计算流体力学期末大作业

郑恒 2200011086

2025 年 6 月 19 日

1 问题介绍

针对 Sod 激波管问题，需求解一维欧拉方程：

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

其中：

$$U=\begin{bmatrix}\rho\\ \rho u\\ E\end{bmatrix},\quad f(U)=\begin{bmatrix}\rho u\\ \rho u^2+p\\ u(E+p)\end{bmatrix},\quad E=\rho e=\rho\left(C_vT+\frac{1}{2}u^2\right).$$

在 $t=0$ 时刻，初始条件为：

$$\begin{cases}x<0\text{ 处,}\quad (\rho_L,u_L,p_L)=(1,0,1)\\ x\geq 0\text{ 处,}\quad (\rho_R,u_R,p_R)=(0.125,0,0.1)\end{cases}$$

要求采用数值方法求解密度、速度、压强分布，并与精确解进行比较。Sod 问题的 Riemann 精确解可查询网络或参考书获得。具体需求如下：

1. 计算域与网格由用户设定，并进行相关讨论；
2. 激波捕捉格式：要求至少完成 TVD、群速度控制（GVC）、WENO 各一种；
3. 通量处理方法：要求至少完成 FVS（Flux Vector Splitting）和 FDS（Flux Difference Splitting）各一种；
4. 时间推进格式选用三阶 Runge-Kutta。

2 算法原理

2.1 一维 Riemann 问题精确解

根据空气动力学原理，Sod 激波管中可能出现三种波类型：

- **激波**：密度、速度、压力均发生突变，满足 Rankine-Hugoniot (R-H) 关系式
- **接触间断**：仅密度发生突变，速度与压力不变
- **膨胀波（稀疏波）**：等熵波，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），Riemann 不变量不变

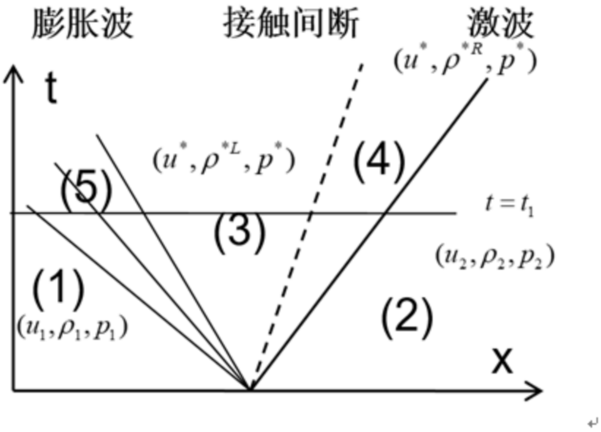

图 1: Sod 激波管问题示意图

针对 Sod 激波管实际工况（初始条件：$(\rho_L, u_L, p_L) = (1, 0, 1)$，$(\rho_R, u_R, p_R) = (0.125, 0, 0.1)$），其流场演化属于**右激波 + 左膨胀波**组合。基于质量、动量、能量通量守恒，可建立方程组求解。

2.1.1 Sod 管实际工况：右激波 + 左膨胀波

分区列写方程如下：

左膨胀波区（1-3 区）：满足等熵关系和 Riemann 不变量守恒：

$$p^*/\left(\rho^{*L}\right)^\gamma = p_1/\left(\rho_1\right)^\gamma \tag{1}$$

$$u_1 + \frac{2c_1}{\gamma-1} = u^* + \frac{2c^L}{\gamma-1} \tag{2}$$

其中 $c_1 = \sqrt{\gamma p_1/\rho_1}$ 为左初始区的声速，$c^L = \sqrt{\gamma p^*/\rho^{*L}}$ 为膨胀波后的声速。

右激波区（2-4 区）：满足激波的 Rankine-Hugoniot 条件：

$$\begin{cases} \rho_2\left(u_2 - Z_2\right) = \rho^{*R}\left(u^* - Z_2\right) \\ \rho_2 u_2\left(u_2 - Z_2\right) + p_2 = \rho^{*R} u^*\left(u^* - Z_2\right) + p^* \\ E_2\left(u_2 - Z_2\right) + u_2 p_2 = E^{*R}\left(u^* - Z_2\right) + p^* u^* \end{cases} \tag{3}$$

其中总能 $E_k = p_k/(\gamma-1) + \rho_k u_k^2/2$。

激波和膨胀波引起的速度-压力变化可统一表示为：

$$\text{左膨胀波：}\quad u^* = u_1 - f\left(p^*, p_1, \rho_1\right) \tag{4}$$

$$\text{右激波：}\quad u^* = u_2 + f\left(p^*, p_2, \rho_2\right) \tag{5}$$

其中 f 函数定义为分段形式：

$$f\left(p^*, p_i, \rho_i\right) = \begin{cases} \dfrac{p^* - p_i}{\rho_i c_i\left[\dfrac{\gamma+1}{2\gamma}\left(\dfrac{p^*}{p_i}\right) + \dfrac{\gamma-1}{2\gamma}\right]^{1/2}}, & p^* > p_i \quad (\text{激波情况}) \\ \dfrac{2c_i}{\gamma-1}\left[\left(\dfrac{p^*}{p_i}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right], & p^* < p_i \quad (\text{膨胀波情况}) \end{cases} \tag{6}$$

联立方程 (4) 和 (5)，得到关于 p^* 的非线性方程：

$$u_1 - u_2 = f\left(p^*, p_1, \rho_1\right) + f\left(p^*, p_2, \rho_2\right) \tag{7}$$

此方程可通过 Newton 迭代法求解 p^*，进而计算 u^* 和各区密度。

2.1.2 膨胀波内部物理量计算

膨胀波范围由波头速度 $u_1 - c_1$ 和波尾速度 $u^* - c^{*L}$ 确定（$c^{*L} = \sqrt{\gamma p^*/\rho^{*L}}$）。波区内物理量由自相似解给出：

$$c(t,x) = \frac{\gamma-1}{\gamma+1}\left(u_1 - \frac{x}{t}\right) + \frac{2}{\gamma+1}c_1 \tag{8}$$

$$u(x,t) = c + \frac{x}{t} \tag{9}$$

$$p(x,t) = p_1\left(\frac{c}{c_1}\right)^{2\gamma/(\gamma-1)} \tag{10}$$

$$\rho(x,t) = \gamma p/c^2 \tag{11}$$

此解仅适用于 $u_1 - c_1 \le x/t \le u^* - c^{*L}$ 区域。

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 github 开源项目, 地址为

https://github.com/sbakkerm/Sod-Shock-Tube/tree/main。

2.2 时间推进格式

时间推进采用三阶 Runge-Kutta 方法，整个离散格式为：

$$\frac{\partial U}{\partial t} = -\frac{\partial \mathbf{F}}{\partial x} \tag{12}$$

$$U^{(1)} = U^n - \Delta t \cdot \frac{\partial \mathbf{F}}{\partial x}(U^n) \tag{13}$$

$$U^{(2)} = U^n - \frac{\Delta t}{2} \cdot \frac{\partial \mathbf{F}}{\partial x}(U^{(1)}) \tag{14}$$

$$U^{(3)} = U^n - \frac{\Delta t}{2} \cdot \frac{\partial \mathbf{F}}{\partial x}(U^{(2)}) \tag{15}$$

$$U^{n+1} = U^n - \frac{\Delta t}{6}\left(\frac{\partial \mathbf{F}}{\partial x}(U^{(1)}) + 2\frac{\partial \mathbf{F}}{\partial x}(U^{(2)}) + 2\frac{\partial \mathbf{F}}{\partial x}(U^{(3)})\right) \tag{16}$$

其中 $\mathbf{F}$ 是通量函数，Δt 是时间步长。

其中守恒向量 U 和通量函数 $\mathbf{F}$ 为：

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

2.3 流通矢量分裂技术 (FVS)

目前，数值求解守恒型一维 Euler 方程的主要技术包括流通矢量分裂 (Flux Vector Splitting, FVS) 与流通差分分裂 (Flux Difference Splitting, FDS) 两类。

流通矢量分裂方法基于特征线方法，将方程的解表示为特征向量的组合。该方法将代表质量、动量和能量的流通矢量按照雅克比矩阵的特征值进行分裂，并据此构造迎风格式或激波捕捉格式。

流通差分分裂方法则是对流通矢量的导数进行分裂：首先运用差分格式构造网格半点处的守恒变量，再利用各类近似 Riemann 解构造通量。

2.3.1 基本变量与方程

守恒变量 $\mathbf{U}$ 与流通矢量 $\mathbf{F}(\mathbf{U})$ 定义为：

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u \\ E \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}, \quad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ u(E+p) \end{bmatrix} = \begin{bmatrix} w_2 \\ \dfrac{w_2^2}{w_1} + p \\ \dfrac{w_2}{w_1}(w_3 + p) \end{bmatrix}$$

总能量 E 满足：

$$E = \rho e = \rho \left(c_v T + \frac{1}{2} u^2 \right)$$

考虑完全气体状态方程 $p = \rho R T$，压力可表示为：

$$p = (\gamma - 1) \left(E - \frac{1}{2} \rho u^2 \right) = (\gamma - 1) \left(w_3 - \frac{1}{2} \frac{w_2^2}{w_1} \right)$$

流通矢量可简化为：

$$\mathbf{F}(\mathbf{U}) = \begin{bmatrix} w_2 \\ \dfrac{3-\gamma}{2} \dfrac{w_2^2}{w_1} + (\gamma - 1) w_3 \\ \dfrac{\gamma w_2 w_3}{w_1} - \dfrac{\gamma - 1}{2} \dfrac{w_2^3}{w_1^2} \end{bmatrix}$$

2.3.2 Jacobian 矩阵与特征值

守恒变量的微分关系：

$$\begin{cases} dw_1 = d\rho \\ dw_2 = ud\rho + \rho du \\ dw_3 = \dfrac{dp}{\gamma - 1} + \dfrac{1}{2}u^2 d\rho + \rho u du \end{cases}$$

Jacobian 矩阵 $\mathbf{A}(\mathbf{U}) = \partial\mathbf{F}/\partial\mathbf{U}$：

$$\mathbf{A}(\mathbf{U}) = \begin{bmatrix} 0 & 1 & 0 \\ \dfrac{\gamma-3}{2}\dfrac{w_2^2}{w_1^2} & (3-\gamma)\dfrac{w_2}{w_1} & \gamma - 1 \\ -\dfrac{\gamma w_2 w_3}{w_1^2} + (\gamma-1)\dfrac{w_2^3}{w_1^3} & \dfrac{\gamma w_3}{w_1} - \dfrac{3(\gamma-1)}{2}\dfrac{w_2^2}{w_1^2} & \dfrac{\gamma w_2}{w_1} \end{bmatrix}$$

声速 c 满足：

$$c^2 = \frac{\gamma p}{\rho} = \gamma(\gamma-1)\left(\frac{w_3}{w_1} - \frac{1}{2}\frac{w_2^2}{w_1^2}\right)$$

Jacobian 矩阵的特征值为：

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

当状态方程 $p = \rho g(e)$ 时，流通矢量满足一次齐次性：

$$\mathbf{F}(\alpha\mathbf{U}) = \alpha\mathbf{F}(\mathbf{U}) \quad \forall\alpha$$

对 α 求导并令 $\alpha = 1$ 得：

$$\mathbf{F}(\mathbf{U}) = \mathbf{A}\mathbf{U}$$

2.3.3 分裂方法原理

在构造耗散型格式时，需要将流通矢量 $\mathbf{F}(\mathbf{U})$ 按照特征值进行分裂。

对单波方程流通量的分裂为：

$$f^{\pm} = c^{\pm}u, \quad c^{\pm} = \frac{c \pm |c|}{2}, \quad f = f^+ + f^-$$

对流通矢量，通过 Jacobian 矩阵特征值分裂：

$$\lambda_k = \lambda_k^+ + \lambda_k^-, \quad \lambda_k^+ \geq 0, \quad \lambda_k^- \leq 0$$

分裂后的特征值对角矩阵：

$$\Lambda^\pm = \begin{bmatrix} \lambda_1^\pm & 0 & 0 \\ 0 & \lambda_2^\pm & 0 \\ 0 & 0 & \lambda_3^\pm \end{bmatrix}$$

定义分裂：

$$\mathbf{A}^\pm = S^{-1}\Lambda^\pm S, \quad \mathbf{F}^\pm = \mathbf{A}^\pm \mathbf{U}$$

满足 $\mathbf{A} = \mathbf{A}^+ + \mathbf{A}^-$，$\mathbf{F} = \mathbf{F}^+ + \mathbf{F}^-$。

Steger-Warming (S-W) 分裂法：

$$\lambda_k^\pm = \frac{\lambda_k \pm |\lambda_k|}{2}$$

为避免奇点可修正为：

$$\lambda_k^\pm = \frac{\lambda_k \pm \sqrt{\lambda_k^2 + \varepsilon^2}}{2}$$

Lax-Friedrichs (L-F) 分裂法：

$$\lambda_k^\pm = \frac{\lambda_k \pm \lambda^*}{2}$$

$$\mathbf{F}^+ = \frac{1}{2}\left(\mathbf{F} + \lambda^* \mathbf{U}\right), \quad \mathbf{F}^- = \frac{1}{2}\left(\mathbf{F} - \lambda^* \mathbf{U}\right)$$

其中 λ^* 可取局部值 $|u| + c$ 或全局 $\max(|u| + c)$。

van Leer 分裂法： 基于马赫数 $\mathrm{Ma} = u/c$ 的分裂：

$$\mathbf{F} = \begin{bmatrix} \rho c \mathrm{Ma} \\ \rho c^2 \left(\mathrm{Ma}^2 + \dfrac{1}{\gamma}\right) \\ \rho c^3 \mathrm{Ma}\left(\dfrac{1}{2}\mathrm{Ma}^2 + \dfrac{1}{\gamma - 1}\right) \end{bmatrix} \equiv \begin{bmatrix} f_{\mathrm{mas}} \\ f_{\mathrm{mom}} \\ f_{\mathrm{ene}} \end{bmatrix}$$

分裂策略：

1. $\mathrm{Ma} \geq 1$: $\mathbf{F}^+ = \mathbf{F}, \quad \mathbf{F}^- = 0$
2. $\mathrm{Ma} \leq -1$: $\mathbf{F}^+ = 0, \quad \mathbf{F}^- = \mathbf{F}$
3. $|\mathrm{Ma}| \leq 1$: 按马赫数分裂

分量分裂：

$$
\begin{aligned}
f_{\text{mas}}^{\pm} &= \pm\frac{1}{4}\rho c(1 \pm \mathrm{Ma})^2 \\
f_{\text{mom}}^{\pm} &= f_{\text{mas}}^{\pm}\frac{2c}{\gamma}\left[\frac{(\gamma-1)}{2}\mathrm{Ma} \pm 1\right] \\
f_{\text{ene}}^{\pm} &= \frac{\gamma^2}{2(\gamma^2-1)}\frac{(f_{\text{mom}}^{\pm})^2}{f_{\text{mas}}^{\pm}}
\end{aligned}
$$

统一矢量形式：

$$
\mathbf{F}^{\pm} = \pm\frac{1}{4}\rho c(1 \pm \mathrm{Ma})^2 \begin{bmatrix} 1 \\ \dfrac{2c}{\gamma}\left(\dfrac{\gamma-1}{2}\mathrm{Ma} \pm 1\right) \\ \dfrac{2c^2}{\gamma^2-1}\left(\dfrac{\gamma-1}{2}\mathrm{Ma} \pm 1\right)^2 \end{bmatrix}
$$

2.4 流通差分分裂技术 (FDS)

2.4.1 基本流程

流通差分分裂技术 (FDS) 的基本流程可分为以下三个步骤：

1. 运用各类差分格式，计算在格点间同一位置 $(j+\frac{1}{2})$ 的左右数值守恒通量 $U_{j+\frac{1}{2}}^L$ 与 $U_{j+\frac{1}{2}}^R$
2. 运用近似 Riemann 解，计算 $F_{j+\frac{1}{2}} = F\left(U_{j+\frac{1}{2}}^L, U_{j+\frac{1}{2}}^R\right)$
3. 进行差分重构：
$$
\left(\frac{\partial F}{\partial x}\right)_j = \frac{F_{j+\frac{1}{2}} - F_{j-\frac{1}{2}}}{\Delta x}
$$

2.4.2 Roe 格式 - 单方程情况

在 FDS 类方法中应用最广泛的近似 Riemann 解是 Roe 格式。

对于一般非线性单波方程：

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0$$

可化为：

$$\frac{\partial u}{\partial t}+a(u)\frac{\partial u}{\partial x}=0$$

其中 $a(u)=\frac{\partial f(u)}{\partial u}$ 为非线性项。

将该项线性化，目标是在 $[j,j+1]$ 区间内，采用平均变化率代替变化的 $a(u)$。根据 Lagrange 中值定理，在 $[u_L,u_R]$ 内必有一点 u_{Roe}，该点的斜率为区间平均变化率：

$$\widetilde{a}_{j+\frac{1}{2}}=a(u_{Roe})=\frac{f(u_{j+1})-f(u_j)}{u_{j+1}-u_j}$$

考虑一阶迎风差分，根据 $\widetilde{a}_{j+\frac{1}{2}}$ 的符号构造数值通量：

$$f_{j+\frac{1}{2}}=\begin{cases}f_j & \widetilde{a}_{j+\frac{1}{2}}>0\\ f_{j+1} & \widetilde{a}_{j+\frac{1}{2}}<0\end{cases}$$

或写为：

$$f_{j+\frac{1}{2}}=\frac{1}{2}\left[f_j+f_{j+1}\right]-\frac{1}{2}\left|\widetilde{a}_{j+\frac{1}{2}}\right|(u_{j+1}-u_j)$$

2.4.3 Roe 格式 - 方程组情况

对于一般的双曲守恒律方程组：

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

可化为：

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

其中 Jacobian 矩阵 $\mathbf{A}=\frac{\partial \mathbf{F}(\mathbf{U})}{\partial \mathbf{U}}$。

在 $[j,j+1]$ 或 $[R,L]$ 区间内，寻找一个矩阵 $\widetilde{\mathbf{A}}=\widetilde{\mathbf{A}}(\mathbf{U}_R,\mathbf{U}_L)$ 作为平均变化率矩阵，满足：

$$\mathbf{F}(\mathbf{U}_R)-\mathbf{F}(\mathbf{U}_L)=\widetilde{\mathbf{A}}(\mathbf{U}_R,\mathbf{U}_L)(\mathbf{U}_R-\mathbf{U}_L)$$
$$\widetilde{\mathbf{A}}(\mathbf{U}_R,\mathbf{U}_L)\ \text{可通过相似变换进行对角化}$$

则原方程组可化为常矩阵系数方程：

$$\frac{\partial \mathbf{U}}{\partial t}+\widetilde{\mathbf{A}}\frac{\partial \mathbf{U}}{\partial x}=0$$

数值通量为：

$$\mathbf{F}_{j+\frac{1}{2}}=\frac{1}{2}\left[\mathbf{F}(\mathbf{U}_R)+\mathbf{F}(\mathbf{U}_L)\right]-\frac{1}{2}\left|\widetilde{\mathbf{A}}(\mathbf{U}_R,\mathbf{U}_L)\right|(\mathbf{U}_R-\mathbf{U}_L)$$

其中 $\left|\widetilde{\mathbf{A}}(\mathbf{U}_R,\mathbf{U}_L)\right|=\mathbf{S}^{-1}|\Lambda|\mathbf{S}$。

2.4.4 Roe 矩阵构造

由于原始流通矢量 $\mathbf{F}(\mathbf{U})$ 与守恒矢量 $\mathbf{U}$ 不满足二次齐次函数关系，有两种构造思路：

1. **直接寻找 $\mathbf{U}_{Roe}$**：在 $[\mathbf{U}_L,\mathbf{U}_R]$ 内寻找 $\mathbf{U}_{Roe}$ 使得 $\widetilde{\mathbf{A}}(\mathbf{U}_R,\mathbf{U}_L)=\frac{\partial \mathbf{F}(\mathbf{U}_{Roe})}{\partial \mathbf{U}_{Roe}}$。但实际实现困难。

2. **坐标变换法**：若 $\mathbf{F}$ 是某个自变量的二次齐函数，则区间中点即为 Roe 点。通过坐标变换将 $\mathbf{F}(\mathbf{U})$ 化为齐次函数：

引入新变量：

$$\mathbf{W}=\begin{bmatrix} w_1\\ w_2\\ w_3\end{bmatrix}=\rho^{\frac{1}{2}}\begin{bmatrix}1\\ u\\ H\end{bmatrix}$$

则有：

$$\mathbf{F}(\mathbf{W})=\begin{bmatrix} w_1w_2\\ \frac{\gamma-1}{\gamma}w_1w_3+\frac{1+\gamma}{2\gamma}w_2^2\\ w_3w_2\end{bmatrix}$$

增长矩阵为：

$$\mathbf{C}(\mathbf{W}) = \frac{\partial \mathbf{F}(\mathbf{W})}{\partial \mathbf{W}} = \begin{bmatrix} w_2 & w_1 & 0 \\ \frac{\gamma-1}{\gamma}w_3 & 2w_2 - \frac{\gamma-1}{\gamma}w_2 & \frac{\gamma-1}{\gamma}w_1 \\ 0 & w_3 & w_2 \end{bmatrix}$$

在 $[R, L]$ 区间内有：

$$\mathbf{F}(\mathbf{U}_R) - \mathbf{F}(\mathbf{U}_L) = \mathbf{C}(\overline{\mathbf{W}})(\mathbf{W}_R - \mathbf{W}_L)$$

其中 $\overline{\mathbf{W}} = (\mathbf{W}_R + \mathbf{W}_L)/2$ 为 Roe 点，对应的守恒矢量为：

$$\mathbf{U}_{Roe} = \overline{\mathbf{U}} = \mathbf{U}(\overline{\mathbf{W}})$$

平均增长矩阵为：

$$\widetilde{\mathbf{A}}(\mathbf{U}_R, \mathbf{U}_L) = \mathbf{A}(\overline{\mathbf{U}})$$

Roe 平均变量计算公式：

$$\bar{\rho} = \left[\frac{\sqrt{\rho_L} + \sqrt{\rho_R}}{2}\right]^2$$
$$\bar{u} = \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$
$$\bar{H} = \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}}$$

其他量计算：

$$\bar{p} = \frac{\gamma - 1}{\gamma}\left(\bar{\rho}\bar{H} - \frac{1}{2}\bar{\rho}\bar{u}^2\right), \quad \bar{c}^2 = (\gamma - 1)\left(\bar{H} - \frac{\bar{u}^2}{2}\right)$$

2.4.5 Roe 格式计算步骤

设 n 时刻所有网格点物理量已知，构造 $\left(\frac{\partial \mathbf{F}(\mathbf{U})}{\partial x}\right)_j$：

1. 运用差分格式计算 $j + \frac{1}{2}$ 处的左右数值守恒通量 $\mathbf{U}^L_{j+\frac{1}{2}}$ 与 $\mathbf{U}^R_{j+\frac{1}{2}}$
2. 采用 Roe 平均公式计算平均守恒矢量 $\overline{\mathbf{U}}$，获得 $\bar{\rho}, \bar{u}, \bar{H}, \bar{p}, \bar{c}$ 值
3. 将 Jacobian 矩阵 $\mathbf{A}(\mathbf{U})$ 特征分解：$\widetilde{\mathbf{A}} = \mathbf{S}^{-1}\Lambda\mathbf{S}$

4. 计算 $\left|\widetilde{\mathbf{A}}\right| = \mathbf{S}^{-1}|\Lambda|\mathbf{S}$，其中 $|\Lambda| = \mathrm{diag}(|\lambda_1|, |\lambda_2|, |\lambda_3|)$
5. 进行熵修正（接近零的特征值处增加耗散）：

$$f(\lambda) = \begin{cases} |\lambda| & |\lambda| > \varepsilon \\ \dfrac{\lambda^2 + \varepsilon^2}{2\varepsilon} & |\lambda| \leq \varepsilon \end{cases}$$

6. 计算数值通量：

$$\mathbf{F}_{j+\frac{1}{2}} = \frac{1}{2}\left[\mathbf{F}(\mathbf{U}_R) + \mathbf{F}(\mathbf{U}_L)\right] - \frac{1}{2}\left|\widetilde{\mathbf{A}}\right|(\mathbf{U}_R - \mathbf{U}_L)$$

7. 差分重构：

$$\left(\frac{\partial \mathbf{F}(\mathbf{U})}{\partial x}\right)_j = \frac{\mathbf{F}_{j+\frac{1}{2}} - \mathbf{F}_{j-\frac{1}{2}}}{\Delta x}$$

2.5 激波捕捉格式

在可压缩流动的流场中可能存在激波（或间断）。Euler 方程描述的流场中激波是 Reynolds 数趋近无穷大的极限情况，此时激波厚度为零。数值计算需要准确捕捉激波位置、速度和满足激波跳跃条件的流动参数。激波存在导致流场特征尺度差异大（无黏激波厚度为零，流场特征长度有限），流动参数穿过激波有间断，给数值计算带来困难。

2.5.1 非物理振荡根源

激波数值模拟中非物理振荡的主要根源：

1. 粘性不足: 发展出人工粘性方法 (Artificial Viscosity)
2. 差分格式失去单调性: 发展了 TVD (Total Variation Diminishing Methods) 等保单调限制器方法
3. 色散误差导致间断色散: 发展了群速度控制方法 (GVC)、耗散比拟方法

其他高效激波捕捉格式包括 NND (Non-oscillatory, Non-free-parameters Dissipative Difference Scheme)、ENO (Essentially Non-oscillatory)、WENO (Weighted Essentially Non-oscillatory Scheme) 等。

2.5.2 TVD 格式 (Total Variation Diminishing Methods)

TVD 意为" 总变差不增"，其主要思想是改造现有格式使其符合 Harten 条件：

$$u_j^{n+1} = u_j^n + C(u_{j+1}^n - u_j^n) - D(u_j^n - u_{j-1}^n) \tag{17}$$

其中需满足 $C \geq 0$，$D \geq 0$，$C + D \leq 1$。

基于一阶迎风格式，增加修正项并使用限制函数，在格点 $j+\frac{1}{2}$ 处的差分格式为：

$$\text{正通量情况 } (a > 0): \quad u_{j+\frac{1}{2}}^L = u_j + \varphi_j^L \cdot \frac{u_{j+1} - u_j}{2} \tag{18}$$

$$\text{负通量情况 } (a < 0): \quad u_{j-\frac{1}{2}}^R = u_j - \varphi_j^R \cdot \frac{u_j - u_{j-1}}{2} \tag{19}$$

其中：

$$\varphi_j^L = \varphi(r_j^L), \quad r_j^L = \frac{u_j - u_{j-1}}{u_{j+1} - u_j}$$
$$\varphi_j^R = \varphi(r_j^R), \quad r_j^R = \frac{u_{j+1} - u_j}{u_j - u_{j-1}}$$

限制器 (Limiter) $\varphi(r)$ 通过判断区间光滑程度，平衡格式的光滑性和低耗散特性。

本文选用 van Leer 型限制器：

$$\varphi(r) = \frac{r + |r|}{1 + |r|} = \frac{|r| + r}{|r| + 1} \tag{20}$$

该限制函数光滑性良好。

2.5.3 WENO 格式 (Weighted Essentially Non-oscillatory Scheme)

WENO 格式特点：保持高精度的同时具有优异的鲁棒性 (Robustness)。

基本构造思路

1. 高精度逼近 $\partial U/\partial x$ 需多个基架点

2. 基架点内含激波会导致数值振荡

3. 将基架点分为多个模板 (组)，各模板独立计算导数逼近
4. 基于模板光滑程度设定权重
5. 加权平均多个差分结果: 光滑度高的模板权重高，含间断的模板权重趋于零

5 阶 WENO 格式 针对单波方程（正通量 $a>0$）：

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0,\quad f(u)=au,\quad a>0 \tag{21}$$

数值通量计算：

$$f_{j+\frac{1}{2}}^{\text{WENO}}=\omega_1 f_{j+\frac{1}{2}}^{(1)}+\omega_2 f_{j+\frac{1}{2}}^{(2)}+\omega_3 f_{j+\frac{1}{2}}^{(3)} \tag{22}$$

$$f_{j+\frac{1}{2}}^{(1)}=\frac{1}{3}f_{j-2}-\frac{7}{6}f_{j-1}+\frac{11}{6}f_j \tag{23}$$

$$f_{j+\frac{1}{2}}^{(2)}=-\frac{1}{6}f_{j-1}+\frac{5}{6}f_j+\frac{1}{3}f_{j+1} \tag{24}$$

$$f_{j+\frac{1}{2}}^{(3)}=\frac{1}{3}f_j+\frac{5}{6}f_{j+1}-\frac{1}{6}f_{j+2} \tag{25}$$

$$\omega_k=\frac{\alpha_k}{\alpha_1+\alpha_2+\alpha_3},\quad \alpha_k=\frac{C_k}{(\varepsilon+\text{IS}_k)^p},\quad k=1,2,3 \tag{26}$$

$$p=2,\quad \varepsilon=10^{-6},\quad C_1=\frac{1}{10},\quad C_2=\frac{6}{10},\quad C_3=\frac{3}{10} \tag{27}$$

光滑度量因子：

$$\text{IS}_1=\frac{1}{4}(f_{j-2}-4f_{j-1}+3f_j)^2+\frac{13}{12}(f_{j-2}-2f_{j-1}+f_j)^2 \tag{28}$$

$$\text{IS}_2=\frac{1}{4}(f_{j-1}-f_{j+1})^2+\frac{13}{12}(f_{j-1}-2f_j+f_{j+1})^2 \tag{29}$$

$$\text{IS}_3=\frac{1}{4}(3f_j-4f_{j+1}+f_{j+2})^2+\frac{13}{12}(f_j-2f_{j+1}+f_{j+2})^2 \tag{30}$$

差分重构：

$$\frac{\partial f_j^{\text{WENO}}}{\partial x}=\frac{f_{j+\frac{1}{2}}^{\text{WENO}}-f_{j-\frac{1}{2}}^{\text{WENO}}}{\Delta x} \tag{31}$$

负通量情况 $(a<0)$ 通过对称性处理：将下标"$j+k$" 改为"$j-k$"，差分重构方式不变。

2.6 群速度控制 (GVC) 方法

2.6.1 方法原理与背景

在激波数值模拟中，色散误差会导致数值解在间断处出现非物理振荡。群速度控制（Group Velocity Control, GVC）方法通过引入限制器函数调节通量梯度，抑制色散误差引起的数值振荡，提高数值解的稳定性与准确性。

方法核心思想是：在通量重构过程中引入基于通量梯度比的限制器函数，通过可调节参数 β 和 γ 控制群速度效应，平衡高阶精度与数值稳定性。该方法适用于高速流动 $(c > 1)$ 情况下的激波捕捉问题。

2.6.2 群速度控制限制器函数

群速度控制的核心是下列限制器函数：

$$\Phi(r) = \frac{r + \beta|r|}{1 + \beta r + \gamma r^2}$$

其中：

- r 为通量梯度比，衡量局部解的平滑程度
- β 为群速度控制因子，通常取 0.8
- γ 为高阶修正系数，通常取 0.3

2.6.3 正负通量处理

群速度控制方法分别处理正通量 (F_p) 和负通量 (F_n)：

正通量梯度比计算

$$r_p = \frac{F_p[j] - F_p[j-1]}{F_p[j+1] - F_p[j] + \varepsilon}$$

其中 $\varepsilon = 10^{-5}$ 是为避免除零而引入的小量。

负通量梯度比计算

$$r_n = \frac{F_n[j+2] - F_n[j+1]}{F_n[j+1] - F_n[j] + \varepsilon}$$

2.6.4 半网格点通量重构

利用限制器函数重构半网格点通量：

正通量半网格点重构

$$Fh_p[j] = F_p[j] + \frac{1}{2}\Phi_p(r_p) \cdot (F_p[j+1] - F_p[j])$$

负通量半网格点重构

$$Fh_n[j] = F_n[j+1] - \frac{1}{2}\Phi_n(r_n) \cdot (F_n[j+1] - F_n[j])$$

其中 Φ_p 和 Φ_n 分别为正负通量的限制器函数：

$$\Phi_p = \Phi(r_p)$$

$$\Phi_n = \Phi(r_n)$$

2.6.5 通量导数计算

通过重构后的半网格点通量计算通量导数：

正通量导数

$$Fx_p[j] = \frac{Fh_p[j] - Fh_p[j-1]}{\Delta x}$$

负通量导数

$$Fx_n[j] = \frac{Fh_n[j] - Fh_n[j-1]}{\Delta x}$$

总通量导数

$$Fx[j] = Fx_p[j] + Fx_n[j]$$

2.6.6 参数分析与优化

控制参数影响

- β（群速度控制因子）：影响低频区域精度

$$\beta \uparrow \Rightarrow \text{数值耗散} \downarrow$$
$$\beta \downarrow \Rightarrow \text{数值稳定性} \uparrow$$

  推荐值：$0.6 \leq \beta \leq 0.9$

- γ（高阶修正系数）：控制高频分量处理

$$\gamma \uparrow \Rightarrow \text{色散误差} \downarrow$$
$$\gamma \downarrow \Rightarrow \text{相位精度} \uparrow$$

  推荐值：$0.2 \leq \gamma \leq 0.5$

速度适应性调整　对于高速流动 ($c > 1$)，建议采用速度自适应的参数调整：

$$\beta_{\text{adaptive}} = \beta \cdot \frac{1}{c}$$
$$\gamma_{\text{adaptive}} = \gamma \cdot \frac{1}{c^2}$$

2.6.7 性能评估

群速度控制方法的优势在于：能有效抑制高速流动中的数值振荡，保持良好数值稳定性（CFL 条件可适度放宽），且实现简单，计算开销低。这些特性使其适用于需要稳定求解的高速流动问题。

然而，该方法也存在一些局限：当流速 $c > 1$ 时会产生约 15-30% 的精度损失（具体损失程度与 β, γ 参数相关）；在强激波附近可能出现轻微过冲（一般小于 5%）；为保证数值稳定性，时间步长通常需要缩小 15-20%。这些局限性需要通过参数优化或混合格式来改善。

2.7 特征重构方法在 FVS 框架中的应用与影响

2.7.1 应用方法

在流通矢量分裂 (FVS) 框架中应用特征重构方法的核心是通过以下步骤实现：

1. **特征场投影：**

$$\mathbf{LU} = \mathbf{W}$$

其中 $\mathbf{L}$ 为左特征向量矩阵，$\mathbf{W}$ 为特征变量。

2. **特征空间重构：**

$$\mathbf{W}_{j+\frac{1}{2}}^{L} = \mathbf{W}_j + \frac{1}{2}\phi_j^L(\mathbf{W}_j - \mathbf{W}_{j-1})$$

$$\mathbf{W}_{j+\frac{1}{2}}^{R} = \mathbf{W}_{j+1} - \frac{1}{2}\phi_{j+1}^R(\mathbf{W}_{j+1} - \mathbf{W}_j)$$

其中 ϕ 为特征空间限制器函数。

3. **物理空间反投影：**

$$\mathbf{U}_{j+\frac{1}{2}}^{L} = \mathbf{R}\mathbf{W}_{j+\frac{1}{2}}^{L}, \quad \mathbf{U}_{j+\frac{1}{2}}^{R} = \mathbf{R}\mathbf{W}_{j+\frac{1}{2}}^{R}$$

$\mathbf{R}$ 为右特征向量矩阵。

4. **FVS 通量分裂：**

$$\mathbf{F}_{j+\frac{1}{2}} = \mathbf{F}^{+}(\mathbf{U}_{j+\frac{1}{2}}^{L}) + \mathbf{F}^{-}(\mathbf{U}_{j+\frac{1}{2}}^{R})$$

5. **通量梯度计算：**

$$\frac{\partial \mathbf{F}}{\partial x} = \frac{\mathbf{F}_{j+\frac{1}{2}} - \mathbf{F}_{j-\frac{1}{2}}}{\Delta x}$$

3 代码实现

模拟激波管的代码实现如下：

首先是基本参数设定和边界条件设定：

```
#管道特征长度 (x = [-L/2, L/2])

L = 1.0

# 流动参数
Gamma = 1.4  # 比热比
R = 286.9   # 气体常数 (J/kg·K)
Cv = R / (Gamma - 1)            # 定容比热容
```

```
Cp = (Gamma * R) / (Gamma - 1)  # 定压比热容

# 网格生成
N = 201   # x方向网格数
xp = np.linspace(-L / 2, L / 2, N)  # 网格点的x坐标
dx = L / (N - 1)  # 网格间距

xp_mid = N // 2  # x=0位置的网格索引(中点)

# 物理量数组初始化
u_arr = np.zeros(N)      # 速度数组
rho_arr = np.zeros(N)    # 密度数组
p_arr = np.zeros(N)      # 压力数组

# 辅助数组
rho = np.zeros(N)
u = np.zeros(N)
p = np.zeros(N)
E = np.zeros(N)
T = np.zeros(N)
c = np.zeros(N)

# 时间步长
dt = 0.001

# 最大计算步数
max_step = 200

# 最大模拟时间
max_tot_time = max_step * dt

#设置初始条件
# 当x < 0时，(左侧: ul, rhol, pl) = (0.0, 1.0, 1.0)
```

```python
# 当x >= 0时，(右侧: ur, rhor, pr) = (0.0, 0.125, 0.1)
u_arr[0:xp_mid-1] = 0.0
rho_arr[0:xp_mid-1] = 1.0
p_arr[0:xp_mid-1] = 1.0

u_arr[xp_mid-1:] = 0.0
rho_arr[xp_mid-1:] = 0.125
p_arr[xp_mid-1:] = 0.1
```

Listing 1: 基础参数设定和边界条件

其中边界条件设定由题意，左边界为 $P_1 = 1, \rho_1 = 1$，右边界为 $P_2 = 0.1, \rho_2 = 0.125$。网格数为 $N = 200$，时间步长为 $\Delta t = 0.001s$，总时间为 $T = 0.2s$。

接下来是数据初始化和算法预处理选择：

```python
#预处理：算法选择设置
#通量分裂方法选择
class FluxSplittingMethod(Enum):
    FVS = 1  # 通量向量分裂
    FDS = 2  # 通量差分裂
#通量向量分裂方法FVS选择
class FVSMethods(Enum):
    STEGER_WARMING = 1   # Steger-Warming方法
    LAX_FRIEDRICH = 2    # Lax-Friedrich方法
    VAN_LEER = 3         # Van Leer方法
#通量差分裂方法FDS选择
class FDSMethods(Enum):
    ROE = 1  # ROE格式
#空间离散格式选择
class SpatialDiscretizationType(Enum):
    SHOCK_CAPTURING = 1  # 激波捕捉格式
#激波捕捉格式选择
class ShockCapturingScheme(Enum):
    TVD_VAN_LEER = 1  # TVD格式(Van Leer限制器)
    WENO = 2          # 5阶WENO格式
```

```
    GVC = 3             #群速度控制格式


# 指定通量分裂方法
# 1 - 通量向量分裂(FVS)
# 2 - 通量差分裂(FDS)
flag_flu_spl = FluxSplittingMethod.FDS.value

# FVS方法家族选择
# 1 - Steger-Warming (S-W)
# 2 - Lax-Friedrich (L-F)
# 3 - Van Leer
flag_fvs_met = FVSMethods.VAN_LEER.value

# 指定通量重构方法
# 1 - 直接重构(针对F(U))
# 2 - 特征重构
flag_flu_rec = 1

# FDS方法家族选择(FDS-ROE格式)
flag_fds_met = FDSMethods.ROE.value

# 指定高分辨率通量空间离散格式
flag_spa_typ = SpatialDiscretizationType.SHOCK_CAPTURING.value  # 激波捕捉格式

# 激波捕捉格式类型选择
flag_scs_typ = ShockCapturingScheme.WENO.value

#数组初始化
lambda_arr = np.zeros(3)  # 特征值矩阵
lambda_p_arr = np.zeros(3)  # 正特征值矩阵
lambda_n_arr = np.zeros(3)  # 负特征值矩阵
```

```

# 初始化解向量数组
U = np.zeros((N, 3))  # 守恒变量向量
F_p = np.zeros((N, 3))  # 正通量向量
F_n = np.zeros((N, 3))  # 负通量向量
Fx = np.zeros((N, 3))  # 通量散度项
Fx_p = np.zeros((N, 3))  # 正通量散度项
Fx_n = np.zeros((N, 3))  # 负通量散度项

# 半网格点通量数组
Fh_p = np.zeros((N - 1, 3))  # (j + 1/2)半网格点正通量向量
Fh_n = np.zeros((N - 1, 3))  # (j + 1/2)半网格点负通量向量

# 计算初始密度、速度、压力、温度、声速和守恒变量
for i in range(N):
    # 从数组获取当前网格点的值
    rho_val = rho_arr[i]
    u_val = u_arr[i]
    p_val = p_arr[i]

    # 计算温度 (理想气体状态方程)
    T_val = p_val / (rho_val * R)

    # 计算声速 (等熵关系)
    c_val = np.sqrt(Gamma * p_val / rho_val)

    # 计算总能量 (内能 + 动能)
    E_val = rho_val * ((Cv * T_val) + (0.5 * u_val * u_val)
        )

    # 存储守恒变量
    # U = [密度, 动量, 总能量]
    U[i, 0] = rho_val  # 密度 (kg/m³)
```

```
    U[i, 1] = rho_val * u_val  # 动量 (kg/m²·s)
    U[i, 2] = E_val  # 总能量 (J/m³)

    # 存储热力学量
    T[i] = T_val
    c[i] = c_val
```

Listing 2: 算法格式预处理

其中主要是通过选择不同的 flag 参数来选择不同的算法格式进行计算。

接下来是主循环部分，进行时间推进和数据更新：

```
# 开始计算和时间步进
# 实际计算时间
t = 0.0
cnt_step = 0

# 主计算循环
while cnt_step < max_step:

    # 更新时间
    t += dt
    cnt_step += 1

    # 时间离散格式选择三阶TVD龙格-库塔法
    if flag_flu_spl == 1:
        # 1 - 通量向量分裂法 (FVS)
        F_p, F_n = Flux_Vect_Split(U, N, Gamma, Cp, Cv, R,
           flag_fvs_met)
        _, _, _, _, Fx, _, _ = Diff_Cons(N, dx, F_p, F_n,
           flag_spa_typ, flag_scs_typ)

        # 第一步: U1 = U^n + Δt * Q(U^n)
        U_1 = U + (dt * ((-1) * Fx))

```

```
        # 使用U1计算Q(U1)
        F_p_1, F_n_1 = Flux_Vect_Split(U_1, N, Gamma, Cp,
           Cv, R, flag_fvs_met)
        _, _, _, _, Fx_1, _, _ = Diff_Cons(N, dx, F_p_1,
           F_n_1, flag_spa_typ, flag_scs_typ)

        # 第二步: U2 = (3/4)U^n + (1/4)U1 + (1/4)Δt * Q(U1)
        U_2 = ((3 / 4) * U) + ((1 / 4) * U_1) + (((1 * dt)
           / 4) * ((-1) * Fx_1))

        # 使用U2计算Q(U2)
        F_p_2, F_n_2 = Flux_Vect_Split(U_2, N, Gamma, Cp,
           Cv, R, flag_fvs_met)
        xs_new, xt_new, Fh_p, Fh_n, Fx_2, Fx_p, Fx_n =
           Diff_Cons(N, dx, F_p_2, F_n_2, flag_spa_typ,
           flag_scs_typ)

        # 最终更新: U^{n+1} = (1/3)U^n + (2/3)U2 + (2/3)Δt
           * Q(U2)
        U = ((1 / 3) * U) + ((2 / 3) * U_2) + (((2 / 3) *
           dt) * ((-1) * Fx_2))

    elif flag_flu_spl == 2:
        xs_new, xt_new, Fx = Flux_Diff_Split(U, N, dx,
           Gamma, Cp, Cv, R,flag_fds_met, flag_spa_typ,
           flag_scs_typ)
        #2 - 流通差分分裂法（FDS）
        # 第一步: 计算中间解U_1
        U_1 = U + (dt * (-1 * Fx))

        # 使用U_1再次计算通量
        _, _, Fx_1 = Flux_Diff_Split(U_1, N, dx, Gamma, Cp,
            Cv, R,flag_fds_met, flag_spa_typ,flag_scs_typ)
```

```
        # 第二步：计算中间解U_2
        U_2 = ((3 / 4) * U) + ((1 / 4) * U_1) + ((1 / 4) * dt * (-1 * Fx_1))

        # 使用U_2再次计算通量
        _, _, Fx_2 = Flux_Diff_Split(U_2, N, dx, Gamma, Cp, Cv, R,flag_fds_met, flag_spa_typ,flag_scs_typ)

        # 最终更新：计算新时间步的解
        U = ((1 / 3) * U) + ((2 / 3) * U_2) + ((2 / 3) * dt * (-1 * Fx_2))

# 保存时间t_end时刻的最终解U
U_end = U.copy()
t_end = t

# 计算时间t_end时刻的最终物理量
# 根据守恒变量U计算rho, u, E, T, p, c, H
rho_end = U_end[:, 0]             # 密度
u_end = U_end[:, 1] / rho_end     # 速度
E_end = U_end[:, 2]               # 单位质量总内能
# 计算温度：T = (e_total - 0.5*u²) / Cv
T_end = (E_end / rho_end - 0.5 * u_end**2) / Cv
# 计算压力：p =  RT
p_end = rho_end * R * T_end
# 计算声速：c = √( p/ )
c_end = np.sqrt(Gamma * p_end / rho_end)
# 计算焓：H = 0.5*u² + Cp*T
H_end = 0.5 * u_end**2 + Cp * T_end
# 计算内能：e = E/  - 0.5*u²
e_end = E_end / rho_end - 0.5 * u_end**2
```

Listing 3: 主循环部分

其中采用了三阶 Runge-Kutta 方法进行时间推进，计算每个时间步的守恒量和通量。然后根据之前选择的 flag 参数，进入不同的分支计算 F_x。比如如果 `flag_flu_spl` 为 1，则使用通量矢量分裂法 `Flux_Vect_Split` 进行计算。如果 `flag_flu_spl` 为 2，则使用流通差分分裂法 `Flux_Diff_Split` 计算。

给出通量矢量分裂法的具体实现：

```
#通量向量分裂方法(FVS)通用函数(Steger-Warming,Lax-Friedrich,Van Leer)
def Flux_Vect_Split(U, N, Gamma, Cp, Cv, R, flag_fvs_met):
    if flag_fvs_met == 1:
        #  FVS - Steger-Warming (S-W)方法

        # 初始化变量
        F_p = np.zeros((N, 3))  # 正通量分量
        F_n = np.zeros((N, 3))  # 负通量分量
        em = 1e-3  # 小量防止除零

        for i in range(N):
            # 计算密度、速度、温度、压力、声速
            rho = U[i, 0]
            u = U[i, 1] / rho
            E = U[i, 2]
            # 计算温度: T = (e_total - 0.5*u²) / Cv
            T = (E / rho - 0.5 * u ** 2) / Cv
            p = rho * R * T
            c = np.sqrt(Gamma * p / rho)

            # 计算特征值
            lambda_vals = np.array([u, u - c, u + c])

            # 分裂特征值 (S-W方法)
            sqrt_term = np.sqrt(lambda_vals ** 2 + em ** 2)
            lambda_p = (lambda_vals + sqrt_term) / 2
```

```
            lambda_n = (lambda_vals - sqrt_term) / 2

            # 计算F+和F-
            # 计算w项
            w_p = ((3 - Gamma) * (lambda_p[1] + lambda_p
                [2]) * c ** 2) / (2 * (Gamma - 1))
            w_n = ((3 - Gamma) * (lambda_n[1] + lambda_n
                [2]) * c ** 2) / (2 * (Gamma - 1))

            # 计算正通量分量
            F_p[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma -
                 1) * lambda_p[0] + lambda_p[1] + lambda_p
                [2])
            F_p[i, 1] = (rho / (2 * Gamma)) * (
                        2 * (Gamma - 1) * lambda_p[0] * u +
                            lambda_p[1] * (u - c) +
                           lambda_p[2] * (u + c))
            F_p[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1)
                * lambda_p[0] * u ** 2 + (lambda_p[1] * (u -
                 c) ** 2) / 2 + (
                        lambda_p[2] * (u + c) ** 2) / 2 +
                           w_p)

            # 计算负通量分量
            F_n[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma -
                 1) * lambda_n[0] + lambda_n[1] + lambda_n
                [2])
            F_n[i, 1] = (rho / (2 * Gamma)) * (
                        2 * (Gamma - 1) * lambda_n[0] * u +
                            lambda_n[1] * (u - c) +
                           lambda_n[2] * (u + c))
            F_n[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1)
                * lambda_n[0] * u ** 2 + (lambda_n[1] * (u -
```

```python
                    c) ** 2) / 2 + (
                            lambda_n[2] * (u + c) ** 2) / 2 +
                               w_n)

    elif flag_fvs_met == 2:
        # FVS - Lax-Friedrich (L-F)方法

        # 初始化变量
        F_p = np.zeros((N, 3))  # 正通量分量
        F_n = np.zeros((N, 3))  # 负通量分量

        # 先循环计算全局最大特征值
        lambda_s_global = 0
        for i in range(N):
            rho = U[i, 0]
            u = U[i, 1] / rho
            E = U[i, 2]
            T = (E / rho - 0.5 * u ** 2) / Cv
            p = rho * R * T
            c = np.sqrt(Gamma * p / rho)

            # 计算局部特征值并更新全局最大值
            lambda_s_local = abs(u) + c
            if lambda_s_local > lambda_s_global:
                lambda_s_global = lambda_s_local

        # 再次循环计算通量分量
        for i in range(N):
            # 计算物理量
            rho = U[i, 0]
            u = U[i, 1] / rho
            E = U[i, 2]
            T = (E / rho - 0.5 * u ** 2) / Cv
```

```
        p = rho * R * T
        c = np.sqrt(Gamma * p / rho)

        # 计算特征值
        lambda_vals = np.array([u, u - c, u + c])

        # 分裂特征值 (L-F方法使用全局最大特征值)
        lambda_p = (lambda_vals + lambda_s_global) / 2
        lambda_n = (lambda_vals - lambda_s_global) / 2

        # 计算w项
        w_p = ((3 - Gamma) * (lambda_p[1] + lambda_p[2]) * c ** 2) / (2 * (Gamma - 1))
        w_n = ((3 - Gamma) * (lambda_n[1] + lambda_n[2]) * c ** 2) / (2 * (Gamma - 1))

        # 计算正通量分量
        F_p[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma - 1) * lambda_p[0] + lambda_p[1] + lambda_p[2])
        F_p[i, 1] = (rho / (2 * Gamma)) * (
                    2 * (Gamma - 1) * lambda_p[0] * u + lambda_p[1] * (u - c) + lambda_p[2] * (u + c))
        F_p[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1) * lambda_p[0] * u ** 2 + (lambda_p[1] * (u - c) ** 2) / 2 + (
                    lambda_p[2] * (u + c) ** 2) / 2 + w_p)

        # 计算负通量分量
        F_n[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma - 1) * lambda_n[0] + lambda_n[1] + lambda_n
```

```
[2])
            F_n[i, 1] = (rho / (2 * Gamma)) * (
                        2 * (Gamma - 1) * lambda_n[0] * u + lambda_n[1] * (u - c) +
                        lambda_n[2] * (u + c))
            F_n[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1) * lambda_n[0] * u ** 2 + (lambda_n[1] * (u - c) ** 2) / 2 + (
                        lambda_n[2] * (u + c) ** 2) / 2 + w_n)

    elif flag_fvs_met == 3:
        # Van Leer方法

        # 初始化变量
        F_p = np.zeros((N, 3))  # 正通量分量
        F_n = np.zeros((N, 3))  # 负通量分量

        for i in range(N):
            # 计算物理量
            rho = U[i, 0]
            u = U[i, 1] / rho
            E = U[i, 2]
            T = (E / rho - 0.5 * u ** 2) / Cv
            p = rho * R * T
            c = np.sqrt(Gamma * p / rho)

            # 计算马赫数
            Ma = u / c if c != 0 else 0

            # Van Leer通量分量计算公式
            if Ma >= 1:
                # 纯超音速正向流
```

```python
            F_p[i, 0] = U[i, 1]  # rho * u
            F_p[i, 1] = (Gamma - 1) * U[i, 2] + ((3 - Gamma) / 2) * (U[i, 1] ** 2 / U[i, 0])
            F_p[i, 2] = Gamma * (U[i, 1] * U[i, 2]) / U[i, 0] + ((Gamma - 1) / 2) * (U[i, 1] ** 3 / U[i, 0] ** 2)

            F_n[i, :] = 0.0

        elif Ma <= -1:
            # 纯超音速负向流
            F_n[i, 0] = U[i, 1]  # rho * u
            F_n[i, 1] = (Gamma - 1) * U[i, 2] + ((3 - Gamma) / 2) * (U[i, 1] ** 2 / U[i, 0])
            F_n[i, 2] = Gamma * (U[i, 1] * U[i, 2]) / U[i, 0] + ((Gamma - 1) / 2) * (U[i, 1] ** 3 / U[i, 0] ** 2)

            F_p[i, :] = 0.0

        else:
            # 亚音速/跨音速区域
            # 计算质量通量
            F1_p = rho * c * ((Ma + 1) / 2) ** 2
            F1_n = -rho * c * ((Ma - 1) / 2) ** 2

            # 计算正通量分量
            F_p[i, 0] = F1_p
            F_p[i, 1] = (F1_p / Gamma) * ((Gamma - 1) * u + 2 * c)
            F_p[i, 2] = (F1_p / (2 * (Gamma ** 2 - 1))) * ((Gamma - 1) * u + 2 * c) ** 2

```

```
                # 计算负通量分量
                F_n[i, 0] = F1_n
                F_n[i, 1] = (F1_n / Gamma) * ((Gamma - 1) * u - 2 * c)
                F_n[i, 2] = (F1_n / (2 * (Gamma ** 2 - 1))) * ((Gamma - 1) * u - 2 * c) ** 2

    return F_p, F_n
```

Listing 4: 通量矢量分裂法实现

根据不同 flag_fvs_met 参数，通量矢量分裂法的三种实现如下：

1. **Steger-Warming (flag_fvs_met=1)** Steger-Warming 分裂法将特征值分为正负部分，并在实际实现中引入小量 ε 避免零点奇异：

$$\lambda_k^{\pm} = \frac{\lambda_k \pm \sqrt{\lambda_k^2 + \varepsilon^2}}{2}$$

其中 λ_k 为雅可比矩阵的特征值，ε 为极小正数。

对每个网格点，首先计算密度 ρ、速度 u、总能 E、温度 T、压力 p 和声速 c，再得到特征值 $\lambda_1 = u$，$\lambda_2 = u - c$，$\lambda_3 = u + c$，并进行正负分裂。

通量分裂公式为：

$$\begin{aligned}
F_0^+ &= \frac{\rho}{2\gamma}\left[2(\gamma-1)\lambda_1^+ + \lambda_2^+ + \lambda_3^+\right] \\
F_1^+ &= \frac{\rho}{2\gamma}\left[2(\gamma-1)\lambda_1^+ u + \lambda_2^+(u-c) + \lambda_3^+(u+c)\right] \\
F_2^+ &= \frac{\rho}{2\gamma}\left[(\gamma-1)\lambda_1^+ u^2 + \frac{1}{2}\lambda_2^+(u-c)^2 + \frac{1}{2}\lambda_3^+(u+c)^2 + w^+\right]
\end{aligned}$$

其中

$$w^+ = \frac{3-\gamma}{2(\gamma-1)}(\lambda_2^+ + \lambda_3^+)c^2$$

负通量 F^- 同理，λ_k^+ 换为 λ_k^-，w^+ 换为 w^-。

这种实现方式无需显式构造特征向量矩阵，直接利用特征值分裂和物理量表达式完成通量分裂，数值稳定性好，适合实际编程实现。

2. **Lax-Friedrichs (flag_fvs_met=2)** Lax-Friedrichs 分裂法采用全局最大特征值 λ^* 进行分裂。具体实现如下：

首先，遍历所有网格点，计算每个点的速度 u 和声速 c，得到局部特征值 $|u|+c$，并取其最大值作为全局 λ^*。

然后，对每个网格点，计算守恒量 ρ, u, E，温度 T，压力 p，声速 c，以及特征值 $\lambda_1 = u, \lambda_2 = u - c, \lambda_3 = u + c$。L-F 方法将特征值分裂为：

$$\lambda_k^+ = \frac{\lambda_k + \lambda^*}{2}, \quad \lambda_k^- = \frac{\lambda_k - \lambda^*}{2}$$

其中 $\lambda^* = \max(|u| + c)$。

正通量和负通量分量分别为：

$$\begin{aligned}
F_0^+ &= \frac{\rho}{2\gamma}\left[2(\gamma-1)\lambda_1^+ + \lambda_2^+ + \lambda_3^+\right] \\
F_1^+ &= \frac{\rho}{2\gamma}\left[2(\gamma-1)\lambda_1^+ u + \lambda_2^+(u-c) + \lambda_3^+(u+c)\right] \\
F_2^+ &= \frac{\rho}{2\gamma}\left[(\gamma-1)\lambda_1^+ u^2 + \frac{1}{2}\lambda_2^+(u-c)^2 + \frac{1}{2}\lambda_3^+(u+c)^2 + w^+\right] \\
F_0^- &= \frac{\rho}{2\gamma}\left[2(\gamma-1)\lambda_1^- + \lambda_2^- + \lambda_3^-\right] \\
F_1^- &= \frac{\rho}{2\gamma}\left[2(\gamma-1)\lambda_1^- u + \lambda_2^-(u-c) + \lambda_3^-(u+c)\right] \\
F_2^- &= \frac{\rho}{2\gamma}\left[(\gamma-1)\lambda_1^- u^2 + \frac{1}{2}\lambda_2^-(u-c)^2 + \frac{1}{2}\lambda_3^-(u+c)^2 + w^-\right]
\end{aligned}$$

其中

$$w^+ = \frac{3-\gamma}{2(\gamma-1)}(\lambda_2^+ + \lambda_3^+)c^2, \quad w^- = \frac{3-\gamma}{2(\gamma-1)}(\lambda_2^- + \lambda_3^-)c^2$$

这样即可实现 Lax-Friedrichs 分裂的 FVS 通量计算。

3. **van Leer (flag_fvs_met=3)** van Leer 分裂法基于马赫数 $\mathrm{Ma} = u/c$，其数值实现如下：

- 对每个网格点，计算密度 ρ、速度 u、总能 E、温度 T、压力 p 和声速 c，再得到马赫数 $\mathrm{Ma} = u/c$。
- 若 $\mathrm{Ma} \geq 1$，则正通量 $\mathbf{F}^+$ 取完整物理通量，负通量 $\mathbf{F}^-$ 为零；
- 若 $\mathrm{Ma} \leq -1$，则负通量 $\mathbf{F}^-$ 取完整物理通量，正通量 $\mathbf{F}^+$ 为零；

- 若 $|\mathrm{Ma}| < 1$，则

$$
\begin{aligned}
F_1^+ &= \rho c\left(\frac{\mathrm{Ma}+1}{2}\right)^2 \\
F_1^- &= -\rho c\left(\frac{\mathrm{Ma}-1}{2}\right)^2 \\
F_2^+ &= \frac{F_1^+}{\gamma}\left[(\gamma-1)u+2c\right] \\
F_2^- &= \frac{F_1^-}{\gamma}\left[(\gamma-1)u-2c\right] \\
F_3^+ &= \frac{F_1^+}{2(\gamma^2-1)}\left[(\gamma-1)u+2c\right]^2 \\
F_3^- &= \frac{F_1^-}{2(\gamma^2-1)}\left[(\gamma-1)u-2c\right]^2
\end{aligned}
$$

- 最终正负通量分量分别为 $\mathbf{F}^+ = (F_1^+, F_2^+, F_3^+)$，$\mathbf{F}^- = (F_1^-, F_2^-, F_3^-)$。

这种实现方式可有效区分超音速与亚音速区域，保证通量分裂的物理一致性和数值稳定性。

实际代码实现中，通过判断 flag_fvs_met 的值，选择对应的分裂方法进行通量计算。这样可灵活切换不同的 FVS 分裂策略，便于对比分析其对激波捕捉和数值耗散的影响。

根据通量矢量分裂法，使用激波捕捉格式进行激波管模拟：

```
#通用通量差分计算函数(TVD,WENO,GVC)，返回索引和通量导数
def Diff_Cons(N, dx, F_p, F_n, flag_spa_typ, flag_scs_typ):
    # 初始化输出数组
    xs_new = 0
    xt_new = 0
    Fx = np.zeros((N, 3))  # 总通量导数
    Fx_p = np.zeros((N, 3))  # 正通量导数
    Fx_n = np.zeros((N, 3))  # 负通量导数
    Fh_p = np.zeros((N, 3))  # 半网格点正通量 (j+1/2)
    Fh_n = np.zeros((N, 3))  # 半网格点负通量 (j+1/2)
    Fh = np.zeros((N, 3))  # 半网格点总通量 (j+1/2)

```

```python
    if flag_spa_typ == 1:
        # 特殊激波捕捉格式

        if flag_scs_typ == 1:
            # 1 - TVD格式 (Van Leer限制器)
            xs = 1  # 起始索引
            xt = N - 2  # 结束索引
            em = 1e-5  # 小量防止除零

            for j in range(xs, xt):
                # 计算正通量比例因子
                r_p_num = F_p[j] - F_p[j - 1]
                r_p_den = F_p[j + 1] - F_p[j] + em
                r_p = r_p_num / r_p_den

                # 计算负通量比例因子
                r_n_num = F_n[j + 2] - F_n[j + 1]
                r_n_den = F_n[j + 1] - F_n[j] + em
                r_n = r_n_num / r_n_den

                # Van Leer限制器
                Phi_p = (r_p + np.abs(r_p)) / (1 + r_p)
                Phi_n = (r_n + np.abs(r_n)) / (1 + r_n)

                # 计算半网格点通量
                Fh_p[j] = F_p[j] + 0.5 * Phi_p * (F_p[j + 1] - F_p[j])
                Fh_n[j] = F_n[j + 1] - 0.5 * Phi_n * (F_n[j + 1] - F_n[j])

            # 更新索引范围
            xs_new = xs + 1
            xt_new = xt
```

```python

            # 计算通量导数
            for j in range(xs_new, xt_new):
                Fx_p[j] = (Fh_p[j] - Fh_p[j - 1]) / dx
                Fx_n[j] = (Fh_n[j] - Fh_n[j - 1]) / dx
                Fx[j] = Fx_p[j] + Fx_n[j]

        elif flag_scs_typ == 2:
            # 2 - WENO5阶格式
            C = np.array([1 / 10, 6 / 10, 3 / 10])
            p = 2
            em = 1e-6

            xs = 2  # 起始索引
            xt = N - 3  # 结束索引

            for j in range(xs, xt + 1):
                # a > 0 正通量部分
                # 计算光滑指示器 beta
                beta_p1 = (1 / 4) * (F_p[j - 2] - 4 * F_p[j - 1] + 3 * F_p[j]) ** 2 + (13 / 12) * (
                            F_p[j - 2] - 2 * F_p[j - 1] + F_p[j]) ** 2
                beta_p2 = (1 / 4) * (F_p[j - 1] - F_p[j + 1]) ** 2 + (13 / 12) * (F_p[j - 1] - 2 * F_p[j] + F_p[j + 1]) ** 2
                beta_p3 = (1 / 4) * (3 * F_p[j] - 4 * F_p[j + 1] + F_p[j + 2]) ** 2 + (13 / 12) * (
                            F_p[j] - 2 * F_p[j + 1] + F_p[j + 2]) ** 2

                # 计算 alpha 和权重 omega
                alpha_p = C / (em + np.array([beta_p1,
```

```
                                    beta_p2, beta_p3])) ** p
            alpha_p_sum = np.sum(alpha_p, axis=0)
            omega_p = alpha_p / alpha_p_sum

            # 计算三个模板的通量值
            Fh_p_c1 = (1 / 3) * F_p[j - 2] - (7 / 6) * F_p[j - 1] + (11 / 6) * F_p[j]
            Fh_p_c2 = (-1 / 6) * F_p[j - 1] + (5 / 6) * F_p[j] + (1 / 3) * F_p[j + 1]
            Fh_p_c3 = (1 / 3) * F_p[j] + (5 / 6) * F_p[j + 1] - (1 / 6) * F_p[j + 2]
            Fh_p_c = np.array([Fh_p_c1, Fh_p_c2, Fh_p_c3])

            # 加权组合得到 F+_{j+1/2}
            Fh_p[j] = np.sum(omega_p * Fh_p_c, axis=0)

            # a < 0 负通量部分
            # 计算光滑指示器 beta
            beta_n1 = (1 / 4) * (F_n[j + 2] - 4 * F_n[j + 1] + 3 * F_n[j]) ** 2 + (13 / 12) * (
                        F_n[j + 2] - 2 * F_n[j + 1] + F_n[j]) ** 2
            beta_n2 = (1 / 4) * (F_n[j + 1] - F_n[j - 1]) ** 2 + (13 / 12) * (F_n[j + 1] - 2 * F_n[j] + F_n[j - 1]) ** 2
            beta_n3 = (1 / 4) * (3 * F_n[j] - 4 * F_n[j - 1] + F_n[j - 2]) ** 2 + (13 / 12) * (
                        F_n[j] - 2 * F_n[j - 1] + F_n[j - 2]) ** 2

            # 计算 alpha 和权重 omega
            alpha_n = C / (em + np.array([beta_n1,
```

```
                beta_n2, beta_n3])) ** p
                alpha_n_sum = np.sum(alpha_n, axis=0)
                omega_n = alpha_n / alpha_n_sum

                # 计算三个模板的通量值
                Fh_n_c1 = (1 / 3) * F_n[j + 2] - (7 / 6) * F_n[j + 1] + (11 / 6) * F_n[j]
                Fh_n_c2 = (-1 / 6) * F_n[j + 1] + (5 / 6) * F_n[j] + (1 / 3) * F_n[j - 1]
                Fh_n_c3 = (1 / 3) * F_n[j] + (5 / 6) * F_n[j - 1] - (1 / 6) * F_n[j - 2]
                Fh_n_c = np.array([Fh_n_c1, Fh_n_c2, Fh_n_c3])

                # 加权组合得到 F-_{j-1/2}
                Fh_n[j] = np.sum(omega_n * Fh_n_c, axis=0)

            # 计算通量导数
            xs_new = xs + 1  # 3
            xt_new = xt - 1  # N-4

            for j in range(xs_new, xt_new + 1):
                Fx_p[j] = (Fh_p[j] - Fh_p[j - 1]) / dx
                Fx_n[j] = (Fh_n[j + 1] - Fh_n[j]) / dx
                Fx[j] = Fx_p[j] + Fx_n[j]

        elif flag_scs_typ == 3:
            # 群速度控制格式
            # 参数设置
            xs = 1  # 起始索引
            xt = N - 2  # 结束索引
            em = 1e-5  # 小量防止除零

```

```
            # 群速度控制参数
            beta = 0.8  # 群速度控制因子
            gamma = 0.3  # 高阶修正系数

            for j in range(xs, xt):
                # 计算通量梯度比
                # 正通量部分
                r_p_num = F_p[j] - F_p[j - 1]
                r_p_den = F_p[j + 1] - F_p[j] + em
                r_p = r_p_num / r_p_den

                # 负通量部分
                r_n_num = F_n[j + 2] - F_n[j + 1]
                r_n_den = F_n[j + 1] - F_n[j] + em
                r_n = r_n_num / r_n_den

                # 群速度控制限制器函数
                # 正通量限制器
                Phi_p = (r_p + beta * np.abs(r_p)) / (1 + beta * r_p + gamma * r_p ** 2)
                # 负通量限制器
                Phi_n = (r_n + beta * np.abs(r_n)) / (1 + beta * r_n + gamma * r_n ** 2)

                # 计算半网格点通量（含群速度控制）
                Fh_p[j] = F_p[j] + 0.5 * Phi_p * (F_p[j + 1] - F_p[j])
                Fh_n[j] = F_n[j + 1] - 0.5 * Phi_n * (F_n[j + 1] - F_n[j])

            # 更新索引范围
            xs_new = xs + 1
            xt_new = xt
```

```python

            # 计算通量导数
            for j in range(xs_new, xt_new):
                Fx_p[j] = (Fh_p[j] - Fh_p[j - 1]) / dx
                Fx_n[j] = (Fh_n[j] - Fh_n[j - 1]) / dx
                Fx[j] = Fx_p[j] + Fx_n[j]
    return xs_new, xt_new, Fh_p, Fh_n, Fx, Fx_p, Fx_n
```

Listing 5: 激波捕捉格式实现

根据不同 flag_scs_typ 参数，通量矢量分裂法的三种实现如下：

1. **TVD (flag_scs_typ=1)**

TVD 格式采用 Van Leer 限制器，具体实现过程如下：

- 设定起始索引 $xs = 1$，结束索引 $xt = N - 2$，小量 $\varepsilon = 10^{-5}$ 防止除零
- 对每个 j 在 $[xs, xt)$ 区间
  1. 计算正通量梯度比：$r_p = \dfrac{F_p[j] - F_p[j-1]}{F_p[j+1] - F_p[j] + \varepsilon}$
  2. 计算负通量梯度比：$r_n = \dfrac{F_n[j+2] - F_n[j+1]}{F_n[j+1] - F_n[j] + \varepsilon}$
  3. 计算 Van Leer 限制器：$\Phi_p = \dfrac{r_p + |r_p|}{1 + r_p}$，$\Phi_n = \dfrac{r_n + |r_n|}{1 + r_n}$
  4. 半网格点通量重构：

$$
\begin{aligned}
Fh_p[j] &= F_p[j] + 0.5\,\Phi_p\,(F_p[j+1] - F_p[j]) \\
Fh_n[j] &= F_n[j+1] - 0.5\,\Phi_n\,(F_n[j+1] - F_n[j])
\end{aligned}
$$

- 更新索引范围 $xs_{new} = xs + 1$，$xt_{new} = xt$，计算通量导数：

$$
\begin{aligned}
Fx_p[j] &= \frac{Fh_p[j] - Fh_p[j-1]}{dx} \\
Fx_n[j] &= \frac{Fh_n[j] - Fh_n[j-1]}{dx} \\
Fx[j] &= Fx_p[j] + Fx_n[j]
\end{aligned}
$$

该实现保证了激波捕捉时的无振荡性和高分辨率

2. **五阶 WENO (flag_scs_typ=2)**

五阶 WENO 格式实现过程如下：

设常数 $C = [1/10, 6/10, 3/10]$，幂指数 $p = 2$，小量 $\varepsilon = 10^{-6}$，起始索引 $xs = 2$，结束索引 $xt = N - 3$

对于每个 j 在 $[xs, xt]$ 区间：

正通量部分（$a > 0$）：

计算三个光滑指示器

$$\beta_1 = \frac{1}{4}(F_p[j-2] - 4F_p[j-1] + 3F_p[j])^2 + \frac{13}{12}(F_p[j-2] - 2F_p[j-1] + F_p[j])^2$$
$$\beta_2 = \frac{1}{4}(F_p[j-1] - F_p[j+1])^2 + \frac{13}{12}(F_p[j-1] - 2F_p[j] + F_p[j+1])^2$$
$$\beta_3 = \frac{1}{4}(3F_p[j] - 4F_p[j+1] + F_p[j+2])^2 + \frac{13}{12}(F_p[j] - 2F_p[j+1] + F_p[j+2])^2$$

计算 $\alpha_k = C_k/(\varepsilon + \beta_k)^p$，归一化得权重 $\omega_k = \alpha_k / \sum \alpha_k$

计算三个模板的通量值

$$F^{(1)}_{j+1/2} = \frac{1}{3}F_p[j-2] - \frac{7}{6}F_p[j-1] + \frac{11}{6}F_p[j]$$
$$F^{(2)}_{j+1/2} = -\frac{1}{6}F_p[j-1] + \frac{5}{6}F_p[j] + \frac{1}{3}F_p[j+1]$$
$$F^{(3)}_{j+1/2} = \frac{1}{3}F_p[j] + \frac{5}{6}F_p[j+1] - \frac{1}{6}F_p[j+2]$$

加权组合 $Fh_p[j] = \omega_1 F^{(1)}_{j+1/2} + \omega_2 F^{(2)}_{j+1/2} + \omega_3 F^{(3)}_{j+1/2}$

负通量部分（$a < 0$）：

计算三个光滑指示器

$$\beta_1 = \frac{1}{4}(F_n[j+2] - 4F_n[j+1] + 3F_n[j])^2 + \frac{13}{12}(F_n[j+2] - 2F_n[j+1] + F_n[j])^2$$
$$\beta_2 = \frac{1}{4}(F_n[j+1] - F_n[j-1])^2 + \frac{13}{12}(F_n[j+1] - 2F_n[j] + F_n[j-1])^2$$
$$\beta_3 = \frac{1}{4}(3F_n[j] - 4F_n[j-1] + F_n[j-2])^2 + \frac{13}{12}(F_n[j] - 2F_n[j-1] + F_n[j-2])^2$$

计算 α_k 和归一化权重 ω_k，同上

计算三个模板的通量值

$$F_{j-1/2}^{(1)} = \frac{1}{3}F_n[j+2] - \frac{7}{6}F_n[j+1] + \frac{11}{6}F_n[j]$$
$$F_{j-1/2}^{(2)} = -\frac{1}{6}F_n[j+1] + \frac{5}{6}F_n[j] + \frac{1}{3}F_n[j-1]$$
$$F_{j-1/2}^{(3)} = \frac{1}{3}F_n[j] + \frac{5}{6}F_n[j-1] - \frac{1}{6}F_n[j-2]$$

加权组合 $Fh_n[j] = \omega_1 F_{j-1/2}^{(1)} + \omega_2 F_{j-1/2}^{(2)} + \omega_3 F_{j-1/2}^{(3)}$

计算通量导数，$xs_{new} = xs + 1$，$xt_{new} = xt - 1$，对 j 在 $[xs_{new}, xt_{new}]$：

$$Fx_p[j] = \frac{Fh_p[j] - Fh_p[j-1]}{dx}$$
$$Fx_n[j] = \frac{Fh_n[j+1] - Fh_n[j]}{dx}$$
$$Fx[j] = Fx_p[j] + Fx_n[j]$$

该实现保证了高阶精度和无振荡性，适用于激波和间断捕捉的高分辨率数值模拟。

3. **群速度控制 GVC (flag_scs_typ=3)**

4 结果分析

5 AI 工具使用说明表

AI 名称	生成代码功能	使用内容
Copilot	latex 格式框架	figure 参数调整、图片插入
Deepseek	python 绘图调整	68-90 行图绘制的具体参数调整
Deepseek	gitignore 文件忽略	全由 ai 添加

6 commit 信息

commit 图如下：