arXiv:2302.03983v2 [cs.CE] 1 Feb 2024

X-Mesh：一种模拟具有清晰界面的两相流的新方法

Antoine Quiriny[a]、Jonathan Lambrechts[a]、Nicolas Mo¨es[b, c]、Jean-Franc¸ois Remacle[a]

[a]力学、材料与土木工程研究所（*iMMC*），乔治·勒梅特大道 *4* 号，*1348* 新鲁汶，比利时[b]南特大学´、 ´南特中央理工学院，*No¨e* 街，*44321* 南特，法国[c]法国大学研究院（*IUF*）

摘要

运动边界和界面的精确建模是计算力学中许多情形面临的难题。我们采用极端网格变形方法（X-Mesh），结合有限元方法模拟两种不互溶流体之间的相互作用，同时无需重新划分网格，即可准确、清晰地描述界面。在这一新方法中，网格会在局部发生变形，以始终贴合界面，但这可能导致退化单元。通过在界面处施加压力跳跃条件，将两种流体之间的表面张力纳入计算；将其与 X-Mesh 框架结合后，即可实现完全清晰的界面。如果数值格式无法恰当地平衡表面张力与压力梯度，就会产生称为虚假流或寄生流的数值伪影。本文提出的方法具有良好的平衡性，可将这类流动降低至机器精度水平。

1. 引言

自20世纪70年代初以来，有限元方法一直是计算力学中的主流方法。除了有限元建立在稳健的数学理论之上这一事实之外，有限元广受欢迎的一个根本原因是，它能够使用非结构网格，从而精确地离散化边界或界面，例如两种不同材料之间的界面。当这些界面事先已知时，可以将其表示在待建模部件的数字蓝图中。现代自动网格生成器（如 Gmsh）能够生成符合这些界面的网格

。当物理模型涉及不连续的材料属性时，界面可能形成或萌生、扩展、改变拓扑结构或消失。根据材料粒子是否随界面移动，这些界面可以是物质界面，也可以是非物质界面。这些界面的位置和速度事先并不明确，而是计算的一部分。

本文研究两种互不相溶流体之间的物质界面，其中黏度或密度等材料属性是不连续的。表面张力的存在还会导致两种流体交界面处出现压力跃变。关于使用有限元模拟两相流，已有大量文献 [3, 4, 5]。这些方法可分为两大类：界面追踪方法和界面捕捉方法。

对于界面追踪方法，网格负责追踪界面，并在界面移动过程中始终与其保持贴合。此类方法中最精确的是 ALE 方法 [6]，其中网格节点随界面速度移动。ALE 方法具有出色的守恒特性 [7]。这类方法精确且相对容易实现，但其最大的缺点是——至少在纯粹的形式下——无法处理界面的大幅变形或拓扑变化。一些论文 [8] 提出将网格自适应与 ALE 相结合，但频繁进行网格自适应会导致解在时间上不连续。通过仅进行少量网格自适应并使用不动点算法 [9]，可以缓解这一问题。尽管如此，这些技术引入了随时间进行网格自适应的复杂性和相对脆弱性，因而偏离了 ALE 方法原本的简洁性。

界面捕捉方法通常采用固定网格，并在该固定网格上离散一个指示函数，以表示界面的位置 [10,11]。例如，在水平集方法中， [12], 指示函数是到界面的有符号距离，该函数的零等值线表示界面。界面捕捉方法与界面追踪方法各有优缺点。它们允许拓扑变化，但需要对有限元公式进行重大修改。

需要注意的是，界面追踪和

捕捉方法都有锐界面版本和弥散界面版本。弥散方法是对两种流体的物理性质进行正则化，也就是说，在厚度为 ϵ 的区域内平滑两种流体的黏度和密度。可以证明，弥散物理性质所引入的误差为 $\sqrt{\epsilon}$ [13] 阶，这意味着只有在界面附近对网格进行加密（必要时采用各向异性加密），弥散方法才能达到较高精度。

最近，我们开发了一种追踪方法，符合以下 X-Mesh 规范：i）保持界面清晰，ii）在固定拓扑网格上运行，iii）允许界面发生大幅拓扑变化。第一篇 X-Mesh 论文探讨了使用斯特藩模型 [14]计算相变过程中的非物质界面。本文旨在将 X-Mesh 应用于物质界面，尤其是求解两相流问题。

X-Mesh 的核心思想是允许单元变形至测度为零。例如，三角形可以变形成一条边，甚至一个点。这一想法相当大胆，彻底重塑了网格剖分领域与计算领域之间的互动方式；数十年来，双方一直致力于通过精美的网格开展合作。允许单元的测度为零，可以使网格随时间连续变形，同时实现：i）接力传递：将界面从一个节点传递到另一个位于相同位置的节点，从而使界面传播；ii）播种：将一个单元缩减为一个点，由此成核生成界面，随后界面展开（之后再将前沿传递至外部节点）；iii）湮灭：两个界面接触时，可以将界面从节点上分离，以模拟界面合并。

2. 控制方程

我们考虑两种互不相溶、不可压缩的牛顿流体通过界面 Γ相互作用时的流动。每种流体的速度和压力随时间的演化

流体随时间的演化由不可压缩纳维-斯托克斯方程描述：

$$\rho\left(\partial_t \mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f} \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0$$

其中 $\mathbf{u}$ 是速度场（u，v）^{T}，p 是压力，ρ 是密度，$\mathbf{f}$ 是超距力（重力），μ 是动力黏度。

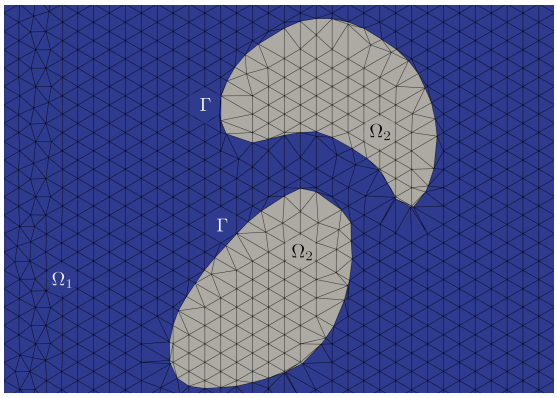


图1：两相流示意图。

两种流体相分别记为 Ω_1 和 Ω_2（见图1），其动力黏度和密度分别为（ρ_1, μ_1）和（ρ_2, μ_2）。因此，这些材料属性在界面 Γ处是不连续的。Navier–Stokes 方程在两个子域上同时求解，并且需要在界面处施加边界条件。方程 1 分别对应动量守恒和质量守恒。这些守恒定律必须在两种流体之间的界面处得到满足。质量守恒要求界面处法向方向上的速度连续，从而防止物质在两相之间传递：

$$[\mathbf{u}]\cdot\mathbf{n} = \mathbf{0}$$

其中，括号 [·]表示界面处的跃变。

除了界面的不可渗透条件外，还必须满足界面垂直方向上的动量守恒。为满足这一条件，法向应力的跳跃由表面张力抵消

$$\left[-pI+\mu\frac{1}{2}\left(\nabla\mathbf{u}+\nabla\mathbf{u}^{T}\right)\right]\cdot\mathbf{n}=\sigma\kappa\mathbf{n} \tag{2}$$

其中 I 为恒等算子， σ 为表面张力系数， κ 为界面的曲率。

在两相流中，主要挑战是准确计算界面位置。所谓准确的界面位置，包含两方面含义：我们必须能够计算拓扑发生变化的复杂移动界面；理想情况下，还应计算出能够使两种流体各自的物质守恒的界面位置。在这篇首篇 X-Mesh 多相论文中，我们着重讨论这两项挑战中的第一项，并将展示如何利用连通性固定的网格追踪拓扑发生变化的复杂前沿。本文所述方法无法确保质量守恒，这一点将在结论中简要讨论；不过，这是我们希望在未来工作中解决的问题。为描述界面位置，我们在此采用一种经典方法。指示函数$\phi(x,y)$在与用于离散化流体压力和速度相同的网格上进行离散化。该水平集函数 [15]通常表示到Γ的有符号距离。因此，在标准设定下，水平集函数 ϕ 随时间的演化以及界面的演化由对流方程控制：

$$\partial_t\phi+\mathbf{u}\cdot\nabla\phi=0 \tag{3}$$

其中，**u**对应于由方程（1）得到的流体速度。

3. 有限元求解器

本节介绍我们用于求解两相流的数值方法。我们选择采用有限元方法，具体而言，采用

Galerkin方法和稳定化P1单元来求解水平集方程 3 和 Navier–Stokes 方程 1。随后，我们将在第 3.3 节介绍变形网格以追随界面的算法。最后，我们将介绍这些步骤之间的顺序耦合。

3.1. Navier–Stokes 求解器

我们的方法针对的不同流动可能由对流主导，因此必须对连续Galerkin有限元法进行稳定化，以避免伪振荡。为此，人们已开发出多种稳定化方法。本文采用流线迎风/Petrov-Galerkin（SUPG）方法 [16]。为尽量减小待求解线性系统的规模，我们将未知量设置在网格的节点上。然而，这不满足Babuška-Brezzi条件 [17]，会导致压力场中出现高频成分。我们采用常用的压力稳定/Petrov-Galerkin（PSPG）稳定化 [18]来解决这一问题。

在我们的 X-Mesh 方法中，使用的是连通性固定的网格：只有顶点位置随时间变化。因此，网格顶点的坐标是一个变量，记为$\mathbf{x}(t)$。通过变形网格使其与界面吻合，网格既不是固定的，也不是以拉格朗日方式运动的。我们采用任意拉格朗日-欧拉（ALE）方法，在 Navier–Stokes 方程中考虑网格速度。如方程（4）所示，我们采用非守恒ALE形式，在相同的网格构型上进行积分，并从对流速度中减去网格速度$\mathbf{u}_{mesh}$。时间采用恒定时间步长 Δt进行离散。对于依赖于时间的变量——例如网格位置$\mathbf{x}(t)$——其离散时间点记为$\mathbf{x}_n = \mathbf{x}(n\Delta t)$。因此，网格速度为$\mathbf{u}_{mesh} = (\mathbf{x}_n - \mathbf{x}_{n-1})/\Delta t$。时间积分采用隐式欧拉格式，并通过有限元方法得到离散形式。

设$\mathbf{S_u}$和$\mathbf{S}^p$分别为$\mathbf{u}_{n+1}$和p_{n+1}的解空间，其检验函数为（$\mathbf{v}_{n+1}$,q_{n+1}）$\in \mathbf{V_u} \times \mathbf{V}^p$。

式（1）的全离散形式是求解 $(\mathbf{u}_{n+1}, p_{n+1}) \in \mathbf{S_u} \times \mathbf{S}^p$，使得对于

任意（$\mathbf{v}_{n+1}$,q_{n+1}）$\in \mathbf{V_u} \times \mathbf{V}_p$:

$$\int_{\Omega_n} \rho \left((\mathbf{u}_{n+1} - \mathbf{u}_n) \frac{1}{\Delta t} + \mathbf{u}_{n+1} \cdot \nabla \mathbf{u}_{n+1} - \mathbf{u}_{mesh} \cdot \nabla \mathbf{u}_n \right) \cdot \mathbf{v}_{n+1} d\Omega$$
$$+ \int_{\Omega_n} \mu \nabla \mathbf{u}_{n+1} : \nabla \mathbf{v}_{n+1} d\Omega - \int_{\Omega_n} \nabla p_{n+1} \cdot \mathbf{v}_{n+1} d\Omega = \int_{\Omega_n} \rho \, \mathbf{g}_{n+1} \cdot \mathbf{v}_{n+1} d\Omega + \text{SUPG} \quad (4)$$

$$\int_{\Omega_n} \nabla \cdot \mathbf{u}_{n+1} \, q_{n+1} d\Omega + \text{PSPG} = 0$$

其中，SUPG 项表示 SUPG 稳定化项，PSPG 项表示 PSPG 稳定化项，它们可以表示为方程残差 $\mathcal{R}$ 的函数:

$$\text{SUPG} = \int_{\Omega_n} \tau_{\text{SUPG}} \mathcal{R}_{n+1} \left(\mathbf{u}_{n+1} - \mathbf{u}_{mesh} \right) \cdot \nabla \mathbf{v}_{n+1} d\Omega$$
$$\text{PSPG} = \int_{\Omega_n} \tau_{\text{PSPG}} \nabla q_{n+1} \mathcal{R}_{n+1} d\Omega$$
$$\mathcal{R}_{n+1} = \left(\rho \, (\mathbf{u}_{n+1} - \mathbf{u}_n) \frac{1}{\Delta t} + \rho \, ((\mathbf{u}_{n+1} - \mathbf{u}_{mesh}) \cdot \nabla \mathbf{u}_{n+1}) - \nabla p_{n+1} - \rho \, \mathbf{g}_{n+1} \right)$$

其中 τ_{SUPG} 和 τ_{PSPG} 分别为 SUPG 和 PSPG稳定化的系数。

经典的有限元空间离散会为 $\mathbf{u}_{n+1}$ 和 p_{n+1} 生成一个非线性方程组，可使用牛顿格式求解。对于 Navier–Stokes 求解器，我们使用了开源软件 Migflow [19]。

3.2. 水平集求解器

两种流体之间界面的位置由水平集函数的等值线$\phi = 0$ 描述，其随时间的演化由对流方程 3 决定。对于 Navier–Stokes 方程的求解，我们通过 SUPG 项 [16] 对流进行稳定化。方程在固定网格上求解，因此省略计算域 Ω 的指标记法，并采用 Crank–Nicolson 方法进行时间积分。令$\mathbf{S}_\phi$ 为ϕ_{n+1} 的解空间，$w_{n+1} \in \mathbf{V}_\phi$ 为与 ϕ_{n+1}相关联的检验函数。离散形式可写为:

求 $\phi_{n+1} \in \mathbf{S}^{\phi}$，使得对于任意 $\psi \in \mathbf{V}^{\phi}$:

$$\int_{\Omega} (\phi_{n+1}\psi - \phi_n \psi) \frac{1}{\Delta t} d\Omega + \frac{1}{2} \int_{\Omega} (\mathbf{u} \cdot \nabla \phi_n \psi + \phi_n \nabla \cdot \mathbf{u} \psi) \, d\Omega$$
$$+ \frac{1}{2} \int_{\Omega} (\mathbf{u} \cdot \nabla \phi_{n+1} \psi + \phi_{n+1} \nabla \cdot \mathbf{u} \psi) \, d\Omega + \mathrm{SUPG} = 0$$

SUPG 稳定化项取决于对流方程的残差，其表达式为:

$$\mathrm{SUPG} = \int_{\Omega} \tau_{\mathrm{SUPG}} \, \mathbf{u} \cdot \nabla \psi \, \mathcal{R} d\Omega$$
$$\mathcal{R} = \left((\phi_{n+1} - \phi_n) \frac{1}{\Delta t} + \frac{1}{2} (\mathbf{u} \cdot \nabla \phi_n + \phi_n \nabla \cdot \mathbf{u}) + \frac{1}{2} (\mathbf{u} \cdot \nabla \phi_{n+1} + \phi_{n+1} \nabla \cdot \mathbf{u}) \right) \psi$$

随后，有限元方法使我们能够对区域进行空间离散化，并得到一个线性方程组。由于在求解对流方程时没有考虑网格位移，因此按照下一节介绍的变形算法移动网格后，节点处的水平集值并不正确。为计算水平集在节点处的新值，我们使用快速行进算法 [20]，计算到前沿的近似有符号距离。由于界面完全由网格的边嵌入，因此算法中水平集值从中向网格传播的种子，就是图4(e)所示的前沿节点。

3.3. 极端网格变形——前沿传递

X-Mesh 方法背后的理念是 [14] 通过节点连续移动来变形网格，使其始终与关注的界面相匹配，即使流体区域发生拓扑变化时也是如此。不过，网格会保持固定拓扑。为实现这一目标，该方法允许单元退化，也就是说，三角形可以变形为一条边，甚至一个点。这样网格就能随时间连续变形，并确保前沿接力。界面会从一个节点传递到另一个位于相同位置的节点，从而像接力赛中的接力棒一样向前传播。

三角形退化为一条边有两种方式。第一种情况下，三角形的三条边之一收缩为一个点，因此与该边相对的角退化为 0 弧度。这样的单元称为针形单元。第二种情况下，三角形的一个顶点趋向于其对边上的一个点，因此与该节点相关的角趋向于 π 弧度。这样的单元称为帽形单元。

使用退化或准退化单元会带来两个具体问题：条件性和稳定性。

3.3.1. 条件性

使用退化或准退化单元时出现的第一个问题是有限元矩阵条件性不佳。设有一个三角形，其3个顶点为i、j 和k，内角为 θ_i、 θ_j 和 θ_k （见图2）。

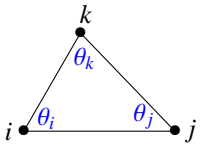


图2：一个三角形及其三个内角 θ_i、θ_j 和 θ_k。

与 $-\nabla^2$ 算子的离散化相对应的局部刚度矩阵 $[K]$可写成其余切形式：

$$[K] = \begin{bmatrix} \cot(\theta_j) + \cot(\theta_k) & -\cot(\theta_k) & -\cot(\theta_j) \\ -\cot(\theta_k) & \cot(\theta_i) + \cot(\theta_k) & -\cot(\theta_i) \\ -\cot(\theta_k) & -\cot(\theta_i) & \cot(\theta_i) + \cot(\theta_j) \end{bmatrix}$$

无论是针形单元还是帽形单元，退化三角形的角$\theta_{i,j,k}$ 中至少有一个将趋于

趋于零，这意味着该角的余切将趋于无穷大。更确切地说，刚度矩阵 $[K]$ 必然包含一个零特征值 $\lambda_1 = 0$，它对应于刚体模态或常数模态$v_1 = (1, 1, 1)$。当该单元退化时，另一个特征值 λ_2 也趋于零，而第三个特征值 λ_3 则趋于无穷大。

设想一种针形单元：顶点 i 和顶点 k 最终重合。对应于零特征值 λ_2 的模态是 $v_2 = (1,0,-1)$。模态 v_2 是除 v_1 （刚体模态）之外唯一能够存在于有限元解中的模态，因为任何涉及 v_3 的其他组合都会导致无限大的能量。因此，在

零测度极限下，有限元解中未知量在节点 i和 k 处的节点值相同。

现在来构造一个帽形单元：顶点k 移动到位置

$$\mathbf{x}_k = \mathbf{x}_i\alpha + \mathbf{x}_k(1-\alpha) \ , \ \alpha \in]0,1[.$$

对应于零特征值的模态 λ_2 是$v_2 = (\alpha, 1-\alpha, -1)$，这意味着在零测度极限下，有限元解沿直线$ikj$呈线性。

为什么要区分针形单元和帽形单元？在这两种情况下，全局刚度矩阵都会出现病态性，但线性求解器——即使是迭代求解器——在特征值非常大的情况下也能表现良好。任何预条件器都能解决这个问题。本文采用与 [14]相同的方法。我们只需通过限制退化三角形的最小面积，来限制 λ_3 的最大值。在后续工作中，我们将为 X-Mesh 提出稳健的预条件器。针形单元与帽形单元的区别在于稳定性，下面将对此进行详细说明。针形单元在两个自由度之间施加局部约束，而帽形单元则将三角形的三个自由度耦合起来。

3.3.2. 稳定性

如果失去稳定性，有限元方法的收敛性也会随之丧失。从历史上看，有限元领域从 Babuˇska 和 Aziz 的开创性论文 [21] 中得知，要确保有限元收敛，只需生成三角形不含过多钝角的网格。因此，稳定性问题本质上与网格中帽形单元的存在有关。许多人认为，这一角度条件是有限元收敛的必要条件，但事实并非如此：它只是充分条件——Babuˇska 和 Aziz 从未说过相反的话。角度条件可以大幅放宽 [22]。网格中孤立的帽形单元不会引发稳定性问题。只有当帽形单元连成较长的带状区域时，才会出现稳定性问题。如果使用常规 P1 三角形，很容易证明，由 n 个帽形单元组成、每个单元长度为 h （$h = \|\mathbf{x}_j - \mathbf{x}_i\|$，见图2）的带状区域会阻碍非线性

解沿整个带长度nh 方向的变化（锁定），从而使收敛阶从h^2 降至$(nh)^2$。另一方面，正如我们上文所示，针形单元会对解施加局部约束，因此这些约束不会传播到很远的地方。因此，我们可以使用任意多个针形单元，而不必担心稳定性。

从一开始，我们的网格变形/传递算法就考虑到了不生成封盖带这一约束。假设有一个三角形网格，能够精确表示某一时间t 的前沿（图3左上图）。前沿以恒定速度v 向右移动，持续时间为 Δt ，因此我们可以将前沿顶点向右移动$v\Delta t$ （图3顶部中央图），然后再移动$v\Delta t$ （图3右上图）。此时可以看到，传递基本上会发生在形状呈帽形的三角形上，而且帽状单元排列成带状区域。显然，这种算法会导致稳定性问题，应当避免。一般来说，如果不够谨慎，网格变形算法自然会倾向于生成这些封盖带。

如果网格顶点沿当前网格的边移动，会怎么样（图3底部各图）？这样会生成大量针状单元。也可能生成孤立帽状单元，但其数量只取决于前沿上游和下游顶点数量之差。该算法的另一个要点是，只将顶点向前沿上游移动。这样可以让传递过程平稳进行，避免下游顶点朝着与前沿移动方向相反的方向移动，从而造成某种交通堵塞 [14]。最后，当某个前沿顶点完成向另一个顶点的传递后，它会逐渐返回初始位置，从而使前沿上游保持高质量网格。

———————

3.3.3. 前沿传递

图4展示了用于在两个时间步之间变形网格的算法，使网格始终贴合新界面。该界面由 $\phi = 0$ 定义， ϕ 的符号决定流体相态。在图4的图像（a）和（b）中，两个连续时间步t 和$t + \Delta t$ 对应的流体相以蓝色和灰色表示。X-Mesh 的基本思想是，两个时刻t 和$t + \Delta t$ 的网格都必须贴合

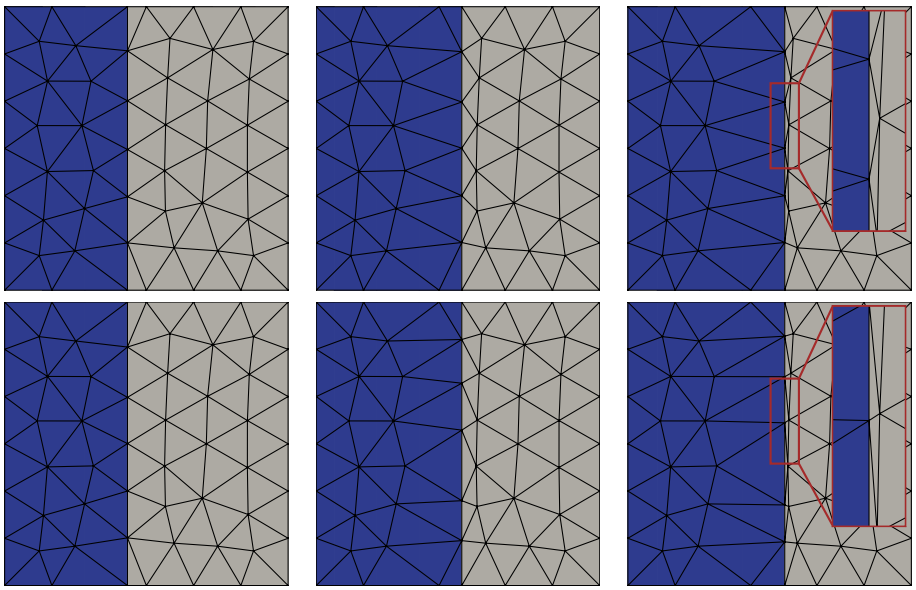
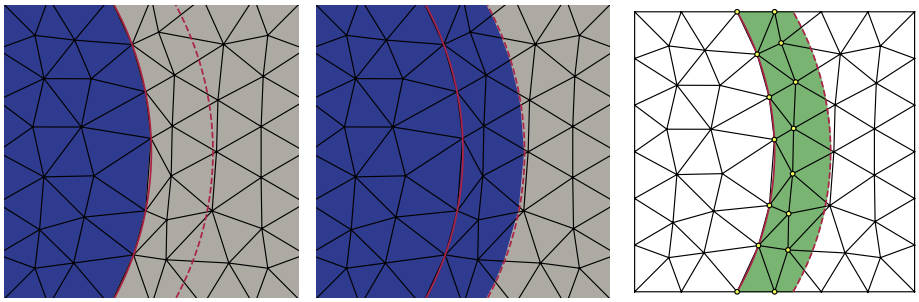


图3：沿速度方向（上）或沿边（下）移动节点。

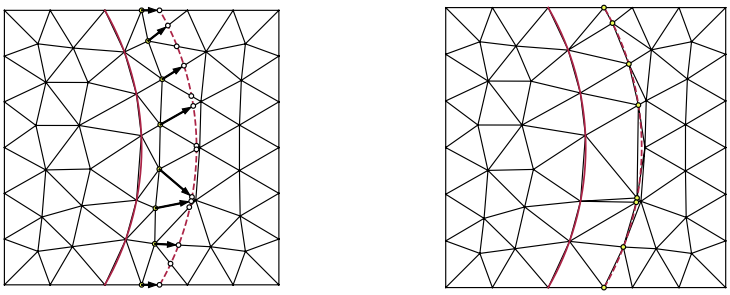
界面。图像（b）显示了时间$t + \Delta t$ 时、位于时间t网格上的各相。时间$t + \Delta t$ 时的新界面（以虚线表示）与网格不贴合。

前沿传递算法的目标是变形时间t时已有的网格，使网格与新界面相合：我们希望时间$t+\Delta t$ 时的界面完全由网格的边表示。此外，出于稳定性考虑，算法必须尽可能限制封盖带的产生。如上所述，沿网格的边移动顶点可以避免产生封盖带。我们还希望避免顶点在前沿下游聚集。因此，只有位于前沿上游的节点才应向前沿移动。这一出发点使我们能够构建稳健的前沿传递算法。图4的（c）中绿色区域表示t 与$t+\Delta t$之间相位发生变化的区域。因此，该区域位于前沿上游，应从此区域中选取需要移动的顶点。我们将水平集 ϕ在此时间步内符号发生变化的顶点称为活跃顶点。活跃顶点包括前一时刻前沿上的顶点，这些顶点的水平集值为 0，在图4的（c）中以黄色标示。

如果在时间步t，一条边（ij）的两个节点符号不同，则该边为切割边



(a) 时间t时的相　(b) 时间 $t + \Delta t$时的相　(c) 活跃顶点



(d) 潜在目标和移动中的顶点　(e) 新网格和新前沿顶点

图4：网格变形算法。

对于时间t ϕ 的水平集 $+ \Delta t$。因此，在切割边（ij）上存在一个潜在目标点，水平集在此处为零，即新前沿将在$t + \Delta t$处位于此处。属于切割边的所有活跃顶点都将移动：我们称其为移动顶点（见图4中的图d）。现在，一个移动顶点可能属于多条切割边，因此我们必须从潜在目标中选择一个。位于域边界上的移动顶点（图4d中最高和最低的顶点）必须移动到位于边界上的潜在目标，以保持计算域的形状。请注意，本文仅考虑多边形计算域，非平凡几何形状需要特殊处理。对于区域内部的移动顶点，我们通过实验观察到，当选择距离节点最近的潜在目标时，网格质量更好（图4d中的箭头）。图4中的图e显示了时间$t + \Delta t$ 处与新界面相一致的网格。

该网格变形方法如算法 1 所示。在此算法中，水平集-

水平集和旧水平集分别表示水平集的新值和旧值。名为前沿的表用于为每个节点存储一个布尔值，以指示该节点是否位于先前的界面上。第二个表边界标记用于指明节点是否位于域边界上，若是，则需要额外处理。

如果该节点不在边界上，则调用函数 移动内部节点 ，通过函数 获取邻居节点 找出节点 i 的所有邻居。然后，我们可以使用函数确定最近目标点*(i,* 邻居*)*计算最近目标的位置及其对应的邻居。这些目标对应于节点 i 与其邻居节点之间网格边上水平集的 0 值位置。

对于边界节点，我们调用 移动边界节点 ，并且必须区分边界上的邻居与体域中的邻居。如果可能，我们将节点 i 放置在边界上的一个目标位置，以保留区域边界。如果无法做到这一点，我们会查看体域内部的邻居，并计算最近目标点。随后，将相应的邻居放置在该目标位置。

算法 1： 移动前沿算法

/* 函数 determine closest target(i,neighbours) 返回节点 i 与其邻居之间边上的最近水平集 0 值目标的位置。 */ 1 函数 移动前沿： 2 对于位于 x_i 的每个节点 i， 执行 3 如果 front[i] 为 True，且 sign(旧水平集[i]) ≠sign(水平集 [i])，则 4 active[i] ← True 5 对于位于 x_i 的每个节点 i， 执行 6 如果 active[i] 为 True，则 7 如果 边界标记[i] 为 True，则 8 移动边界节点 (i) 9 否则 10 移动内部节点 (i)

11 **函数** 移动内部节点 (i)： 12 邻居 ← 获取邻居节点(i)13 $\boldsymbol{x}_{\text{closest}}$，邻居 ←确定最近目标点(i,neighbours)14 **如果** $\boldsymbol{x}_{\text{closest}}$ **不为** Null **则**15 $\boldsymbol{x}^{\text{new}}_{\text{i}}$ ←$\boldsymbol{x}_{\text{best}}$16 前沿^{new}[i] ← True

17 **函数** 移动边界节点：

18 边界邻居 ←获取边界邻居(i)

19 体域邻居 ← 获取体域邻居(i)

20 $\boldsymbol{x}_{\text{closest}}$，邻居 ←确定最近目标点(i,边界邻居)

21 **如果** $\boldsymbol{x}_{\text{closest}}$ **不为**空值 **则**

22 $\boldsymbol{x}_{\text{i}} \leftarrow \boldsymbol{x}_{\text{closest}}$

23 前沿^{new}[i] ← True

24 **else**

25 $\boldsymbol{x}_{\text{closest}}$，邻居 ← 确定最近目标点(i,体域 邻居)

26 **如果** $\boldsymbol{x}_{\text{closest}}$ **不为** Null，**则**

27 $\boldsymbol{x}_{\text{neighbour}} \leftarrow \boldsymbol{x}_{\text{closest}}$

28 前沿^{new}[邻居] ←True

本文的补充材料中提供了实现该算法的 Python 代码。一个小型测试用例展示了一个圆形边界沿螺旋线路径移动并返回的过程，可用于观察该算法在区域内部的经典情形下的表现，以及其到达网格边界时的表现。

前沿传递算法的最后一个要素，是逐步将已经离开前沿的顶点移回其原始位置。这些节点只需按照下方的方程移动：

$$\mathbf{x}_{n+1} = (1 - \alpha)\mathbf{x}_n + \alpha\mathbf{x}_0$$

其中，$\mathbf{x}_0$ 表示节点在初始网格中的位置（即任何网格变形发生之前的位置），α 是一个介于 0 和 1 之间的用户参数，用于调整节点返回初始位置的速度。当前沿通过后，此功能可以恢复前沿上游的网格，使其保持良好状态。

随后可以利用水平集函数确定每个单元所属的相。由于网格与边界共形，水平集的零等值由前沿节点之间的边表示。因此，网格变形后，没有网格边会切过前沿。这样，所有单元顶点上的水平集值都不会出现异号。由此，可以根据单元各顶点的水平集值确定其所属的相。如果三角形的 3 个顶点都位于前沿上（因此水平集值均为 0），则无法确定该单元所属的相。我们称这类单元为三零单元。当相对于单元特征长度而言，前沿曲率较大，或发生拓扑变化时，就会出现这种情况。处理这类单元的一种简单方法是查看它们在前一个时间步所属的相。如果该单元此前已经是三零单元，则保持其所属的相不变，以免这些单元在不同时间步之间来回振荡。如果该单元此前不是三零单元，则改变其所属的相。这样就可以实现相位合并。

3.4. 耦合

本节介绍用于耦合前文所述不同部分的算法：Navier–Stokes 方程的求解、平流方程的求解、采用为 X-Mesh 提出的算法进行网格位移，以及水平集的重新初始化。本文的所有结果均采用算法 1 所示的顺序算法获得。一个时间步的模拟由依次求解这些不同步骤组成。首先求解 Navier–Stokes 方程，以获得新的速度场。利用该速度场对水平集进行平流，从而获得由位置 $\mathbf{x}_n$定义的网格上的新水平集值。随后对网格进行变形，使其与水平集定义的新界面相匹配。由于网格发生了移动，节点处的水平集值不再准确。然后，以$n+1$ 中的前沿节点为种子，应用快速行进法 [20]，获得网格新位置$\mathbf{x}_{n+1}$ 处的新水平集。对于具有大量拓扑变化的测试案例，这种耦合方法在实践中表现出很强的鲁棒性。为提高格式的精度，人们可能会尝试在网格变形与 Navier–Stokes 方程求解之间进行迭代。这样便可在t_n 与t_{n+1}之间的方程中，直接考虑t_n 与t_{n+1} 之间的网格变形。然而，这并非易事，简单的不动点法无法快速收敛。网格位置可能会在两种状态之间振荡而无法收敛，尤其是在构型突然变化时，例如发生拓扑变化时。

算法 2：单个时间步的顺序耦合

1 $u_{mesh} = (x_n - x_{n-1}) / \Delta t$ 2 u_{n+1} = *Navier-Stokes* (u_n, ϕ_n, u_{mesh})3 ϕ_{adv} = 水平集 (u_{n+1}, ϕ_n) 4 x_{n+1}, 前沿$_{n+1}$ = *X-Mesh* $(x_n, \phi_{adv}, \phi_n$, 前沿$_n)$5 ϕ_{n+1} = *Fastmarching* $(x_{n+1}$, 前沿$_{n+1})$ 6 $t = t + \Delta t$

4. 表面张力

考虑两相流时，根据两种流体的材料属性以及所研究的问题，表面张力并非总是可以忽略。工业过程中有大量与气泡动力学相关的流动，这种情况尤为常见。要正确模拟这些流动，准确表示表面张力至关重要。根据方程 2，我们可以直接将表面张力与界面处的动量守恒条件联系起来。事实上，当沿界面法向考察这一条件时，可表示为：

$$\begin{aligned}\mathbf{n}\cdot\left[-pI+\mu\,\frac{1}{2}\left(\nabla\mathbf{u}+\nabla\mathbf{u}^{T}\right)\right]\cdot\mathbf{n}&=\mathbf{n}\cdot\sigma\kappa\mathbf{n}\\ [p]&=-\sigma\kappa+\mathbf{n}\cdot\left[\mu\,\frac{1}{2}\left(\nabla\mathbf{u}+\nabla\mathbf{u}^{T}\right)\right]\cdot\mathbf{n}\\ [p]&=-\sigma\kappa\end{aligned}\tag{5}$$

项$\mathbf{n}\cdot\left[\mu\ \frac{1}{2}\left(\nabla\mathbf{u}+\nabla\mathbf{u}^{T}\right)\right]\cdot\mathbf{n}$恰好为 0，因为两种流体都被视为牛顿流体且不可压缩。因此，可以根据界面的曲率和表面张力系数求得界面处的压降。为了实现表面张力的影响，我们将采用幽灵流体法 [23] [24]，通过在界面处施加该压力跃变。在这种固定网格方法中，对于被界面截穿的单元，压力梯度算子 ∇p 会经过修正，以便在考虑压力跃变的同时计入经典压力。因此，人们可能会认为该方法是锐界面的，因为压力跃变被准确地定位在界面处；然而，界面的厚度相当于一个单元的尺寸。实际上，当界面位移小于 $\pm\frac{h}{2}$ 与h 单元尺寸 [23]时，该方法对界面位移并不敏感。我们的方法与幽灵流体法类似，因为我们同样显式地添加压力跃变来模拟表面张力的影响。不同之处在于，X-Mesh 的网格节点位于界面处，因此我们需要直接在节点处添加压力跃变。方程 4 中p 算子所使用的修正压力 $\bar{p}$ 仅在边界节点处有所不同，其取值为

取值为：

$$\bar{p}_i = \begin{cases} p_i & \text{if } \Omega_e \in \Omega_1 \\ p_i + \sigma\kappa_i & \text{if } \Omega_e \in \Omega_2 \end{cases}$$

其中，p_i 表示节点i处的压力，κ_i 表示节点i 处的界面曲率，Ω_e 表示所考虑的单元。

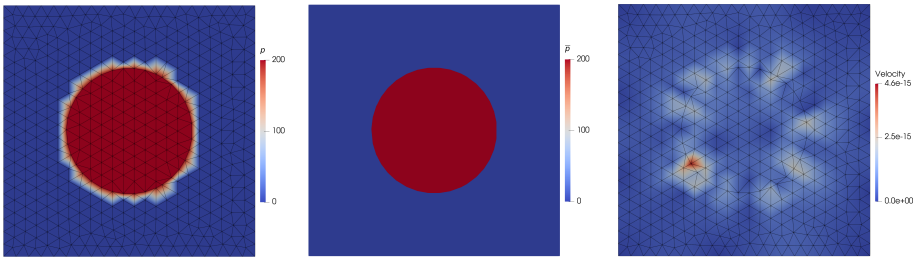
因此，所得的表面张力表示具有精确的锐界面；与固定网格上的幽灵流体法不同，界面厚度为零。界面的任意微小位移都会引起网格位移，从而改变压力梯度算子，进而影响表面张力。

以静态气泡为例，可以在图5中观察到这种压力跃变：经典压力（a）和修正压力（b）在除界面处之外的所有位置都几乎保持恒定。对表面张力进行建模时，一个非常重要的步骤是测量界面曲率。由于我们知道位于界面上的节点位置，因此可以通过计算界面上连续三个节点所构成的外接圆半径来近似曲率。这种非常局部的方法有一个优点：对于圆形而言，它是精确的，而圆形正是静态气泡的平衡状态。图5展示了静态拉普拉斯气泡的压力、修正压力和速度场。在这项验证中，模拟了一个无重力的圆形气泡。气泡内部的预期压力根据拉普拉斯公式解析计算，结果为：

$$p_{bubble} = \frac{\sigma}{R}$$

其中 R 为气泡半径。该表达式假设气泡外部的压力为零。

该气泡处于平衡状态，压力梯度应与表面张力效应相平衡，但这种简单的平衡状态在数值上很难复现。由于数值不准确性或表面张力数值建模不佳，伪流



(a) 压力 p (b) 修正压力 $\bar{p}$ (c) 寄生流

图5：静态气泡。

或寄生流可能会出现。图5 c显示了采用本文所述方法针对静态气泡测试案例获得的速度范数 $|\mathbf{v}|$的色彩图。这些伪流在整个界面周围振荡，但这些流的范数仅为 10^{-15} 的量级，即机器精度的量级。图5的前两幅图还分别表示了连续压力 p和修正后的（不连续）压力 $\bar{p}$ 。在此情况下，所考虑气泡的半径为 $R = 0.5\,[m]$ ，所考虑的表面张力为 $\sigma = 100\,[N/m]$。气泡内部获得的压力为 $p = 200\,[Pa]$ ，符合拉普拉斯公式的预期。观测压力和速度场的时间 t_{obs} 是该问题特征时间的250倍$t_{char} = D\mu\sigma$ 。

—

5. 结果

在本节中，我们将该方法应用于晃荡验证案例以及多个测试用例，以检验其性能，具体包括：黏性溃坝、瑞利-泰勒不稳定性、单气泡上升和双气泡合并。

5.1. 求解器验证——晃荡

晃荡问题是计算液体在容器中自由振荡的问题。这个问题基本上可以用来验证我们的求解器是否求解了正确的方程，因为在平面界面发生小扰动的情况下，这是少数存在解析解的情形之一 [25]。在解析设定中，界面

足够简单，可以用高度函数 $\eta(x,t)$ 表示，该函数的初始条件为

$$\eta(x,0) = d + \eta_0 \sin k\,(0.5 - x)$$

其中，$d = 1.0$ 为平均高度，$\eta_0 = 0.01$ 为初始振荡振幅，k 为波数。

自由表面最大高度的解析演化 η 取决于$\nu = \mu\rho$ 运动黏度以及无量纲参数 $\kappa = g\nu^2k^3$：

$$\frac{\eta(t)}{\eta_0} = 1 - \kappa e^{-\nu k^2 t} f(\nu k^2 t)$$

其中，$f(t)$ 是该函数的拉普拉斯逆变换

$$F(s) = \frac{1}{(s-1)\left((s+1)^2 - 4s^{\frac{1}{2}} + \kappa\right)}.$$

所考虑的计算域为一个尺寸为 $[0,d] \times [0, 1.3d]$ 的矩形，四面墙均采用自由滑移条件。

图6显示了扰动 η 随时间的演变。可以看出，模拟结果与解析解吻合良好。由于该解析解是根据线性化纳维-斯托克斯方程得到的，初始扰动 $\eta_0 = 10^{-2}$ 相对于$d = 0.5$ 不能太大。

要使线性化纳维-斯托克斯方程的参考解有效，晃荡振幅必须足够小。然而，此次模拟所用的网格相当粗糙，只有 3900 个顶点。模拟开始时，单元的特征尺寸小于晃荡振幅，因此前沿会在不同顶点之间接力传递。但很快，晃荡振幅便会减小，模拟也就接近于不进行界面接力传递的经典任意拉格朗日-欧拉模拟

5.2. 溃坝

溃坝测试案例描述了容器中液柱的坍塌过程。这对应于坝体突然溃决以及蓄水流出的情形。

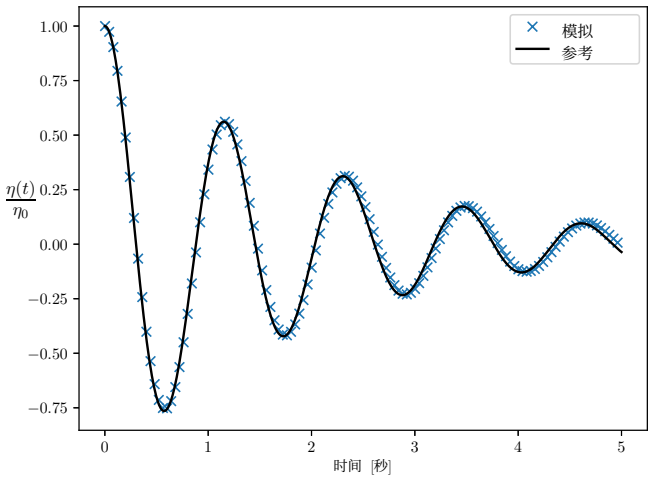


图6：晃荡问题中的波面升高演化。

该问题中的界面演变过程复杂，涉及多次拓扑变化。由于其复杂性，该案例已成为验证两相流模拟的参考测试案例。所考虑液柱的高度为 $H = 0.4\,[m]$，宽度为 $L = 0.4\,[m]$。液柱在尺寸为 $1.4 \times 1.1\,[m^2]$ 的计算域中坍塌。4 面壁面采用自由滑移边界条件。该测试案例所用的两种流体可用其密度 ρ 和运动黏度 ν来描述：

$$\rho_0 = 10 \qquad \rho_1 = 1000 \qquad \nu_0 = \nu_1 = 10^{-3}$$

所用网格包含 75,950 个顶点。图7展示了自由表面在不同无量纲时间$t = 0.5$、1.75、3、5.5、6.5 和 22.5 时的演化，其中所采用的参考时间为$t_{ref} = \sqrt{h/g}$。尽管溃坝过程中会发生许多拓扑变化，该方法仍足够稳健，能够模拟此类复杂流动。

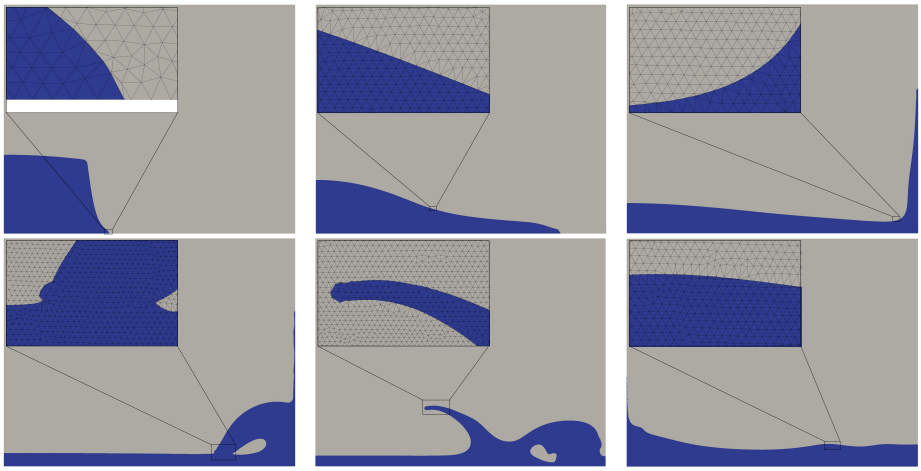


图7：溃坝模拟：无量纲时间 t =0.5、1.75、3、5.5、6.5 和 22.5 时的自由表面位置。

5.3. 瑞利-泰勒

瑞利-泰勒不稳定性是指将较重流体置于较轻流体之上。这种位置构成不稳定平衡，且对任何扰动都很敏感。验证两相流求解器的经典基准测试，是在初始化时触发单模态扰动。这种不稳定性在重力场作用下发展，常见于广泛的物理现象中。在$t = 0$ 时，界面位置处初始化一个扰动，其初始位置为：

$$y = 2.0 + 0.05 \cos 2\pi x$$

初始时，速度场设为 0，压力为静水压。两种流体随时间的演化由两个无量纲数决定，即阿特伍德数 (At) 和雷诺数 (Re)，在本例中定义如下：

$$At = \frac{\rho_1 - \rho_2}{\rho_1 + \rho_2}$$
$$Re = \frac{\sqrt{Wg}W}{\nu}$$

其中 W 为运河的宽度。

图8展示了 $\mathrm{At} = 0.5$ 和 $\mathrm{Re} = 256$ 情况下两相的演化过程。我们在三种不同的网格上进行了模拟：网格 a 约有 123 000 个节点，网格 b 有 31 000 个节点，第三种非常粗的网格 c 有 8250 个节点。我们将结果与 He等人 [21] 采用格子玻尔兹曼方法、在约 262,000 个单元的网格上得到的结果进行了比较。图1中的白色虚线对应于他们获得的界面。所采用方法的尖锐界面以及使用的细网格1，使流动特征能够长时间得到保持。如第 3.3.3 节所述，如果一个单元的 3 个节点位于界面上，则该单元的相位是不明确的。每当一个相合并或分裂时都会发生这种情况，因为水平集能够自然地处理单元尺度上的拓扑变化。因此，合并过程取决于网格，而网格的不对称性会在流动中产生轻微的不对称，并不断传播，最终形成完全不对称的流动，如图9（左）所示。网格特征长度的影响如图8（底部）所示，其中对相同的瑞利-泰勒不稳定性进行了模拟，但使用的是粗 4 倍的网格 b。线性化方程存在一个解析解，并且该解适用于不稳定性发展的线性阶段 [26]:

$$h = h_0 e^{\hat{\alpha} t}$$

其中h 表示扰动尺寸，h_0 表示$t = 0$ 时的扰动，$\hat{\alpha}$ 表示增长率。对于该模拟参数，解析增长率为$\hat{\alpha}_a = 9.3$，观测到的增长率为$\hat{\alpha}_o = 9.2$。

比较不同网格细化程度下的质量守恒情况也很有意义。图9展示了瑞利–泰勒模拟过程中灰色流体相对质量的变化。可以看出，质量变化幅度不超过1%，且较粗网格下的变化更为混乱。图中的点表示流动不再被网格完全捕捉的时刻。可以看到，到达该时刻后，质量开始大幅波动。将网格节点放置在边界上，会自然地截去边界的角部，从而造成质量损失

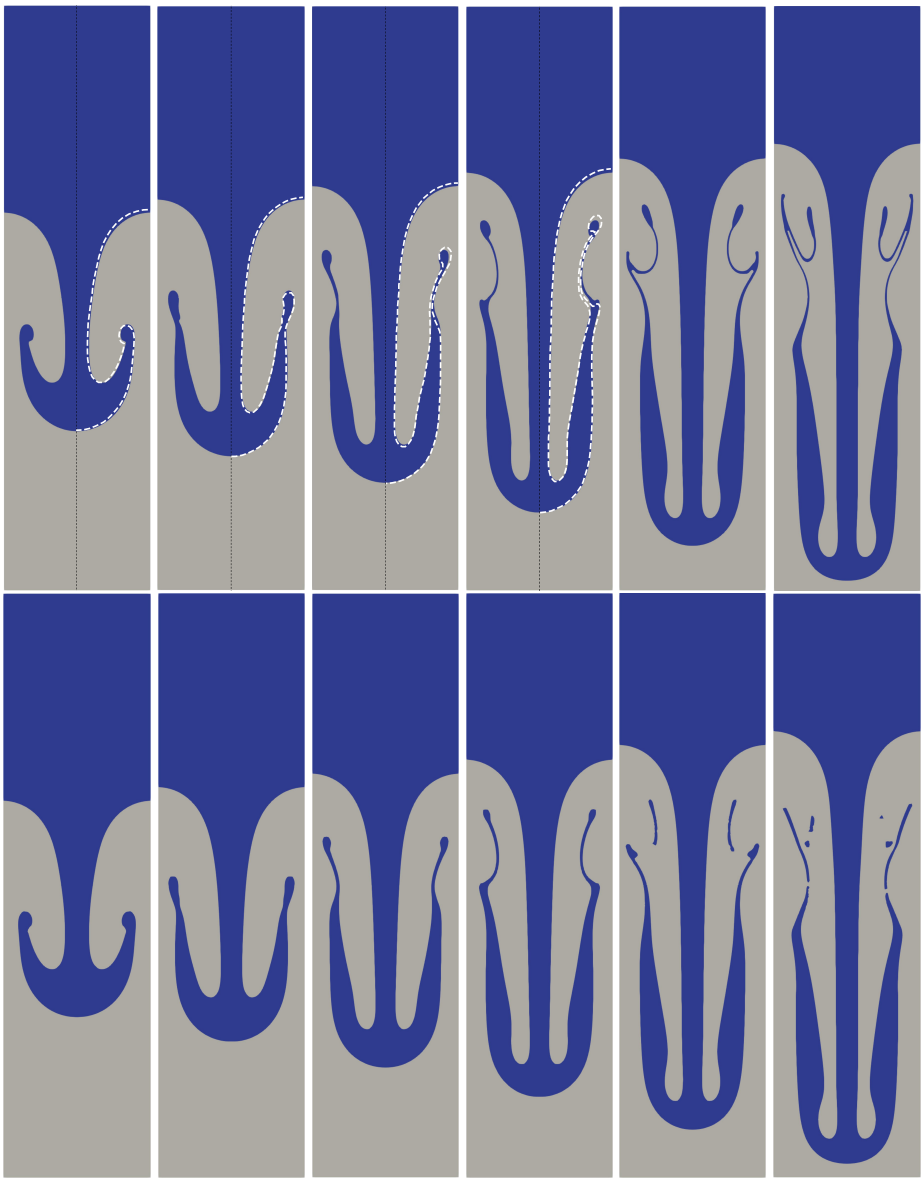


图8：雷诺数 = 256、阿特伍德数 = 0.5 时的瑞利-泰勒不稳定性；在时间 = 3、3.5、4、4.5、5、5.5 时，采用细网格（上）和较粗网格（下）。

对于流体1或流体2，具体取决于边界是凸形还是凹形。瑞利-泰勒不稳定性的界面有些地方凸起，有些地方凹陷。这会导致相反的质量损失效应，从而解释了上升和

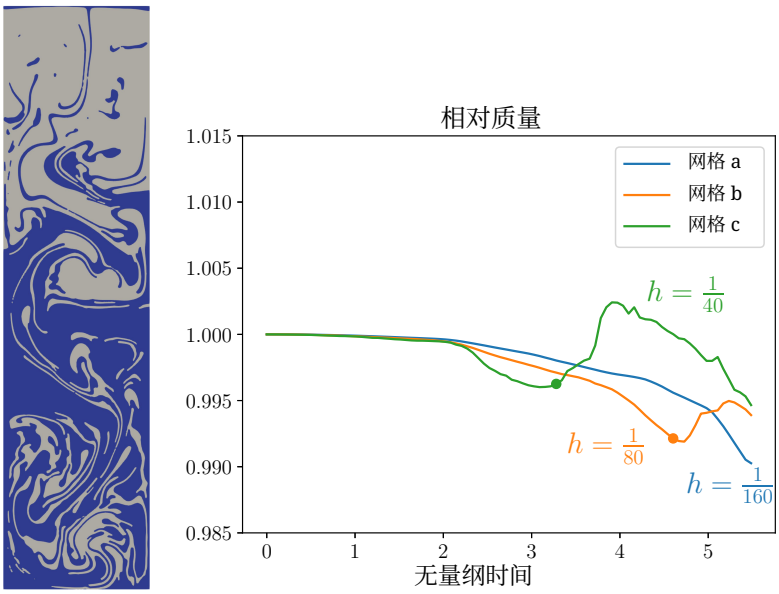


图9：（左）使用网格 a 时，时间为 $t = 15$ 的瑞利-泰勒不稳定性。（右）瑞利–泰勒模拟过程中灰色流体的相对质量，展示了 3 种不同网格的结果

图9中曲线的下降部分。

5.4. 单气泡上升

气泡动力学常见于各种工业过程或自然流动中。对于这类问题，正确表示界面至关重要，因为表面张力与界面的曲率成正比。Grace等人 [27]于1976年描述了不同流体中单气泡上升实验所得到的最终形状。图11总结了他们的观察结果。为了验证我们的方法在气泡动力学中的有效性，我们尝试重现这些结果。图11的纵轴对应雷诺数，横轴对应邦德数（或E¨ötvös数）。气泡的这些数值可按以下方式计算：

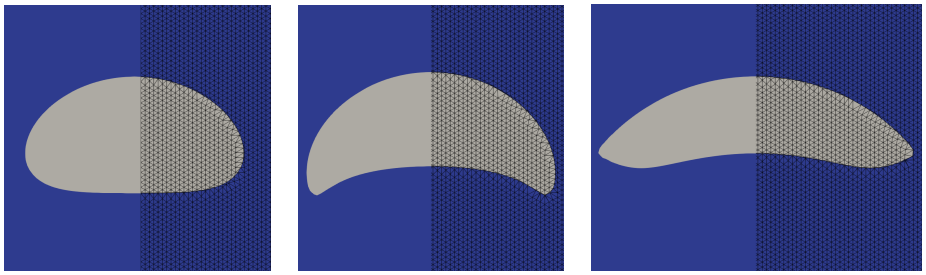
$$Re = \frac{\rho_1 \sqrt{g} D^{3/2}}{\mu_1}$$
$$Bo = \frac{\rho_1 g D^2}{\sigma}$$

其中D 为气泡直径，下标1表示较重流体。

我们考虑三种情况，其雷诺数和邦德数的不同组合如图10和图11中的 Grace 的图示所示。所考虑的密度比和黏度比为 $\frac{\rho_1}{\rho 2} = 1000$ 和 $\frac{1}{\mu_2} = 100_{\mu}$。气泡直径固定为$D = 1\,[m]$。为避免侧壁对气泡动力学产生影响，所考虑的计算区域宽$W = 6\,[m]$、高$H = 12\,[m]$。显式施加表面张力会对时间步产生稳定性条件。时间步必须满足以下条件：

$$\Delta t < \sqrt{\frac{(\rho_1 + \rho_2)\, h^3}{4\pi\sigma}}$$

其中，h 为网格的特征尺寸。



(a)雷诺数 = 10，邦德数 = 10　(b)雷诺数 = 10，邦德数 = 50　(c)雷诺数 = 100，邦德数 = 50

图10：自由上升气泡的最终形态

观察气泡的最终形状，并将其与 Grace 的实验图或 Hua 等人 [28], 的结果进行比较，可以看出模拟呈现出相同的趋势。验证结果的一种有效方法是使用 Hygwing 等人提出的二维基准测试进行比较。这是一个自由上升气泡，其两种流体具有以下性质：

$\rho_1 = 1000$　$\rho_2 = 100$　$\mu_1 = 10$　$\mu_2 = 1$　$g = 0.98$　$\sigma = 24.5$

除了比较气泡的最终形状外，本文还提出比较

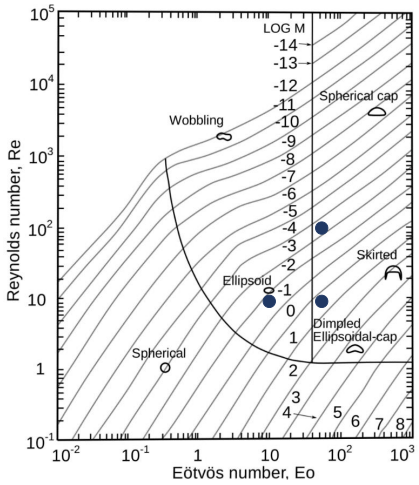


图11：Grace 的图示 [27]。

几项基准测试量：气泡的质心、气泡的平均速度以及圆度。圆度定义为与气泡面积相同的圆的周长与气泡周长之比。我们还将加入气泡的质量（面积）。为观察该方法的收敛性，采用不同程度的网格细化进行模拟：$h = \frac{1}{40}$、$\frac{1}{80}$ 和 $\frac{1}{160}$，其中h 是单元的特征尺寸。此外还显示了参考解，即 TP2D 求解器 [29, 30] 在网格尺寸$h = \frac{1}{320}$下获得的解。TP2D 是二维传输现象（Transport Phenomena in 2D）的简称，由多特蒙德工业大学开发，是 Featflow 不可压缩流求解器的扩展，用于通过水平集方法处理不相溶流体。

如图 12 所示，气泡界面很快收敛，因为 $h = \frac{1}{80}$ 和 $h = \frac{1}{160}$ 对应的界面位置难以区分。可以将气泡的质心垂直位置、速度以及圆度与参考解进行比较。质心位置与参考值吻合良好，即使网格较粗，我们的方法得到的解也接近参考解。以气泡内部的速度积分衡量的气泡速度似乎并未收敛，但所得解已几乎与参考解重合。圆度是一个更复杂的基准

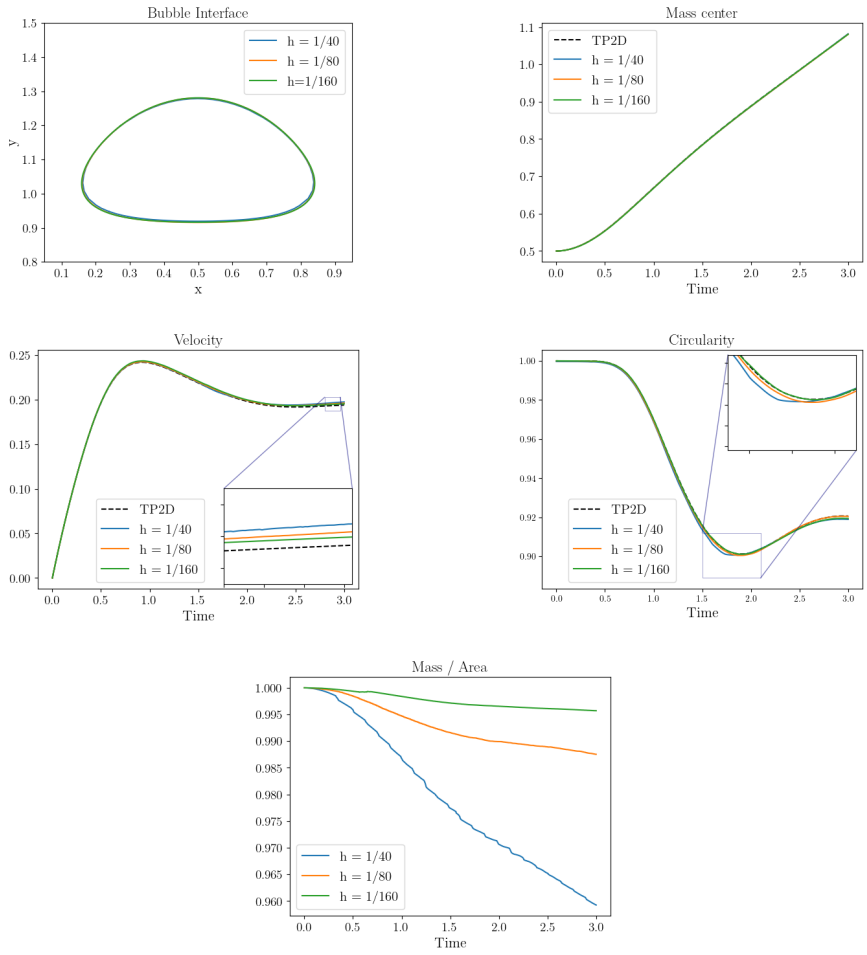


图 12：Hysing 等人的自由上升气泡基准测试结果。

量。我们从图 12 中可以看出，我们的方法收敛到了参考解，但仍存在显著差异。最后，我们可以计算模拟过程中气泡相对质量的变化。可以清楚地看到网格对质量守恒的影响：当特征网格尺寸为 $h = \frac{1}{40}$ 时，变化约为 4%；而当尺寸为 $h = \frac{1}{160}$ 时，变化为 0.25%。

5.5. 气泡合并

为了验证所采用的表面张力处理方法在流体相发生拓扑变化时是否有效，我们关注一个测试案例：两个气泡上下叠置

于具有自由表面的重流体柱体中。计算域为 6 乘 3[米] 的正方形，两个气泡的中心位于水平方向上，其中心分别位于高度 $y = 1$ [米] 和 $y = 2$ [米] 处，半径分别为 0.4 米和 0.5 米。两种流体的参数选取如下：

$$\frac{\rho_1}{\rho_0} = 1000 \qquad \frac{\mu_1}{\mu_0} = 2 \qquad Re = 104 \qquad Bo = 313$$

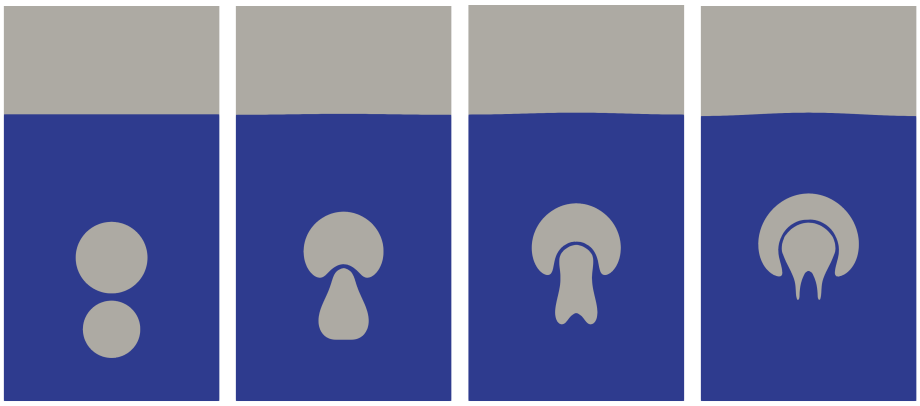
其中，Re和Bo 数均以较大气泡的直径为基准定义。

下方的气泡受到上方气泡的阻力作用而被吸引，因此在$t = 0.4$ [s]时被后者吞并。两个气泡一同上升，直到$t = 0.95$ [s], 时，柱体中心更强的流将它们推向坍塌。这导致拓扑发生多次变化，最终形成两个上升至自由表面的气泡，以及两个被两侧向下流动携带至底部的细长气泡。

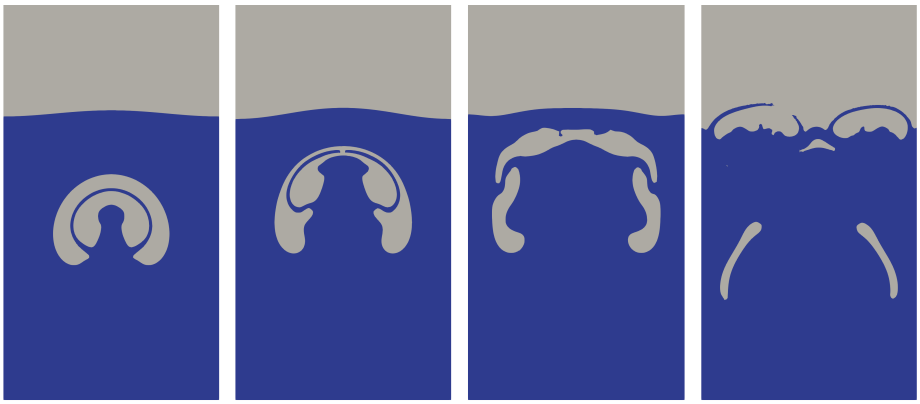
6. 结论

本文介绍了一种利用 X-Mesh 模拟两相流的方法：以清晰且低成本的方式对界面进行建模。直接在网格中表示界面，使其保持清晰。与 ALE 方法类似，该方法能够表示界面处速度导数的不连续性。通过水平集函数更新界面位置，可以轻松考虑流体相拓扑的变化。前沿接力通过仅对界面附近的网格进行局部变形，实现了对界面大幅位移的良好表示。本文提出的变形算法能够保留原始网格，从而保持其拓扑结构。

得益于将节点布置在界面上，并像幽灵流体法那样施加压力跃变，表面张力模型实现了精确的界面捕捉。通过利用连续三个节点的外接圆计算界面曲率，寄生流已降低至机器精度量级。最后，顺序算法能够将各个步骤耦合起来。



(a)$t = 0$ (b)$t = 0.2$ (c)$t = 0.3$ (d)$t = 0.4$



(e)$t = 0.6$ (f)$t = 0.95$ (g)$t = 2.2$ (h)$t = 3.85$

即使流体相拓扑发生了众多变化，也能以稳健的方式进行模拟。

然而，模拟中观察到了质量变化。尽管水平集对流格式能够精确守恒水平集场的全局积分，但该积分并不具有任何物理意义。即使水平集积分能够得到精确守恒，我们仍可能观察到较大的质量变化。这是水平集方法固有的问题，因此我们希望在未来的工作中通过更改界面表示来加以改进。还需指出，PSPG稳定化会在质量守恒方程中引入误差，导致所得速度场并非严格无散。

致谢

本项目获得了欧盟 Horizon研究与创新计划下欧洲研究理事会（ERC）的资助（资助协议编号：101071255）。

作者感谢 Michel Henry 维护 MigFlow 中开发的功能，并感谢 Nicolas Chevaugeon 协助实现快速行进算法。

参考文献

[1] S. C. Brenner、L. R. Scott、L. R. Scott，《有限元方法的数学理论》，第 3 卷，施普林格，2008。[2] C. Geuzaine 、 J.-F. Remacle，Gmsh：一款具有内置前处理和后处理功能的三维有限元网格生成器，《国际工程数值方法期刊》79（2009）1309–1331。[3] A. Prosperetti、G. Tryggvason，《多相流的计算方法》，剑桥大学出版社，2009。[4] T. Frachon、S. Zahedi，一种用于不可压缩两相纳维–斯托克斯流的切割有限元法，《计算物理学杂志》384（2019）77–98。[5] S. Nagrath、K. E. Jansen、R. T. Lahey，采用稳定化有限元水平集方法计算不可压缩气泡动力学，《应用力学与工程中的计算方法》194（2005）4565–4587。[6] T. J. Hughes、W. K. Liu、T. K. Zimmermann，用于不可压缩黏性流的拉格朗日–欧拉有限元公式，《应用力学与工程中的计算方法》29（1981年）329–349。

[7] M. Lesoinne, C. Farhat，运动边界和可变形网格流动问题的几何守恒律及其对气动弹性计算的影响，《应用力学与工程中的计算方法》134 (1996) 71–90。

[8] F. Alauzet, P. J. Frey, P.-L. George, B. Mohammadi，时变问题的三维瞬态不动点网格自适应：在计算流体力学模拟中的应用，《计算物理学杂志》222 (2007) 592–623。[9] F. Alauzet, P. L. George, B. Mohammadi, P. Frey, H. Borouchaki，基于瞬态不动点的非结构网格自适应，《国际流体数值方法杂志》43 (2003) 729–745。[10] J. E. Pilliod, E. G. Puckett，用于追踪物质界面的二阶精度流体体积算法，《计算物理学杂志》199 (2004) 465–502。[11] C. Hirt, B. Nichols，用于自由边界动力学的流体体积（VOF）方法，《计算物理学杂志》39 (1981) 201–225。[12] E. Marchandise, P. Geuzaine, N. Chevaugeon, J.-F. Remacle，采用不连续水平集方法计算气泡动力学的稳定化有限元方法，《计算物理学杂志》225 (2007) 949–974。[13] M. Azaiez, F. Jelassi, M. Mint Brahim, J. Shen，采用平滑焓的两相斯特藩问题，《数学科学通讯》14 (2016) 1625–1641。[14] N. Moes, J. Remacle, J. Lambrechts, B. Le，用于斯特藩相变模型的极端网格变形方法（X-MESH），CoRR abs/2111.04179 (2021)。[15] J. A. Sethian, P. Smereka 等，流体界面的水平集方法，《流体力学年度评论》35 (2003) 341–372。[16] A. N. Brooks, T. J. Hughes，针对对流占优流动的流线迎风/Petrov-Galerkin公式，特别侧重于不可压缩纳维-斯托克斯方程，《应用力学与工程中的计算方法》32 (1982) 199–259。

[17] F. Brezzi，关于由拉格朗日乘子产生的鞍点问题的存在性、唯一性与近似，雷恩数学与信息学出版物 (1974) 1–26。

[18] T. J. Hughes, L. P. Franca, M. Balestra，《用于计算流体力学的新型有限元公式：V. 绕过 Babuška-Brezzi 条件：适用于等阶插值的稳定 Stokes问题 Petrov-Galerkin 公式》，《应用力学与工程中的计算方法》 59 （ 1986 ） 85–99。[19] M. Constant, F. Dubois, J. Lambrechts, V. Legat，《用于求解浸没颗粒流的未解析稳定化 FEM–DEM 模型的实现》，《计算颗粒力学》6（2019）213–226。[20] J. A. Sethian，《一种用于单调推进前沿的快速行进水平集方法》，《美国国家科学院院刊》93（1996）1591–1595。[21] I. Babuška, A. K. Aziz，《有限元方法中的角度条件》，《SIAM数值分析杂志》13（1976）214–226。[22] V. Kučera，《有限元收敛的充要条件》，arXiv 预印本 arXiv:1601.02942（2016）。[23] S. Popinet，《表面张力的数值模型》，《流体力学年度评论》50（2018）49–75。[24] M. Kang, R. Fedkiw, X. Liu，《一种用于多相不可压缩流动的边界条件捕捉方法》，《科学计算杂志》15（2000）323–360。[25] G. Wu, R. Eatock Taylor, D. Greaves，《黏度对二维水槽中瞬态自由表面波的影响》，《工程数学杂志》40（2001）77–90。[26] P. G. Drazin, W. H. Reid，《流体动力稳定性》，剑桥大学出版社，2004年。

[27] J. Grace, T. Wairegi, T. Nguyen, 《单个液滴和气泡在不互溶液体中自由运动时的形状和速度》, Trans Inst Chem Eng 54（1976）167–173。[28] J. Hua, J. Lou, 《黏性液体中气泡上升的数值模拟》, 《计算物理学杂志》222（2007）769–795。[29] S. Osher, J. A. Sethian, 《以依赖曲率的速度传播的前沿：基于 Hamilton–Jacobi 公式的算法》, 《计算物理学杂志》79（1988）12–49。[30] S. Turek, 《不可压缩纳维-斯托克斯方程的离散投影方法：一种算法方法》, 《应用力学与工程中的计算方法》143（1997）271–288。