密级 _____

中国科学院大学
University of Chinese Academy of Sciences

博士学位论文

可压缩流动中间断问题的高分辨率数值求解方法及应用

作者姓名 刘利

指导教师 申义庆 研究员

中国科学院力学研究所

学位类别 理学博士

学科专业 流体力学

培养单位 中国科学院力学研究所

2017 年 4 月

Typeset by LaTeX 2$_\varepsilon$ at August 30, 2017

With package `CASthesis` v0.2 of CTEX.ORG

A Study of High-Resolution Algorithms for Discontinuous Problems in Complex Compressible Flows and Their Applications

By

Li Liu

A Dissertation Submitted to
University of Chinese Academy of Sciences
In partial fulfillment of the requirement
For the degree of
Doctor of Fluid Mechanics

Institute of Mechanics
University of Chinese Academy of Sciences

April, 2017

目录

表格

插图

第一章　双信息保存（DIP）方法

界面作为最为常见的物理现象之一，广泛的存在于力学、化学、生物工程、材料科学和计算机图形学等多个学科领域。界面的准确模拟对于多相流动、晶体生长、火焰的发展和传播等等很多方面的研究都有重要意义。

1.1　不同的界面类型和界面方法

界面方法最早可以追溯到 1958 年的洛斯 · 阿拉莫斯国家实验室发展的 Particle-In-Cell(PIC) 方法 [?,?] 。在第一章 ?? 节中，对不同的界面方法都有较详细的介绍。本章中，我们将从方程的角度来分析界面问题。

忽略力学模型，界面的运动可以抽象为求解对流方程

$$\frac{\partial z}{\partial t} + \boldsymbol{V} \cdot \nabla z = 0 \tag{1.1}$$

其中 $\boldsymbol{V}$ 为速度场，z 为区分界面两侧物质的物理量，如比热比 γ、密度 ρ 等。下面将分别对不同类型的界面进行分析。

1.1.1　传统界面

尽管方程 (1.1) 是最简单的对流方程，然而，如果我们关注于界面的运动时，想要实时的得到几何面（线）是异常困难的。以一维图 1.1 为例，如果直接对方程（1.1) 进行求解，例如采用差分方法，随着间断被耗散，界面将无法识别。

因此，绝大多数界面方法并不直接求解方程 (1.1) 本身，而是采用一种追踪的视角进行模拟，图 1.2 给出了不同界面方法的一维示意图。其中锋面追踪方法直接追踪界面；MAC 方法在界面一侧添加标记点；而 VOF 方法在每一个网格引入一个体积分数函数，将界面的运动转化为体积分数的变化。只有 level set 方法求解方程 (1.1)，但是也并不直接求解间断函数 z 本身，而是以到界面距离为新的函数 ϕ，求解

$$\frac{\partial \phi}{\partial t} + \boldsymbol{V} \cdot \nabla \phi = 0$$

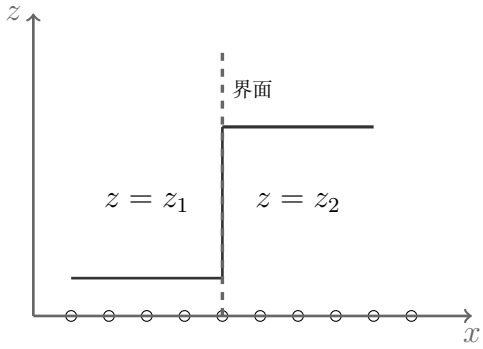


图 1.1: 界面的一维示意图

如图 1.2.d 所示，level set 方法将 $\phi = 0$ 的位置识别为界面位置。

这些界面方法明显可以分成两类，一类全计算域计算的界面捕捉方法，如 VOF 和 level set 方法，另外一类只对部分区域进行 Lagrangian 追踪的界面追踪方法，如锋面追踪、MAC 方法等。两类方法各有优势，全域计算的方法更适合处理光滑的几何形状，并且易于处理由于由于流体压缩性导致的膨胀过程，如图 1.3.a 所示；Lagrangian 局部追踪的方法更易于处理锋利夹角、大变形以及界面破碎等问题，如图 1.3.b 所示。由于各有明显的优势和缺点，有学者尝试将两种类型方法结合，如 level set-粒子方法，VOF-粒子方法等方法。

1.1.2 多相界面和耗散界面

除了经常研究的两相界面外，在工业、化学、生物等很多领域中存在三相甚至更多相物质之间的相互作用，我们可以将这种问题称为多相界面问题。相比两相界面丰富的研究，多相界面的研究无论在理论还是数值方面都很少 [?]。我们在第一章中对耗散界面做过简单介绍，耗散界面同样也是较难处理的一类界面问题。

传统界面类方法求解多相界面和耗散界面困难的根本原因是由于这些方法都不是针对某一真实的物理量进行求解，如 γ，ρ，而只是从拓扑角度计算界面的运动。这一问题导致传统界面类方法无法直接求解如图 1.4 所示的多相界面和具有界面厚度 δ 的耗散界面。

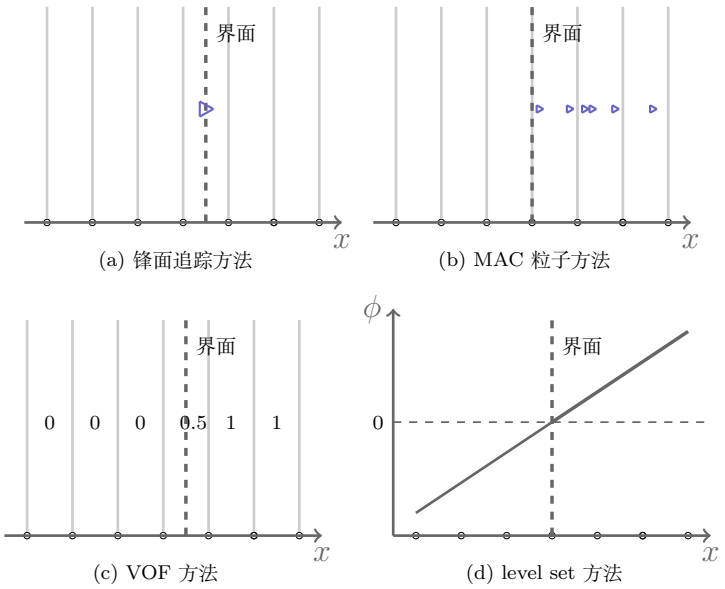


图 1.2: 不同界面方法一维示意图

1.2　双信息保存方法基本思想

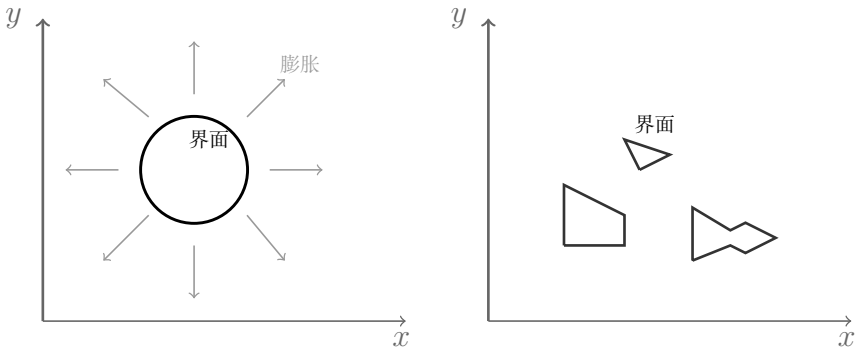
首先我们考虑两个问题，

1, 界面捕捉类方法由于采用全计算域求解，可以更好的处理界面膨胀（图 1.4.a）、界面张力等光滑函数问题；界面追踪方法由于采用拉格朗日运动思想可以更好的保持界面形状、计算界面大变形和破裂等问题。我们能否同时具备两方面的优点呢?

2, 传统界面方法都不是直接求解方程（1.1）中的和物理相关的量 z ，因此较难处理多相界面和耗散界面。界面方法能否针对 z 直接求解?

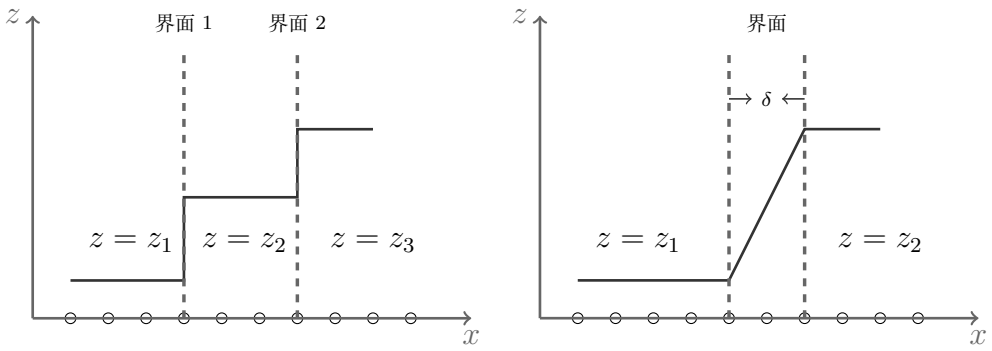
1.2.1　信息点

粒子方法是一种最直观的界面方法，具有优良的 Lagrangian 特性，但是由于它明显的不足，学者们更愿意将它作为一种辅助手段，如用作示踪粒子，或者和其它方法结合，用粒子来修正计算结果，如 level set-粒子方法，VOF-粒子方法等。粒子方法最大的问题是粒子的离散性和分布的随意性，难以保证全域的覆盖，这样就无法像网格方法一样准确的得到任意位置的信息。粒子类界面



(a) 界面捕捉方法　　(b) 界面追踪方法

图 1.3: 不同类型界面方法适合求解的工况二维示意图

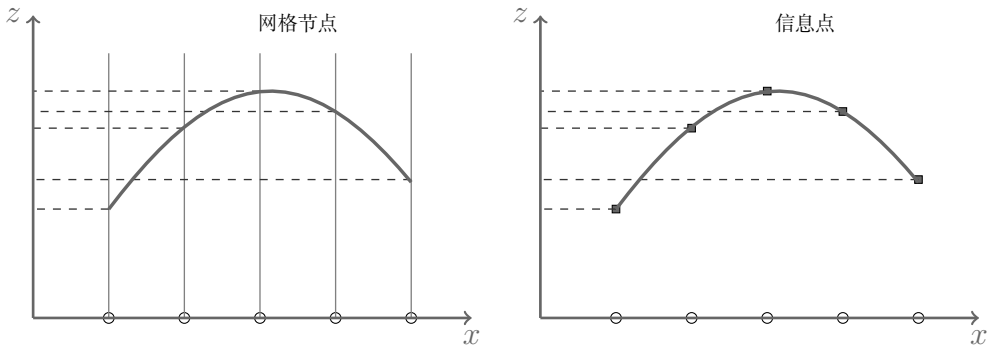


(a) 多相界面 1D 示意图　　(b) 耗散界面 1D 示意图

图 1.4: 多相界面和耗散界面一维示意图

方法主要包括早期的 Particle-In-Cell（PIC）方法和目前仍在使用的 Marker-In-Cell（MAC）方法。PIC 方法采用携带有质量的真实粒子模拟流体的运动，根据粒子携带的信息可以判断界面位置，然而，由于真实粒子无法人为的生成和抹去，无法保证全计算域每个网格都含有粒子。Marker-and-cell (MAC) 方法采用标记粒子，粒子除位置以外不含有其它信息，是完全的虚拟粒子，增加和减少都不会影响流场的物理性质，但是由于它完全不携带信息，只能通过有粒子和没有粒子判断界面，仍然无法做到覆盖计算域。

我们知道网格是对真实流场的离散，这种离散其实是真实流场信息在网格点上的映射，我们可以随时加密网格、减少网格或移动网格，尽管这可能影响计算精度，但是并不会改变所研究问题本质。如图 1.5 类比于网格点，我们可



(a) 网格节点映射　(b) 信息点映射

图 1.5: 信息点和网格节点类比一维示意图

以引入一套和网格点相似的、携带有输运值 z 在该空间位置投影的粒子，这种粒子可以根据需要增加、减少和移动。我们将这种携带有输运值的 Lagrangian 粒子称为**信息点**。

本文的方法中同时含有两类信息点，因此将该方法称为双信息保存方法，下文将分别介绍这两类信息点。

1.2.2　单元信息点（单元点）

对于固定网格方法，网格点位置是确定的，我们每个时刻都能得到确定的空间点上的计算值，并可以通过高阶插值运算得到空间任何点的近似值；对于运动网格，如自适应加密网格，我们也可以人为的控制网格加密规律，避免过分畸形的网格分布和网格形状。然而，粒子点的运动完全是由速度场决定的，所以粒子点的分布具有很高的随意性。MAC 方法也遇到相同的困难，某单元原本含有粒子，但是随着流场膨胀，单元不再含有粒子，但这并不代表该单元产生了相变。MAC 方法的解决策略是在初始计算时，在相应状态的单元内放置多达 16 个粒子来避免这种错误的发生。但是这样即增加计算和存储，又不能从根本上避免粒子分布随意性这一问题。

对于信息点可以随时生成和抹去这一特点，我们为什么不构造一个限定在单元内的信息点呢？既可以解决粒子分布问题，又有利于单元上输运值和粒子上输运值之间的传递。我们将这种每个单元内唯一的信息点，称为单元信息点，简称为**单元点（cell-point）**。下面我们将从一维来介绍单元点求解方程 (1.1) 的步骤。

一. **初始化**

如图 1.6 我们首先将 $[i-1/2, i+1/2]$ 定义为第 i 单元。开始计算时，在每个单元引入一个单元点。由于单元点是和每个单元关联的，所以单元点无需特殊编号进行区分，单元点的位置可以根据在单元内相对坐标 $X(i)$ 给出，为了简便，下面简写为 X。单元点的初始条件为，

$$X(i) = 0, \quad \bar{z}(i) = z(i) \tag{1.2}$$

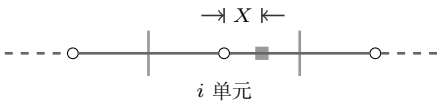


图 1.6: 单元信息点一维示意图

二. **运动追踪**

单元点的运动速度是由所在的速度场决定的。通常计算中，例如在流体计算中，可以通过其它方程得到的网格点上的速度。然后通过插值可以近似求得粒子点所在位置的速度。理论上可以构造任意高阶的插值，为了简便，我们只采用两个单元的线性插值，以 i 单元的信息点为例

$$\bar{u}(i) = (1 - |X|)u(i) + |X|u(i + s_x) \tag{1.3}$$

其中 $s_x = \text{sign}(X)$，通过相对坐标的符号决定是向正向还是向负向插值。

当粒子开始运动后我们需要每一步确定 i 单元点是否运动出单元，如果运动出单元新的位置，我们需要确定单元点新的位置。单元点 i 经过 Δt 时间内运动到新坐标

$$L_x = X + \Delta t\bar{u}/\Delta x \tag{1.4}$$

所在单元 M 为

$$M = i + \text{floor}(L_x + 0.5) \tag{1.5}$$

其中 floor 为向上取整函数，由于 M 和 i 可能不是同一个单元，相对坐标需要更新为

$$\bar{X} = L_x - \text{floor}(L + 0.5) \tag{1.6}$$

三. 单元点的更新

为了保证单元和单元点的一一对应，我们需要在每一时间步结束时对单元点进行更新，将含有多余单元点的和不含单元点的单元进行处理。

我们首先考虑这样一个问题，对于图 1.7 中所示的情况，i 单元点上 $z=1$，$i+1$ 单元点上 $z=0$，那么界面究竟是图中哪种情况呢？这其实是无法确定的，需要进行统一的规定。根据所研究的问题是否涉及界面厚度分成无厚度界面和耗散界面两类。

1). 无厚度界面问题中单元点的消去方法

如果我们研究的是无厚度的传统界面或多相界面，我们规定界面位于 z 值较小的信息点位置，既图 1.7 中第二种情况。

如果一个单元内有多个信息点，如图 1.8，根据上面界面位置的规定，我们选择输运值 z 较小的单元点。

2). 无厚度界面问题中单元点的生成方法

在 $k+1$ 时间步，对于没有单元点的单元 i，我们需要在单元中心引入新的单元点，单元点上对流信息 z 可以通过**回溯生成法**得到。如图 1.11，回溯生成法的做法如下：

i. 首先在计算时刻 t_{k+1} 的上一时间步 t_k，假设我们曾在某一位置引入了一个新的单元点 P，该点速度为 u'；

ii. 经过时间 Δt，在 $k+1$ 时间步该单元点正好运动到 i 单元中心 $X=0$；

iii. 由于我们不知道速度 u'，我们假设 u' 为 $k+1$ 时间步 i 单元中心的速度：

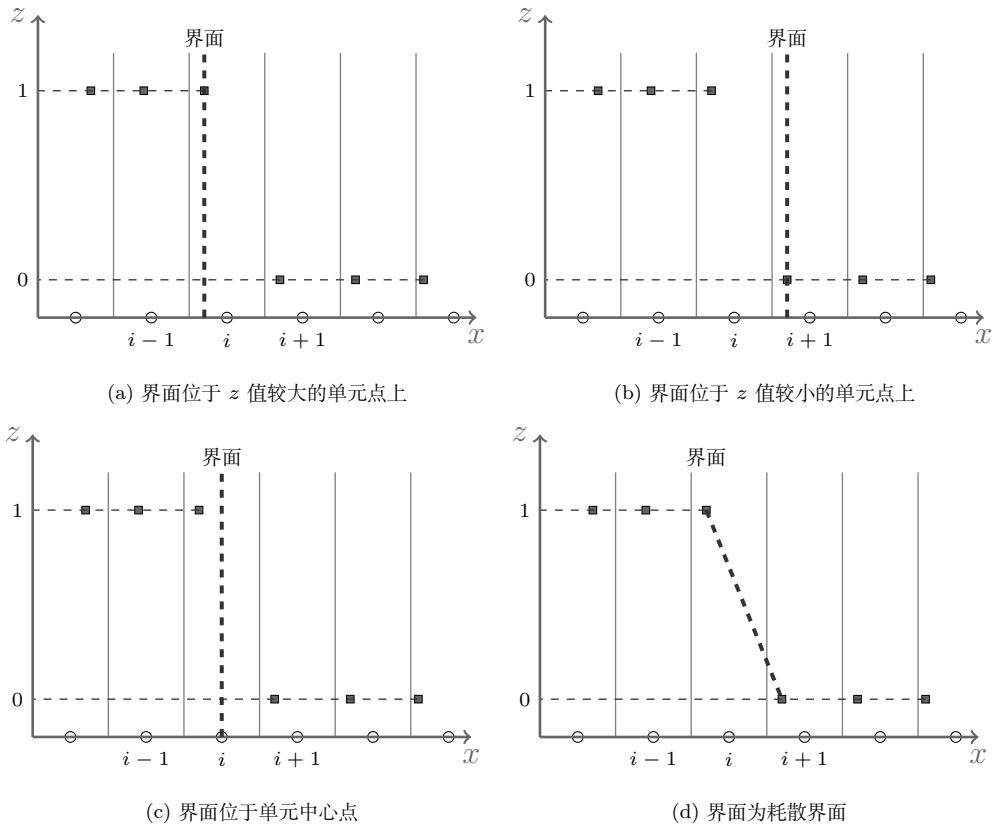
$$u' = u^{k+1}(i)$$

iv. 通过逆速度，可以得到临时坐标为

$$L_x = -u'\Delta t/\Delta x \tag{1.7}$$

根据公式 (1.5) 和 (1.6) 可以知道 P 在 k 时间步的单元为

$$M = i + \mathrm{floor}(L_x + 0.5)$$



(a) 界面位于 z 值较大的单元点上

(b) 界面位于 z 值较小的单元点上

(c) 界面位于单元中心点

(d) 界面为耗散界面

图 1.7: 界面在单元点间可能的分布情况一维示意图

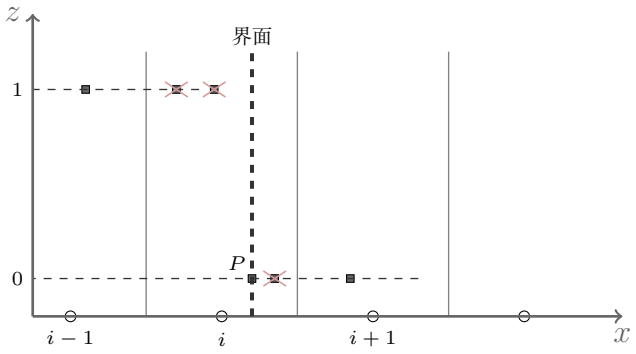


图 1.8: 无厚度界面情况中单元点的消去方法一维示意图

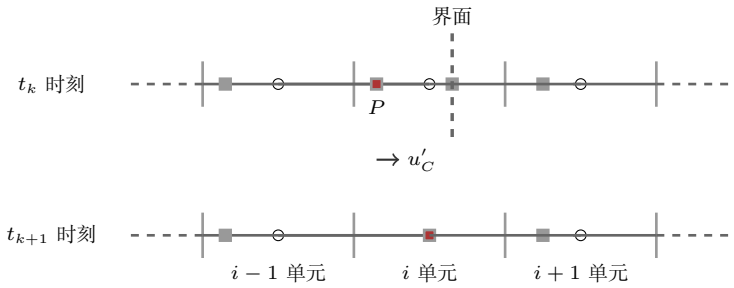


图 1.9: 单元点的回溯生成法一维示意图

由于 k 时间步是已知的，根据求得的位置就可以知道该单元输运值，即

$$\bar{z}^{k+1}(i) = \bar{z}^{k}(M)$$

3). 耗散界面问题中单元点的消去方法

如果我们研究的是耗散界面，则选择图 1.7 中第四种情况做为界面真实情况。由于耗散界面存在过渡值，如图 1.10 ，如果单元内含有多个单元点，我们采取所有单元点位置和输运值 z 算数平均的形式。

4). 耗散界面问题中单元点的生成方法

在耗散界面问题中由于没有清晰的分界面，允许存在一定的耗散，因此我们采用另外一种插值的方法生成单元点。如图 1.11 所示，如果单元 i 没有单元点，我们在单元中心生成新的单元点，单元点上的输运值可以采用前后临近单元点加权平均的方式得到，权重反比于到 i 单元中心的距离：

$$\hat{z}_i = \frac{\omega_1 \bar{z}_{i-1} + \omega_2 \bar{z}_{i+1}}{\omega_1 + \omega_2} \tag{1.8}$$

其中，$\omega_1 = 1/L_1$，$\omega_2 = 1/L_2$。

每一时间步，经过单元点运动和更新的计算，我们可以得到每个单元中单元点的位置和输运值，然后将单元点的输运值作为该单元输运值。

1.2.3　粒子信息点（粒子点）

上文中，我们引入了单元信息点，通过单元信息点可以直接求解方程 (1.1)，无论是传统的无厚度界面、多于两相物质的多相界面还是具有界面厚度的耗散

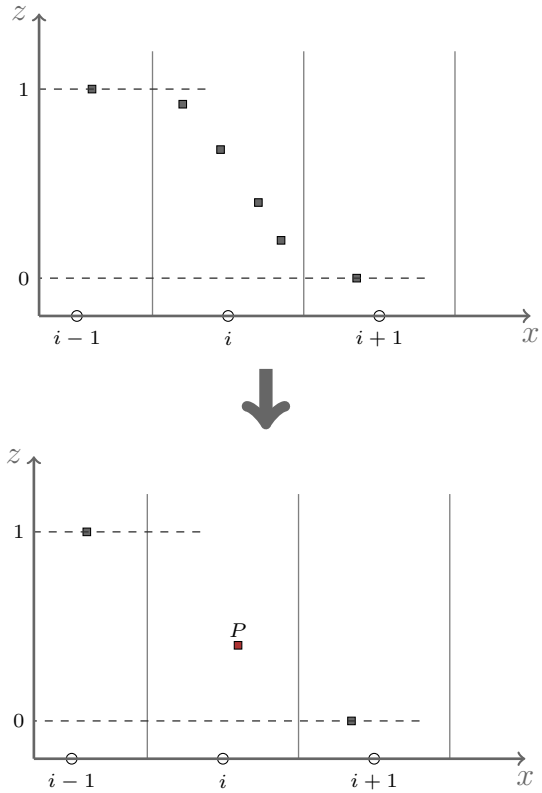


图 1.10: 耗散界面情况中单元点的消去方法一维示意图

界面都可以进行模拟。

单元信息点只具备一半的 Lagrangian 特性，尽管它弥补了粒子点不能全域覆盖、无法计算耗散界面等缺点，但由于每步的更新，新的单元点会不断替代旧的单元点，而新的单元点的生成存在误差，导致对界面结构的保持并不像纯粒子方法那么好，这一点将会在算例中展示。为了改进这一不足，我们引入了另外一套全域追踪的信息点，在每一步对单元点进行修正。将这种全域追踪、具有纯粒子点特性的单元点，称为**粒子信息点**，简称为粒子点。下面我们给出通过粒子点修正单元点的过程。

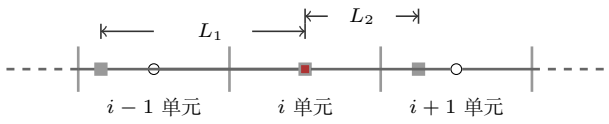


图 1.11: 耗散界面问题中单元点的生成方法一维示意图

一. **初始化**

初始化时，在每个单元中心引入一个粒子点，由于粒子点是全域追踪的，我们需要为每一个粒子点建立一个无冲突的标记，这里以最开始所在单元作为标记。对于某一标记为 i 的粒子点，我们需要给出两个数组分别记录它所在单元 $ip(i)$ 和单元相对坐标 $X_p(i)$。初始化过程和单元点相似

$$ip(i) = i, \quad X_p(i) = 0, \quad \bar{z}_p = z(i) \tag{1.9}$$

二. **运动追踪**

粒子点的运动和单元点相同，我们首先根据粒子点 i 所在单元 ip 和 X_p 线性插值出粒子点的运动速度

$$\bar{u}_p = (1 - |X_p|)u(ip) + |X_p|u(ip + s_x) \tag{1.10}$$

其中 $s_x = \text{sign}(X_p)$。Δt 时间后单元点坐标更新为

$$L_x = X_p + u_p \Delta t / \Delta x \tag{1.11}$$

新的单元位置和相对坐标变化为

$$ip = ip_{\text{old}} + \text{floor}(L_x + 0.5) \tag{1.12}$$

$$X_p = L_x - \text{floor}(L_x + 0.5) \tag{1.13}$$

三. **修正单元点**

在 t_k 时刻，假设我们已经更新了单元点，如果单元 ip 内含有粒子点，我们则采用粒子点重新更新单元点，关于新的单元点更新策略，无论是对于无厚度界面还是对耗散界面来说，都和通过单元点更新完全相同，只是用粒子点取代了原本的单元点。

1.2.4　边界处理

粒子方法的边界条件都很容易实现，下面我们分别介绍几种常用的边界条件。如图 1.12，首先假设我们计算从第 0 单元到 Nx 单元，DIP 方法由于在计算

中可能需要临近网格的信息，因此在左右分别构造一个虚拟网格 -1 和 $Nx+1$。

图 1.12: 边界和虚拟网格一维示意图

一. **周期边界条件**

对于单元点，将 1 单元内单元点直接赋值给 $Nx+1$ 虚拟单元，将 $Nx-1$ 单元内单元点直接赋值给 -1 单元

$$\begin{aligned} &X(-1)=X(Nx-1),\quad X(Nx+1)=X(1),\\ &\bar{z}(-1)=\bar{z}(Nx-1),\quad \bar{z}(Nx+1)=\bar{z}(1) \end{aligned} \tag{1.14}$$

对于粒子点，如果有粒子流出，即 $ip(i)>Nx$ 或 $ip(i)<0$，则

$$\begin{aligned} &ip(i)=ip(i)-Nx,\quad \text{if}\quad ip(i)>Nx\\ &ip(i)=ip(i)+Nx,\quad \text{if}\quad ip(i)<0 \end{aligned} \tag{1.15}$$

而 $X_p(i)$ 保持不变。

二. **入流条件**

假设左侧为入流，对于单元点，我们令左侧虚拟单元 -1 中的单元点始终放在单元中心，输运值为入流值。

$$X(-1)=0,\quad \bar{z}(-1)=z_{\text{in}} \tag{1.16}$$

对于粒子点随着流体流入，我们在入流中添加新的粒子点，粒子点的总数会出现浮动，假设粒子点的数量用 N_p 表示，初始时 $N_p=Nx$。当 0 单元中不存在粒子点时，将新的粒子点放置在单元中心，粒子点的输运值为入流输运值，粒子点总数加 1。

$$\begin{cases} N_p = N_p + 1 \\ ip(N_p) = 0 \\ X_p(N_p) = 0 \\ \bar{z}_p(N_p) = z_{\text{in}} \end{cases} \quad \text{if} \quad \text{Mrk}(0) = 0 \tag{1.17}$$

其中 Mrk 为单元中粒子点数目标记函数。

三. **出流条件**

和入流相似，假设右侧为出流条件，对于单元点，我们令虚拟单元 $Nx+1$ 中的单元点始终放在单元中心，输运值为 Nx 单元点输运值。

$$X(Nx+1) = 0, \quad \bar{z}(Nx+1) = z(Nx) \tag{1.18}$$

对于粒子点，随着流出，粒子点的总数会减少，当有粒子点 $ip(i) > Nx$ 时，我们将它去掉，并调整序号。

$$\begin{cases} ip(i) = ip(N_p) \\ X_p(i) = X_p(N_p) \\ \bar{z}_p(i) = \bar{z}_p(N_p) \\ N_p = N_p - 1 \end{cases} \quad \text{if} \quad ip(i) > Nx \tag{1.19}$$

四. **固壁反射条件**

假设图 1.7 中，左边为固壁，对于单元点有边界条件

$$X(0) = 0, \quad X(-1) = -X(1), \quad \bar{z}(-1) = \bar{z}(1) \tag{1.20}$$

其中 $\bar{z}(0)$ 不做特殊处理。

对于粒子点有，边界无法穿透，因此如果有粒子穿越边界，我们将它做反射

$$\begin{cases} \text{tmp} = ip(i) + X_p \\ ip(i) = -\text{floor}(\text{tmp} + 0.5) \\ X_p(i) = -(\text{tmp} + ip(i)) \end{cases} \quad \text{if} \quad \text{tmp} < 0 \tag{1.21}$$

1.2.5 DIP 方法一维伪代码

为了更清晰的演示 DIP 方法的求解过程，下面我们给出 DIP 方法在一维中的伪代码。其中方框内为求解耗散界面时需要用到的代码。

一维双信息保存方法（DIP）伪代码

初始化

```
！单元点初始化
DO  i = 0, Nx
    X(i) = 0
    z̄(i) = z(i)                          ！其中 −0.5 < X(i) ≤ 0.5 为单元内坐标
ENDDO

！粒子点初始化
Np = Nx                                  ！Np 为粒子点总数，会因为出流和入流变化
DO  i = 0, Np
    ip(i) = i
    X_p(i) = 0                           ！其中 N_p 为粒子点总数，会因为出流和入流变化
    z̄_p(i) = z(i)
ENDDO

DO it = 1, NT ！时间循环
    Mrk(i) = 0
    Mrk_p(i) = 0                         ！用来标记第 i 单元中临时的单元点数量和粒子点数量
```

1. 单元点的运动追踪

```
DO i = −1, Nx + 1
    s_x = sign(X(i))
    ū(i) = (1 − |X|)u(i) + |X|u(i + s_x)          ！速度插值
    L_x = X(i) + ū(i)Δt/Δx,
    X(i) = L_x − floor(L_x + 0.5)
    M(i) = i + floor(L_x + 0.5)                   ！i 单元点移动到 M 单元
```

2. 单元点的更新

```
IF Mrk(M) = 0 THEN
    X′(M) = X(i)
    z̄′(M) = z̄(i)
ELSE
    IF z̄(i) < z̄′(M) THEN
        X′(M) = X(i)                              ！多个单元点，选取 z 值小的点
        z̄′(M) = z̄(i)
    ENDIF
ENDIF
X′(M) = (X(i) + Mrk(M)X′(M))/(Mrk(M) + 1)         ！多个单元点，进行平均
z̄′(M) = (z̄(i) + Mrk(M)z̄′(M)/(Mrk(M) + 1)
Mrk(M) = Mrk(M) + 1
END DO
```

3. 粒子点的运动追踪

```
DO i = 0, Np
```

$s_x = sign(X_p(i))$
$\bar{u}_p(i) = (1 - |X_p|)u(ip) + |X|u(ip + s_x)$　　！速度插值
$L_x = X_p(i) + \bar{u}_p(i)\Delta t/\Delta x$
$X_p(i) = L_x - floor(L_x + 0.5)$　　！i 粒子点新位置
$ip(i) = ip(i) + floor(L_x + 0.5)$

4. 粒子点修正单元点

IF $\mathrm{Mrk}_p(ip) = 0$ THEN
$X'(ip) = X_p(i)$
$\bar{z}'(ip) = \bar{z}_p(i)$
ELSE
IF $\bar{z}_p(i) < \bar{z}'(ip)$ THEN
$X'(ip) = X_p(i)$
$\bar{z}'(ip) = \bar{z}_p(i)$　　！多个粒子点，选取 z 值小的点
ENDIF
ENDIF
$X'(ip) = (X_p(i) + \mathrm{Mrk}_p(ip)X'_p(ip))/(\mathrm{Mrk}_p(ip) + 1)$
$\bar{z}'(ip) = (\bar{z}_p(i) + \mathrm{Mrk}_p(ip)\bar{z}'_p(ip))/(\mathrm{Mrk}_p(ip) + 1)$　　！多个粒子点，取平均
$\mathrm{Mrk}_p(ip) = \mathrm{Mrk}_p(ip) + 1$
END DO

5. 在空单元生成单元点

DO $i = 0, Nx$
IF $\mathrm{Mrk}(i) + \mathrm{Mrk}_p(i) = 0$ THEN
$L_x = -u(i)\Delta t/\Delta x$
$M = i + \mathrm{floor}(L_x + 0.5)$
$X'(i) = 0$
$\bar{z}'(i) = \bar{z}(M)$　　！回溯生成单元点
$L_1 = |X'(i + 1) + 1|$
$L_2 = |X'(i - 1) - 1|$
$\bar{z}'(i) = (\bar{z}'(i + 1)/L_1 + \bar{z}'(i - 1)/L_2)/(1/L_1 + 1/L_2)$
ENDIF
END DO
DO $i = 0, Nx$
$\bar{z}(i) = \bar{z}'(i)$
$X(i) = X'(i)$
$z(i) = \bar{z}(i)$
END DO

1.2.6　二维 DIP 方法

粒子方法具有极好的高维拓展性，我们只需要增加 Y 方向的位置函数即可求解二维问题。下面以单元点为例简单的介绍下 DIP 方法求解二维方程 (1.22)

的实现过程，详细内容可见附录。

$$\frac{\partial z}{\partial t}+u\frac{\partial z}{\partial x}+v\frac{\partial z}{\partial y}=0 \tag{1.22}$$

一. **初始化**

初始化除了维度增加以外完全和一维相同：

$$X(i,j)=0,\quad Y(i,j)=0,\quad \bar{z}(i,j)=z(i,j) \tag{1.23}$$

二. **单元点的运动追踪**

首先通过一维线性插值的方式得到单元点速度：

$$\begin{cases}\bar{u}(i,j)=(1-|X|)u(i,j)+|X|u(i+s_x,j)\\ \bar{v}(i,j)=(1-|Y|)v(i,j)+|Y|v(i,j+s_y)\end{cases} \tag{1.24}$$

为了表达简单，偶尔在不混淆的情况下将 $X(i,j)$ 和 $Y(i,j)$ 简写为 X 和 Y。其中

$$\begin{cases}s_x=\mathrm{sign}(X)\\ s_y=\mathrm{sign}(Y)\end{cases}$$

临时坐标变化：

$$\begin{cases}L_x=X(i,j)+\bar{u}(i,j)\Delta t/\Delta x\\ L_y=Y(i,j)+\bar{v}(i,j)\Delta t/\Delta y\end{cases}$$

所在单元变化和相对坐标变化为

$$\begin{cases}M(i,j)=i+\mathrm{floor}(L_x+0.5)\\ N(i,j)=j+\mathrm{floor}(L_y+0.5)\\ X(i,j)=L_x-\mathrm{floor}(L_x+0.5)\\ Y(i,j)=L_y-\mathrm{floor}(L_y+0.5)\end{cases}$$

三. **单元点的更新**

单元点的更新过程和一维中相同，其中耗散型界面的单元点生成仍采

用临近单元单元点加权平均的形式，

$$\bar{z}(i,j)=\sum_{i_1,j_1}\left[\omega_{i_1,j_1}\bar{z}(i+i_1,j+j_1)\right]/\sum_{i_1,j_1}\omega_{i_1,j_1}$$

其中 $i_1=-1,1$，$j_1=-1,1$。权重函数反比于距离，$\omega_{i_1,j_1}=1/L_{i_1,j_1}$ ，而距离很容易求得：

$$L_{i_1,j_1}=\sqrt{(X(i+i_1,j+j_1)+i_1)^2+(Y(i+i_1,j+j_1)+j_1)^2}$$

1.3　数值算例

下面我们将通过一维和二维算例验证新方法的表现。

1.3.1　一维数值算例

1. 均匀速度场算例对于均匀速度场，粒子点和单元点是完全重合的，信息点间的距离不会变化，所以也不会出现一个单元存在多个或没有单元点的情况。

   我们首先测试一个在 $u=1$ 的均匀速度场中间断函数的传播问题。初始条件为

   $$z=\begin{cases}1, & \text{if}\quad x<0.2\\ 0, & \text{if}\quad x\geq 0.2\end{cases}$$

   计算域为 $[0,1]$，边界采用周期边界条件。计算采用的网格为 $N=20$，计算时间步长为 $\Delta t=0.005$，图 1.13 给出 23 个时间步后单元点的分布情况。从该算例可以看出，粒子类方法由于追踪的连续性，即便在极其稀疏的网格中，也能对每个信息点上的信息进行准确的追踪。

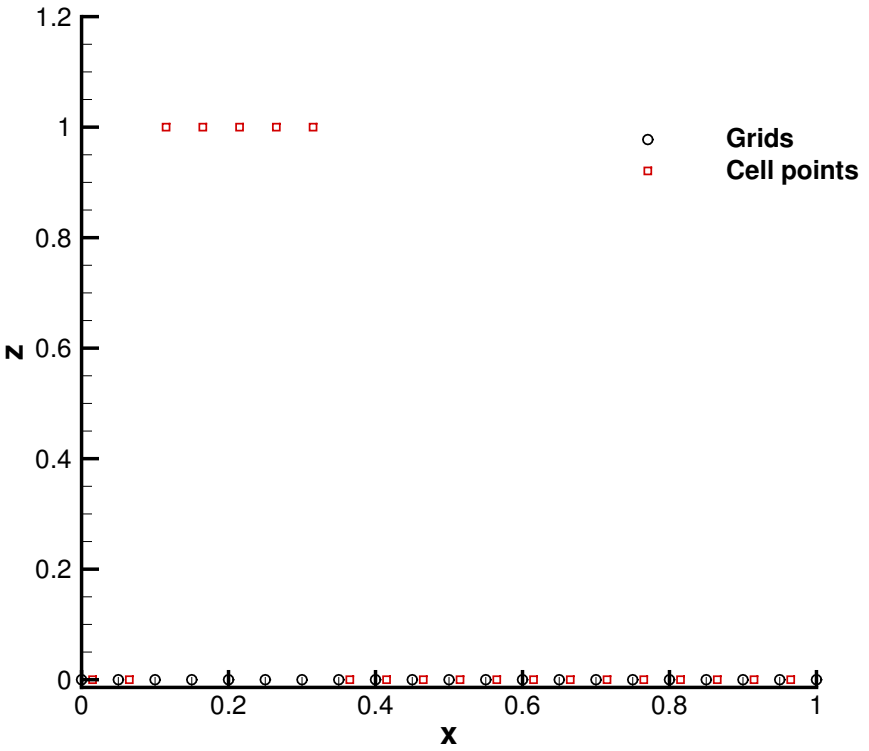


图 1.13: 一维线性对流算例，均匀速度场 $u = 1$，时间步 $Nt = 23$