

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：

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

在 $t = 0$ 时刻，初始条件为：

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

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。具体要求如下：

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

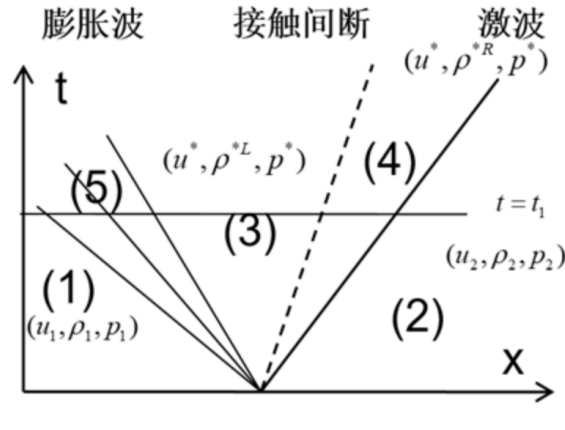


图 1: Sod 激波管问题示意图

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:

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

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

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:

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

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:

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

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

其中 f 函数定义为分段形式:

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

联立方程 (4) 和 (5), 得到关于 p^* 的非线性方程:

$$u_1 - u_2 = f(p^*, p_1, \rho_1) + f(p^*, p_2, \rho_2) \quad (7)$$

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：

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

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

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

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

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

$$\frac{\partial U}{\partial t} = -\frac{\partial \mathbf{F}}{\partial x} \quad (12)$$

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

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

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

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

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可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：

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

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：

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

2.3.2 Jacobian 矩阵与特征值

守恒变量的微分关系：

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

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

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

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：

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

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

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 $\text{Ma} = u/c$ 的分裂：

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

分裂策略：

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

分量分裂:

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

统一矢量形式:

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

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 $F_{j+\frac{1}{2}} = F(U_{j+\frac{1}{2}}^L, U_{j+\frac{1}{2}}^R)$
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：

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

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

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

或写为：

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

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 $\tilde{\mathbf{A}} = \tilde{\mathbf{A}}(\mathbf{U}_R, \mathbf{U}_L)$ 作为平均变化率矩阵, 满足:

$$\mathbf{F}(\mathbf{U}_R) - \mathbf{F}(\mathbf{U}_L) = \tilde{\mathbf{A}}(\mathbf{U}_R, \mathbf{U}_L)(\mathbf{U}_R - \mathbf{U}_L)$$

$\tilde{\mathbf{A}}(\mathbf{U}_R, \mathbf{U}_L)$ 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:

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

数值通量为:

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

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

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 $\tilde{\mathbf{A}}(\mathbf{U}_R, \mathbf{U}_L) = \frac{\partial \mathbf{F}(\mathbf{U}_{Roe})}{\partial \mathbf{U}_{Roe}}$ 。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:

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

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：

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

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

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

平均增长矩阵为：

$$\tilde{\mathbf{A}}(\mathbf{U}_R, \mathbf{U}_L) = \mathbf{A}(\bar{\mathbf{U}})$$

Roe 平均变量计算公式：

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

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 $\mathbf{U}_{j+\frac{1}{2}}^L$ 与 $\mathbf{U}_{j+\frac{1}{2}}^R$
2. 采用 Roe 平均公式计算平均守恒矢量 $\bar{\mathbf{U}}$ ，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： $\tilde{\mathbf{A}} = \mathbf{S}^{-1}\Lambda\mathbf{S}$

4. 计算 $\left| \tilde{\mathbf{A}} \right| = \mathbf{S}^{-1} |\Lambda| \mathbf{S}$, 其中 $|\Lambda| = \text{diag}(|\lambda_1|, |\lambda_2|, |\lambda_3|)$

5. 进行熵修正 (接近零的特征值处增加耗散):

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

6. 计算数值通量:

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

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：

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

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：

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

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

其中：

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

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：

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

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 $a > 0$):

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

数值通量计算:

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

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

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

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

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

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

光滑度量因子:

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

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

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

差分重构:

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

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

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

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

$$\text{推荐值: } 0.6 \leq \beta \leq 0.9$$

- γ (高阶修正系数): 控制高频分量处理

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

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

$$\text{推荐值: } 0.2 \leq \gamma \leq 0.5$$

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:

$$\mathbf{L}\mathbf{U} = \mathbf{W}$$

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:

```

1 #管道特征长度 (x = [-L/2, L/2])
2
3 L = 1.0
4
5 # 流动参数
6 Gamma = 1.4 # 比热比
7 R = 286.9 # 气体常数 (J/kg·K)
8 Cv = R / (Gamma - 1) # 定容比热容

```

```

9  Cp = (Gamma * R) / (Gamma - 1) # 定压比热容
10
11 # 网格生成
12 N = 201 # x方向网格数
13 xp = np.linspace(-L / 2, L / 2, N) # 网格点的x坐标
14 dx = L / (N - 1) # 网格间距
15
16 xp_mid = N // 2 # x=0位置的网格索引(中点)
17
18 # 物理量数组初始化
19 u_arr = np.zeros(N) # 速度数组
20 rho_arr = np.zeros(N) # 密度数组
21 p_arr = np.zeros(N) # 压力数组
22
23 # 辅助数组
24 rho = np.zeros(N)
25 u = np.zeros(N)
26 p = np.zeros(N)
27 E = np.zeros(N)
28 T = np.zeros(N)
29 c = np.zeros(N)
30
31 # 时间步长
32 dt = 0.001
33
34 # 最大计算步数
35 max_step = 200
36
37 # 最大模拟时间
38 max_tot_time = max_step * dt
39
40 # 设置初始条件
41 # 当x < 0时, (左侧: u1, rho1, p1) = (0.0, 1.0, 1.0)

```

```

42 # 当x >= 0时, (右侧: ur, rhor, pr) = (0.0, 0.125, 0.1)
43 u_arr[0:xp_mid-1] = 0.0
44 rho_arr[0:xp_mid-1] = 1.0
45 p_arr[0:xp_mid-1] = 1.0
46
47 u_arr[xp_mid-1:] = 0.0
48 rho_arr[xp_mid-1:] = 0.125
49 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:

```

1 #预处理: 算法选择设置
2 #通量分裂方法选择
3 class FluxSplittingMethod(Enum):
4     FVS = 1 # 通量向量分裂
5     FDS = 2 # 通量差分裂
6 #通量向量分裂方法FVS选择
7 class FVSMETHODS(Enum):
8     STEGER_WARMING = 1 # Steger-Warming方法
9     LAX_FRIEDRICH = 2 # Lax-Friedrich方法
10    VAN_LEER = 3 # Van Leer方法
11 #通量差分裂方法FDS选择
12 class FDSMETHODS(Enum):
13    ROE = 1 # ROE格式
14 #空间离散格式选择
15 class SpatialDiscretizationType(Enum):
16    SHOCK_CAPTURING = 1 # 激波捕捉格式
17 #激波捕捉格式选择
18 class ShockCapturingScheme(Enum):
19    TVD_VAN_LEER = 1 # TVD格式(Van Leer限制器)
20    WENO = 2 # 5阶WENO格式

```

```

21     GVC = 3                #群速度控制格式
22
23
24 # 指定通量分裂方法
25 # 1 - 通量向量分裂(FVS)
26 # 2 - 通量差分裂(FDS)
27 flag_flu_spl = FluxSplittingMethod.FDS.value
28
29 # FVS方法家族选择
30 # 1 - Steger-Warming (S-W)
31 # 2 - Lax-Friedrich (L-F)
32 # 3 - Van Leer
33 flag_fvs_met = FVSMETHODS.VAN_LEER.value
34
35 # 指定通量重构方法
36 # 1 - 直接重构(针对F(U))
37 # 2 - 特征重构
38 flag_flu_rec = 1
39
40 # FDS方法家族选择(FDS-ROE格式)
41 flag_fds_met = FDSMETHODS.ROE.value
42
43 # 指定高分辨率通量空间离散格式
44 flag_spa_typ = SpatialDiscretizationType.SHOCK_CAPTURING.
    value # 激波捕捉格式
45
46 # 激波捕捉格式类型选择
47 flag_scs_typ = ShockCapturingScheme.WENO.value
48
49 #数组初始化
50 lambda_arr = np.zeros(3) # 特征值矩阵
51 lambda_p_arr = np.zeros(3) # 正特征值矩阵
52 lambda_n_arr = np.zeros(3) # 负特征值矩阵

```

```

53
54 # 初始化解向量数组
55 U = np.zeros((N, 3)) # 守恒变量向量
56 F_p = np.zeros((N, 3)) # 正通量向量
57 F_n = np.zeros((N, 3)) # 负通量向量
58 Fx = np.zeros((N, 3)) # 通量散度项
59 Fx_p = np.zeros((N, 3)) # 正通量散度项
60 Fx_n = np.zeros((N, 3)) # 负通量散度项
61
62 # 半网格点通量数组
63 Fh_p = np.zeros((N - 1, 3)) # (j + 1/2)半网格点正通量向量
64 Fh_n = np.zeros((N - 1, 3)) # (j + 1/2)半网格点负通量向量
65
66 # 计算初始密度、速度、压力、温度、声速和守恒变量
67 for i in range(N):
68     # 从数组获取当前网格点的值
69     rho_val = rho_arr[i]
70     u_val = u_arr[i]
71     p_val = p_arr[i]
72
73     # 计算温度 (理想气体状态方程)
74     T_val = p_val / (rho_val * R)
75
76     # 计算声速 (等熵关系)
77     c_val = np.sqrt(Gamma * p_val / rho_val)
78
79     # 计算总能量 (内能 + 动能)
80     E_val = rho_val * ((Cv * T_val) + (0.5 * u_val * u_val))
81
82     # 存储守恒变量
83     # U = [密度, 动量, 总能量]
84     U[i, 0] = rho_val # 密度 (kg/m³)

```

```

85     U[i, 1] = rho_val * u_val # 动量 (kg/m2 · s)
86     U[i, 2] = E_val # 总能量 (J/m3)
87
88     # 存储热力学量
89     T[i] = T_val
90     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：

```

1  # 开始计算和时间步进
2  # 实际计算时间
3  t = 0.0
4  cnt_step = 0
5
6  # 主计算循环
7  while cnt_step < max_step:
8
9      # 更新时间
10     t += dt
11     cnt_step += 1
12
13     # 时间离散格式选择三阶TVD龙格-库塔法
14     if flag_flu_spl == 1:
15         # 1 - 通量向量分裂法 (FVS)
16         F_p, F_n = Flux_Vect_Split(U, N, Gamma, Cp, Cv, R,
17                                     flag_fvs_met)
18         _, _, _, _, Fx, _, _ = Diff_Cons(N, dx, F_p, F_n,
19                                             flag_spa_typ, flag_scs_typ)
20
21         # 第一步:  $U_1 = U^n + \Delta t * Q(U^n)$ 
22         U_1 = U + (dt * ((-1) * Fx))

```

```

22     # 使用U1计算Q(U1)
23     F_p_1, F_n_1 = Flux_Vect_Split(U_1, N, Gamma, Cp,
24                                     Cv, R, flag_fvs_met)
25
26     # 第二步:  $U_2 = (3/4)U^n + (1/4)U_1 + (1/4)\Delta t * Q(U_1)$ 
27     U_2 = ((3 / 4) * U) + ((1 / 4) * U_1) + (((1 * dt)
28         / 4) * ((-1) * Fx_1))
29
30     # 使用U2计算Q(U2)
31     F_p_2, F_n_2 = Flux_Vect_Split(U_2, N, Gamma, Cp,
32                                     Cv, R, flag_fvs_met)
33     xs_new, xt_new, Fh_p, Fh_n, Fx_2, Fx_p, Fx_n =
34         Diff_Cons(N, dx, F_p_2, F_n_2, flag_spa_typ,
35                 flag_scs_typ)
36
37     # 最终更新:  $U^{n+1} = (1/3)U^n + (2/3)U_2 + (2/3)\Delta t$ 
38     #  $* Q(U_2)$ 
39     U = ((1 / 3) * U) + ((2 / 3) * U_2) + (((2 / 3) *
40         dt) * ((-1) * Fx_2))
41
42     elif flag_flu_spl == 2:
43         xs_new, xt_new, Fx = Flux_Diff_Split(U, N, dx,
44                                             Gamma, Cp, Cv, R, flag_fds_met, flag_spa_typ,
45                                             flag_scs_typ)
46
47     #2 - 流通差分分裂法 (FDS)
48     # 第一步: 计算中间解U_1
49     U_1 = U + (dt * (-1 * Fx))
50
51     # 使用U_1再次计算通量
52     _, _, Fx_1 = Flux_Diff_Split(U_1, N, dx, Gamma, Cp,
53                                   Cv, R, flag_fds_met, flag_spa_typ, flag_scs_typ)

```



```

44
45     # 第二步：计算中间解U_2
46     U_2 = ((3 / 4) * U) + ((1 / 4) * U_1) + ((1 / 4) *
47         dt * (-1 * Fx_1))
48
49     # 使用U_2再次计算通量
50     _, _, Fx_2 = Flux_Diff_Split(U_2, N, dx, Gamma, Cp,
51         Cv, R, flag_fds_met, flag_spa_typ, flag_scs_typ)
52
53     # 最终更新：计算新时间步的解
54     U = ((1 / 3) * U) + ((2 / 3) * U_2) + ((2 / 3) * dt
55         * (-1 * Fx_2))
56
57     # 保存时间t_end时刻的最终解U
58     U_end = U.copy()
59     t_end = t
60
61     # 计算时间t_end时刻的最终物理量
62     # 根据守恒变量U计算rho, u, E, T, p, c, H
63     rho_end = U_end[:, 0] # 密度
64     u_end = U_end[:, 1] / rho_end # 速度
65     E_end = U_end[:, 2] # 单位质量总内能
66     # 计算温度：T = (e_total - 0.5*u^2) / Cv
67     T_end = (E_end / rho_end - 0.5 * u_end**2) / Cv
68     # 计算压力：p = RT
69     p_end = rho_end * R * T_end
70     # 计算声速：c = sqrt(p / )
71     c_end = np.sqrt(Gamma * p_end / rho_end)
72     # 计算焓：H = 0.5*u^2 + Cp*T
73     H_end = 0.5 * u_end**2 + Cp * T_end
74     # 计算内能：e = E / rho - 0.5*u^2
75     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：

```

1 #通量向量分裂方法(FVS)通用函数(Steger-Warming,Lax-Friedrich
  ,Van Leer)
2 def Flux_Vect_Split(U, N, Gamma, Cp, Cv, R, flag_fvs_met):
3     if flag_fvs_met == 1:
4         # FVS - Steger-Warming (S-W)方法
5
6         # 初始化变量
7         F_p = np.zeros((N, 3)) # 正通量分量
8         F_n = np.zeros((N, 3)) # 负通量分量
9         em = 1e-3 # 小量防止除零
10
11        for i in range(N):
12            # 计算密度、速度、温度、压力、声速
13            rho = U[i, 0]
14            u = U[i, 1] / rho
15            E = U[i, 2]
16            # 计算温度: T = (e_total - 0.5*u^2) / Cv
17            T = (E / rho - 0.5 * u ** 2) / Cv
18            p = rho * R * T
19            c = np.sqrt(Gamma * p / rho)
20
21            # 计算特征值
22            lambda_vals = np.array([u, u - c, u + c])
23
24            # 分裂特征值 (S-W方法)
25            sqrt_term = np.sqrt(lambda_vals ** 2 + em ** 2)
26            lambda_p = (lambda_vals + sqrt_term) / 2

```

```

27     lambda_n = (lambda_vals - sqrt_term) / 2
28
29     # 计算F+和F-
30     # 计算w项
31     w_p = ((3 - Gamma) * (lambda_p[1] + lambda_p
32             [2]) * c ** 2) / (2 * (Gamma - 1))
33     w_n = ((3 - Gamma) * (lambda_n[1] + lambda_n
34             [2]) * c ** 2) / (2 * (Gamma - 1))
35
36     # 计算正通量分量
37     F_p[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma -
38         1) * lambda_p[0] + lambda_p[1] + lambda_p
39         [2])
40     F_p[i, 1] = (rho / (2 * Gamma)) * (
41         2 * (Gamma - 1) * lambda_p[0] * u +
42         lambda_p[1] * (u - c) +
43         lambda_p[2] * (u + c))
44     F_p[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1)
45         * lambda_p[0] * u ** 2 + (lambda_p[1] * (u -
46         c) ** 2) / 2 + (
47         lambda_p[2] * (u + c) ** 2) / 2 +
48         w_p)
49
50     # 计算负通量分量
51     F_n[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma -
52         1) * lambda_n[0] + lambda_n[1] + lambda_n
53         [2])
54     F_n[i, 1] = (rho / (2 * Gamma)) * (
55         2 * (Gamma - 1) * lambda_n[0] * u +
56         lambda_n[1] * (u - c) +
57         lambda_n[2] * (u + c))
58     F_n[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1)
59         * lambda_n[0] * u ** 2 + (lambda_n[1] * (u -
60         c) ** 2) / 2 + (
61         lambda_n[2] * (u + c) ** 2) / 2 +
62         w_n)

```

```

46         c) ** 2) / 2 + (
47             lambda_n[2] * (u + c) ** 2) / 2 +
48             w_n)
49
50 elif flag_fvs_met == 2:
51     # FVS - Lax-Friedrich (L-F) 方法
52
53     # 初始化变量
54     F_p = np.zeros((N, 3)) # 正通量分量
55     F_n = np.zeros((N, 3)) # 负通量分量
56
57     # 先循环计算全局最大特征值
58     lambda_s_global = 0
59     for i in range(N):
60         rho = U[i, 0]
61         u = U[i, 1] / rho
62         E = U[i, 2]
63         T = (E / rho - 0.5 * u ** 2) / Cv
64         p = rho * R * T
65         c = np.sqrt(Gamma * p / rho)
66
67         # 计算局部特征值并更新全局最大值
68         lambda_s_local = abs(u) + c
69         if lambda_s_local > lambda_s_global:
70             lambda_s_global = lambda_s_local
71
72     # 再次循环计算通量分量
73     for i in range(N):
74         # 计算物理量
75         rho = U[i, 0]
76         u = U[i, 1] / rho
77         E = U[i, 2]
78         T = (E / rho - 0.5 * u ** 2) / Cv

```

```

77     p = rho * R * T
78     c = np.sqrt(Gamma * p / rho)
79
80     # 计算特征值
81     lambda_vals = np.array([u, u - c, u + c])
82
83     # 分裂特征值 (L-F方法使用全局最大特征值)
84     lambda_p = (lambda_vals + lambda_s_global) / 2
85     lambda_n = (lambda_vals - lambda_s_global) / 2
86
87     # 计算w项
88     w_p = ((3 - Gamma) * (lambda_p[1] + lambda_p
89               [2]) * c ** 2) / (2 * (Gamma - 1))
89     w_n = ((3 - Gamma) * (lambda_n[1] + lambda_n
90               [2]) * c ** 2) / (2 * (Gamma - 1))
91
92     # 计算正通量分量
93     F_p[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma -
94               1) * lambda_p[0] + lambda_p[1] + lambda_p
95               [2])
96     F_p[i, 1] = (rho / (2 * Gamma)) * (
97               2 * (Gamma - 1) * lambda_p[0] * u +
98               lambda_p[1] * (u - c) +
99               lambda_p[2] * (u + c))
100     F_p[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1)
101               * lambda_p[0] * u ** 2 + (lambda_p[1] * (u -
102               c) ** 2) / 2 + (
103               lambda_p[2] * (u + c) ** 2) / 2 +
104               w_p)
105
106     # 计算负通量分量
107     F_n[i, 0] = (rho / (2 * Gamma)) * (2 * (Gamma -
108               1) * lambda_n[0] + lambda_n[1] + lambda_n

```

```

[2])
100     F_n[i, 1] = (rho / (2 * Gamma)) * (
101         2 * (Gamma - 1) * lambda_n[0] * u +
            lambda_n[1] * (u - c) +
            lambda_n[2] * (u + c))
102     F_n[i, 2] = (rho / (2 * Gamma)) * ((Gamma - 1)
        * lambda_n[0] * u ** 2 + (lambda_n[1] * (u -
            c) ** 2) / 2 + (
103         lambda_n[2] * (u + c) ** 2) / 2 +
            w_n)
104
105     elif flag_fvs_met == 3:
106         # Van Leer方法
107
108         # 初始化变量
109         F_p = np.zeros((N, 3)) # 正通量分量
110         F_n = np.zeros((N, 3)) # 负通量分量
111
112         for i in range(N):
113             # 计算物理量
114             rho = U[i, 0]
115             u = U[i, 1] / rho
116             E = U[i, 2]
117             T = (E / rho - 0.5 * u ** 2) / Cv
118             p = rho * R * T
119             c = np.sqrt(Gamma * p / rho)
120
121             # 计算马赫数
122             Ma = u / c if c != 0 else 0
123
124             # Van Leer通量分量计算公式
125             if Ma >= 1:
126                 # 纯超音速正向流

```

```

127         F_p[i, 0] = U[i, 1] # rho * u
128         F_p[i, 1] = (Gamma - 1) * U[i, 2] + ((3 -
129             Gamma) / 2) * (U[i, 1] ** 2 / U[i, 0])
130         F_p[i, 2] = Gamma * (U[i, 1] * U[i, 2]) / U
131             [i, 0] + ((Gamma - 1) / 2) * (U[i, 1] **
132                 3 / U[i, 0] ** 2)
133
134         F_n[i, :] = 0.0
135
136     elif Ma <= -1:
137         # 纯超音速负向流
138         F_n[i, 0] = U[i, 1] # rho * u
139         F_n[i, 1] = (Gamma - 1) * U[i, 2] + ((3 -
140             Gamma) / 2) * (U[i, 1] ** 2 / U[i, 0])
141         F_n[i, 2] = Gamma * (U[i, 1] * U[i, 2]) / U
142             [i, 0] + ((Gamma - 1) / 2) * (U[i, 1] **
143                 3 / U[i, 0] ** 2)
144
145         F_p[i, :] = 0.0
146
147     else:
148         # 亚音速/跨音速区域
149         # 计算质量通量
150         F1_p = rho * c * ((Ma + 1) / 2) ** 2
151         F1_n = -rho * c * ((Ma - 1) / 2) ** 2
152
153         # 计算正通量分量
154         F_p[i, 0] = F1_p
155         F_p[i, 1] = (F1_p / Gamma) * ((Gamma - 1) *
156             u + 2 * c)
157         F_p[i, 2] = (F1_p / (2 * (Gamma ** 2 - 1)))
158             * ((Gamma - 1) * u + 2 * c) ** 2

```

```

152         # 计算负通量分量
153         F_n[i, 0] = F1_n
154         F_n[i, 1] = (F1_n / Gamma) * ((Gamma - 1) *
            u - 2 * c)
155         F_n[i, 2] = (F1_n / (2 * (Gamma ** 2 - 1)))
            * ((Gamma - 1) * u - 2 * c) ** 2
156
157     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：

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

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：

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

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 $\text{Ma} = u/c$ ，其数值实现如下：

在计算过程中，对于每个网格点，首先计算密度 ρ 、速度 u 、总能 E 、温度 T 、压力 p 和声速 c ，进而得到马赫数 $\text{Ma} = u/c$ 。根据马赫数范围进行通量分裂处理：

当 $\text{Ma} \geq 1$ 时，正通量 \mathbf{F}^+ 取完整物理通量，即 $\mathbf{F}^+ = \mathbf{F}$ ，同时负通量 \mathbf{F}^- 设为零；当 $\text{Ma} \leq -1$ 时，负通量 \mathbf{F}^- 取完整物理通量 $\mathbf{F}^- = \mathbf{F}$ ，同时正通量 \mathbf{F}^+ 设为零。

当 $|\text{Ma}| < 1$ 时，通量按以下公式分裂：

$$F_1^+ = \rho c \left(\frac{\text{Ma} + 1}{2} \right)^2, \quad F_1^- = -\rho c \left(\frac{\text{Ma} - 1}{2} \right)^2$$

$$F_2^+ = \frac{F_1^+}{\gamma} [(\gamma - 1)u + 2c], \quad F_2^- = \frac{F_1^-}{\gamma} [(\gamma - 1)u - 2c]$$

$$F_3^+ = \frac{F_1^+}{2(\gamma^2 - 1)} [(\gamma - 1)u + 2c]^2, \quad F_3^- = \frac{F_1^-}{2(\gamma^2 - 1)} [(\gamma - 1)u - 2c]^2$$

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

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：

```

1 #通用通量差分计算函数(TVD,WENO,GVC)，返回索引和通量导数
2 def Diff_Cons(N, dx, F_p, F_n, flag_spa_typ, flag_scs_typ):
3     # 初始化输出数组
4     xs_new = 0
5     xt_new = 0
6     Fx = np.zeros((N, 3)) # 总通量导数
7     Fx_p = np.zeros((N, 3)) # 正通量导数
8     Fx_n = np.zeros((N, 3)) # 负通量导数
9     Fh_p = np.zeros((N, 3)) # 半网格点正通量 (j+1/2)
10    Fh_n = np.zeros((N, 3)) # 半网格点负通量 (j+1/2)
11    Fh = np.zeros((N, 3)) # 半网格点总通量 (j+1/2)
12
13    if flag_spa_typ == 1:
14        # 特殊激波捕捉格式
15
16        if flag_scs_typ == 1:
17            # 1 - TVD格式 (Van Leer限制器)
18            xs = 1 # 起始索引
19            xt = N - 2 # 结束索引

```

```

20     em = 1e-5 # 小量防止除零
21
22     for j in range(xs, xt):
23         # 计算正通量比例因子
24         r_p_num = F_p[j] - F_p[j - 1]
25         r_p_den = F_p[j + 1] - F_p[j] + em
26         r_p = r_p_num / r_p_den
27
28         # 计算负通量比例因子
29         r_n_num = F_n[j + 2] - F_n[j + 1]
30         r_n_den = F_n[j + 1] - F_n[j] + em
31         r_n = r_n_num / r_n_den
32
33         # Van Leer限制器
34         Phi_p = (r_p + np.abs(r_p)) / (1 + r_p)
35         Phi_n = (r_n + np.abs(r_n)) / (1 + r_n)
36
37         # 计算半网格点通量
38         Fh_p[j] = F_p[j] + 0.5 * Phi_p * (F_p[j +
39             1] - F_p[j])
40         Fh_n[j] = F_n[j + 1] - 0.5 * Phi_n * (F_n[j
41             + 1] - F_n[j])
42
43     # 更新索引范围
44     xs_new = xs + 1
45     xt_new = xt
46
47     # 计算通量导数
48     for j in range(xs_new, xt_new):
49         Fx_p[j] = (Fh_p[j] - Fh_p[j - 1]) / dx
50         Fx_n[j] = (Fh_n[j] - Fh_n[j - 1]) / dx
51         Fx[j] = Fx_p[j] + Fx_n[j]

```

```

51 elif flag_scs_typ == 2:
52     # 2 - WENO5 阶格式
53     C = np.array([1 / 10, 6 / 10, 3 / 10])
54     p = 2
55     em = 1e-6
56
57     xs = 2 # 起始索引
58     xt = N - 3 # 结束索引
59
60     for j in range(xs, xt + 1):
61         # a > 0 正通量部分
62         # 计算光滑指示器 beta
63         beta_p1 = (1 / 4) * (F_p[j - 2] - 4 * F_p[j
64             - 1] + 3 * F_p[j]) ** 2 + (13 / 12) * (
65             F_p[j - 2] - 2 * F_p[j - 1] +
66             F_p[j]) ** 2
67         beta_p2 = (1 / 4) * (F_p[j - 1] - F_p[j +
68             1]) ** 2 + (13 / 12) * (F_p[j - 1] - 2 *
69             F_p[j] + F_p[j + 1]) ** 2
70         beta_p3 = (1 / 4) * (3 * F_p[j] - 4 * F_p[j
71             + 1] + F_p[j + 2]) ** 2 + (13 / 12) * (
72             F_p[j] - 2 * F_p[j + 1] + F_p[j
73             + 2]) ** 2
74
75         # 计算 alpha 和权重 omega
76         alpha_p = C / (em + np.array([beta_p1,
77             beta_p2, beta_p3])) ** p
78         alpha_p_sum = np.sum(alpha_p, axis=0)
79         omega_p = alpha_p / alpha_p_sum
80
81         # 计算三个模板的通量值
82         Fh_p_c1 = (1 / 3) * F_p[j - 2] - (7 / 6) *
83             F_p[j - 1] + (11 / 6) * F_p[j]

```

```

76     Fh_p_c2 = (-1 / 6) * F_p[j - 1] + (5 / 6) *
           F_p[j] + (1 / 3) * F_p[j + 1]
77     Fh_p_c3 = (1 / 3) * F_p[j] + (5 / 6) * F_p[
           j + 1] - (1 / 6) * F_p[j + 2]
78     Fh_p_c = np.array([Fh_p_c1, Fh_p_c2,
           Fh_p_c3])
79
80     # 加权组合得到  $F_{+}\{j+1/2\}$ 
81     Fh_p[j] = np.sum(omega_p * Fh_p_c, axis=0)
82
83     # a < 0 负通量部分
84     # 计算光滑指示器 beta
85     beta_n1 = (1 / 4) * (F_n[j + 2] - 4 * F_n[j
           + 1] + 3 * F_n[j]) ** 2 + (13 / 12) * (
86         F_n[j + 2] - 2 * F_n[j + 1] +
           F_n[j]) ** 2
87     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
88     beta_n3 = (1 / 4) * (3 * F_n[j] - 4 * F_n[j
           - 1] + F_n[j - 2]) ** 2 + (13 / 12) * (
89         F_n[j] - 2 * F_n[j - 1] + F_n[j
           - 2]) ** 2
90
91     # 计算 alpha 和权重 omega
92     alpha_n = C / (em + np.array([beta_n1,
           beta_n2, beta_n3])) ** p
93     alpha_n_sum = np.sum(alpha_n, axis=0)
94     omega_n = alpha_n / alpha_n_sum
95
96     # 计算三个模板的通量值
97     Fh_n_c1 = (1 / 3) * F_n[j + 2] - (7 / 6) *
           F_n[j + 1] + (11 / 6) * F_n[j]

```

```

98         Fh_n_c2 = (-1 / 6) * F_n[j + 1] + (5 / 6) *
          F_n[j] + (1 / 3) * F_n[j - 1]
99         Fh_n_c3 = (1 / 3) * F_n[j] + (5 / 6) * F_n[
          j - 1] - (1 / 6) * F_n[j - 2]
100        Fh_n_c = np.array([Fh_n_c1, Fh_n_c2,
          Fh_n_c3])
101
102        # 加权组合得到  $F_{-}\{j-1/2\}$ 
103        Fh_n[j] = np.sum(omega_n * Fh_n_c, axis=0)
104
105        # 计算通量导数
106        xs_new = xs + 1 # 3
107        xt_new = xt - 1 # N-4
108
109        for j in range(xs_new, xt_new + 1):
110            Fx_p[j] = (Fh_p[j] - Fh_p[j - 1]) / dx
111            Fx_n[j] = (Fh_n[j + 1] - Fh_n[j]) / dx
112            Fx[j] = Fx_p[j] + Fx_n[j]
113
114        elif flag_scs_ttyp == 3:
115            # 群速度控制格式
116            # 参数设置
117            xs = 1 # 起始索引
118            xt = N - 2 # 结束索引
119            em = 1e-5 # 小量防止除零
120
121            # 群速度控制参数
122            beta = 0.8 # 群速度控制因子
123            gamma = 0.3 # 高阶修正系数
124
125            for j in range(xs, xt):
126                # 计算通量梯度比
127                # 正通量部分

```

```

128         r_p_num = F_p[j] - F_p[j - 1]
129         r_p_den = F_p[j + 1] - F_p[j] + em
130         r_p = r_p_num / r_p_den
131
132         # 负通量部分
133         r_n_num = F_n[j + 2] - F_n[j + 1]
134         r_n_den = F_n[j + 1] - F_n[j] + em
135         r_n = r_n_num / r_n_den
136
137         # 群速度控制限制器函数
138         # 正通量限制器
139         Phi_p = (r_p + beta * np.abs(r_p)) / (1 +
140             beta * r_p + gamma * r_p ** 2)
141         # 负通量限制器
142         Phi_n = (r_n + beta * np.abs(r_n)) / (1 +
143             beta * r_n + gamma * r_n ** 2)
144
145         # 计算半网格点通量（含群速度控制）
146         Fh_p[j] = F_p[j] + 0.5 * Phi_p * (F_p[j +
147             1] - F_p[j])
148         Fh_n[j] = F_n[j + 1] - 0.5 * Phi_n * (F_n[j
149             + 1] - F_n[j])
150
151         # 更新索引范围
152         xs_new = xs + 1
153         xt_new = xt
154
155         # 计算通量导数
156         for j in range(xs_new, xt_new):
157             Fx_p[j] = (Fh_p[j] - Fh_p[j - 1]) / dx
158             Fx_n[j] = (Fh_n[j] - Fh_n[j - 1]) / dx
159             Fx[j] = Fx_p[j] + Fx_n[j]
160     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 $xs = 1$ 和结束索引 $xt = N - 2$, 其中引入小量 $\varepsilon = 10^{-5}$

以防止除零情况。对于每个索引 j 在区间 $[xs, xt)$ 内执行以下计算步骤:

首先计算正通量梯度比: $r_p = \frac{F_p[j] - F_p[j-1]}{F_p[j+1] - F_p[j] + \varepsilon}$;

接着计算负通量梯度比: $r_n = \frac{F_n[j+2] - F_n[j+1]}{F_n[j+1] - F_n[j] + \varepsilon}$;

随后应用 Van Leer 限制器函数: $\Phi_p = \frac{r_p + |r_p|}{1 + r_p}$ 和 $\Phi_n = \frac{r_n + |r_n|}{1 + r_n}$;

最后进行半网格点通量重构:

$$Fh_p[j] = F_p[j] + 0.5 \Phi_p (F_p[j+1] - F_p[j]), \quad Fh_n[j] = F_n[j+1] - 0.5 \Phi_n (F_n[j+1] - F_n[j])$$

完成所有 j 的计算后, 更新索引范围: $xs_{\text{new}} = xs + 1$, $xt_{\text{new}} = xt$ 。在新索引范围 $[xs_{\text{new}}, xt_{\text{new}}]$ 内计算通量导数:

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

以及总通量导数:

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

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

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

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

$$\begin{aligned} F_{j+1/2}^{(1)} &= \frac{1}{3}F_p[j-2] - \frac{7}{6}F_p[j-1] + \frac{11}{6}F_p[j] \\ F_{j+1/2}^{(2)} &= -\frac{1}{6}F_p[j-1] + \frac{5}{6}F_p[j] + \frac{1}{3}F_p[j+1] \\ F_{j+1/2}^{(3)} &= \frac{1}{3}F_p[j] + \frac{5}{6}F_p[j+1] - \frac{1}{6}F_p[j+2] \end{aligned}$$

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

负通量部分 ($a < 0$):

计算三个光滑指示器

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

计算 α_k 和归一化权重 ω_k ，同上

计算三个模板的通量值

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

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 $[xs_{new}, xt_{new}]$:

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

3 群速度控制 GVC (flag_scs_tpy=3)

群速度控制 (GVC) 格式的实现过程如下:

设定起始索引 $xs = 1$ ，结束索引 $xt = N - 2$ ，并引入小量 $em = 10^{-5}$ 以防止除零情况。同时设置群速度控制参数： $\beta = 0.8$ 作为群速度控制因子， $\gamma = 0.3$ 作为高阶修正系数。

对于每个索引 j 在区间 $[xs, xt)$ 内进行如下计算：首先计算正通量梯度比：

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

接着计算负通量梯度比：

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

然后计算群速度控制限制器函数：

$$\Phi_p = \frac{r_p + \beta|r_p|}{1 + \beta r_p + \gamma r_p^2}, \quad \Phi_n = \frac{r_n + \beta|r_n|}{1 + \beta r_n + \gamma r_n^2}$$

最后计算含群速度控制的半网格点通量：

$$Fh_p[j] = F_p[j] + 0.5 \Phi_p (F_p[j+1] - F_p[j]), \quad Fh_n[j] = F_n[j+1] - 0.5 \Phi_n (F_n[j+1] - F_n[j])$$

完成上述计算后，更新索引范围为 $xs_{\text{new}} = xs + 1$ ， $xt_{\text{new}} = xt$ 。在新索引区间 $[xs_{\text{new}}, xt_{\text{new}})$ 内计算通量导数：

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

以及总通量导数：

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

接下来是根据流通差分分裂法（FDS），实现激波模拟算法：

```

1 #使用 ROE 格式实现通量差分裂法（FDS）
2 def Flux_Diff_Split(U, N, dx, Gamma, Cp, Cv, R,
   flag_fds_met, flag_spa_typ, flag_scs_typ):
3     # Roe格式通量差分分裂
4     if flag_fds_met == 1:
5         # 初始化数组
6         Fh_l = np.zeros((N, 3)) # 左界面通量
7         Fh_r = np.zeros((N, 3)) # 右界面通量

```

```

8      U_ave = np.zeros((N, 3)) # Roe平均守恒变量
9      F_ave = np.zeros((N, 3)) # Roe平均通量
10     Fh = np.zeros((N, 3)) # 界面通量
11     Fx = np.zeros((N, 3)) # 空间导数
12
13     em = 1e-5 # 熵修正参数
14
15     # 调用守恒变量差分函数
16     xs, xt, Uh_l, Uh_r, _, _, _ = Diff_Cons(
17         N, dx, U, U, flag_spa_typ, 2
18     )
19
20     # WENO格式
21     if flag_spa_typ == 1 and flag_scs_typ == 2:
22         xt = xt - 1 # 调整计算域结束索引
23         # 右状态值向右平移一位
24         for j in range(xs, xt + 1):
25             Uh_r[j] = Uh_r[j + 1]
26
27     # 遍历所有网格界面
28     for j in range(xs, xt + 1):
29         # 计算左状态物理量
30         rho_l = Uh_l[j, 0] # 左侧密度
31         u_l = Uh_l[j, 1] / rho_l # 左侧速度
32         E_l = Uh_l[j, 2] # 左侧总能
33         # 左侧温度
34         T_l = ((E_l / rho_l) - 0.5 * u_l ** 2) / Cv
35         p_l = rho_l * R * T_l # 左侧压力
36         H_l = 0.5 * u_l ** 2 + Cp * T_l # 左侧总焓
37
38         # 左侧通量计算
39         Fh_l[j, 0] = rho_l * u_l # 质量通量
40         Fh_l[j, 1] = rho_l * u_l ** 2 + p_l # 动量通量

```

```

41     Fh_l[j, 2] = u_l * (E_l + p_l) # 能量通量
42
43     # 计算右状态物理量
44     rho_r = Uh_r[j, 0] # 右侧密度
45     u_r = Uh_r[j, 1] / rho_r # 右侧速度
46     E_r = Uh_r[j, 2] # 右侧总能
47     # 右侧温度
48     T_r = ((E_r / rho_r) - 0.5 * u_r ** 2) / Cv
49     p_r = rho_r * R * T_r # 右侧压力
50     H_r = 0.5 * u_r ** 2 + Cp * T_r # 右侧总焓
51
52     # 右侧通量计算
53     Fh_r[j, 0] = rho_r * u_r # 质量通量
54     Fh_r[j, 1] = rho_r * u_r ** 2 + p_r # 动量通量
55     Fh_r[j, 2] = u_r * (E_r + p_r) # 能量通量
56
57     # 计算Roe平均量
58     sqrt_rho_l = np.sqrt(rho_l)
59     sqrt_rho_r = np.sqrt(rho_r)
60     # Roe平均密度
61     rho_ave = ((sqrt_rho_l + sqrt_rho_r) / 2) ** 2
62     # Roe平均速度
63     u_ave = (sqrt_rho_l * u_l + sqrt_rho_r * u_r) /
64             (np.sqrt(rho_ave) * 2)
65     # Roe平均总焓
66     H_ave = (sqrt_rho_l * H_l + sqrt_rho_r * H_r) /
67             (np.sqrt(rho_ave) * 2)
68     # Roe平均压力
69     p_ave = ((Gamma - 1) / Gamma) * (rho_ave *
70             H_ave - 0.5 * rho_ave * u_ave ** 2)
71     # Roe平均声速
72     c_ave = np.sqrt((Gamma - 1) * (H_ave - 0.5 *
73             u_ave ** 2))

```

```

70     # Roe平均总能
71     E_ave = rho_ave * H_ave - p_ave
72
73     # 存储Roe平均守恒变量
74     U_ave[j] = [rho_ave, rho_ave * u_ave, E_ave]
75     # 存储Roe平均通量
76     F_ave[j] = [rho_ave * u_ave, rho_ave * u_ave **
77                 2 + p_ave, u_ave * (E_ave + p_ave)]
78
79     # 构造Jacobian矩阵
80     A_ave = np.array([
81         [0, 1, 0],
82         [(-(3 - Gamma) / 2) * u_ave ** 2, (3 -
83         Gamma) * u_ave, Gamma - 1],
84         [((Gamma - 2) / 2) * u_ave ** 3 - u_ave *
85         c_ave ** 2 / (Gamma - 1),
86         c_ave ** 2 / (Gamma - 1) + (3 - Gamma) / 2
87         * u_ave ** 2,
88         Gamma * u_ave]
89     ])
90
91     # 特征分解
92     D, V = np.linalg.eig(A_ave)
93     S = np.linalg.inv(V)
94
95     # 熵修正处理特征值
96     D_abs = np.zeros(3)
97     for i in range(3):
98         if abs(D[i]) > em:
99             D_abs[i] = abs(D[i])
100         else:
101             # Harten修正公式
102             D_abs[i] = (D[i] ** 2 + em ** 2) / (2 *

```

```

em)

99
100     # 构造绝对Jacobian矩阵
101     A_ave_abs = V @ np.diag(D_abs) @ S
102
103     # 计算界面通量
104     diff = (Uh_r[j] - Uh_l[j]).reshape(3, 1) # 状态差向量
105     # Roe通量公式
106     roe_flux = 0.5 * (A_ave_abs @ diff).flatten()
107     Fh[j] = 0.5 * (Fh_l[j] + Fh_r[j]) - roe_flux
108
109     #计算空间导数
110     xs_new = xs + 1 # 新计算域起始索引
111     xt_new = xt # 新计算域结束索引
112     for j in range(xs_new, xt_new + 1):
113         # 中心差分格式
114         Fx[j] = (Fh[j] - Fh[j - 1]) / dx
115
116     return xs_new, xt_new, Fx

```

Listing 6: 流通差分分裂法实现

目前只有 ROE 格式的实现，ROE 格式的实现过程如下：

1. 界面左右状态变量重构通过高阶差分（如 WENO）在每个界面 $j + \frac{1}{2}$ 处获得左右守恒变量 $\mathbf{U}_{j+\frac{1}{2}}^L$ 和 $\mathbf{U}_{j+\frac{1}{2}}^R$ 。
2. 物理量计算对于每个界面，分别计算左右状态的密度 $\rho_{L,R}$ 、速度 $u_{L,R}$ 、总能 $E_{L,R}$ 、温度 $T_{L,R}$ 、压力 $p_{L,R}$ 、总焓 $H_{L,R}$ ：

$$T_{L,R} = \frac{E_{L,R}/\rho_{L,R} - 0.5u_{L,R}^2}{C_v}$$

$$p_{L,R} = \rho_{L,R}RT_{L,R}$$

$$H_{L,R} = 0.5u_{L,R}^2 + C_pT_{L,R}$$

3. Roe 平均量计算

$$\begin{aligned}\tilde{\rho} &= \left(\frac{\sqrt{\rho_L} + \sqrt{\rho_R}}{2} \right)^2 \\ \tilde{u} &= \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\ \tilde{H} &= \frac{\sqrt{\rho_L}H_L + \sqrt{\rho_R}H_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \\ \tilde{p} &= \frac{\gamma - 1}{\gamma} \left(\tilde{\rho}\tilde{H} - \frac{1}{2}\tilde{\rho}\tilde{u}^2 \right) \\ \tilde{c}^2 &= (\gamma - 1) \left(\tilde{H} - \frac{1}{2}\tilde{u}^2 \right)\end{aligned}$$

4. **Roe 平均 Jacobian 矩阵特征分解构造 Roe 平均 Jacobian 矩阵 $\mathbf{A}(\tilde{\mathbf{U}})$** ，进行特征分解：

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

其中 $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \lambda_3)$ 为特征值对角阵。

5. **熵修正**对每个特征值 λ_i ，采用 Harten 熵修正：

$$|\lambda_i|_{\text{mod}} = \begin{cases} |\lambda_i|, & |\lambda_i| > \varepsilon \\ \frac{\lambda_i^2 + \varepsilon^2}{2\varepsilon}, & |\lambda_i| \leq \varepsilon \end{cases}$$

构造绝对 Jacobian 矩阵 $|\mathbf{A}| = \mathbf{V}|\mathbf{\Lambda}|_{\text{mod}}\mathbf{V}^{-1}$ 。

6. Roe 界面通量计算

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

7. **空间导数离散**用中心差分格式计算：

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

上述步骤即为 Roe 格式通量差分分裂法的完整实现流程。

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：