

# 水平集方程数值求解方法

黄胤强

June 5, 2025

# 目录

- 1 水平集方法概述
- 2 有限差分离散化
- 3 隐式时间离散化方案
- 4 重初始化与数值实现
- 5 算法实现与优化
- 6 EDA 竞赛应用实例
- 7 总结与展望
- 8 总结与展望

# 水平集方法基本概念

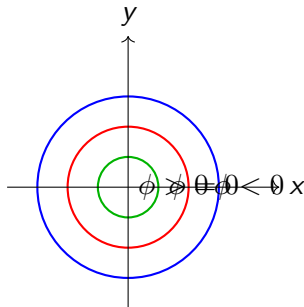
## 水平集函数定义

对于  $n$  维空间中的  $(n-1)$  维界面  $\Gamma$ , 水平集函数  $\phi(\mathbf{x}, t)$  定义为:

$$\phi(\mathbf{x}, t) = \begin{cases} > 0 & \text{if } \mathbf{x} \text{ is inside } \Gamma \\ = 0 & \text{if } \mathbf{x} \text{ is on } \Gamma \\ < 0 & \text{if } \mathbf{x} \text{ is outside } \Gamma \end{cases}$$

## 核心优势

- 隐式表示: 避免参数化复杂界面
- 自动处理拓扑变化 (分裂、合并)
- 数值稳定: 固定网格上的欧拉方程
- 扩展性好: 高维问题处理直观



水平集函数等值线示意

# 水平集方程的物理意义

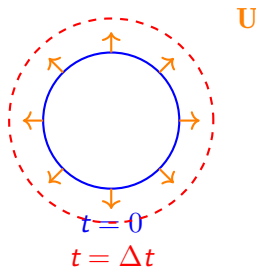
## Hamilton-Jacobi 型水平集方程

$$\frac{\partial \phi}{\partial t} + \mathbf{U} \cdot \nabla \phi = 0$$

其中  $\mathbf{U} = (U_x, U_y, U_z)$  是速度场

几何解释:

- $\nabla \phi$ : 水平集函数的梯度
- $\mathbf{U}$ : 速度场
- 零水平集  $\{\phi = 0\}$  随时间演化



界面在速度场作用下的演化

## 核心挑战

水平集方程是双曲型 PDE，需要满足熵条件以保证解的唯一性

一阶迎风格式：

$$\begin{aligned} \mathbf{U} \cdot \nabla \phi \approx & \begin{cases} U_x \frac{\phi_i - \phi_{i-1}}{\Delta x} & \text{if } U_x > 0 \\ U_x \frac{\phi_{i+1} - \phi_i}{\Delta x} & \text{if } U_x < 0 \end{cases} \\ & + \begin{cases} U_y \frac{\phi_j - \phi_{j-1}}{\Delta y} & \text{if } U_y > 0 \\ U_y \frac{\phi_{j+1} - \phi_j}{\Delta y} & \text{if } U_y < 0 \end{cases} \end{aligned}$$

其中：

速度符号决定差分方向

# 高阶 WENO 格式

## WENO 基本思想

加权本质无振荡格式 (Weighted Essentially Non-Oscillatory) 通过自适应权重选择光滑模板, 在光滑区域达到高精度, 在间断附近保持稳定性。

五阶 WENO 重构 (正向差分):

$$\phi_{i+1/2}^+ = \sum_{k=0}^2 \omega_k^+ \phi_{i+1/2}^{(k)}$$

三个候选模板:

$$\phi_{i+1/2}^{(0)} = \frac{1}{3}\phi_{i-2} - \frac{7}{6}\phi_{i-1} + \frac{11}{6}\phi_i$$

$$\phi_{i+1/2}^{(1)} = -\frac{1}{6}\phi_{i-1} + \frac{5}{6}\phi_i + \frac{1}{3}\phi_{i+1}$$

$$\phi_{i+1/2}^{(2)} = \frac{1}{3}\phi_i + \frac{5}{6}\phi_{i+1} - \frac{1}{6}\phi_{i+2}$$

权重计算:

$$\omega_k = \frac{\alpha_k}{\sum_j \alpha_j}$$

$$\alpha_k = \frac{d_k}{(\epsilon + \beta_k)^2}$$

其中  $\beta_k$  是光滑性指示子,  $d_k$  是理想权重。

# 时间离散化方案

## 显式 Runge-Kutta 方法

三阶 TVD Runge-Kutta 格式:

$$\begin{aligned}\phi^{(1)} &= \phi^n + \Delta t L(\phi^n) \\ \phi^{(2)} &= \frac{3}{4}\phi^n + \frac{1}{4}\phi^{(1)} + \frac{1}{4}\Delta t L(\phi^{(1)}) \\ \phi^{n+1} &= \frac{1}{3}\phi^n + \frac{2}{3}\phi^{(2)} + \frac{2}{3}\Delta t L(\phi^{(2)})\end{aligned}$$

CFL 稳定性条件:

$$\Delta t \leq C \frac{\Delta x}{\max |\mathbf{U}|}$$

隐式格式 (Backward Euler):

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \mathbf{U} \cdot \nabla \phi^{n+1} = 0$$

方法	精度	稳定性
Forward Euler	$O(\Delta t)$	CFL 限制
RK3-TVD	$O(\Delta t^3)$	CFL 限制
Backward Euler	$O(\Delta t)$	无条件稳定

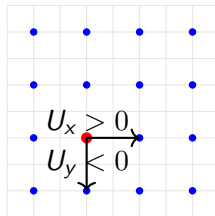
## 离散化方程

$$(I - \Delta t \cdot L)\phi^{n+1} = \phi^n$$

$$L(\phi) = \mathbf{U} \cdot \nabla \phi$$

### 矩阵组装关键步骤:

- ① 边界点处理: 对角元素设为 1.0
- ② 内部点处理: 根据速度方向添加非对角元素
- ③ OpenMP 并行组装: 线程局部 Triplet 列表
- ④ 合并线程局部数据





# 双时间步方法实现

## 双时间步进策略

- 物理时间步：大时间步长  $\Delta t$
- 伪时间步：小时间步长  $\tau$
- 内迭代：每个物理时间步内进行伪时间迭代

离散方程：

$$\left(\frac{1}{\tau} + \frac{1 + \gamma}{\Delta t}\right) \Delta \phi + L(\phi) = \frac{\phi_m}{\tau} + \frac{(1 + \gamma)\phi_n + \gamma(\phi_n - \phi_{n-1})}{\Delta t}$$

矩阵特点：

- 包含物理时间和伪时间项
- 迎风通量离散
- 数值粘性项  $\epsilon$  增强稳定性

# 重初始化的必要性

## 问题提出

水平集函数在演化过程中会逐渐偏离符号距离函数 (SDF), 导致:

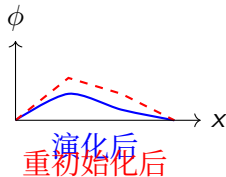
- 梯度  $|\nabla \phi| \neq 1$ , 数值误差累积
- 界面厚度变化, 精度下降

重初始化方程:

$$\frac{\partial \psi}{\partial \tau} + \text{sign}(\phi)(|\nabla \psi| - 1) = 0$$

数值实现:

- 伪时间步进方法
- 迭代求解至稳态



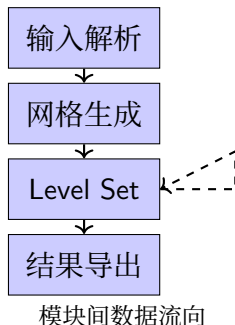
# 核心模块架构

## 主要模块:

- **DFISEParser**: 文件解析与数据结构构建
- **LevelSetMethod**: 水平集数值方法核心
- **BackwardEulerScheme**: 隐式时间推进
- **implicitLUSGS**: 双时间步进方法
- **GeometryProcessor**: 几何处理与格式转换

## 数据流:

输入 → 网格 → 演化 → 输出



# 稀疏矩阵并行组装

## 优化策略:

- ① 使用 Triplet 列表存储非零元素
- ② OpenMP 多线程并行计算
- ③ 线程局部存储避免锁竞争
- ④ 预分配内存减少开销



矩阵非零元素分布

## 关键代码

```
#pragma omp parallel
thread_triplets[thread_id].reserve(...)
#pragma omp for nowait
tripletList.insert(tripletList.end(), ...)
```

## 整体流程概览

- ① **输入处理**: DF-ISE/OBJ 结构文件解析, 几何信息提取
- ② **数据预处理**: 网格生成, SDF 初始化, 材料属性映射
- ③ **数值演化**: Level Set 方法, 界面追踪, 几何约束处理
- ④ **结果导出**: 界面提取, 网格重构, 格式转换

## 主要技术挑战:

- 多材料界面的精确追踪
- 复杂几何结构的网格适应
- 大规模稀疏系统高效求解
- 数值稳定性与精度平衡

## 核心模块:

- DFISEParser
- LevelSetMethod
- BackwardEulerScheme
- GeometryProcessor

# 数值求解流程实现

## 主程序执行步骤

- ① 解析输入文件: `DFISEParser parser(inputFile);  
parser.parse();`
- ② 网格与 SDF 初始化: `levelSet.generateGrid();  
levelSet.initializeSDF();`
- ③ 材料属性设置: `levelSet.setMaterialProperties();`
- ④ 速度场计算: `levelSet.updateVelocityField();`
- ⑤ 数值演化: `levelSet.evolve();` (内部调用时间推进方案)
- ⑥ 结果导出: `levelSet.extractSurfaceMesh();  
ConvertToDFISE();`

## 关键技术实现

稀疏矩阵并行组装 (OpenMP) + BiCGSTAB 迭代求解 (Eigen 库) + 多格式转换支持

# 水平集方法优缺点总结

## 主要优势

- **拓扑灵活性**: 自动处理分裂与合并
- **隐式格式稳定**: 大时间步长, 无条件稳定
- **并行性好**: OpenMP 加速关键计算
- **扩展性强**: 易于推广至高维问题

## 主要挑战

- **内存消耗**: 三维问题内存需求大
- **收敛速度**: 迭代求解器效率问题
- **重初始化**: 额外计算开销
- **边界处理**: 复杂边界条件实现

## 未来工作方向

- GPU 加速计算
- 自适应网格细化 (AMR)
- 混合显式-隐式格式
- 机器学习加速求解