDS 方法，HLL 和 Lax-Wendroff 等通量差分分裂方法在处理激波和膨胀波时的表现如图11所示。可以看到，不同分裂方法在激波和膨胀波的形状和位置上存在差异。观察可知：

- HLL 方法在处理激波和膨胀波时表现出较好的精度。
- Lax-Wendroff 方法在处理激波时表现出较低的数值耗散，能够保持激波和膨胀波的细节，但是接触间断处可能出现数值振荡现象。

3.5 综合讨论

综合上述讨论，可以得出如果在适宜计算成本下想得到较为精确的 Sod 激波管问题数值解，建议使用以下组合：

- 计算域： $x \in [-5, 5]$, $t \in [0, 2.0]$
- 网格密度：200
- 激波捕捉格式：WENO-Z

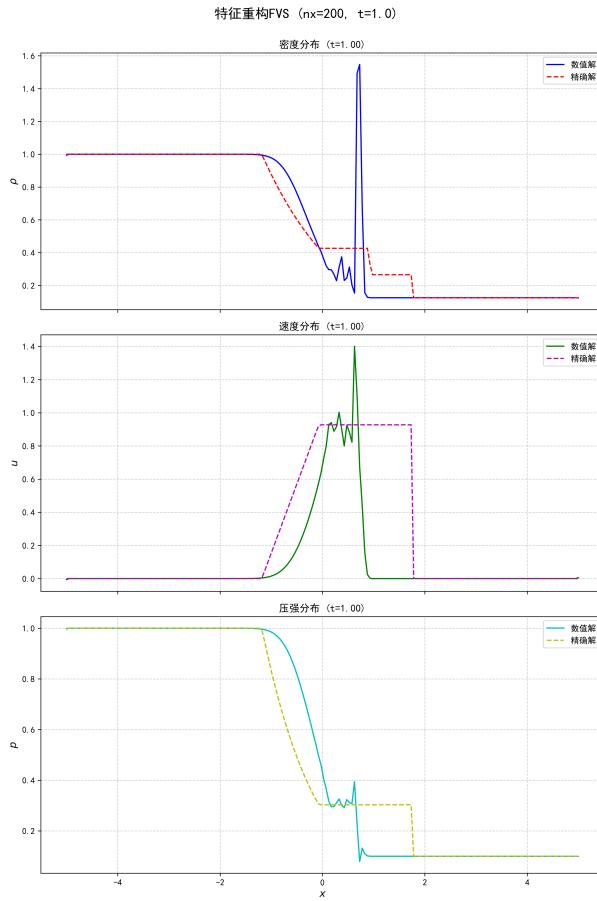


图 12: 特征重构方法在 FVS 框架中的应用结果示例

- 通量处理方法: FVS(van Leer) 或 FDS(HLL)
- 时间推进格式: 三阶 Runge-Kutta (SSP-RK3)

这种组合能够较好地平衡计算精度和效率，捕捉激波。

3.6 附加题：特征重构方法在 FVS 框架中的应用

我们希望通过特征重构方法在 FVS 框架中实现更高精度的激波捕捉。特征重构方法能够有效减少接触间断处的数值耗散，提高密度台阶的分辨率，并保持激波捕捉能力。但实际上，对于运行结果图像12所示，特征重构方法在 FVS 框架中的应用并未显著提高激波捕捉精度，反而在接触间断处出现了较大的数值振荡现象。我认为这可能由以下原因造成：

1. 特征重构实现复杂度高，涉及多次矩阵变换和高频次的矩阵运算，容易积累数值误差。
2. 数值稳定性问题，在接触间断附近（特别是低密度区域），特征矩阵计算容易出现病态条件，声速接近零时，特征矩阵的逆计算不稳定。
3. WENO 重构中的高阶导数计算对噪声敏感，可能导致接触间断处的数值振荡现象。

因此，虽然特征重构方法在理论上具有优势，但在实际应用中需要谨慎处理数值稳定性和误差积累问题。未来可以考虑改进特征矩阵计算方法，或者结合其他数值格式以提高激波捕捉精度。

A AI 工具使用声明表

使用内容	AI 工具用量	使用目的
doc/figure/picture_generate.py	100%	使用 AI 生成文档中所需要的示意性图片
doc/final.tex	20%	省略插入图片和写格式控制语句的重复性工作
ReadMe.md	80%	AI 生成一个框架，在此基础上增加而来
.gitignore	100%	针对于 python 和 latex 的自动生成.gitignore 文件
src/utils/visualization.py	30%	使用 AI 解决了中文显示问题
test/	70%	测试代码，只是调用函数，本身不涉及核心代码，用 AI 生成
src/ 下其余代码	10%	自己编写，用 AI debug 和编写注释

B

参考文献

- [1] John D. Anderson. *Computational Fluid Dynamics: The Basics with Applications*. McGraw-Hill, 2nd edition, 2019.
- [2] Rafael Borges, Monique Carmona, Bruno Costa, and Wai Sun Don. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. *Journal of Computational Physics*, 227(6):3191–3211, 2008.
- [3] fantaz. Simple shock tube calculator, 2021. Accessed: 2025-06-17. URL: https://gitlab.com/fantaz/simple_shock_tube_calculator.
- [4] Ami Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49(2):357–393, 1983.
- [5] Zhenhua Jiang and Chaokun Wu. A high-order scheme for cfd. *Journal of Scientific Computing*, 14(1):1–18, 1999.
- [6] Guang-Shan Jiang and Chi-Wang Shu. Efficient implementation of weighted eno schemes. *Journal of Computational Physics*, 126(1):202–228, 1996.
- [7] Bram Van Leer. Flux-vector splitting for the euler equations. In *Lecture Notes in Physics*, volume 170, pages 507–512. Springer, 1982.
- [8] Meng-Sing Liou. A sequel to ausm: Ausm+. *Journal of Computational Physics*, 129(2):364–382, 1996.
- [9] Chi-Wang Shu and Stanley Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77(2):439–471, 1988.