无痛苦
N-S方程笔记

李东岳

dyfluid.com

裂痕是什么
　　那是阳光照进来的地方

本笔记电子版首发于 2019 年 2 月 14 日，最新修订于 2025 年 7 月 2 日。

本笔记为李东岳博士讲授的CFD 课程预习资料。可作为相关科研人员的参考资料。也可为自编 CFD 软件开发人员提供算法角度的参考。作者水平有限，一直对算法保持敬畏，难免有不妥和错误之处，敬请各位老师同学指正。本笔记采用 LaTeX 制作。勘误、增补等可前往CFD 中文网，或邮件联系 li.dy@dyfluid.com

笔记在撰写过程中耗费了大量精力。笔记之所以免费发放，得益于其他方面获得的收入。若觉得本笔记有帮助，可通过CFD 项目、CFD 课程、CFD 服务器、CFD 加特林的方式建立合作。

第一版前言　CFD 糅合物理、数值方法和计算机科学于一体，用于模拟流动状态。早期工作可追溯到 1917 年，Lewis Richardson 尝试手算偏微分方程组来预测天气预报。1943 年，Los Alamos 实验室，意图通过求解偏微分方程来预测飞行器的激波位置并用于军事行动。这被认为是 CFD 的雏形。当时，CFD 被 Los Alamos 实验室当做一种机密黑科技。到了 70 年代，得益于计算机技术的高速发展，CFD 被大量地应用于航空航天的流场模拟中。在航空行业的大力支持下，大量的湍流模型顺势而生。从某种程度上说，早期 CFD 的发展和计算机的发展直接相关。多亏大型计算集群，1980 年代，已经有了计算二维势流和三维欧拉方程的能力。随后，一些加速数值算法也被提出，如多重网格技术 [26]。80 年代中后期，CFD 已经可以用于求解粘性 N-S 方程。与此同时，大量的涡粘模型被提出。随后，直接模拟（Direct Numerical Simulation, DNS）以及大涡模拟（Large Eddy Simulation, LES）成为了 CFD 中一种解析度非常高的模拟技术。今天，连智能手机都配备了 8G 内存，4 核心的处理器几乎成为最低配置，甚至有人用手机进行小型的 CFD 计算。目前，计算机算力大大提升。CFD 与各个领域相结合，成为了一种辅助工程设计的技术。学术界，CFD 也与各种学科相互交叉，研究成果日新月异，层出不穷。呈现一种蓬勃发展的态势。

在这里，有必要区分一下流体力学与 CFD。二者之间存在不可割舍的联系，但差异也非常明显。从 CFD 的名字上来理解，Computational 译为 “计算”，Fluid Dynamics 则为 “流体动力学”。因此 CFD 即为计算流体力学。流体力学着重研究如何构建偏微分方程，CFD 着重研究如何计算偏微分方程。因此，CFD 也可以理解为首先通过流体力学研究构建对应的数学模型，然后通过数值方法对模型进行求解的过程。流体力学侧重建模，CFD 侧重数值求解。没有流体力学构建的模型 CFD 无用武之地，流体力学构建模型之后没有 CFD 则不可能获得解。举例，多相流体力学的研究者可能对两个气泡之间如何破碎感兴趣，因此这些研究学者的工作可能是通过实验监控气泡周围的流场数据（如湍流动能），构建一个破碎频率关于湍流动能的关系式。多相计算流体力学的研究者在拿到这个关系式之后，需要用数值方法去对其进行求解。另一方面，流体力学研究者离不开实验，CFD 研究者可以无实验进行研究。流体力学的顶级期刊是 Journal of Fluid Mechanics 等，CFD 的顶级期刊是 Journal of Computational Physics 等。流体力学与 CFD，也存在一种所属关系。CFD 属于流体力学研究，但不能认为流体力学研究就是 CFD。同时，Journal of Fluid Mechanics 经常会看到偏 CFD 的研究。但 Journal of Computational Physics 上流体力学的研究要少的多。所以，CFD 应该属于流体力学研究的一个小类。正如 Ferziger 等人所说 [51]：

> *“CFD* 可以看做是流体力学或者数值算法的一个分支。*CFD* 从业者需要在这俩方面具备牢固的背景知识。如果偏科，很有可能会得到错误的模拟结果。我们希望读者能重视这一点，并学习相关的知识。”

在生活中，流体的流动也吸引了科学家广泛的关注并进行了深入的研究。Bourouiba

等针对人们打喷嚏的现象，详细的研究了喷嚏液滴的流动状态，得出喷嚏飞沫可以漂浮 8 米远的结论，在 2020 年的新冠疫情中，对普通百姓防疫起了重要作用。Gillespie and Hamilton对一种烤肉架进行 CFD 模拟分析，认为经过 CFD 优化后的烤肉架炙烤后的菜品更加美味并申请了美国专利。Centaur 公司通过将 CFD 温控技术加入粮食储备系统，可以实时监控粮食仓储系统的温湿度，防止粮食变质，并拿下 3000 万人民币的融资。著名卫浴领导品牌Roca，为了提升马桶的用户体验，通过 CFD 对马桶进行了模拟，研究了水管的形状、弯度、管道直径、需水量等影响因素对流体流动的影响。此外，CFD 在汽车领域的研究已属家常便饭，蔚来、比亚迪汉、甚至宾利都通过 CFD 来对新能源汽车进行设计，力图减少风阻增加续航。Wallevik 等采用开源 CFD 软件，通过两相流模型，研究混凝土搅拌车的搅拌效果。除了地上跑的汽车，美国宇航局甚至用自编 CFD 代码研究太空飞的陨石，力图降低陨石在高空爆炸引起的危害。更为新奇的，有人用 CFD 的方法研究精子运动与人类受孕，研究喷墨打印机的喷口流动，Rabino甚至用 CFD 商业软件来分析女性平胸与巨乳的阻力系数，得到了一些令人瞠目结舌的结果。

总而言之，CFD 已经成为了一个上可飞天，下可落地的技术。各个行业的工程师，但凡涉及到流体的学问，必然都会诉求 CFD 来解决实际问题。然而，CFD 并不像 word 软件，拿来就能用，随便就能学。大量的从业者表示，CFD 学习的曲线非常陡峭。商业软件的 CFD 学习尚且相对容易。开源软件的 CFD 学习，以及 CFD 算法的学习，则是难上加难。但是，要将 CFD 学成真本领，还真得从算法出发，上升至应用，才能将 CFD 真正掌握，玩弄于股掌。同时，商业 CFD 软件与自编 CFD 软件基本上是两条路。商业软件侧重 CFD 应用，自编/开源 CFD 软件侧重算法开发。我国数值传热学泰斗、中国科学院院士陶文铨教授，曾为宇波教授等人著作的《数值传热学实训》做序，其中写道 [206]：

> “如果我们培养的大学生甚至研究生，只会使用商业软件，而不会开发自己的，特别考虑到目前广为使用的商业软件均是国外的产品，那么多年后我们国家的工程技术和研究人员的开发水平会是什么样子？不堪想象！”

2019 年起，中美贸易摩擦频繁发生，除了百姓的必需品大豆之外，连芯片及工业软件（如 MATLAB），均被限制出口。华为、中兴的业务惨遭锁喉。以计算机芯片为例，国外的芯片设备商掌握了复杂的芯片制造工艺，随着工艺越来越高精尖，芯片设备商的话语权一步一步的提升。因此，在过去的长达几十年里，美国一直以各种手段，保证掌控芯片制作工艺，保证自己在设备领域的绝对领导地位。

类似于芯片，CFD 软件，作为 CAE 大系中的一个小类，一直都是中国卡脖子的技术难点。国外的开源软件 OpenFOAM，用了 40 年的时间发展到了今天。中国的 CFD 大牛并不少，甚至涌现出很多世界一流的 CFD 研究学者，在这些学者的领导下，出现了一些 in-house 的 CFD 求解器。另一方面，很多优秀的普适性 CFD 求解器甚至都开源了。但是为什么中国做不出一个类似 OpenFOAM 的强大的普适性 CFD 求解器。这与我们如何看

待知识产权的态度、如何评价科研成果等有直接关系。要做中国自主的 CFD 软件，需要 5 年、10 年，甚至几代人的努力。任正非曾经感叹：

> “我们国家修桥、修路、修房子，已经习惯了砸钱就行。但是芯片砸钱不行，得砸数学家、物理学家、化学家。”

因此，如果这份笔记，能让我国的 CFD 人才少走弯路，或激发相关人员研究 CFD 的兴趣，也算作为一个 CFD 从业者，为我国整个 CFD 行业应尽的一份责任。

2019 年年末

李东岳

北京

第二版前言　第一版前言撰写后 5 年的今天，CFD 有了空前的发展。结合机器学习，少有任何一个学科能在 5 年内有如此盛况。这甚至在 CFD 的整个发展历史中也是从来没出现过的。回顾历史，在 1940 年代二战期间，Los Alamos 实验室尝试使用 CFD 来进行原子弹爆炸的冲击波模拟。这被认为是使用电子计算机进行 CFD 求解的首次尝试。可见 CFD 在诞生时就是针对可压缩流动而生。随后的 1970 年代前后是经典 CFD（或者说可压缩 CFD）的研究黄金时期。大量的可用于求解跨音速可压缩欧拉方程的高精度数值格式涌现。目前在用的主流对流格式或相关算法都大体诞生在这个黄金时代（如 vanLeer 格式、Godunov 格式、Roe 格式、中心格式、HLL 格式等）。1980 年代，英国帝国理工大学 Spalding 课题组大力的将不可压缩 CFD 推广至工业界，Spalding 还向工业界输送了大量的 CFD 科研人才，随后全世界开始了通过 CFD 来进行工业设计的浪潮。从这个角度来说，没有 Spalding 在工业界大力推广 CFD 以及在学术界的鼎力传授，就没有 CFD 在工业界的全行业发展，Spalding 当之无愧的是 CFD 之父。因此整个 CFD 的发展大体可以分为 1970 年前的可压缩流求解，到 80 年代的可压缩高精度格式、同位网格不可压缩流拓展，到 2000 年前的湍流模型研发，以及在此之间穿插的 CFD 多相流、燃烧、传热等拓展，再到随后的 CFD 工业化。虽然最开始 CFD 诞生于可压缩流，目前不可压缩 CFD 的行业应用（化工、建筑、暖通、风电、汽车、船舶等）远远超出了可压缩 CFD 的应用范畴。

最近几年，得益于 GPU 算力的发展，数据驱动 CFD 横空出世，以不可阻挡之势横扫各个 CFD 的分支，大量的代理模型涌现，工业界也出现了 AI-CFD、AI for Science 相关热点。2019 年 2 月见刊的物理信息神经网络 [131] 在 2023 年单单一年被引共计 3200 余次，随后相关作者在 Science 提出一个新的流体力学分支：隐藏流体力学（Hidden Fluid Dynamics）[132]。世界所有相关 CFD 研究人员都有目睹了一个新兴科研分支的诞生。结合目前 GPU 算力的爆发，数据驱动 CFD 在未来几年依然会是国际上的研究热点。问题是为什么数据驱动 CFD 成了天选之子？为什么偏偏是 CFD 与机器学习结合之后有了空前的盛世？原因有几点，且很好理解。CFD 最基本的元素是通过算法来对偏微分方程组进行求解。以数据驱动 CFD 中的物理信息神经网络举例，物理信息神经网络主要处理的问题是通过机器学习来求解偏微分方程组。偏偏这些方程组也是经典 CFD 尝尝用来验证的（比如对流方程、Burgers 方程）。因此物理信息神经网络求解 CFD 方程可以被看做一个全新的数值求解方法，其本源是数据驱动并进行物理知识的嵌入。一些其他的数据驱动代理模型也可以近似的来理解。例如 1980 年左右比较火爆的湍流模型研究一般从物理或实验中获取相关的经验来构建湍流模型的偏微分方程组。但数据驱动湍流模型从大量的流场数据出发，建立变量的映射关系来对偏微分方程组进行封闭。也即数据驱动。另外一个原因是一些数据驱动 CFD 方法不需要大量的物理知识。经典 CFD 由于离散数学的高度复杂性已经成为了科研人员初期进行研究的阻碍。数据驱动 CFD 从数据出发使得计算求解流程比经典 CFD 简单若干个数量级。由于其相对经典 CFD 更加易于上手，大量科研人员将数据驱动 CFD 作为一个新的科研切入点。

在第一版前言中重点讨论了流体力学与 CFD 的区别。这在当时是很有必要的。在当下，或许应该讨论数据驱动 CFD 与机器学习的区别。数据驱动 CFD 是 CFD 的一个分支。可以理解为将机器学习相关算法应用于 CFD。传统的机器学习算法跟 CFD 是没有任何关系的，相关人员用机器学习主要是进行人脸识别、语言翻译、自动驾驶等。数据驱动 CFD 本质是用机器学习的算法来求解 CFD 方程组。且数据驱动 CFD 大量的依赖 CFD 数据集。因此数据驱动 CFD 被归类于 CFD 一个分支，而不是普适性机器学习的分支是必然的。同时，针对数据驱动 CFD 的研究也存在一些不一样的声音。Spalart 在 2023 年发表文章，公开抨击数据驱动湍流模型的若干问题 [152]。很多人也认为数据驱动 CFD 从数据出发，其中缺乏物理的限定这完全不可理解。然而经测试数据驱动 CFD 确实能攻克经典 CFD 难以实现的技术难点。正确看待数据驱动 CFD 与经典 CFD 的长处与短处保持客观很有必要。

值得欣慰的是，目前不仅仅是数据驱动 CFD，也包含经典 CFD，依然是一个蓬勃发展的学科。越来越多的课题组通过 CFD 来进行研究。15 年前的国内，使用 CFD 来进行研究看起来还相对是一个比较时髦的领域（不可否认国外使用 CFD 研究要早很多年）。2024 年的今天，基本所有流体力学研究课题组都进行了 CFD 研究。在未来我相信在新兴算法以及 CPU/GPU 算力增长的加持下，CFD 依然会是国际研究的强势学科，并渗入到工业界的各个领域。

2024 年的今天，长期的中美科技之争已经成为定局。然而在第一版前言撰写后的今天，经历了新冠病毒疫情、国际政治局势的变化、俄乌战争以及中东地区的冲突，这期间发生了太多不可预计变化。我已经没有心态写出之前的话语。唯独不变的是 CFD 算法无止境。本笔记在 2019 年免费线上发布后，最初是作为 CFD 算法编程课的预习资料，在后来随着时间的增长内容一致在大量的扩充。虽没有进行统计，但经私下交流可以证实的是本笔记已经在国内的 CFD 界产生了重要的影响力。这 5 年来笔记一直在更新，涵盖 CFD 的前沿热点以及方方面面。我也想类似其他书籍一样来感谢帮助撰写的一些学生。但很遗憾本笔记是我自己一人所写，因此并没有一个致谢名单。但是我要感谢所有阅读本笔记，并认真学习甚至勘误的同行们。您的仔细阅读表示了您对本笔记的认可。也感谢那些一直关注我的朋友们。

2024 年 8 月 9 日

李东岳

Baillargues

勘误：

190705 陈佳 增加脚注 3

190619 谢鹏 更正若干笔误

190610 沈学峰 更正若干笔误

190504 Frank0514 更正若干笔误

190610 刘威 更正 8 个问题

190607 汪洋 更正若干笔误

190314 灿 更正交叉引用

190222 金国庆 (2.26)左侧第二项 $\frac{\partial}{\partial t}$ 更正为 $\frac{\partial}{\partial x}$

191023 蔡欣 更正方程(3.67)下标

191204 梁钢 更正方程(3.34)、(3.35)、(3.36)

191221 陈东林 更正若干笔误

200207 文洋 增加方程(6.239)中的 Δx

200318 赵鹏 更正若干笔误

200326 周佳其 更正若干笔误

200716 Yongbo 更正方程索引

200826 姚卫 更正方程(3.69)的 k 项

210102 王飞 更正若干笔误

210219 ZY1905417 更正若干笔误

210225 王忠琨 更正若干笔误

210317 张仕钊 更正方程(4.296)正负号

210319 方筑 更正若干笔误

210325 warnerchang 更正若干笔误

210818 袁超 更正若干笔误

210903 Kingkong 更正若干笔误

210921 向北 更正若干笔误

211021 王飞 更正若干笔误

211209 袁宝强 更正若干笔误

220320 Tong 更正一些概念问题

220406 管策 更正一处笔误

220527 周星光 更正雷诺应力扩散项和湍动能耗散率扩散项中少写了一个点

220711 郑亮 增加方程(3.67)的脚注

220802 沙小涵 修正脚注 3 的时间项

220802 路天庆 修正泰勒公式若干标识

220827 J. Hu 修正泰勒公式若干标识

221020 王东旭 修正方程(6.244)的 ΔV
221125 EZY 修正 ShihQuadraticKE 的产生项 G
230104 淮文 修正若干笔误
230105 陈铃伟 修正若干笔误
230803 匿名 修正方程(6.107)、(6.108)中的 A_f
230906 夏花秋叶 修正方程 $\mathcal{W}_l^r, i = 1, 2, 3$ 为 $\mathcal{W}_i^r, i = 1, 2, 3$
231117 田园 修正附录若干方程中的 ϕ_w 系数为 2
231123 陈铃伟 修正若干笔误
231123 王飞 增加 SpalartAllmaras 湍流模型适用性
231201 thegame 修正笔误
240430 孙逸凡 修正 kOmegaSSTDES 模型
240716 剑金锋 修正两处笔误
240829 余港龙 脚注 11 更改正负号
240901 line 更正方程 5.3，5.4 的编号
240902 dragonbin 修正脚注 23，修正 Lax-Friedrichs 等格式的数值扩散项系数
240903 彭锋 更正“潜”水方程为“浅”水方程
241202 娄喻森 删除二阶张量一些基本操作
241202 Banbor 修正浮力 kEpsilon 方程
250116 李成立 更正图6.16中的序号
250213 Wonderstruck 修正方程 3.18 的引用
250314 武锦芝 更正若干笔误
250509 范准 更正若干笔误

DYFLUID 创立于 2014 年，是国内首家以非 CFD 软件代理形式运营、专注算法编程的 CFD 咨询公司。DYFLUID 旨在鼓励 CFD 从业者开展算法研究，转变 CFD 商软的用户习惯，倡导 CFD 商业环境新秩序。2017 年 DYFLUID 首次推出手推方程现场编程的 CFD 课程，2020 年首次提出 CFD 工作站以性能为交付标准而非硬件参数，首次提出 CFD 超算优劣要以加速比为判定标准。中的 D 表示 CFD 中的有限控制体及导数的首字母，左右的弧线表示流线。目前 DYFLUID 是国内最大的 CFD/OpenFOAM 网站、CFD 中文网为国内最大的 CFD 论坛。DYFLUID 扎根于 CFD 咨询，提供下至硬件上至软件等服务的 CFD 全业态公司。DYFLUID 在 CFD 领域持续引领前沿。DYFLUID 不效仿不跟随，DYFLUID 制定 CFD 行业新规则。

目录

第一章 百万美元大奖的 **N-S** 方程 **1**

1.1 奇妙 N-S 方程 1

1.2 百万美元的问题是什么？ 2

第二章 方程标识 **5**

2.1 方程的各种写法 5

2.2 索引标识法 7

2.3 偏导标识法 8

2.4 运算符标识法 9

2.5 **∇U** 的写法 10

第三章 **N-S** 控制方程 **11**

3.1 泰勒公式 13

3.2 网格、有限控制体、无穷小微团 13

3.3 连续性方程 15

3.3.1 微分/导数形式 15

3.3.2 积分形式 17

3.3.3 通量与速度散度 19

3.3.4 拉格朗日观点 23

3.3.5 连续性方程小结 23

3.4 动量方程 24

3.4.1 受力分析 24

3.4.2 动量守恒 27

3.4.3 守恒/非守恒转化 28

3.4.4 封闭 29

3.4.5 积分观点 31

3.5 N-S 方程的展开与思考 33

第四章 经典 **CFD** **35**
4.1 湍流 35
4.1.1 直接模拟 DNS 36
4.1.2 大涡模拟 LES 37
4.1.3 雷诺平均 RANS 43
4.1.4 二维 DNS/LES 45
4.1.5 RANS-LES 混合模型、DES 模型 45
4.2 多相流 48
4.2.1 微观模型 49
4.2.2 宏观模型 52
4.2.3 介尺度模型 54
4.2.4 多尺度模型的适用性 55
4.3 气体动理学与矩方法 57
4.3.1 分子的自由度 57
4.3.2 速度分布函数、玻尔兹曼方程 57
4.3.3 矩方程 61
4.3.4 欧拉方程、五矩方程、十矩方程 63
4.4 Von Neumann 稳定性分析与截断误差 65
4.4.1 变量分离法 65
4.4.2 抛物线方程稳定性分析、无条件稳定 67
4.4.3 截断误差 68
4.5 守恒法则与双曲系统 69
4.5.1 守恒法则 70
4.5.2 线性双曲系统 71
4.5.3 黎曼问题 72
4.5.4 积分形式与守恒方程 75
4.5.5 封闭与模板 76
4.5.6 中心格式 76
4.5.7 Lax-Friedrichs 格式 76
4.5.8 Kurganov-Tadmor 格式 77
4.5.9 再看 Roe 格式 77
4.5.10 Godunov 格式 77
4.6 化学反应、ODE 求解器 79
4.6.1 复杂反应、基元反应、可逆反应 79
4.6.2 反应速率 RR、质量作用定律 80

4.6.3 化学反应实例 82
4.6.4 刚性方程组、ODE 求解器 83
4.7 大气边界层与大气稳定度 85
4.7.1 湍流动能自保持 86
4.7.2 地转风与科氏力 88
4.7.3 中性、稳定以及不稳定大气环境 90
4.7.4 下垫面 92
4.8 多孔介质模型 93
4.8.1 多孔介质平均技术 93
4.8.2 Darcy 定律 94
4.8.3 Forschheimer-Darcy 定律 95
4.8.4 一个错误 96
4.8.5 多相多孔介质、IMPES 算法 96
4.8.6 Darcy-Brinkman-Stokes 方程、固相连续介质模型 98
4.9 结构力学、可变形的固体 99
4.9.1 两相固液多孔介质模型 99
4.10 稀疏线性系统求解器 100
4.10.1 基本迭代求解器 101
4.11 CFD 小问题 102
4.11.1 物质导数 102
4.11.2 结构网格与非结构网格 104
4.11.3 体膨胀系数 $-\frac{2}{3}\mu(\nabla\cdot\mathbf{U})\mathbf{I}$ 105
4.11.4 二阶张量不变量 Invariants 106
4.11.5 应力的 Deviatoric 与 Hydrostatic 106
4.11.6 湍流的涡粘近似、$\frac{2}{3}\rho k\mathbf{I}$，$P=\tau_{ij}\frac{\partial u_i}{\partial x_j}$，$\sqrt{2S_{ij}S_{ij}}$ 107
4.11.7 变量有界性 110
4.11.8 守恒与非守恒、守恒变量与原始变量 111
4.11.9 分离式求解器与耦合式求解器 112
4.11.10 方程奇异 113
4.11.11 显性离散、隐性离散 114
4.11.12 Jacobian 矩阵 114
4.11.13 非牛顿流体 115
4.11.14 Stokes 流动 117
4.11.15 算子分裂法 117
4.11.16 总温、驻点温度 118

4.11.17 Sutherland 法则 118
4.11.18 波的传输与边界条件 118
4.11.19 计算气动声学 119
4.11.20 Partial Elimination 算法 120
4.11.21 阻力、切向力、法向力、阻力系数 121
4.11.22 三维下的库朗数计算 122

第五章 数据驱动 CFD 125
5.1 监督学习 126
5.2 流场重组、超分辨率、扩散模型 133
5.3 数据驱动加速求解 137
5.4 数据驱动湍流模型 139
5.5 数据驱动多相计算流体力学 144
5.6 循环神经网络与瞬态流场预测 145
5.7 正向逆向 PINN、隐藏流体力学 147
5.8 POD、DMD、降阶类方法 154
5.9 PyTorch 与 libtorch 158
5.10 数据驱动 CFD 侧重点 160

第六章 OpenFOAM 算法速查 163
6.1 OpenFOAM 中的非结构网格 163
6.1.1 polyMesh 中的 points 164
6.1.2 polyMesh 中的 faces 165
6.1.3 polyMesh 中的 owner、neighbour 169
6.1.4 虚拟的 cells，体心，体积 172
6.1.5 line 和 edge 177
6.1.6 偏斜度 177
6.2 OpenFOAM 中的 TVD 格式 178
6.2.1 非结构网格 TVD 格式 178
6.2.2 附加限制器的数值格式 180
6.2.3 显性修正的数值格式 181
6.3 OpenFOAM 中的 fvc 显性离散 182
6.3.1 fvc::ddt 显性时间项计算 182
6.3.2 fvc::div 显性对流项离散 183
6.3.3 fvc::laplacian 显性拉普拉斯项计算 185

6.3.4 fvc::snGrad 显性面法向梯度项计算 185
6.3.5 fvc::grad 显性梯度项计算 187
6.3.6 fvc 操作符的单位 188
6.4 OpenFOAM 中的 fvm 隐性离散 188
6.4.1 稀疏线性系统、`lduMatrix`，`fvMatrix` 等 188
6.4.2 fvm::ddt 隐性时间项离散 191
6.4.3 fvm::laplacian 隐性拉普拉斯项离散 194
6.4.4 拉普拉斯项非正交修正 197
6.4.5 fvm::div 隐性对流项离散 197
6.4.6 fvm 操作符的单位 200
6.5 OpenFOAM 中的 fvMatrix 稀疏线性系统 200
6.5.1 fvMatrix 之 Gauss-Seidel 求解器 200
6.5.2 fvMatrix 之 PCG、PBiCG 求解器 202
6.5.3 fvMatrix 的 H()、H1() 204
6.5.4 fvMatrix 的操作符重载 206
6.5.5 fvMatrix 的 correction() 207
6.5.6 fvMatrix 中的 setValue() 209
6.5.7 fvMatrix 的单位 211
6.5.8 Bandwidth 带宽 212
6.6 OpenFOAM 中的 MULES 214
6.6.1 寻找最大值最小值 214
6.6.2 反扩散通量 216
6.6.3 可流入/流出的最大反扩散通量、反扩散通量限制器 218
6.6.4 标准 FCT 限制器 219
6.6.5 限制器迭代更新算法 222
6.7 OpenFOAM 中的热物理模型 224
6.7.1 能量方程 224
6.7.2 速度压力密度耦合、psiThermo 与 rhoThermo 226
6.7.3 更新 ρ 的 EOS 状态方程 228
6.7.4 更新 T 的 sensibleInternalEnergy 与 sensibleEnthalpy 228
6.7.5 更新 C_p, C_v 的 thermo 热模型、JANAF 229
6.7.6 正压模型 230
6.8 OpenFOAM 中的多重参考系 230
6.9 OpenFOAM 中的多孔介质模型 232
6.10 OpenFOAM 中的 IATE 模型 234

6.11 OpenFOAM 中的化学反应 237
6.11.1 化学反应 237
6.11.2 ODE 求解器 239
6.12 OpenFOAM 中的边界条件 242
6.12.1 边界条件计算方法 242
6.12.2 OpenFOAM 边界条件速查 243
6.13 OpenFOAM 常用代码 253
6.13.1 div(U) = div(phi)? 253
6.13.2 laplacian(ϕ) = div(grad(ϕ))? 254
6.13.3 fvm::ddt(alpha1) - fvc::ddt(alpha1)? 255
6.13.4 -fvc::flux(-phir,..) = fvc::flux(phir,..)? 256
6.13.5 pEqn.flux() 与 fluxRequired? 257
6.13.6 fvc::div(phiHbyA) 259
6.13.7 fvc::reconstruct() 260
6.13.8 fvc::smooth() 261
6.13.9 deltaCoeffs() 262
6.13.10 R、sigma、prime2Mean、wallShearStress 263
6.13.11 fvm::ddt(psi, p) 与 psi*correction(fvm::ddt(p))? 264
6.13.12 bound() 265
6.13.13 adjustPhi() 265
6.13.14 pressureReference() 或 setReference() 266
6.13.15 constrainHbyA() 268
6.13.16 continuityErrs 与 compressibleContinuityErrs 269
6.13.17 limitPressure 与 limitTemperature 269
6.13.18 fixedValueConstraint 270
6.13.19 constrainPressure 270
6.13.20 correctPhi 271
6.14 OpenFOAM 中的湍流模型 272
6.14.1 RANS 湍流模型 272
6.14.2 LES 湍流模型 284
6.14.3 RANS-LES 混合模型 286
6.14.4 壁面函数 288

附录 A 洞见偏微分方程 295
A.1 稳态扩散方程 295

A.2 非均一源项 297
A.3 扩散系数的影响 299
A.4 全局守恒与边界条件限定 302
A.5 OpenFOAM 离散的矩阵系数 304

附录 B 新版 OpenFOAM 之模块化求解器 307
B.1 版本适配 307
B.2 代码实例 307

第一章　百万美元大奖的 N-S 方程

1.1　奇妙 N-S 方程

谈到 CFD，大部分人最开始想到的就是 Navier-Stokes 方程（N-S 方程）。N-S 方程从守恒定律推导而来：

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0, \tag{1}$$

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) = -\nabla p + \nabla \cdot \boldsymbol{\tau}, \tag{2}$$

其中 ρ 为密度，$\mathbf{U}$ 为速度，p 为压力，$\boldsymbol{\tau}$ 为剪切应力。N-S 方程具有以下特点：

- 方程(2)中左边第二项是关于 $\mathbf{U}$ 的偏导数，这种未知量和未知量乘积的问题构成非线性问题，CFD 对非线性问题需要特殊处理。一方面，非线性的双曲问题的解可能会存在间断（如激波）。另一方面，非线性项也是湍流在数学方程中的体现；

- 方程(2)的数学特征复杂。不同数学特征的问题需要调用不同的时间/空间离散格式。若方程(2)中省略若干项则会改变方程的数学特征，例如若将方程右侧置为 0，则变为双曲特征的欧拉方程。欧拉方程得益于其双曲特性，可采用迎风类显性算法推进，各种基于有限体积法的高分辨率格式因此而生（交错网格中心格式、中心-迎风格式等）。同时，动量方程的对流项和扩散项的相对强弱，也会影响边界条件的设置（如 DNS 的出口无反射边界条件）；

- 在马赫数较大时（如大于 0.3），方程(1)可用来求解密度，方程(2)可用来求解速度，同时附加能量方程求解温度以及状态方程求解压力，即密度基求解器。在马赫数较小时，并没有单独的压力方程，并且方程(1)缺少主要求解变量。这导致压力的求解需要特殊的策略。这也是 CFD 中压力基 SIMPLE/PISO 算法要处理的问题。在构建离散矩阵的情况下，对速度和压力整合处理还是单独处理，是分离求解与耦合算法的重要问题；

- N-S 方程为宏观方程，调用了宏观假定，其可从玻尔兹曼方程附加 Chapman-Enskog 展开推导而来。N-S 方程也可以看做从介尺度模型演化的宏观二阶矩模型。在无压力无粘性的条件下具备弱双曲特征。由于失去了高阶矩的统计学特征，因此 N-S 方程在某些情况下是不适用的，比如稀薄流、微流动、以及气固多相流中的颗粒轨迹交叉现

象都不能用 N-S 方程来描述。其数学本源在于 N-S 方程假定流体为一种近似均衡状态，在某一个网格点存在一个单值的分布。然而，在分子/颗粒数量非常小的情况下，这些分子/颗粒基本无碰撞，会导致多值问题的产生。因此，在这种情况下，需要对统计学分布的高阶矩进行追踪，N-S 方程作为二阶矩模型是往往不够的。

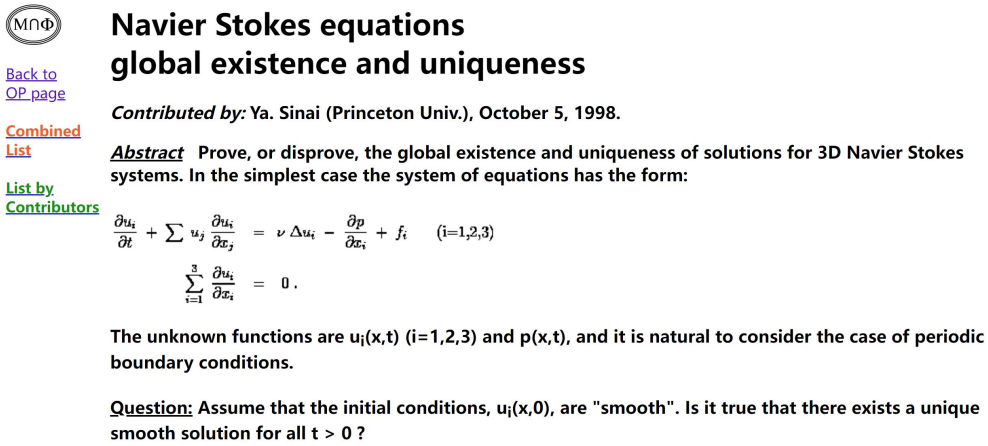
1.2 百万美元的问题是什么？

克雷数学研究所曾经在 2000 年提出七个千禧年问题，若有人解决任意一个，在 2000 年就可以获得 100 万美元奖金。在 2017 年甚至有一部电影叫“天才少女”，电影虚构了一个小女孩，其母亲在自杀之前，就致力于解决千禧年问题。小女孩出生后也继承了母亲的数学天分，在 7 岁的时候就证明了千禧年问题，但其数学天分一直被扼杀的故事。

既然这个问题这么复杂，那么这个问题到底是什么？其原始问题可以简述如下 [50]：

- 证明或反证无体积力的三维 N-S 方程，给定任意的光滑的散度为 0 的初始速度场，在无穷大或周期的计算域内，在所有的时间周期内，存在一个光滑的、有界的解；

N-S 方程千禧年问题原本一共 4 个问题。且存在一个所谓的“爆掉了”的定义。也即解的速度的平方（能量）在全场加和后无穷大。克雷数学研究所抛出的数学问题的描述是极为严谨的且需要各种方程假定。在这里尽可能的将其归纳无数学方程的可理解的 1 个问题。可能略失严谨性，但更容易让人理解。



Back to OP page

Combined List

List by Contributors

Navier Stokes equations
global existence and uniqueness

Contributed by: **Ya. Sinai (Princeton Univ.), October 5, 1998.**

Abstract **Prove, or disprove, the global existence and uniqueness of solutions for 3D Navier Stokes systems. In the simplest case the system of equations has the form:**

$$\frac{\partial u_i}{\partial t} + \sum u_j \frac{\partial u_i}{\partial x_j} = \nu \Delta u_i - \frac{\partial p}{\partial x_i} + f_i \qquad (i=1,2,3)$$

$$\sum_{i=1}^{3} \frac{\partial u_i}{\partial x_i} = 0 .$$

The unknown functions are $u_i(x,t)$ (i=1,2,3) and p(x,t), and it is natural to consider the case of periodic boundary conditions.

Question: Assume that the initial conditions, $u_i(x,0)$, are "smooth". Is it true that there exists a unique smooth solution for all t > 0 ?

图 1.1: 普林斯顿大学在 1998 年提出的 N-S 方程解的唯一性以及存在性的公开问题。

历史上，其他学者在 2000 年之前也提出过类似的问题。如图1.1所示，普林斯顿大学在 1998 年曾抛出一个问题：

- 在周期边界条件下给定一个光滑的初始条件，在所有的时间周期内，证明或反证三维 N-S 方程的解的唯一性以及存在性；

在 2000 年，历史上也存在了一些阶段性的进展。如：

- 在 1969 年，数学家 Ladyzhenskaya 证明了在二维情况下，N-S 方程的解的存在性以及光滑性 [82]。但是这些证明对三维情况下的 N-S 方程的解的证明毫无帮助 [50]；
- 在三维情况下，如果初始的速度比较小的时候[1]，也可以证明 N-S 方程在有限的时间内，解的存在性以及光滑性。这个有限的时间点，也即“爆掉”时间 [50]；
- 1934 年，Leray 证明了三维 N-S 方程永远存在弱解，但唯一性目前不可证明 [91]；但欧拉方程存在弱解的不唯一性；
- 一些文献发现，一些特定的流动，解存在不唯一性。在这里要注意，仅限于一些特殊的流动状态，如顶盖驱动流 [143]、Rayleigh-Benard 对流 [161, 71]；
- 2016 年陶哲轩发表文章，其并没有直接回答 N-S 方程的千禧年问题，而是构建了一套自己的理论，认为其可能会通过该方法构建出一个 N-S 方程的“爆掉”的解 [164]；
- 在 arxiv 进行 N-S 方程解的唯一性以及存在性的关键词搜索，会搜出大量的相关文章，在这里很难做全面相关介绍；

最后的问题是，如果 N-S 方程解的唯一性、光滑性、以及存在性都没有被证明，那么 CFD 求解的是什么？解是否有意义？笔者主观认为，N-S 方程解的特性虽然没有被数学证明，但是实际操作过程中，只要设置合理，CFD 对 N-S 方程的求解都是唯一的（虽然从数学上没有被证明）。且已经被证明的是，在有限的时间内，N-S 方程的解是存在的且光滑的。进行无限长时间的 CFD 计算完全没有意义。这也可能被认为是还没有推进到“爆掉”时间。

正如上面讨论的，N-S 方程中存在大量的数学问题，初学者也看的不明不白。这无关紧要。本章仅仅抛出一块砖头，让大家了解隐藏在 CFD 中的方程的美妙含义。

[1] 在原始文献中，费曼也没有定义多小是比较小。

第二章　方程标识

学习 CFD 理论首先遇到的就是各种各样的偏微分算符。能看懂、拆分 CFD 方程是研究算法的最基本步骤。给出一个 CFD 方程（如动量方程），可以不知道是怎么推出来的，但要能看懂。本章介绍 CFD 方程的各种写法，不涉及到任何的 CFD 算法。

2.1　方程的各种写法

本笔记中主要采用本节讨论的标识法讨论 CFD 方程。给定一个张量，其存在各种不同的表示方法。在矢量标识法中，标量全部采用斜体，如压力 p。矢量采用正体加粗，如速度矢量 $\mathbf{U}$，其具有三个分量 u_1，u_2，u_3 或 u，v，w。二阶张量也采用正体[1]，比如应力张量 $\boldsymbol{\tau}$，其具备 9 个分量，其可表示为：

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix} \tag{2.1}$$

下面将不可压缩流体动量方程为例对其进行拆分，这个方程若采用矢量标识法可以写为：

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) = -\nabla \frac{p}{\rho} + \nabla \cdot (\nu \nabla \mathbf{U}) \tag{2.2}$$

其中的 $\mathbf{U}$ 为速度矢量，p 为压力，ρ 为密度，ν 为粘度。一般来讲，CFD 文献中通常采用方程(2.2)的形式，而并不进行展开进而更加紧凑。下面介绍如何将其展开为 3 个方程。

- 方程(2.2)第一项表示 $\mathbf{U}$ 对时间的偏导数，因为 $\mathbf{U}$ 为矢量，故其导数的分量形式为：

$$\frac{\partial \mathbf{U}}{\partial t} = \begin{bmatrix} \frac{\partial u_1}{\partial t} \\ \frac{\partial u_2}{\partial t} \\ \frac{\partial u_3}{\partial t} \end{bmatrix} \tag{2.3}$$

  其中 u_1 表示 x 方向速度，u_2 表示 y 方向速度，u_3 表示 z 方向速度。这样拆分之后的方程，即为各个方向的速度针对时间的偏导数。如果理解方程(2.3)有困难，那么有必要预习一下《高等数学》（同济大学版）第二章。

[1] 如果难以理解二阶张量的含义，可以这样尝试：矢量是一阶张量，具有三个分量，二阶张量则具有 9 个分量。

- 方程(2.2)第二项 $\nabla\cdot(\mathbf{UU})$ 中的 $\mathbf{UU}$ 是一种简写，完整形式为 $\mathbf{U}\otimes\mathbf{U}$，$\otimes$ 是一个张量运算符。依据 $\otimes$ 的定义，$\mathbf{UU}$ 可以写为：

$$\mathbf{U}\otimes\mathbf{U}=\mathbf{UU}=\mathbf{U}\cdot\mathbf{U}^T=\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix}[u_1,u_2,u_3]=\begin{bmatrix}u_1u_1 & u_1u_2 & u_1u_3\\u_2u_1 & u_2u_2 & u_2u_3\\u_3u_1 & u_3u_2 & u_3u_3\end{bmatrix} \tag{2.4}$$

接下来看 $\nabla\cdot$，其为散度算符，有时用 div 来表示。对一个矢量（1 阶张量）做散度的结果为一个标量（0 阶张量），对一个 2 阶张量做散度的结果为矢量（1 阶张量）。因此，对任意 n 阶张量做散度操作之后，结果为 $n-1$ 阶张量。举例，对一个矢量 $\mathbf{U}$ 做散度有：$\nabla\cdot\mathbf{U}=\frac{\partial u_1}{\partial x}+\frac{\partial u_2}{\partial y}+\frac{\partial u_3}{\partial z}$。因此，方程(2.2)中的第二项 $\nabla\cdot(\mathbf{UU})$ 即为对一个 2 阶张量做散度：

$$\nabla\cdot(\mathbf{UU})=\nabla\cdot\begin{bmatrix}u_1u_1 & u_1u_2 & u_1u_3\\u_2u_1 & u_2u_2 & u_2u_3\\u_3u_1 & u_3u_2 & u_3u_3\end{bmatrix}=\begin{bmatrix}\frac{\partial u_1u_1}{\partial x}+\frac{\partial u_2u_1}{\partial y}+\frac{\partial u_3u_1}{\partial z}\\\frac{\partial u_1u_2}{\partial x}+\frac{\partial u_2u_2}{\partial y}+\frac{\partial u_3u_2}{\partial z}\\\frac{\partial u_1u_3}{\partial x}+\frac{\partial u_2u_3}{\partial y}+\frac{\partial u_3u_3}{\partial z}\end{bmatrix} \tag{2.5}$$

- 方程(2.2)第三项也存在 ∇，这一项中没有了 $\cdot$ 符号。单独的 ∇ 表示梯度运算，有时也被写为 grad。对一个标量（0 阶张量）做梯度的结果为一个矢量（1 阶张量），对一个矢量做梯度的结果为 2 阶张量。因此，对任意 n 阶张量做梯度之后的结果为 $n+1$ 阶张量。举例，对一个标量 p 做梯度有[2]：

$$\nabla p=\begin{bmatrix}\frac{\partial p}{\partial x}\\\frac{\partial p}{\partial y}\\\frac{\partial p}{\partial z}\end{bmatrix} \tag{2.6}$$

方程(2.2)第四项，如果忽略粘度，$\nabla\cdot(\nabla\mathbf{U})$ 为对速度 $\mathbf{U}$ 先做梯度再做散度。$\nabla\cdot\nabla$ 通常也写为 ∇^2，并称为拉普拉斯算子，有时也被写为 laplacian。即 $\nabla\cdot(\nabla\mathbf{U})=\nabla^2\mathbf{U}$。现尝试对其进行展开：

$$\nabla\cdot(\nabla\mathbf{U})=\nabla\cdot\begin{bmatrix}\frac{\partial u_1}{\partial x} & \frac{\partial u_2}{\partial x} & \frac{\partial u_3}{\partial x}\\\frac{\partial u_1}{\partial y} & \frac{\partial u_2}{\partial y} & \frac{\partial u_3}{\partial y}\\\frac{\partial u_1}{\partial z} & \frac{\partial u_2}{\partial z} & \frac{\partial u_3}{\partial z}\end{bmatrix}=\begin{bmatrix}\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_1}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_1}{\partial z}\right)\\\frac{\partial}{\partial x}\left(\frac{\partial u_2}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_2}{\partial z}\right)\\\frac{\partial}{\partial x}\left(\frac{\partial u_3}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_3}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_3}{\partial z}\right)\end{bmatrix} \tag{2.7}$$

结合方程(2.3)、(2.5)、(2.6)、(2.7)这四项，有三个方程。下面仅列出 x 方向（取各个项展开形式中的第一行，且忽略粘度）：

$$\frac{\partial u_1}{\partial t}+\frac{\partial u_1u_1}{\partial x}+\frac{\partial u_2u_1}{\partial y}+\frac{\partial u_3u_1}{\partial z}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_1}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_1}{\partial z}\right) \tag{2.8}$$

[2]一般情况下压力 p 为标量，即为各向同性的。在某些模型中（如假定高斯分布的十矩模型），压力为二阶各向异性张量。

方程(2.8)即为关于 u_1 的方程。仔细观察方程(2.8)，其只不过是各种导数的加和。除了 u_1 之外，如果所有的变量均为已知，那么就可以求出 u_1 关于时间的步进（看第一项速度关于时间的导数）。方程(2.8)的特点可参考第一章论述的内容。

2.2 索引标识法

CFD 中的偏微分方程还可以用索引标识法表示，这种方法在 SCI 论文以及书籍中也非常常见 [186]。在索引标识法中，符号的下标表示张量的阶数。如标量压力 p 可以记为（二者无区别）：

$$p \equiv p \tag{2.9}$$

矢量 $\mathbf{U}$ 可以写为 u_i：

$$u_i = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \equiv \mathbf{U} \tag{2.10}$$

二阶张量 $\boldsymbol{\tau}$ 定义为 τ_{ij}：

$$\tau_{ij} = \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix} \equiv \boldsymbol{\tau} \tag{2.11}$$

一定需要注意，在使用索引标识法的时候，u_i 表示矢量而不是某个具体的分量。举例说明：如果某文章采用索引标识法，则 u_i 表示速度矢量，u_1 表示 x 方向速度分量。这和上一节的写法是不同的。在本笔记中，$\mathbf{U}$ 表示速度矢量，u_i 表示各个方向的分量而不是速度矢量，其中 $i = 1, 2, 3$。因此，遇到 u_i，采用不同的数学表示方法，其含义可能不同。

索引标识法做乘积操作的时候，如果某一项中下标重复，那么则需要进行加和，如：

$$a_i b_i = a_1 b_1 + a_2 b_2 + a_3 b_3 \equiv \mathbf{a} \cdot \mathbf{b} \tag{2.12}$$

$$\begin{aligned} &\tau_{ij}\tau_{ij} = \\ &\tau_{11}\tau_{11} + \tau_{12}\tau_{12} + \tau_{13}\tau_{13} + \tau_{21}\tau_{21} + \tau_{22}\tau_{22} + \tau_{23}\tau_{23} + \tau_{31}\tau_{31} + \tau_{32}\tau_{32} + \tau_{33}\tau_{33} \equiv \boldsymbol{\tau} : \boldsymbol{\tau} \end{aligned} \tag{2.13}$$

$$\tau_{ij} u_j = \begin{bmatrix} \tau_{11}u_1 + \tau_{12}u_2 + \tau_{13}u_3 \\ \tau_{21}u_1 + \tau_{22}u_2 + \tau_{23}u_3 \\ \tau_{31}u_1 + \tau_{32}u_2 + \tau_{33}u_3 \end{bmatrix} \equiv \boldsymbol{\tau} \cdot \mathbf{U} \tag{2.14}$$

方程(2.14)中 $\tau_{ij}u_j =$ 的 j 出现两次，需要加和。i 出现一次，表示矢量。索引标识法在这种情况下也被称之为爱因斯坦求和约定。

对于偏微分方程组也可以进行类似的表示。例如对于下面的方程（即连续性方程）：

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u_1}{\partial x}+\frac{\partial \rho u_2}{\partial y}+\frac{\partial \rho u_3}{\partial z}=\frac{\partial \rho}{\partial t}+\frac{\partial \rho u_i}{\partial x_i}=0 \tag{2.15}$$

其中 $\frac{\partial \rho u_i}{\partial x_i}$ 中出现两次 i，则需要加和。对于下面的方程（即动量方程），其可以表示为：

$$\frac{\partial u_i}{\partial t}+\frac{\partial u_i u_j}{\partial x_j}=-\frac{1}{\rho}\frac{\partial p}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\nu\frac{\partial u_i}{\partial x_j}\right) \tag{2.16}$$

其中 $\frac{\partial u_i u_j}{\partial x_j}$ 中出现两次 j，则需要加和。出现一次 i，则表示分量。因此其为一个矢量的方程。现在随意从文献中抽取一个方程，如文献 [186] 中 $k-\varepsilon$ 模型的湍流动能方程，即方程 (4.9) 为[3]：

$$\rho\frac{\partial k}{\partial t}+\rho u_j\frac{\partial k}{\partial x_j}=\tau_{ij}\frac{\partial u_i}{\partial x_j}-\rho\varepsilon+\frac{\partial}{\partial x_j}\left[(\mu+\mu_t/\sigma_k)\frac{\partial k}{\partial x_j}\right] \tag{2.17}$$

对其分析可知首先其为一个标量方程而非矢量方程组，且其展开形式为：

$$\rho\frac{\partial k}{\partial t}+\sum_{j=1}^{3}\rho u_j\frac{\partial k}{\partial x_j}=\sum_{i=1}^{3}\sum_{j=1}^{3}\tau_{ij}\frac{\partial u_i}{\partial x_j}-\rho\varepsilon+\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left[(\mu+\mu_t/\sigma_k)\frac{\partial k}{\partial x_j}\right] \tag{2.18}$$

2.3 偏导标识法

CFD 一个重要分支是高超音速动力学。在高超音速研究领域，大量的文章和书籍中均采用偏导标识法进行方程分析 [92]。在偏导标识法中，方程(2.3)可以表示为：

$$\mathbf{U}_t\equiv\frac{\partial \mathbf{U}}{\partial t} \tag{2.19}$$

同理，压力 p 对 x 的导数可以写为：

$$p_x\equiv\frac{\partial p}{\partial x} \tag{2.20}$$

一维连续性方程的偏导标识法可以写为：

$$\rho_t+(\rho u_1)_x=0 \tag{2.21}$$

三维连续性方程的偏导标识法可以写为：

$$\rho_t+(\rho u_1)_x+(\rho u_2)_y+(\rho u_3)_z=0 \tag{2.22}$$

[3]读者在这里应该尝试将其写为矢量标识法。例如，对 $\tau_{ij}\frac{\partial u_i}{\partial x_j}$ 展开有（$\boldsymbol{\tau}$ 为对称二阶张量）：

$$\tau_{ij}\frac{\partial u_i}{\partial x_j}=\tau_{11}\frac{\partial u_1}{\partial x_1}+\tau_{12}\frac{\partial u_1}{\partial x_2}+\tau_{13}\frac{\partial u_1}{\partial x_3}+\tau_{21}\frac{\partial u_2}{\partial x_1}+\tau_{22}\frac{\partial u_2}{\partial x_2}+\tau_{23}\frac{\partial u_2}{\partial x_3}+\tau_{31}\frac{\partial u_3}{\partial x_1}+\tau_{32}\frac{\partial u_3}{\partial x_2}+\tau_{33}\frac{\partial u_3}{\partial x_3}=\nabla\mathbf{U}:\boldsymbol{\tau}^T=\nabla\mathbf{U}:\boldsymbol{\tau}$$

因此，$k-\varepsilon$ 的矢量标识法形式为：

$$\rho\frac{\partial k}{\partial t}+\rho\mathbf{U}\cdot\nabla k-\nabla\cdot((\mu+\mu_t/\sigma_k)\nabla k)=\nabla\mathbf{U}:\boldsymbol{\tau}-\rho\varepsilon$$

方程左侧分别为 k 的时间项、对流项以及扩散项，右侧分别为产生项与耗散项。

方程(2.22)在矢量标识法中可以表示为：

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \tag{2.23}$$

另外，在偏导标识法中，偏微分系统通常用下面的方程表示：

$$q_t + Aq_x = 0 \tag{2.24}$$

其中 q 可以为标量，也可以为矢量。如果 q 为矢量，且具备 m 个分量，那么 A 则表示 $m \times m$ 阶矩阵。例如下面的气动声学方程

$$\begin{cases} p_t + \rho c^2 u_x = 0 \\ u_t + (1/\rho) p_x = 0 \end{cases} \tag{2.25}$$

其中未知量为 p 和 u，ρ 和 c 为密度和声速，其可以写为下面更好理解的形式：

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \frac{\partial}{\partial t} \begin{pmatrix} p \\ u \end{pmatrix} + \begin{pmatrix} 0 & c^2\rho \\ 1/\rho & 0 \end{pmatrix} \frac{\partial}{\partial x} \begin{pmatrix} p \\ u \end{pmatrix} = 0 \tag{2.26}$$

在这种情况下有：

$$q = \begin{bmatrix} p \\ u \end{bmatrix}, A = \begin{bmatrix} 0 & c^2\rho \\ 1/\rho & 0 \end{bmatrix} \tag{2.27}$$

偏导标识法的这种写法可以更好的用于分析偏微分方程系统的数学特征。

2.4 运算符标识法

运算符标识法在文章中比较少见，其只不过是将数学符号翻译为相应的英文名字。例如，在笛卡尔坐标系下有

$$\operatorname{div} \mathbf{U} = \nabla \cdot \mathbf{U} \tag{2.28}$$

$$\operatorname{grad} p = \nabla p \tag{2.29}$$

$$\operatorname{laplacian} p = \operatorname{div} \operatorname{grad} p = \nabla \cdot (\nabla p) \tag{2.30}$$

$$\operatorname{curl} \mathbf{U} = \nabla \times \mathbf{U} \tag{2.31}$$

在这里需要注意的是，前文使用的 ∇ 算符在不同的坐标系下需要转换。但运算符标识法不需要转换。

2.5 $\nabla\mathbf{U}$ 的写法

现在回头看速度梯度的数学形式，在一些文章中 [106]，其被写为：

$$\nabla\mathbf{U} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} & \frac{\partial u_1}{\partial z} \\ \frac{\partial u_2}{\partial x} & \frac{\partial u_2}{\partial y} & \frac{\partial u_2}{\partial z} \\ \frac{\partial u_3}{\partial x} & \frac{\partial u_3}{\partial y} & \frac{\partial u_3}{\partial z} \end{bmatrix} \tag{2.32}$$

可以看出，方程(2.32)中的速度梯度与方程(2.7)形式并不一样。在方程(2.7)中，速度梯度被定义为

$$\nabla\mathbf{U} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_2}{\partial x} & \frac{\partial u_3}{\partial x} \\ \frac{\partial u_1}{\partial y} & \frac{\partial u_2}{\partial y} & \frac{\partial u_3}{\partial y} \\ \frac{\partial u_1}{\partial z} & \frac{\partial u_2}{\partial z} & \frac{\partial u_3}{\partial z} \end{bmatrix} \tag{2.33}$$

很明显，两个速度梯度互为转置。这并不是笔误。而是涉及到一个张量概念：分子布局（numerator layout）以及分母布局（denominator layout）。方程(2.32)中的速度梯度采用了分子布局。方程(2.7)以及(2.33)采用的为分母布局。两种布局导致写法不同。维基百科表示目前的教材中确实存在这一问题 [7]。在本书以及开源软件 OpenFOAM 中，使用的是方程(2.7)的形式。

注意，在各种期刊文章中以及教材中，并不会明确的强调是使用分子布局还是分母布局。需要结合上下文进行判断。

第三章　N-S 控制方程

流体通常泛指气体和液体。不同于固体，流体在施加剪切力的时候，会发生形变。且流体没有固定的形态。流体在快速变形的时候，也会存在一种抵抗的力，这种抵抗的力一旦流动停止，也即消失。流体这种抗拒本身发生变形的力称之为粘性力。不同的流体存在不同的粘性力。有些流体的粘性很大，有些流体的粘性很小。大部分日常生活中接触的为牛顿流体，其形变率和剪切力呈现线性的关系。部分流体为非牛顿流体，其剪切力和形变率呈现非线性的关系。

标准状态下，一立方毫米气体包含大约 2.43×10^{16} 个分子。由于气体分子之间的碰撞非常频繁，有效地将气体分子合理的进行了分布。因此这些粒子群表现的并不是单一的粒子而是连续介质。标准情况下，气体分子每秒钟大约进行 10^9 次碰撞，每一次碰撞导致移动的距离非常小（约 5×10^{-8} 米）。前者即所谓的碰撞频率，后者即所谓的分子自由程。分子之间频繁的碰撞的宏观表现，就是存在压力。例如，将手竖直的放置在水中，手背和手面感受到的力就是水分子的压力。对于理想气体，宏观的压力 p 和微观量的关系可以表示为 $p=nkT$，其中 n 表示单个分子占据的平均体积的倒数，其单位为 m^{-3}。这个平均体积主要是分子运动的平均范围。k 为 Boltzmann 常数，其值为 1.38066×10^{-23} J/K。T 表示温度。

液体不同于气体的典型特征为其对体积变化的巨大抵抗力。例如我们不能把一瓶矿泉水压缩成一瓶盖的大小。然而气体对体积变化的抵抗力是相对小的。例如我们可以轻易地用手对一个封闭的空的矿泉水瓶进行压缩。基于流体的密度是否为常数，进一步可以分为不可压缩流体和可压缩流体。液体通常考虑为不可压缩流体。虽然气体比液体的可压缩性高很多，但是是否考虑气体的可压缩性主要取决于具体情况。比如在空气动力学中，如果我们的气流速度很小，气体的可压缩性对汽车的设计影响较小，但是对于飞机，其速度有可能和声速接近或者大于声速，在这种情况下气体的可压缩性对飞机的设计非常重要。有些高压容器中，即使是液体，也应该考虑可压缩性。因为只有考虑了可压缩性，才能解释某些奇怪的物理现象，比如水锤。

特殊的情况下存在一些特殊的现象。比如宇宙中的空气接近真空，这种通常被称之为稀薄气体。稀薄气体由于气体的分子数量太少，因此气体的碰撞并不均衡。另一种为基本无碰撞的颗粒流，在颗粒不相互碰撞的情况下，会发生颗粒轨迹交叉，也就是一个粒子从

左向右移动，另一个粒子从右向左移动，在某个位置，两个粒子彼此穿过并不发生碰撞。以上两种情况为一些极端的情况，这些情况并不能用 Navier-Stokes（N-S）方程来描述。在这里需要表明的是 N-S 方程最重要的前提是连续介质假定。其可以理解为一个网格单元内存在大量的流体分子或固体颗粒。分子/颗粒之间碰撞均衡。这种情况下才能使用 N-S 方程来描述。在 N-S 方程不能描述的情况下，只能调用玻尔兹曼方程。本笔记仅仅考虑碰撞均衡的 N-S 方程[1]。

CFD 是采用计算机求解控制流体流动的偏微分方程组的学科。以日常生活中的电风扇吹风为例。电风扇吹出来的空气的流动看不见、摸不着，但是能明显的感觉吹出来的风不一样。风扇转的越快，风越大越凉快，相对应的能耗也越多。工业上的设计目标就是用最小的能耗，吹出最大的风让一群人都很凉快。在这物理现象的背后，风扇吹出来的风的速度、压力、温度等均遵循相关的物理定律，这个物理定律可以写成数学中偏微分方程的形式，其主要有两个方程：一个是连续性方程（表明流动着的空气质量是守恒的），另一个是动量方程（表明空气的流动满足牛顿第二定律）。在考虑传热的情况下，还应该附加能量方程（表明能量守恒定律）。暂且可以这样理解：求解这两个关于时间的方程，就可以求出风扇吹出来的风在某一时刻的流速。有了流速之后，同时通过风扇叶片表面的受力计算一下风扇的功率，就可以算出耗电量进行相关的设计。如果通过实验，工程师可以设计 100 个叶片的形状，检测风力的大小和能耗并做分析，取最优的叶片形状。通过 CFD，工程师可以生成 100 个叶片的几何模型，用计算机求解这 100 个几何模型，就可以计算出风力的大小和能耗进行分析。在工业应用中，一个机翼的实验工作可能要花费上百万元，一个化工反应器的中试实验工作可能需要上亿。但是通过 CFD 计算模拟，可能需要几万元就可以了。可见，CFD 的目标是让工程师从实验中解脱，并节省实验资金，进而进行工业设计。在有些情况下，可能会听到一些新词汇，如数值风洞、数值水池等，这实际上就是用 CFD 来模拟风洞内的流场、用 CFD 来模拟水池内水的流动。

问题在于，用于描述 CFD 的方程非常难解，甚至目前解析解（方程的准确解，比如 $x=-1$ 是 $x+1=0$ 的解析解）的存在与否都无从考证。目前 CFD 在学术上的研究，即如何获得这些方程的解，这些解通常被称之为数值解。在理想的情况下，数值解无限趋近于解析解。然而，理想很丰满，现实很骨感。用户获得的数值解，总是和解析解存在或多或少的差距。

各种算法犹如各路神仙过河一样，总会通过各种奇葩的方法来获得 CFD 方程的数值解。比如 CFD 中有一种方法为光滑粒子法，其将流体看做是粒子，然后对这些粒子进行跟踪进行求解并获得数值解。最近，还有研究学者通过深度学习来假拟 CFD 的方程并求解。目前 CFD 求解方法中最广泛的，莫过于有限体积法。目前主流的 CFD 软件，均使用的有限体积法进行求解。

[1]若流体的特征长度 L 远远大于平均分子自由程 λ，若流体的分子碰撞次数在标准状态下分子的碰撞频率左右（不远低于），则可用 N-S 方程进行描述。λ 和 L 的比值为粒子克努森数。N-S 方程只在克努森数远远小于 1 的情况下适用。

有限体积法的思想也很简单。举例，如果要分析整个河流的流动状态、整个房间的热流密度、血管内的血流状态，有限体积法是通过从局部到整体的方法进行分析。例如我们把整个房间分成一个个的小格子，如果可以获得每个小格子内的流动速度，那么这些格子构成的整体就是整个房间的流动速度。格子的数量越多，格子的速度场越能代表整个房间的速度场。本文主要介绍有限体积法。

3.1 泰勒公式

在推导 N-S 方程之前，有必要介绍一下泰勒公式。假设有函数 $f(x)$，其用来表示整个 x 的型线。泰勒公式可以理解为函数 $f(x)$ 可以表示为若干个函数的加和，这里 $f(x)$ 为仅仅关于 x 的函数：

$$f(x) = f(x_0) + \left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_0}(x - x_0) + ... \tag{3.1}$$

当 $x = x_1$ 的时候，泰勒公式可用于求 $f(x_1)$ 的值：

$$f(x_1) = f(x_0) + \left.\frac{\mathrm{d}f(x)}{\mathrm{d}x}\right|_{x=x_0}(x_1 - x_0) + ... \tag{3.2}$$

现以 $f(x) = x^3$ 举例，其导数为 $\frac{\mathrm{d}f(x)}{\mathrm{d}x} = 3x^2$。取 $x_0 = 2$，有 $f(x_0) = 8$，则有：

$$f(x_1) = 8 + 3 \times 2^2(x_1 - 2) + ... \tag{3.3}$$

如果 $x_1 = 3$，有 $f(x_1) = 20$。如果 $x_1 = 2.1$，有 $f(x_1) = 9.2$。可见 x_1 越趋向于 x_0，通过泰勒公式计算的值越精准。这在 CFD 中也反映了网格单元越小，结果越精准的特性。

泰勒公式也可用于多元函数，如有函数 $f(x, y, z, t)$，其位于 (x_1, y_1, z_1, t_1) 点的泰勒展开可表示为[2]：

$$\begin{aligned} f(x,y,z,t) = f(x_1,y_1,z_1,t_1) &+ \left.\frac{\partial f(x,y,z,t)}{\partial x}\right|_{x_1,y_1,z_1,t_1}(x - x_1) \\ &+ \left.\frac{\partial f(x,y,z,t)}{\partial y}\right|_{x_1,y_1,z_1,t_1}(y - y_1) + \left.\frac{\partial f(x,y,z,t)}{\partial z}\right|_{x_1,y_1,z_1,t_1}(z - z_1) \\ &+ \left.\frac{\partial f(x,y,z,t)}{\partial t}\right|_{x_1,y_1,z_1,t_1}(t - t_1) + ... \end{aligned} \tag{3.4}$$

3.2 网格、有限控制体、无穷小微团

NS 方程在表述形式上区分为积分形式以及微分形式。通过有限体积法来求解 CFD 需要划分网格。如果使用积分形式的方程，需要定义积分区域。如果使用微分形式的方程，需要使用高斯定律转化为面积分的形式，同样需要定义积分区域。这个积分区域就可以理解

[2]注意这里 f 为多元函数，因此使用偏导数符号 ∂。更详细的关于泰勒公式的内容请参考文献 [205]。

为网格。在数学上可以理解为空间位置固定的有限控制体，也可以理解为无穷小微团。空间位置固定的有限控制体在无限小的时候，就变成了无穷小微团。

如图3.1所示，有限控制体其进一步可以分为空间位置固定的有限控制体，和随流线运动的有限控制体。空间位置固定的有限控制体可以理解为一个空旷的房间（当然控制体的体积要小的多），其会流入、流出某些物理量（如质量、能量等），这些量的产生和消失符合物理学基本定律。举例，这个房间内部充满了人（不能再容纳更多的人），如果挤进来 10 个人，那么必然会挤出去 10 个人。即为质量守恒。随流线运动的有限控制体可以理解为随风飘扬的一个气球，气球内部存在大量的空气，且空气和外界不发生交换（质量不变）。这个气球的运动当然满足物理的基本定律。

在后续的讨论中，可以看出从有限控制体推导的方程为积分形式，这是因为控制体相对下文讨论的微团模型体积更大。且若采用空间位置固定的控制体，方程为守恒型方程。若采用随流线运动的控制体，方程为非守恒型。要透彻的理解 CFD 中守恒和非守恒的概念，目前来讲并不容易。在此只需要知道有不同的方程的概念就可以。

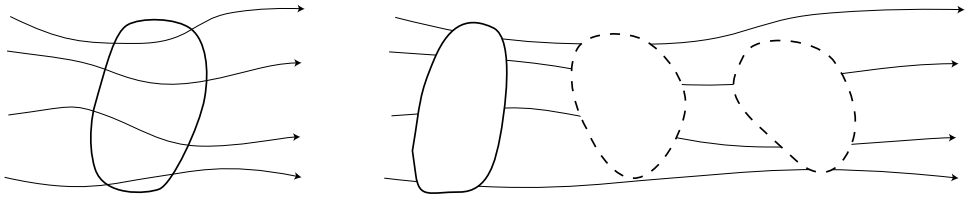


图 3.1: 左图：空间位置固定的有限控制体，流体流入控制体后继续流出。右图：空间位置移动的有限控制体，其沿着流体的流线进行移动。

如图3.2所示[3]，类似有限控制体，无穷小微团也可以进一步分为空间位置固定的无穷小微团，和随流线运动的无穷小微团。无穷小微团和有限控制体的重要区别在于无穷小微团的体积、质量等变量可以用数学中的 $\mathrm{d}V$、$\mathrm{d}m$ 来表示。有限控制体的体积要比无穷小微团要大，也即有限控制体中包含了大量的 $\mathrm{d}V$，需要用积分来表示。从数学的角度出发，$\mathrm{d}V$ 表示这个无穷小微团的体积足够小。需要注意的是，为了使得 CFD 方程适用于连续介质，这个 $\mathrm{d}V$ 还要包含一定量的流体分子[4]。

现在以空间位置固定的有限控制体或流体微团为例。我们要了解一个房间内流体的流动状况。如果房间内每个有限控制体或流体微团的流动均已知，那么我们就可以获得这个房间完整的流体流动图像。更形象的，我们把房间分为 9 个位置并用 9 个有限控制体或流体微团来表示，如果采用实验的方法对这 9 个有限控制体或流体微团检测速度，那么我们就可以认为这 9 个速度的场即为整个房间的速度场。如果我们把这个房间分为 N 块，并

[3]图中仅仅画出一个无穷小微团，若存在多个无穷小微团，每个无穷小微团应彼此相连并填充整个计算空间。也就是说整个计算域内应充满网格单元。

[4]N-S 方程并不是普适性的方程，更为底层的为玻尔兹曼方程。只有满足连续性假定，才能使得 BGK 假定成立，进而将玻尔兹曼方程简化为 N-S 方程。例如对于稀薄气体，由于并不满足连续性假定，因此 N-S 方程并不适用，只能通过数值方法求解玻尔兹曼方程（如矩方法 [160]、动理学格式方法 [192]、蒙特卡洛法 [20] 等）。对于多相流领域同样具有类似的情况，比如颗粒轨迹交叉问题就不满足连续介质假定 [112]。

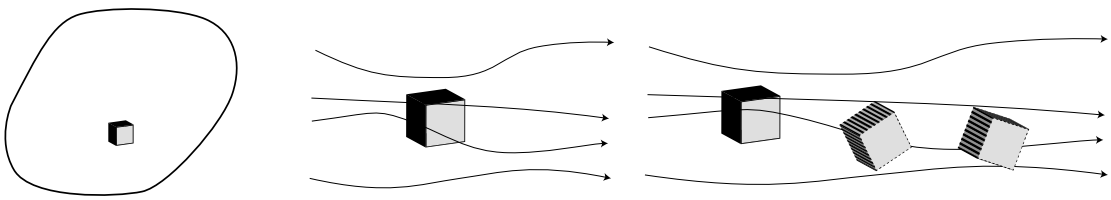


图 3.2: 左图：有限控制体（大圈）和无穷小微团（小正方体）。中图：空间位置固定的无穷小微团，流体流入微团后继续流出。右图：空间位置移动的无穷小微团，其沿着流体的流线进行移动。

用实验的方法检测这 N 个位置的速度，我们会得到一个更精确的速度场。实际上，房间内的 N 块，即可以近似的理解为 CFD 中网格的概念。CFD 要做的，就是通过计算机，求解这些划分的小块上的物理量。

总结：正因为无穷小微团的各个属性可以用数学中的 d... 来表示，同时有限控制体包含了大量的流体微团。因此从有限控制体推导出来的方程为积分形式，从无穷小微团推导出来的方程为微分形式。再次重申，在没有讨论方程的时候，来理解积分形式的 NS 方程以及微分形式的方程是很困难的。回过头来看此内容会更加简单。

3.3 连续性方程

CFD 的控制方程无论具有什么形式，都是建立在流体力学基本控制方程：连续性方程、动量方程、能量方程的基础之上[5]。他们主要来源于质量守恒定律、牛顿第二定律、能量守恒定律。N-S 方程最基本的假设为：把流体看做是连续介质。当从宏观尺度来分析的时候（比如一微米），流体的随机分子运动可以忽略。这样，就可以描述流体的一些宏观物理量如速度、压力、密度和温度等。因为这些物理量可以看做大量分子的平均。

3.3.1 微分/导数形式

连续性方程可以在上文中的 4 种流动模型基础上进行推导而来，本节从空间位置固定的无穷小微团角度出发，对连续性方程进行推导。这种推导形式是最容易理解的，只涉及到高等数学中的泰勒方程。从空间位置固定的角度来分析的方法，通常被称之为欧拉观点。欧拉观点下的质量守恒意味着位置固定无穷小微团质量的变化 = 流入无穷小微团质量 − 流出无穷小微团质量。也即为：位置固定无穷小微团质量的变化率 = (流入无穷小微团质量 − 流出无穷小微团质量) 的变化率。图3.3为一个质量守恒的实例。

下面首先看无穷小微团的质量的变化率。如上文所述，流体微团的体积可以表示为 $\mathrm{d}V = \mathrm{d}x\mathrm{d}y\mathrm{d}z$。同时，有密度 ρ，则对应的无穷小微团质量为

$$\mathrm{d}m = \rho \mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.5}$$

[5]再一次强调，这里假定流动为平衡流，即分子碰撞是频繁的。

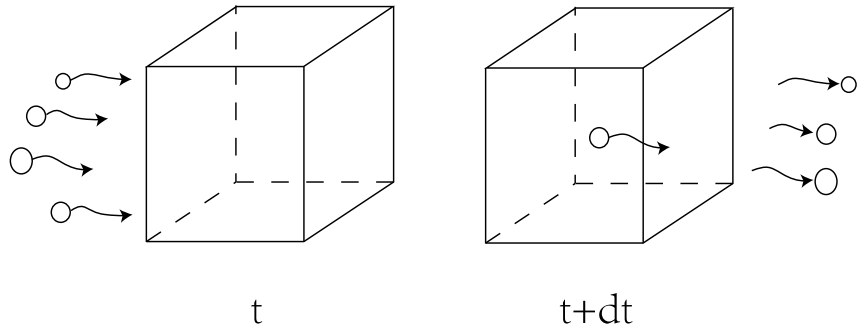


图 3.3: 质量守恒实例。在 t 时刻，有 4 单位的流体要流入无穷小微团（左图）。在 $t+\mathrm{d}t$ 时刻，有 3 单位的流体流出了无穷小微团（右图）。那么这个无穷小微团的变化，即为增加了 1 单位的流体（右图无穷小微团内增加 1 单位流体）。

依据质量变化率的定义（质量的变化除以时间的变化），其可以表示为

$$\frac{\partial \mathrm{d}m}{\partial t} = \frac{\partial \rho \mathrm{d}x\mathrm{d}y\mathrm{d}z}{\partial t} = \frac{\partial \rho}{\partial t}\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.6}$$

下面考虑无穷小微团的流入和流出。在这里介绍一下泰勒方程的另一个形式：如果对于一个函数，x_0 点的值为 $f(x_0)$，那么 x_0+h 的值可以通过下述方程求得[6]：

$$f(x_0+h) = f(x_0) + f'(x_0)h + \mathcal{O}(h) \approx f(x_0) + f'(x_0)h \tag{3.7}$$

现在看图3.4中的无穷小微团，定义立方体左侧单位面积的质量流量为 ρu，其表示单位时

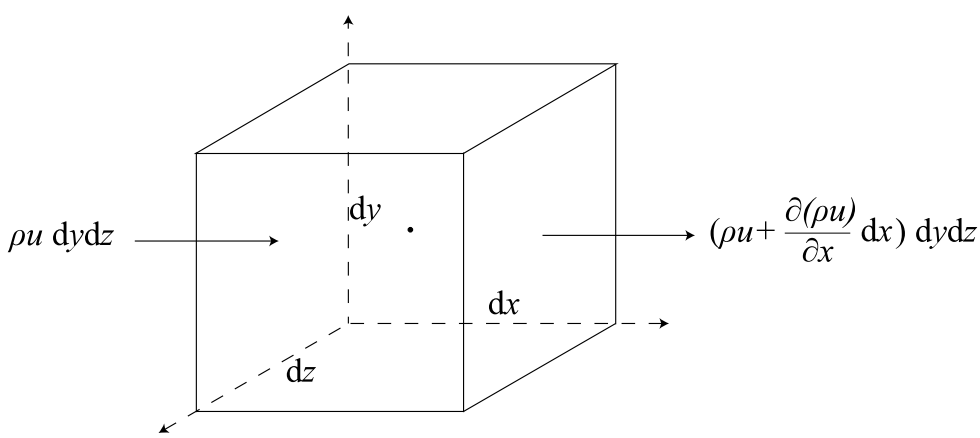


图 3.4: 无穷小微团 x 方向的通量。

间内流入单位面积的质量（注意其中的单位时间和单位面积）。同时立方体左侧的面积为 $\mathrm{d}y\mathrm{d}z$，因此有单位时间内流入的质量为

$$\rho u \mathrm{d}y\mathrm{d}z \tag{3.8}$$

[6]在这里假定函数是连续可微的。然而对于某些特殊问题这种假定并不成立（例如存在激波的问题）。这种情况下需要从积分的角度推导连续性方程（参见下节）。

对于立方体右侧的面，在已知立方体左侧面 ρu 的定义的时候，右侧可以通过泰勒方程求出，即单位时间内流出立方体右侧的单位面积的质量为 $\rho u + \frac{\partial \rho u}{\partial x}\mathrm{d}x$。同样的，立方体右侧的面的面积为 $\mathrm{d}y\mathrm{d}z$，因此有单位时间内流出的质量约为[7]

$$\left(\rho u + \frac{\partial \rho u}{\partial x}\mathrm{d}x\right)\mathrm{d}y\mathrm{d}z \tag{3.9}$$

结合无穷小微团单位时间的流出质量，即方程(3.9)，和无穷小微团单位时间的流入质量，即方程(3.8)，有单位时间的 x 方向的净质量变化率为

$$\rho u\mathrm{d}y\mathrm{d}z - \left(\rho u + \frac{\partial \rho u}{\partial x}\mathrm{d}x\right)\mathrm{d}y\mathrm{d}z = -\frac{\partial \rho u}{\partial x}\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.10}$$

同理，有 y 方向的净质量变化率为

$$-\frac{\partial \rho v}{\partial y}\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.11}$$

z 方向的净质量变化率为

$$-\frac{\partial \rho w}{\partial z}\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.12}$$

将各个方向的质量变化率加和有

$$-\frac{\partial \rho u}{\partial x}\mathrm{d}x\mathrm{d}y\mathrm{d}z - \frac{\partial \rho v}{\partial y}\mathrm{d}x\mathrm{d}y\mathrm{d}z - \frac{\partial \rho w}{\partial z}\mathrm{d}x\mathrm{d}y\mathrm{d}z = -\left(\frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.13}$$

同时，结合方程(3.6)，有

$$\frac{\partial \rho}{\partial t}\mathrm{d}x\mathrm{d}y\mathrm{d}z = -\left(\frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.14}$$

也即为连续性方程：

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0 \tag{3.15}$$

方程(3.15)在 CFD 中至关重要，其直接关系到压力泊松方程的导出。这里应尝试去理解连续性方程推导的本源，即连续性方程表示的是质量守恒。

3.3.2 积分形式

首先提示，若本节难以理解可跳过本节且不失去连续性。

上文中从空间位置固定的无穷小微团推导的连续性方程是最容易理解的。下面考虑从空间位置固定有限控制体推导的连续性方程。本节涉及到高等数学的散度定律。本节只不过是另一种推导连续性方程的途径。要提前强调的是，有限控制体和无穷小微团的关系可以理解为有限控制体包含着很多的无穷小微团（如图3.2左图所示）。因此，有限控制体中

[7] 其中忽略了无穷小量 $\mathcal{O}(\mathrm{d}x)$。

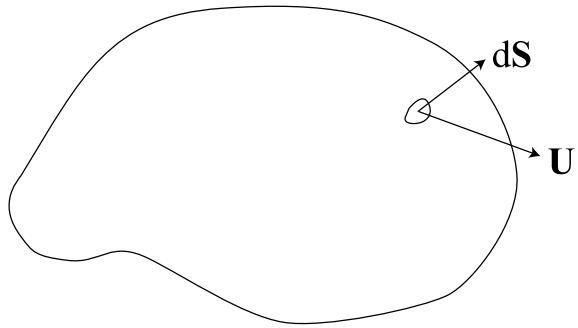


图 3.5: 有限控制体上的速度和面矢量。

的量都需要用积分来进行计算。对于空间位置固定的有限控制体，其质量可通过下面的方程进行计算：

$$\int \rho \mathrm{d}V \tag{3.16}$$

其实际上就是高等数学中，质量的计算方程。其对应的质量变化率为

$$\frac{\partial}{\partial t}\int \rho \mathrm{d}V \tag{3.17}$$

下面考虑空间位置固定的有限控制体由于流入流出引起的质量变化。图3.5中的通量可以定义为

$$\rho \mathbf{U}\cdot \mathrm{d}\mathbf{S} \tag{3.18}$$

那么整个有限控制体上的净通量可以表示为

$$-\int_{\mathbf{S}} \rho \mathbf{U}\cdot \mathrm{d}\mathbf{S} \tag{3.19}$$

其中负号的引入是因为考虑的为流入减去流出的通量（而不是流出减去流入）[8]。结合方程(3.19)和方程(3.17)，有

$$\frac{\partial}{\partial t}\int_V \rho \mathrm{d}V + \int_{\mathbf{S}} \rho \mathbf{U}\cdot \mathrm{d}\mathbf{S} = 0 \tag{3.20}$$

也即[9]

$$\frac{\partial \rho}{\partial t} + \nabla\cdot \rho \mathbf{U} = 0 \tag{3.21}$$

方程(3.21)即为通过空间位置固定的有限控制体推导的连续性方程，其和(3.15)是等价的。在这里需要注意的是，从方程(3.20)推导至(3.21)的时候，需要假定函数是光滑的连续的。因此积分形式的方程(3.20)是比微分形式的方程(3.21)更基础的控制方程。

[8] 流入的通量为负。

[9] 其中要调用高斯定律 $-\int \rho \mathbf{U}\cdot \mathrm{d}\mathbf{S} = -\int \nabla\cdot \rho \mathbf{U}\mathrm{d}V$

方程(3.20)是一种形式的积分形式连续性方程。对其对时间做积分，有另一种形式的积分形式的连续性方程[10]：

$$\int_t^{t+\Delta t} \frac{\partial}{\partial t} \int_V \rho \mathrm{d}V \mathrm{d}t + \int_t^{t+\Delta t} \int_{\mathbf{S}} \rho \mathbf{U} \cdot \mathrm{d}\mathbf{S} \mathrm{d}t' = 0 \tag{3.22}$$

$$\int_V \rho(t+\Delta t)\mathrm{d}V - \int_V \rho(t)\mathrm{d}V = -\int_t^{t+\Delta t} \left(\int_{\mathbf{S}} \rho(t')\mathbf{U}(t') \cdot \mathrm{d}\mathbf{S} \right) \mathrm{d}t' \tag{3.23}$$

方程(3.23)的左边表示在体积 V 上，时间间隔 Δt 上质量的变化，右侧则考虑通量的变化形式，同样表示体积 V 上，时间间隔 Δt 上质量的变化。方程(3.23)与(3.20)是相同的，且是精准的。如果知道方程(3.23)右侧 $\rho(t')\mathbf{U}(t')$ 的准确值，那么结果就是精准的。方程(3.23)主要用于双曲系统来推导高阶通量格式（如 HLL 格式，其通过 V 将黎曼扇区域进行包围并假定 $\rho(t')\mathbf{U}(t')$ 的近似值即可获得网格面上的通量）。

3.3.3 通量与速度散度

在这里有必要介绍通量的概念。因为初学者经常将对流项引申的量理解为通量。然而 CFD 控制方程中离散后很多项都可以称之为通量，比如对流项 $\nabla \cdot (\mathbf{UU})$ 在做线性化后会出现对流通量。扩散项 $\nabla \cdot (\mu \nabla \mathbf{U})$ 离散后同样会出现扩散通量。进一步的，通量可区分为体积通量和质量通量。体积通量的物理意义为单位时间内流经某个网格单元面流体的体积，单位为 m^3/s。质量通量的物理意义为单位时间内流经某个网格单元面流体的质量，单位为 kg/s。不同的场合通常选取不同的通量。对于不可压缩流体，通常选取体积通量，对于可压缩流体，通常选取质量通量。

从数学的角度来理解更为简单：速度 $\mathbf{U}$ 乘以面积 $\mathbf{S}$ 就是体积通量，即每单位时间流经某个网格单元面多少立方米的流体：

$$\phi_f = \mathbf{U}_f \cdot \mathbf{S}_f \tag{3.24}$$

其中 ϕ_f 表示网格单元面 $\mathbf{S}_f$ 上的通量，$\mathbf{S}_f$ 也被称之为面矢量，$\mathbf{U}_f$ 为定义在网格单元面上的速度。在 ϕ_f 的基础上，除以网格面积，为每单位时间每单位面积流经的体积：

$$\frac{\phi_f}{|\mathbf{S}_f|} = \mathbf{U}_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} = \mathbf{U}_f \cdot \mathbf{n}_f \tag{3.25}$$

图3.6中左中右三个网格单元的最上方的网格单元面对应的三个面矢量，方向相同，大小不同。面矢量 $\mathbf{S}_f$ 的大小可通过下式计算：

$$|\mathbf{S}_f| = \sqrt{S_x^2 + S_y^2 + S_z^2} \tag{3.26}$$

其中 S_x, S_y, S_z 表示面矢量 $\mathbf{S}_f$ 的分量。类似的，$\rho_f \mathbf{U}_f \cdot \mathbf{S}_f$ 则表示每单位时间经过某个网格

[10] 右侧的时间 t 以及 $t+\Delta t$ 表示积分限，为固定值，t' 才是真正的积分变量，即时间

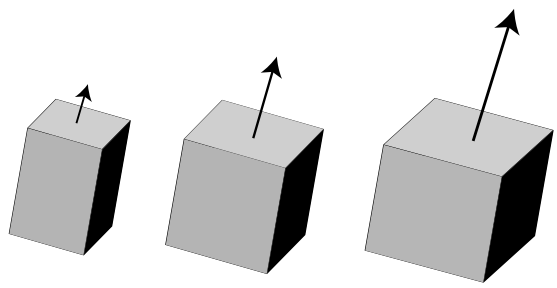


图 3.6: 上图中表示了相同方向，不同大小（箭头长度不同）的面矢量。

面多少千克的流体。

CFD 中另外一个重要的概念是面法向矢量 $\mathbf{n}_f$，其用于定义面的法向，面法向矢量的特点是其模均为 1，计算公式为：

$$\mathbf{n}_f = \frac{\mathbf{S}_f}{|\mathbf{S}_f|} \tag{3.27}$$

如图3.7所示，在一个网格的边界区域，通常由若干个网格单元面构成，每个网格单元面则分别对应一个面矢量和面法向矢量。

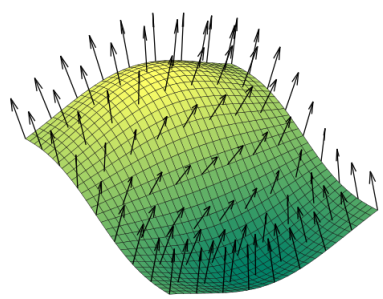


图 3.7: 任意的弯曲几何的面法向矢量。

除了边界面，网格的内部面也存在面矢量。图3.8表示为网格内部面的面矢量定义。图中的 $\mathbf{S}$ 为两个网格连接面的矢量，其垂直于网格面，大小等于面的面积。由宿主网格（图中 P 点）指向相邻网格（图中 N 点）。同时可以看出，连接 P 网格体心和 N 网格体心的矢量方向和 $\mathbf{S}_f$ 的方向并不相同。这会导致梯度的计算差异，也即 CFD 中的非正交修正算法，在此不做介绍。回到通量的计算公式，有：

$$\phi_f = \mathbf{U}_f \cdot \mathbf{S}_f = |\mathbf{U}_f||\mathbf{S}_f|\cos\theta \tag{3.28}$$

其中 θ 为面矢量和面速度矢量的夹角。很明显通量的大小取决于夹角的大小。图3.9形象的表示了这一特性。左图中速度矢量和面矢量方向平行，则其中的虚线表示单位时间内通过

面的体积。中图中的速度矢量和面矢量存在一定的夹角，因此体积有所减小。右图中速度矢量和面矢量垂直，因此没有任何流体通过这个面，因此通量为零。

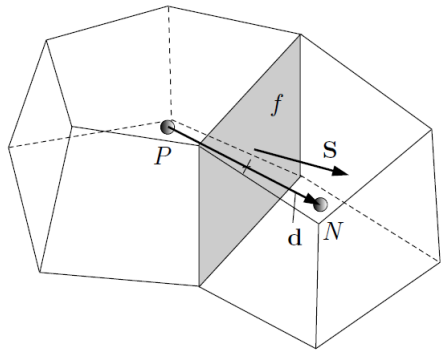


图 3.8: 网格单元内部面面矢量与网格体心点连线示意图。

附加几点有关通量的总结和评价：

- 不仅仅对流存在通量，扩散也可以用通量来表示[11]；
- 通量的计算不仅和面的大小以及速度的大小有关，并且和面的方向以及速度的方向有很大关系；
- 有限体积法中的守恒表示为通量的守恒，然而通量上的面速度和网格单元体心处的速度是有区别的；
- 本文中面速度均为给定，但是在 CFD 计算中，一些通量需要从体心速度插值而来，一些通量需要通过压力速度耦合算法计算而来；
- 对于图3.8中的非矩形网格以及高偏斜网格，需要对应的非正交修正算法，或者采用最小二乘法计算梯度；

下面介绍速度散度的物理意义，需要一些高等数学中积分的内容，若难以理解可不失去连续性的略过后续本节内容[12]。在考虑体积通量 ϕ_f（单位时间内流经某个面的流体体积）

[11] 扩散通量也可以这样理解，通常用 D 表示扩散系数，那么，每单位时间每单位面积由于扩散引致的通量可以表示为：

$$\mathbf{J} = -D\nabla\phi$$

要注意的是，扩散通量为一个矢量，因为在一些情况下，各个方向由于扩散系数的不同，会导致呈现各向异性扩散。在一般情况下，各个方向的梯度也不相同，因此扩散通量为一个分量不同的矢量。同时也可以看出，扩散通量的驱动因素是变量的梯度。类似的，每单位时间由于扩散引致的通量可以表示为：

$$\mathbf{J}_f \cdot \mathbf{S}_f = -D(\nabla\phi)_f \cdot \mathbf{S}_f = -D(\nabla\phi)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|}|\mathbf{S}_f|$$

其中 $(\nabla\phi)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|}$ 表示面法向梯度。上述公式还可以整理为：

$$(\nabla\phi)_f \cdot \mathbf{n}_f = -\frac{\mathbf{J}_f \cdot \mathbf{n}_f}{D}$$

上述公式通常可以用于施加边界条件。

[12] 但3.3.4节以此为基础展开。

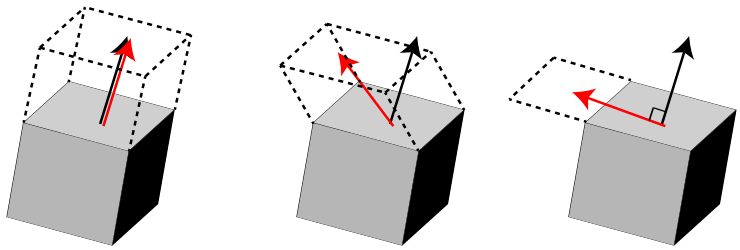


图 3.9: 不同速度矢量对应的通量大小，其中黑色箭头表示面矢量，红色箭头表示速度矢量。

的物理意义的基础上。参考图3.9，如果在体积通量的基础上乘以时间 Δt，则表示体积为 V 的流动控制体，由于其中的面 $\mathrm{d}\mathbf{S}$ 以 $\mathbf{U}$ 的速度移动，在 Δt 时间内体积的变化量：

$$\mathrm{d}V_f = \phi_f \Delta t = \mathbf{U}_f \cdot \mathrm{d}\mathbf{S}_f \Delta t \tag{3.29}$$

因此，整个流动控制体的体积变量需要在方程(3.29)的基础上做积分：

$$\Delta V = \int_V \mathrm{d}V = \int_{\mathbf{S}} \mathbf{U}_f \cdot \mathrm{d}\mathbf{S}_f \Delta t \tag{3.30}$$

体积 V 的物质导数定义为

$$\frac{\mathrm{D}V}{\mathrm{D}t} = \frac{\Delta V}{\Delta t} = \int_{\mathbf{S}} \mathbf{U}_f \cdot \mathrm{d}\mathbf{S}_f \tag{3.31}$$

依据高等数学的散度定律有

$$\int_{\mathbf{S}} \mathbf{U}_f \cdot \mathrm{d}\mathbf{S}_f = \int_V \nabla \cdot \mathbf{U} \mathrm{d}V \tag{3.32}$$

即

$$\frac{\mathrm{D}V}{\mathrm{D}t} = \int_V \nabla \cdot \mathbf{U} \mathrm{d}V \tag{3.33}$$

方程(3.33)对体积 V 做积分。如果考虑体积 V 足够小，即 $V \to \delta V$，方程(3.33)在此 δV 可以表示为：

$$\frac{\mathrm{D}(\delta V)}{\mathrm{D}t} = \int_{\delta V} (\nabla \cdot \mathbf{U}) \mathrm{d}V \tag{3.34}$$

由于体积 δV 足够小，$\nabla \cdot \mathbf{U}$ 在此 δV 上都相等可以提出，即

$$\int_{\delta V} (\nabla \cdot \mathbf{U}) \mathrm{d}V = \nabla \cdot \mathbf{U} \delta V \tag{3.35}$$

结合方程(3.34)和(3.35)有：

$$\nabla \cdot \mathbf{U} = \frac{1}{\delta V} \frac{\mathrm{D}\delta V}{\mathrm{D}t} \tag{3.36}$$

即速度散度的物理意义：每单位体积流动着的控制体体积随时间的变化率。

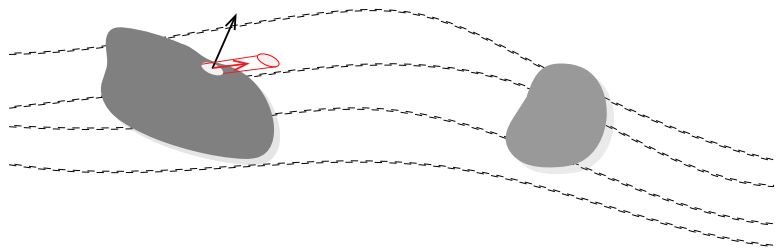


图 3.10: 流动的控制体由于速度的变化导致形状体积的改变，左图中面 d**S** 以 **U** 的速度移动，在 Δt 时间间隔内引起的体积变化为圆柱体体积。右图为最终的控制体形状。

3.3.4 拉格朗日观点

首先提示，若本节难以理解可跳过本节且不失去连续性。

上文是从空间位置固定的有限控制体以及无穷小微团下推导的连续性方程的。下面讨论另一种推导形式：从空间位置移动的无穷小微团推导的连续性方程。这种方式偏向于拉格朗日思想。在拉格朗日框架下，随流线运动无穷小微团质量的变化 $=0$。参考前文物质导数的定义，即

$$\frac{\mathrm{D}}{\mathrm{D}t}\mathrm{d}m = 0 \tag{3.37}$$

其中采用 $\mathrm{d}m$ 是因为跟踪的为无穷小微团。依据物质导数的定义，方程(3.37)可以化简为

$$\frac{\mathrm{D}}{\mathrm{D}t}\mathrm{d}m = \frac{\mathrm{D}\left(\rho \mathrm{d}V\right)}{\mathrm{D}t} = \rho\frac{\mathrm{D}\left(\mathrm{d}V\right)}{\mathrm{D}t} + \mathrm{d}V\frac{\mathrm{D}\rho}{\mathrm{D}t} = 0 \tag{3.38}$$

即

$$\frac{\rho}{\mathrm{d}V}\frac{\mathrm{D}\left(\mathrm{d}V\right)}{\mathrm{D}t} + \frac{\mathrm{D}\rho}{\mathrm{D}t} = 0 \tag{3.39}$$

依据上一节介绍的物质导数和速度散度的物理意义，有

$$\frac{1}{\mathrm{d}V}\frac{\mathrm{D}\left(\mathrm{d}V\right)}{\mathrm{D}t} = \nabla\cdot\mathbf{U} \tag{3.40}$$

代入到方程(3.39)，有

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho\nabla\cdot\mathbf{U} = 0 \tag{3.41}$$

3.3.5 连续性方程小结

方程(3.15)、(3.21)以及(3.41)均表示连续性方程，但各自的推导采用了不同的流动模型。有的模型出发点和推导比较容易理解，有的方法比较复杂。这些方程可以相互的转换。且各种教材中书写并不统一。例如，下面的方程均为采用不同书写方式书写的连续性方程：

$$\frac{\partial\rho}{\partial t} + \nabla\cdot\left(\rho\mathbf{U}\right) = 0 \tag{3.42}$$

$$\frac{\partial\rho}{\partial t} + \mathrm{div}\left(\rho\mathbf{U}\right) = 0 \tag{3.43}$$

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \tag{3.44}$$

$\nabla\cdot$ 和 div 在笛卡尔网格下等价，均表示散度操作。方程(3.44)则是另一种书写方式。初学 CFD 可能对方程中的这些符号感到很生疏，在此记住下面最简单的连续性方程形式即可[13]：

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} + \frac{\partial \rho w}{\partial z} = 0 \tag{3.45}$$

3.4 动量方程

本节讨论动量方程。动量方程依然可以通过上面四种不同的流动模型进行推导。由于篇幅所限本节不再一一推导，仅介绍一种较为容易理解的推导方法，即从空间位置移动的无穷小微团进行分析。在推导之前，需要介绍一些基础的知识。

3.4.1 受力分析

对动量方程进行推导，就需要对无穷小微团进行受力分析。正是因为流体的受力，才会引致流体的流动。无穷小微团的受力可区分为体积力和表面力。表面力为作用在无穷小微团面上的力，如压力、表面张力，海洋表面上的风也可以认为是表面力。体积力作用在无穷小微团全部的体积上（不仅表面有，体积内也有），例如重力、电磁力，或者一切引起旋转的力（科氏力）。在这些力中，最重要的表面力为压力和应力（后续会对应力进行介绍），最重要的体积力为重力（若不考虑其他源项力）[14]。

首先看压力，如图3.11所示，CFD 中的压力用 p 来表示（其实际为压强），且这个压力表示静压。其作用主要导致流体的压缩和膨胀，而不是切应力引致的变形。另外，压力为一种正应力（不同于下文中要介绍的切应力），其永远作用于无穷小微团的面的垂直的方向。 同时，如图3.12所示，无穷小微团还受到剪切应力的作用。其主要体现为流体的粘

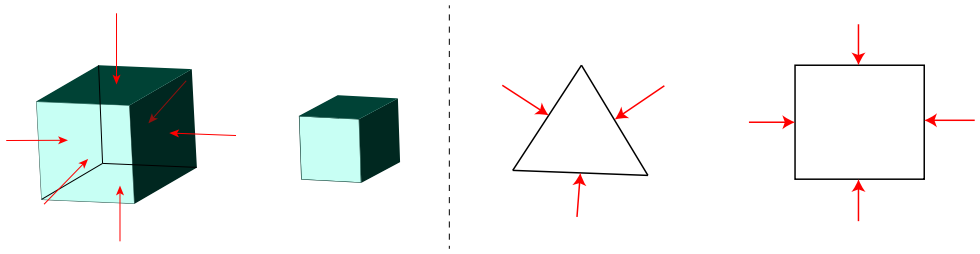


图 3.11: 左图：无穷小微团由于压力的作用导致体积的压缩。右图：三角形网格和四边形网格受到的垂直的压力。

性[15]。可以很明显的注意到，蜂蜜要比水要粘。这就是因为要获得同样大的形变率，施加

[13]注意，积分形式的方程才是更根本的方程。但由于其不太好理解，因此目前这个阶段不做讨论。

[14]CFD 中的一些体积力与表面力需要特殊处理，否则会引起数值震荡，感兴趣的读者可以阅读相关文献了解相关信息 [200, 201]。

[15]剪切应力也可以与摩擦力结合起来。宏观上可以理解的摩擦力，就是施加于流体/固体表面的剪切应力。正是因为剪切应力/摩擦力的存在，我们才可以握住物体。

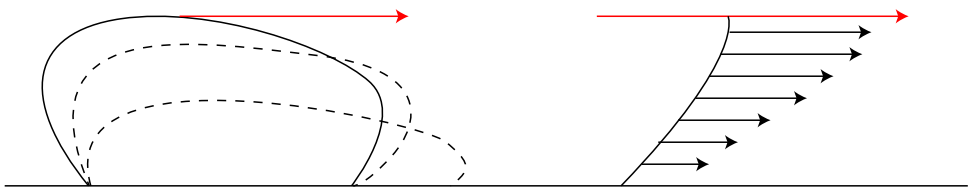


图 3.12: 左图：一块蜂蜜由于剪切应力的作用导致的变形。右图：流体由于上方剪切力作用导致的流速区别，壁面附近速度较小。红色的箭头表示剪切应力，黑色的箭头表示速度大小和方向。

给蜂蜜的剪切应力需要更大。剪切应力的作用导致流体产生形变。这个应力不是和形变成正比，而是和形变的速率成正比。也就是说，应力越大，形变的速度越快。

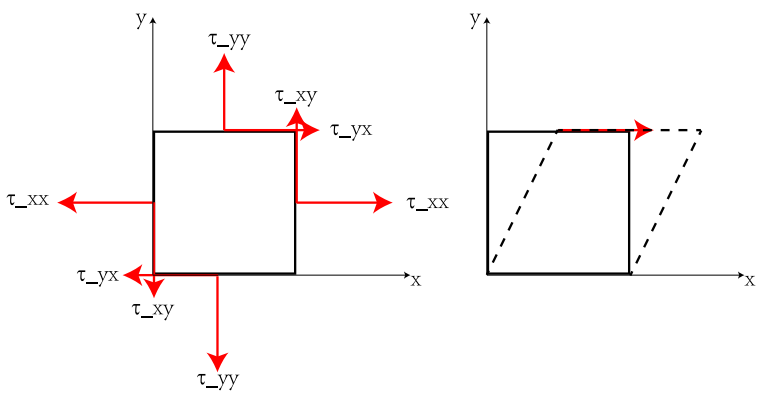


图 3.13: 左图：一个四边形无穷小微团所受的剪切应力。右图：一个四边形有限控制体受到剪切应力导致形变。

如图3.13左图所示，一个二维四边形无穷小微团所受的剪切应力分量分别为

$$\tau_{yy}, \tau_{yx}, \tau_{xx}, \tau_{xy} \tag{3.46}$$

其中的 τ 表示剪切应力，第一个下标表示作用于与某方向垂直的平面，第二个下标表示力的方向。比如 τ_{yy} 表示作用于和 y 垂直的平面的剪切应力分量，其方向为 y 方向。τ_{yx} 表示作用于和 y 垂直的平面的剪切应力分量，其方向为 x 方向。图3.13右图表示了流体由于剪切应力作用发生的形变。图3.14则是三维的无穷小微团受到的剪切应力。在三维的情况下，剪切应力的分量可以表示为

$$\tau_{xx}, \tau_{xy}, \tau_{xz}, \tau_{yx}, \tau_{yy}, \tau_{yz}, \tau_{zx}, \tau_{zy}, \tau_{zz} \tag{3.47}$$

其通常写成下面的形式：

$$\begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} \tag{3.48}$$

同时要注意，剪切应力张量为一个对称的量。只有其为对称的（净扭矩为零），才能防止流体自发产生旋转。

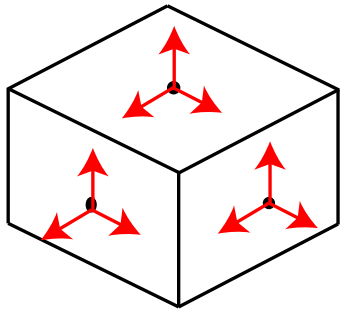


图 3.14: 三维剪切应力。

在这里，读者应该知道方程(3.48)中的剪切应力分量的定义是什么？分别作用在哪个面？哪个方向？其会引起什么样的物理过程。下面来分析一个具体的无穷小微团受到的力。图3.15表示施加在无穷小微团 x 方向上的全部表面力[16]。在面 abcd 上，仅仅存在由切应力引起的 x 方向的分量 $\tau_{yx}\mathrm{d}x\mathrm{d}z$。面 efgh 和面 abcd 的距离为 $\mathrm{d}y$，所以 efgh 面上的 x 方向的切应力为 $(\tau_{yx}+(\partial\tau_{yx}/\partial y)\mathrm{d}y)\mathrm{d}x\mathrm{d}z$。对于面 abcd 和面 efgh 上的切应力，要注意他们的方向。在底面，τ_{yx} 是向左的（与 x 轴方向相反），在顶面，$\tau_{yx}+(\partial\tau_{yx}/\partial y)\mathrm{d}y$ 是向右的（与 x 轴方向相同）。这与下面的约定是一致的：即速度的三个分量的正的增量与坐标轴的正向一致。例如，图3.15中的平面 efgh，因为 u 沿着 y 轴正向是增加的，所以在稍微高于平面 efgh 的地方，速度 u 要比平面 efgh 上的 u 大。于是就形成了“拉”的动作，试图将流体微团向右拉向 x 轴的正向。相反，若考虑平面 abcd，则在稍稍低于平面 abcd 的地方，速度 u 要比平面 abcd 上的 u 小。于是对流体形成了一个阻碍的作用，作用在 x 轴的负向。图3.15中其他剪切应力的方向，都可以用相同的方式进行判断。特别是在面 dcgh 上，τ_{zx} 指向 x 轴负向，而在面 abfe 上，$\tau_{zx}+(\partial\tau_{zx}/\partial z)\mathrm{d}z$ 指向 x 轴正向。在垂直于 x 轴的面 adhe 上，x 方向的力有压力 $p\mathrm{d}y\mathrm{d}z$，指向流体微团的内部；还有沿 x 轴负向的应力 $\tau_{xx}\mathrm{d}y\mathrm{d}z$。依据前面提到的速度增量方向的约定，我们可以解释为什么在图3.15中，面 adhe 上 τ_{xx} 是指向左边的。依据规定，速度 u 的正增量和 x 轴的正向一致，所以稍微离开面 adhe 左面一点点，u 的值比面 adhe 上的 u 要小。因此，正应力的粘性作用在面 adhe 上就好像是一个吸力，产生一个向左拉的作用，想要阻止流体微团的流动。与此相反，在面 bcgf 上，压力 $(p+(\partial p/\partial x)\mathrm{d}x)\mathrm{d}y\mathrm{d}z$ 指向流体微团的内部（沿着 x 轴负向）。而由于在稍微离开面 bcfg 右面一点点的地方，u 的值比面 bcfg 上的 u 要大，就会产生一个由正应力引起的吸力，将流体微团向右拉，这个力的大小为 $(\tau_{xx}+(\partial\tau_{xx}/\partial x))\mathrm{d}y\mathrm{d}z$，方向指向 x 轴正向。如果上面的描述同学们感觉难以理解，下面的结论要知悉。通过对流体微团 x 方向的受力做分析，有 x 方向所受到的表面力（压力和剪切应力贡献）为：

[16] 图3.15看起来是非常复杂的。如果感觉理解有困难，可以进一步参考文献 [209] 第 47 页的描述。

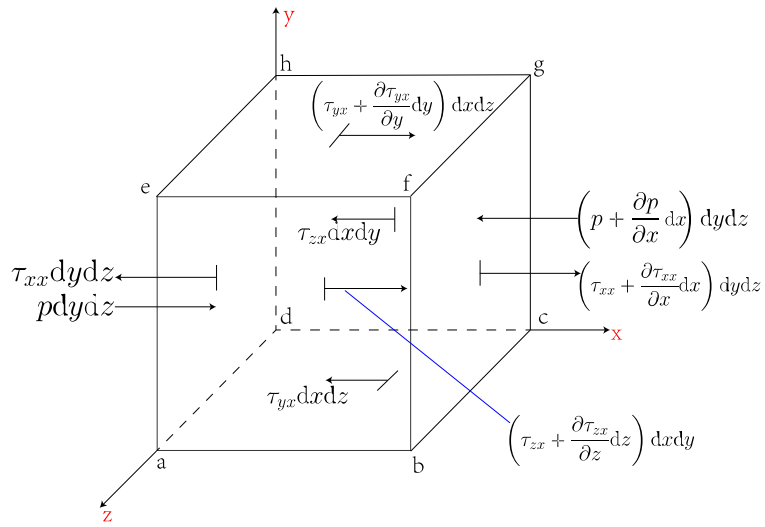


图 3.15: x 方向各个面受到的力。

$$
\begin{aligned}
F_x = &\left(p-\left(p+\frac{\partial p}{\partial x}\mathrm{d}x\right)\right)\mathrm{d}y\mathrm{d}z+\left(\left(\tau_{xx}+\frac{\partial\tau_{xx}}{\partial x}\mathrm{d}x\right)-\tau_{xx}\right)\mathrm{d}y\mathrm{d}z \\
&+\left(\left(\tau_{yx}+\frac{\partial\tau_{yx}}{\partial y}\mathrm{d}y\right)-\tau_{yx}\right)\mathrm{d}x\mathrm{d}z+\left(\left(\tau_{zx}+\frac{\partial\tau_{zx}}{\partial z}\mathrm{d}z\right)-\tau_{zx}\right)\mathrm{d}x\mathrm{d}y
\end{aligned}
\tag{3.49}
$$

即

$$
F_x=\left(-\frac{\partial p}{\partial x}+\frac{\partial\tau_{xx}}{\partial x}+\frac{\partial\tau_{yx}}{\partial y}+\frac{\partial\tau_{zx}}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.50}
$$

同理，有 y 方向和 z 方向：

$$
F_y=\left(-\frac{\partial p}{\partial y}+\frac{\partial\tau_{xy}}{\partial x}+\frac{\partial\tau_{yy}}{\partial y}+\frac{\partial\tau_{zy}}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.51}
$$

$$
F_z=\left(-\frac{\partial p}{\partial z}+\frac{\partial\tau_{xz}}{\partial x}+\frac{\partial\tau_{yz}}{\partial y}+\frac{\partial\tau_{zz}}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.52}
$$

方程(3.50)、(3.51)和(3.52)即为流体微团在三个方向上的受力（为一个矢量）：

$$
\mathbf{F}=\begin{bmatrix}\left(-\frac{\partial p}{\partial x}+\frac{\partial\tau_{xx}}{\partial x}+\frac{\partial\tau_{yx}}{\partial y}+\frac{\partial\tau_{zx}}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \\ \left(-\frac{\partial p}{\partial y}+\frac{\partial\tau_{xy}}{\partial x}+\frac{\partial\tau_{yy}}{\partial y}+\frac{\partial\tau_{zy}}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \\ \left(-\frac{\partial p}{\partial z}+\frac{\partial\tau_{xz}}{\partial x}+\frac{\partial\tau_{yz}}{\partial y}+\frac{\partial\tau_{zz}}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z\end{bmatrix} \tag{3.53}
$$

3.4.2 动量守恒

在分析完受力之后，回到牛顿第二定律，即力等于质量乘加速度。力已经在方程(3.53)中给出，现在来看质量和加速度。在上文中，已经知道流体微团的质量可以表示为 $\mathrm{d}m=$

$\rho \mathrm{d}x\mathrm{d}y\mathrm{d}z$，同时，流体微团的加速度为 $\frac{\mathrm{D}\mathbf{U}}{\mathrm{D}t}$（参考物质导数一节），其分量形式为

$$\left[\frac{\mathrm{D}u}{\mathrm{D}t}, \frac{\mathrm{D}v}{\mathrm{D}t}, \frac{\mathrm{D}w}{\mathrm{D}t}\right]^{\mathrm{T}} \tag{3.54}$$

那么有 x 方向的动量方程为

$$\rho \mathrm{d}x\mathrm{d}y\mathrm{d}z\frac{\mathrm{D}u}{\mathrm{D}t} = \left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z}\right)\mathrm{d}x\mathrm{d}y\mathrm{d}z \tag{3.55}$$

同理，y 方向和 z 方向可以类似导出。即

$$\rho\frac{\mathrm{D}u}{\mathrm{D}t} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \tag{3.56}$$

$$\rho\frac{\mathrm{D}v}{\mathrm{D}t} = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} \tag{3.57}$$

$$\rho\frac{\mathrm{D}w}{\mathrm{D}t} = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} \tag{3.58}$$

方程(3.56)、(3.57)和(3.58)即动量方程。同学们应该结合上文介绍的相关知识，将方程(3.56)、(3.57)和(3.58)左边项写成欧拉形式（参考物质导数一节）。然后，再将这三个分量形式的动量方程写成矢量形式（结合张量一节）[17]：

$$\rho\frac{\mathrm{D}\mathbf{U}}{\mathrm{D}t} = -\nabla p + \nabla\cdot\boldsymbol{\tau} \tag{3.59}$$

也即为

$$\rho\left(\frac{\partial \mathbf{U}}{\partial t} + \mathbf{U}\cdot\nabla\mathbf{U}\right) = -\nabla p + \nabla\cdot\boldsymbol{\tau} \tag{3.60}$$

3.4.3 守恒/非守恒转化

方程(3.60)为动量方程的非守恒形式。这是因为上述动量方程是从空间位置移动的无穷小微团进行推导的。非守恒控制方程可通过结合连续性方程转化为守恒的控制方程。将连续性方程(3.42)左右乘以 $\mathbf{U}$ 有

$$\mathbf{U}\frac{\partial \rho}{\partial t} + \mathbf{U}\nabla\cdot(\rho\mathbf{U}) = 0 \tag{3.61}$$

随后将方程(3.61)与(3.60)相加有

$$\mathbf{U}\frac{\partial \rho}{\partial t} + \mathbf{U}\nabla\cdot(\rho\mathbf{U}) + \rho\frac{\partial \mathbf{U}}{\partial t} + \rho\mathbf{U}\cdot\nabla\mathbf{U} = -\nabla p + \nabla\cdot\boldsymbol{\tau} \tag{3.62}$$

即守恒形式的动量方程：

$$\frac{\partial \rho\mathbf{U}}{\partial t} + \nabla\cdot(\rho\mathbf{U}\mathbf{U}) = -\nabla p + \nabla\cdot\boldsymbol{\tau} \tag{3.63}$$

[17] 另一种常见的写法为

$$\rho\frac{\mathrm{D}\mathbf{U}}{\mathrm{D}t} = \nabla\cdot\sigma$$

其中 $\sigma = -p\mathbf{I} + \boldsymbol{\tau}$。在这种情况下 σ 考虑的为总应力（剪切应力 τ 和正应力 p 的和）。

连续性方程可以进行类似的转化。考虑从空间位置移动的无穷小微团进行推导的连续性方程(3.41)，依据物质导数的定义将其展开有

$$\frac{\mathrm{D}\rho}{\mathrm{D}t}+\rho\nabla\cdot\mathbf{U}=\frac{\partial\rho}{\partial t}+\mathbf{U}\cdot\nabla\rho+\rho\nabla\cdot\mathbf{U}=\frac{\partial\rho}{\partial t}+\nabla\cdot\rho\mathbf{U}=0 \tag{3.64}$$

其中 $\frac{\mathrm{D}\rho}{\mathrm{D}t}+\rho\nabla\cdot\mathbf{U}$ 为非守恒形式，$\frac{\partial\rho}{\partial t}+\nabla\cdot\rho\mathbf{U}$ 为守恒形式。

同理，守恒形式的方程，在假定原始变量光滑的情况下，可以演变为非守恒形式。

3.4.4 封闭

方程(3.56)、(3.57)和(3.58)还不能求解，因为其中的未知量为方程左边的速度，但方程右边还存在未知量 $\boldsymbol{\tau}$，这种待求未知量之外还存在未知量的情况，被称之为封闭问题。为了将其封闭，需要将 $\boldsymbol{\tau}$ 表示为 $\mathbf{U}$ 的函数。剪切应力 $\boldsymbol{\tau}$ 和速度 $\mathbf{U}$ 的关系，被称之为本构关系。他们的方程，被称之为本构方程。17 世纪末牛顿指出，流体的剪切应力和速度的梯度成线性关系。这种流体因此也被称之为牛顿流体。还存在一些不符合牛顿流体定义的流体，被定义为非牛顿流体，会展现一些不常见的流动特性。

下面讨论一个新的 CFD 变量：形变率，其也被称为剪切速率。未知量 $\boldsymbol{\tau}$ 可以表示为形变率的函数，形变率进一步可以表示为速度的函数。这样，方程(3.56)、(3.57)和(3.58)就可以封闭。形变率和剪切应力类似，是一个二阶张量，一般用 $\mathbf{S}$ 来表示。但由于在 CFD 中，形变率本身并不会封闭方程（而是形变率和速度的关系封闭了方程），因此在 CFD 方程中往往不会出现形变率这一项[18]。若 $\mathbf{S}$ 符号表示形变率，即：

$$\mathbf{S}=\begin{bmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{bmatrix} \tag{3.65}$$

对于各向同性流体，形变率的 9 个分量有 6 个是独立的：

$$S_{xy}=S_{yx}, S_{xz}=S_{zx}, S_{yz}=S_{zy} \tag{3.66}$$

当然也存在一些各向异性流体，如一些聚合物。对这些流体的研究会导致问题更加复杂。当然，很多模型都是从各向同性假定出发的。不可压缩流体的形变率进一步的可以表示为速度的变量：

$$\mathbf{S}=\begin{bmatrix} \frac{1}{2}\left(\frac{\partial u_1}{\partial x}+\frac{\partial u_1}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u_1}{\partial y}+\frac{\partial u_2}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u_1}{\partial z}+\frac{\partial u_3}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial u_2}{\partial x}+\frac{\partial u_1}{\partial y}\right) & \frac{1}{2}\left(\frac{\partial u_2}{\partial y}+\frac{\partial u_2}{\partial y}\right) & \frac{1}{2}\left(\frac{\partial u_2}{\partial z}+\frac{\partial u_3}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial u_3}{\partial x}+\frac{\partial u_1}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial u_3}{\partial y}+\frac{\partial u_2}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial u_3}{\partial z}+\frac{\partial u_3}{\partial z}\right) \end{bmatrix} \tag{3.67}$$

[18] $\mathbf{S}$ 这个符号在有限体积法中更倾向于被定义为面矢量。

若写成矢量形式，有[19]：

$$\mathbf{S}=\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}}{2} \tag{3.68}$$

对于牛顿流体，其定义剪切应力和形变率为线性的关系。进一步考虑不可压缩流体（忽略体膨胀率）[20]：

$$\boldsymbol{\tau}=2\mu\mathbf{S}=\mu\left(\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}\right) \tag{3.69}$$

其中 μ 是粘度，其为一个物理属性。将方程(3.69)代入到方程(3.63)中，其中的 $\boldsymbol{\tau}$ 就被速度表示出来进而被封闭（其中的密度 ρ 被除掉）：

$$\frac{\partial\mathbf{U}}{\partial t}+\nabla\cdot(\mathbf{U}\mathbf{U})=-\frac{1}{\rho}\nabla p+\nabla\cdot\left(\nu\left(\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}\right)\right) \tag{3.70}$$

其中 $\nu=\frac{\mu}{\rho}$。如果 $\nu=0$，也即常粘度，方程(3.70)中的 $\nabla\cdot\left(\nu\left(\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}\right)\right)$ 在大量文献中被写为 $\nabla\cdot(\nu\nabla\mathbf{U})$，下面我们来看一下为什么。首先有[21]

$$\nabla\mathbf{U}=\begin{bmatrix}\frac{\partial u_1}{\partial x} & \frac{\partial u_2}{\partial x} & \frac{\partial u_3}{\partial x}\\ \frac{\partial u_1}{\partial y} & \frac{\partial u_2}{\partial y} & \frac{\partial u_3}{\partial y}\\ \frac{\partial u_1}{\partial z} & \frac{\partial u_2}{\partial z} & \frac{\partial u_3}{\partial z}\end{bmatrix} \tag{3.71}$$

对其转置操作有

$$\nabla\mathbf{U}^{\mathrm{T}}=\begin{bmatrix}\frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} & \frac{\partial u_1}{\partial z}\\ \frac{\partial u_2}{\partial x} & \frac{\partial u_2}{\partial y} & \frac{\partial u_2}{\partial z}\\ \frac{\partial u_3}{\partial x} & \frac{\partial u_3}{\partial y} & \frac{\partial u_3}{\partial z}\end{bmatrix} \tag{3.72}$$

然后求散度有[22]

$$\begin{aligned}&\nabla\cdot(\nu\nabla\mathbf{U}^{\mathrm{T}})=\nu\nabla\cdot(\nabla\mathbf{U}^{\mathrm{T}})=\\ &\nu\nabla\cdot\begin{bmatrix}\frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} & \frac{\partial u_1}{\partial z}\\ \frac{\partial u_2}{\partial x} & \frac{\partial u_2}{\partial y} & \frac{\partial u_2}{\partial z}\\ \frac{\partial u_3}{\partial x} & \frac{\partial u_3}{\partial y} & \frac{\partial u_3}{\partial z}\end{bmatrix}=\nu\begin{bmatrix}\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial x}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_3}{\partial x}\right)\\ \frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial y}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_3}{\partial y}\right)\\ \frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial z}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial z}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_3}{\partial z}\right)\end{bmatrix}=0\end{aligned} \tag{3.73}$$

这是因为对于任何一个标量 a，有

$$\frac{\partial}{\partial y}\left(\frac{\partial a}{\partial x}\right)=\frac{\partial}{\partial x}\left(\frac{\partial a}{\partial y}\right) \tag{3.74}$$

---

[19] 方程(3.67)若按项对应应为 $\mathbf{S}=\frac{1}{2}\left(\nabla\mathbf{U}^{\mathrm{T}}+\nabla\mathbf{U}\right)$。

[20] 可压缩流体完整形式为 $\tau=\mu\left(\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}\right)-\lambda(\nabla\cdot\mathbf{U})\mathbf{I}$，其中 λ 为体积粘度，也被称之为第二粘度系数等。Stokes 认为 $\lambda=\frac{2}{3}\mu$，进一步，$\nabla\cdot\mathbf{U}=0$，因此可以化简。

[21] 值得注意的是 $\nabla\mathbf{U}$ 的定义并不唯一，在流体力学中通常采取本文的定义。

[22] 以 x 分量为例，有连续性方程

$$\frac{\partial u_1}{\partial x}+\frac{\partial u_2}{\partial y}+\frac{\partial u_3}{\partial z}=0$$

则

$$\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial x}\right)+\frac{\partial}{\partial z}\left(\frac{\partial u_3}{\partial x}\right)=\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}\right)+\frac{\partial}{\partial x}\left(\frac{\partial u_2}{\partial y}\right)+\frac{\partial}{\partial x}\left(\frac{\partial u_3}{\partial z}\right)=\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}+\frac{\partial u_2}{\partial y}+\frac{\partial u_3}{\partial z}\right)=0$$

因此，动量方程(3.70)可写为

$$\frac{\partial \mathbf{U}}{\partial t}+\nabla\cdot(\mathbf{U}\mathbf{U})=-\frac{1}{\rho}\nabla p+\nabla\cdot(\nu\nabla\mathbf{U}) \tag{3.75}$$

这个封闭的方程若展开，为 3 个方程，且存在未知项速度 $\mathbf{U}$ 和压力 p 共 4 个未知量，即可求解。

对于可压缩流体，结合方程(3.69)的脚注，有完整的动量方程为

$$\frac{\partial \rho\mathbf{U}}{\partial t}+\nabla\cdot(\rho\mathbf{U}\mathbf{U})=-\nabla p+\nabla\cdot\left(\mu\left(\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}\right)-\frac{2}{3}\mu\left(\nabla\cdot\mathbf{U}\right)\mathbf{I}\right) \tag{3.76}$$

注意其中的 $\nabla\mathbf{U}^{\mathrm{T}}$ 并不能化简，因为可压缩流体的 $\nabla\cdot\mathbf{U}\neq 0$[23]。对于粘度可变的流体，$\nabla\cdot(\nu\nabla\mathbf{U}^{\mathrm{T}})$ 可以进一步的写为：

$$\nabla\cdot(\nu\nabla\mathbf{U}^{\mathrm{T}})=\nu\nabla(\nabla\cdot\mathbf{U})+\nabla\nu\cdot\nabla\mathbf{U} \tag{3.77}$$

在开源 CFD 软件 OpenFOAM 的代码中，选择植入 $\nabla\cdot\left(\nu\left(\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}\right)\right)$ 的形式。

3.4.5 积分观点

动量的定义为质量和速度的乘积，若将动量除以体积，即 ρu 表示为动量密度。考虑一维的区域，(x_1,x_2) 的区间范围内的动量可以表示为：

$$\int_{x_1}^{x_2}\rho u\mathrm{d}x \tag{3.78}$$

其变化率可表示为：

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2}\rho u\mathrm{d}x \tag{3.79}$$

(x_1,x_2) 内的动量变化率取决于流入和流出的通量。如何表示动量的通量？参考质量通量的定义，对于质量通量，其为密度和速度的乘积。那么动量的通量即为动量密度和速度的乘积，即 ρuu。另外，压力也会引致动量的变化，因此，$\rho uu+p$ 表示动量的通量，(x_1,x_2) 内流入减去流出的通量为

$$-\left(\rho uu+p\right)\big|_{x_1}^{x_2} \tag{3.80}$$

[23] 另外一种更紧凑的动量方程写法为：

$$\frac{\partial \rho\mathbf{U}}{\partial t}+\nabla\cdot(\rho\mathbf{U}\mathbf{U}+p\mathbf{I}-\rho\boldsymbol{\sigma})=0 \tag{1}$$

其中 $\boldsymbol{\sigma}$ 表示除掉密度后的应力张量 $\boldsymbol{\tau}/\rho$。如果考虑多相流体系，粒子相的方程与方程(1)具有类似的形式（方程右边需要添加动量交换项），这种方程形式更易于理解。例如，对于气固体系，有气相控制方程为：

$$\frac{\partial \alpha_g\rho_g\mathbf{U}_g}{\partial t}+\nabla\cdot(\alpha_g\rho_g\mathbf{U}_g\mathbf{U}_g+p_g\mathbf{I}-\alpha_g\rho_g\boldsymbol{\sigma}_g)=\alpha_g\rho_g\mathbf{g}-\alpha_p\rho_p\mathbf{M} \tag{2}$$

其中 α_g 表示气相相分数，$\mathbf{M}$ 表示动量交换项，$\mathbf{g}$ 表示重力加速度矢量。同时有颗粒相控制方程（在这里考虑颗粒粒子分布为各向异性高斯分布，即十矩模型，更详细的信息请参考4.3.3节）为：

$$\frac{\partial \alpha_p\rho_p\mathbf{U}_p}{\partial t}+\nabla\cdot(\alpha_p\rho_p\mathbf{U}_p\mathbf{U}_p+\mathbf{P}_p+\mathbf{G}_p+\mathbf{Z}_p)=\alpha_p\rho_p\mathbf{g}+\alpha_p\rho_p\mathbf{M} \tag{3}$$

其中 $\mathbf{P}_p$ 表示颗粒压力张量，其也可以分解为各向同性部分以及各向异性部分。各向同性部分为 $\Theta\mathbf{I}$，其中 Θ 表示颗粒温度。各向异性部分为 $\boldsymbol{\sigma}_p$，其也可以通过 Boussinesq 近似进行假定。$\mathbf{G}_p$ 表示碰撞通量，$\mathbf{Z}_p$ 表示摩擦通量。

结合方程(3.79)，有

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2}\rho u\mathrm{d}x=-\left(\rho uu+p\right)|_{x_1}^{x_2} \tag{3.81}$$

上式即为积分观点下推导的一维动量方程。

同样的，考虑质量，(x_1,x_2) 的区间范围内的质量可以表示为：

$$\int_{x_1}^{x_2}\rho\mathrm{d}x \tag{3.82}$$

其变化率为

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2}\rho\mathrm{d}x \tag{3.83}$$

质量的净通量可以表示为：

$$-\left(\rho u\right)|_{x_1}^{x_2} \tag{3.84}$$

因此，有连续性方程：

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2}\rho\mathrm{d}x=-\left(\rho u\right)|_{x_1}^{x_2} \tag{3.85}$$

现在尝试将积分形式转换为微分形式。进一步假定 $\rho uu+p$ 是光滑可导的，方程(3.81)可以写为：

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2}\rho u\mathrm{d}x=-\int_{x_1}^{x_2}\frac{\mathrm{d}}{\mathrm{d}x}(\rho uu+p)\mathrm{d}x \tag{3.86}$$

进一步操作有

$$\int_{x_1}^{x_2}\left(\frac{\mathrm{d}\rho u}{\mathrm{d}t}+\frac{\mathrm{d}}{\mathrm{d}x}(\rho uu+p)\right)\mathrm{d}x=0 \tag{3.87}$$

即

$$\frac{\mathrm{d}\rho u}{\mathrm{d}t}+\frac{\mathrm{d}}{\mathrm{d}x}(\rho uu+p)=0 \tag{3.88}$$

方程(3.88)即不考虑剪切变形（τ 的非对角线元素为 0）的一维的动量方程。本节呈现的是另一种动量方程的推导形式，更好的展现了从积分的角度推导动量方程的思想。

在这里需要注意的是，积分形式的方程和微分形式的方程有重要的区别。积分形式的方程(3.81)可以在控制体内出现间断。因为数学上并没有要求被积分的函数不能出现间断。但微分形式的方程(3.88)要求函数是连续的（上文中引入假定 $\rho uu+p$ 是光滑可导的），否则不可导。积分形式的 CFD 方程更加基础，更加普适性。同时，在网格变形的情况下，自动满足体积守恒方程。积分形式的 CFD 方程也适用于各种不同的网格形状。但之所以将积分形式的方程放在最后是因为积分形式的方程中会出现面积分，例如三维的连续性方程可以写为

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V}\rho\mathrm{d}V+\int_{\mathbf{S}_f}\rho_f\mathbf{U}_f\mathrm{d}\mathbf{S}_f=0 \tag{3.89}$$

看起来对初学者并不友好。

3.5 N-S 方程的展开与思考

上述内容完整详细地推导了 N-S 方程。在算法编程课中，将进一步讲解 N-S 方程的数值离散。现在对 N-S 方程做最终的总结：

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \tag{3.90}$$

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) = -\nabla p + \nabla \cdot \boldsymbol{\tau} \tag{3.91}$$

考虑二维的情况，进一步的假定不可压缩以及常粘度系数，方程左右两边除掉密度（注意下面公式中的压力已经除掉了密度但符号未做变更）：

$$\frac{\partial u_1}{\partial x} + \frac{\partial u_2}{\partial y} = 0 \tag{3.92}$$

$$\frac{\partial u_1}{\partial t} + \frac{\partial u_1 u_1}{\partial x} + \frac{\partial u_1 u_2}{\partial y} = -\frac{\partial p}{\partial x} + \nu \frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x} + \frac{\partial u_1}{\partial x}\right) + \nu \frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}\right) \tag{3.93}$$

$$\frac{\partial u_2}{\partial t} + \frac{\partial u_1 u_2}{\partial x} + \frac{\partial u_2 u_2}{\partial y} = -\frac{\partial p}{\partial x} + \nu \frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\right) + \nu \frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial y} + \frac{\partial u_2}{\partial y}\right) \tag{3.94}$$

动量方程可以进一步展开（同时忽略时间项）：

$$2u_1 \frac{\partial u_1}{\partial x} + u_1 \frac{\partial u_2}{\partial y} + u_2 \frac{\partial u_1}{\partial y} = -\frac{\partial p}{\partial x} + 2\nu \frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}\right) + \nu \frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}\right) \tag{3.95}$$

$$2u_2 \frac{\partial u_2}{\partial y} + u_1 \frac{\partial u_2}{\partial x} + u_2 \frac{\partial u_1}{\partial x} = -\frac{\partial p}{\partial x} + \nu \frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial y} + \frac{\partial u_2}{\partial x}\right) + 2\nu \frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial y}\right) \tag{3.96}$$

读者应该清楚方程(3.90)的第一项、第二项表示什么物理意义，(3.91)的第一项、第二项、第三项、第四项表示什么物理意义。同时，对各项的数学推导过程有大致的印象。N-S 方程是大量算法的根基，例如，直接模拟 DNS 就是调用高阶格式直接求解 N-S 方程，多相流 VOF 模型就是在 N-S 方程的基础上添加表面张力等体积力（体积力在哪一个方程中有体现？在方程的哪一侧？）。欧拉拉格朗日算法就是连续性方程使用 N-S 方程来描述，颗粒相使用拉格朗日算法来描述。湍流模型则是通过数值的方法增加方程的粘度使得易于收敛（粘度在方程中哪里有所体现？）。牛顿流体假定一个恒定的粘度项，非牛顿流体流动则是将粘度表示为可变的变量。相变过程则是在相分数方程和速度方程添加源项来实现等。

虽然基于 N-S 方程可以进行各种模型的演变。但在离散求解中，任何一个小问题都可能会演变为最后的发散，例如，粘度过大可能会导致方程刚性、附加体积力的梯度项可能导致数值震荡、不依附于流线的对流项会导致较大的数值耗散，另外，如何处理温度小于 0 的情况？为什么有些情况湍流粘度过大？湍流模型的参数是否可以改动？速度压力迭代次数是否可以减少提高效率？为什么模拟过程中无法算出湍流？计算中所有的参数在数值上都有什么意义？所有这些内容，都属于 CFD 算法研究的内容。在这里，读者可以重新翻到本笔记的第一章，进一步了解 N-S 方程更加奇妙的数值特性。

第四章　经典 CFD

4.1　湍流

湍流的流动状态是非常不稳定的。如果将速度对时间做数据提取，会发现其呈现一个非常随机的波动状态。且流动中存在大量的涡。湍流过程将具有不同动量的流体单元“打散”且“混合”到一起。同时由于粘性的作用，速度的梯度会趋向于减小，并进一步的减小动能。这些动能被转化为内能。湍流有些情况是让人喜欢的，有些情况却让人厌恶。如果过程中需要加强传递与混合，那么湍流会让研究学者欣喜若狂。另一方面，对动能的掺混需要附加能量，这可能会大幅的增加能量消耗支出。最理想的情况下，是调用尽可能少的能量（如电能），获得最大的湍流效应来增加混合效果。

在下文中，会发现湍流的数值模拟需要引入大量的模型，并且会导致计算不稳定。但为什么大家都对湍流如此感兴趣？这是因为飞机要上天、要对天气预报进行预测，类似的过程都是湍流的。正所谓处处皆湍流。

最早对湍流进行研究的是 Reynolds。1895 年左右，Reynolds 提出了一种时间平均方法，因此，在后期相当一定的时间内，雷诺平均主要指的是时间平均。1877 年，Boussinesq 参考分子梯度扩散的数学形式，引入涡粘的概念。随后，Reynolds 和 Boussinesq 在湍流领域一夜成名，一直在现在，对两人的研究只需要点到为止，都不需要引用。

但是 Reynolds 和 Boussinesq 并没有对湍流模型进行求解。在 19 世纪，湍流模型的结果一直是一个迷。直到 1904 年，Prandtl 发现了边界层的概念（原始文献为德语版），并于 1925 年提出了混合长模型（德语版），并可以通过混合长来计算湍流粘度。在随后的近 30 年中，混合长湍流模型成为了其他湍流模型的理论基础。当代，Prandtl 的混合长模型通常被称之为零方程湍流模型。顾名思义，零方程湍流模型中，不存在传输方程，仅仅通过代数关系式就可以获得湍流粘度。进一步的，Prandtl 在 1945 年，提出了一种一方程湍流模型（德语版），其将湍流粘度与湍流动能的概念建立起来，并组建一个偏微分方程组来对湍流动能进行传输。

与此同时，Kolmogorov 认为除了应该考虑湍流动能，还应该考虑一个变量 ω。这个变量表示的是每单位体积单位时间的能量耗散。ω 的倒数表示湍流的时间尺度。$k\omega$ 则类似于湍流耗散率 ε。1942 年，Kolmogorov 构建了 k 和 ω 的偏微分方程，这是最早的一种两

方程湍流模型。很遗憾，当时并没有计算机能求解这种湍流模型。除了这些湍流模型之外，最早的二阶矩湍流模型可以追溯到 Rotta 在 1951 年的工作（德语版）。二阶矩湍流模型也就是雷诺应力传输模型。所有这些模型都是在 1955 年之前提出的。

1960 年代之后，大量的湍流模型开始涌现，并可以通过计算机进行求解。2020 年，如果将所有的湍流模型进行归类，湍流的数值研究方法主要有三种：直接模拟（Direct Numerical Simulation，DNS），大涡模拟（Large Eddy Simulation，LES）以及雷诺平均法（Raynolds-averaged Navier-Stokes，RANS）。本节对这三种方法进行简述。

4.1.1 直接模拟 DNS

直接模拟是对湍流进行数值研究最精确的方法。直接模拟直接求解 N-S 方程，不附加任何湍流模型。因此，从概念上理解的话，直接模拟是一种最简单的方法。上升到实践，如果采用 OpenFOAM 进行直接模拟的话，只需要将流动设置为层流即可。为了精确的捕获到流动中的湍流，需要注意计算域大小。首先，计算域的整体大小，应该是流动中可能存在的最大的涡的尺度的数倍以上。同时，为了捕获所有的能量耗散，网格尺度还需要与最小的粘性尺度相符合。这种最小的尺度通常被称之为 Kolmogorov 尺度。最重要的，采用的数值格式应该具备非常高的精度，通常采用谱方法或其他同类方法。如果采用 OpenFOAM 直接采用层流来计算 NS 方程，可以认为是一种近似的直接模拟，也被称之为 quasi-DNS。

直接模拟预测的结果包含了大量的信息。例如，图4.1为采用直接模拟捕获的近壁条带。但是，这些信息并不是所有都至关重要。对于这种耗时且极为占据存储空间的数值方法，直接模拟目前主要在学术界采用。随着计算机能力的提升，2020 年，一些个人工作站已经也可以进行一些直接模拟计算。

对于直接模拟的求解方法。有限体积法由于精度问题受限。主流的方法是有限差分法与谱方法。不管采用什么样的方法，需要注意的就是方法的精度。例如，对于一阶迎风格式，其在数学上等于在方程中添加了一个数值耗散项。这种耗散项对算法有稳定性作用。对于稳态问题，这种耗散可能无关紧要。但对于瞬态问题，这种人工添加的耗散可能不会捕获尖锐的速度梯度导致直接模拟的结果失真，丧失准确性。

另外一个极为重要的问题就是直接模拟的边界条件设置。对于初始条件，其必须包含足够的流场细节。最好的情况下，是给定一个从实验获得的真实的流场。但这是非常困难的。同样的，对于边界条件，尤其是进口边界条件，必须包含足够的流场信息来“形成”湍流。

出口边界条件相对简单。通常采用的边界条件就是法向梯度为 0。但，另外一种方式是使用一种对流非稳态出口边界条件。比如，对 ϕ 变量，可以调用下述边界条件：

$$\frac{\partial \phi}{\partial t} + \mathbf{U} \cdot \nabla \phi = 0 \tag{4.1}$$

在开源 CFD 软件 OpenFOAM 中，上述边界条件被称之为无反射边界条件[1]。在壁面处，可以使用无滑移边界条件。但是通常在壁面处，会存在一些细微的非常重要的“条带”结构。这些结构需要非常精细的网格才能处理。另外需要注意的是，对称边界条件通常是不适用于直接模拟与大涡模拟的。因为虽然在雷诺平均中，可以将壁面当做对称面来处理，但是在直接模拟中，对称面的脉动速度必然不是对称的。

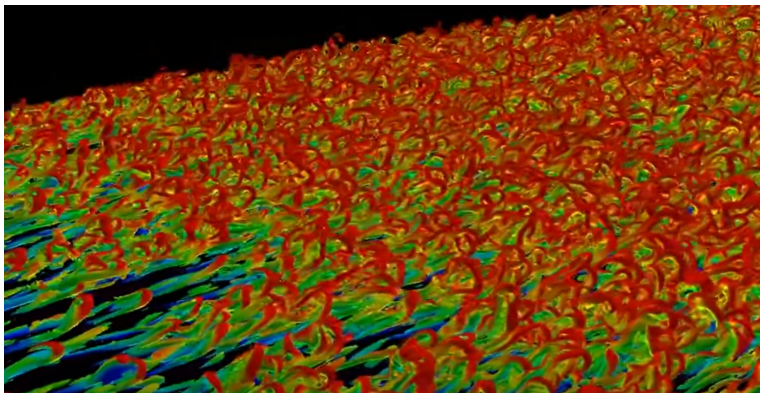


图 4.1: 湍流近壁条带结构 [10]。

4.1.2 大涡模拟 LES

湍流中的大涡往往比小涡更具有能量。这样来看，或许可以直接对大涡进行非模化的跟踪，将小涡模化处理。这就是大涡模拟的思想起源。大涡模拟同样是三维的并且是随时间瞬态变化的。在直接模拟不现实的情况下，通常可以采用大涡模拟来捕获流场结果。在一些领域（如气候科学、湍流燃烧等领域），大涡模拟已经成为不可或缺的研究方法。不同于雷诺平均法，大涡模拟起始于空间滤波技术。各种大涡模拟方法都要确定一个 Δ，其被称之为截断尺度（cutoff width），也称为滤波尺度（filtered width）。大于截断尺度的涡全部被解析，在空间滤波的过程中，小于截断尺度的涡全部被模化。在算法的角度，大涡模拟区分为不同的计算方法，如隐性滤波大涡模拟（Implicit filtered LES）、显性滤波大涡模拟（Explicit filtered LES）等。在隐性滤波大涡模拟中，最终植入的算法并不需要显性的进行滤波操作。最终的计算方程，隐性滤波大涡模拟与非稳态雷诺平均类方法（如大涡模拟的 Smagorinsky 模型与雷诺平均的 kEpsilon 模型），区别仅仅在于湍流粘度的模化。且雷诺平均的湍流粘度的大小远远大于隐性滤波大涡模拟。

大涡模拟需要定义一个速度变量，这个速度变量包含且仅仅包含大尺度的结构。因此，这个速度变量通常被称之为滤波速度（filtering velocity）。滤波速度跟滤波尺度 Δ 有关。大涡模拟认为尺度大于 Δ 就是大涡，尺度小于 Δ 就是小涡。不同于直接模拟，由于大涡模拟需要进行模化，因此需要对控制方程进行推导。大涡模拟中重要的模型在于将亚格子

[1] 在这里有一些关于无反射边界条件在 DNS 模拟中的讨论。

应力进行模化（Subgrid-Scale Stress，SGS）。

对于隐性滤波大涡模拟，在提升网格分辨率的情况下，截断误差以及亚格子应力模化的误差都随之减小。因此，隐性滤波大涡模拟不存在网格无关性这一说。跟直接模拟方法一样，大涡模拟的边界条件以及初始条件均异常敏感，需要复杂的方法进行给定。最后值得一提的是，还有另外一种大涡模拟方法，其被称之为 MILES（Monotone Integrated Large Eddy Simulation）。Boris et al. [22] 最早的提出了 MILES 的概念，其将有界的通量修正算法（Flux Corrected Transport）与三阶格式相结合来实现大涡模拟的目的。需要注意的是，在 MILES 中不需要显性的调用亚格子模型。因此，MILES 与传统的 LES 的区别主要在于：1）MILES 的对流项格式需要通过通量修正算法来计算，2）MILES 不需要调用显性的亚格子模型。在 Fureby and Grinstein 的研究中 [54]，系统的对比了 MILES 与传统 LES 方法的数值等同性本源。本质上，MILES 也是隐性滤波大涡模拟。

接下来我们看大涡模拟的推导过程。在大涡模拟中，需要定义一个滤波函数，目前有限体积法，使用最广也最简单的滤波函数为 Tophat 滤波：

$$
\begin{aligned}
G(\Delta) &= \frac{1}{\Delta^3}, \mathbf{x} \subset \Delta V, \\
G(\Delta) &= 0, \mathbf{x} \notin \Delta V
\end{aligned}
\tag{4.2}
$$

其他还存在如主要应用于有限差分的高斯滤波，主要应用于谱方法的谱滤波方法（能谱在波长为 Δ/π 处精准截断）。大涡模拟的控制方程在推导的过程中，需要对变量进行滤波后并积分。考虑 Tophat 滤波函数，有滤波后的变量的定义：

$$
\overline{\phi} = \int_V G(\Delta)\phi \mathrm{d}V = \frac{1}{\Delta^3}\int_V \phi \mathrm{d}V \tag{4.3}
$$

$$
\overline{\frac{\partial \phi}{\partial x}} = \int_V G(\Delta)\frac{\partial \phi}{\partial x}\mathrm{d}V = \frac{1}{\Delta^3}\int_V \frac{\partial \phi}{\partial x}\mathrm{d}V \tag{4.4}
$$

如果 ΔV 不变，方程(4.4)中可以将微分符号提取出来。但在有限体积法中，网格大小一般是变化的。然而，同样可以将微分符号提取出来，只不多其增加了截断误差 ΔV^2。这是可以接受的，因为本身传统有限体积法也充其量最高二阶精度。因此，方程(4.4)可以继续写为 [58]：

$$
\overline{\frac{\partial \phi}{\partial x}} = \frac{1}{\Delta^3}\int_V \frac{\partial \phi}{\partial x}\mathrm{d}V = \frac{\partial}{\partial x}\left(\frac{1}{\Delta^3}\int_V \phi \mathrm{d}V\right) + O(\Delta V^2) = \frac{\partial \overline{\phi}}{\partial x} \tag{4.5}
$$

注意在方程(4.5)中实际上隐含了离散滤波，在(4.3)隐含了网格滤波。这是因为本身有限体积法求解的 ϕ 变量即为一种体平均变量。因此使用有限体积法网格进行 LES 方程的推导，本身就涵盖了网格滤波。同时在(4.5)中对高精度量的舍去，可以看为一种离散滤波。也正是如此，有限体积法如果不进行二次滤波，那么即为隐性滤波大涡模拟。

因此，不可压缩大涡模拟的控制方程，在进一步进行滤波操作后有：

$$\nabla \cdot \overline{\mathbf{U}} = 0, \tag{4.6}$$

$$\frac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}\mathbf{U}}) = -\frac{1}{\rho}\nabla \overline{p} + \nabla \cdot \overline{\boldsymbol{\tau}}, \tag{4.7}$$

需要注意的是方程(4.7)中的对流项 $\nabla \cdot (\overline{\mathbf{U}\mathbf{U}})$ 的存在是因为 $\overline{\mathbf{U}\mathbf{U}} \neq \overline{\mathbf{U}}\,\overline{\mathbf{U}}$。为了将 $\overline{\mathbf{U}\mathbf{U}}$ 在方程中体现，可以简单的在方程(4.7)左右两边将这一项加上：

$$\frac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}}\,\overline{\mathbf{U}}) = -\frac{1}{\rho}\nabla \overline{p} + \nabla \cdot \overline{\boldsymbol{\tau}} - \nabla \cdot (\overline{\mathbf{U}\mathbf{U}}) + \nabla \cdot (\overline{\mathbf{U}}\,\overline{\mathbf{U}}) \tag{4.8}$$

将亚格子应力写为 $\boldsymbol{\tau}_{sgs}$[2]：

$$\boldsymbol{\tau}_{sgs} = \overline{\mathbf{U}\mathbf{U}} - \overline{\mathbf{U}}\,\overline{\mathbf{U}} \tag{4.9}$$

方程(4.8)可以写为：

$$\frac{\partial \overline{\mathbf{U}}}{\partial t} + \nabla \cdot (\overline{\mathbf{U}}\,\overline{\mathbf{U}}) = -\frac{1}{\rho}\nabla \overline{p} + \nabla \cdot \overline{\boldsymbol{\tau}} - \nabla \cdot \boldsymbol{\tau}_{sgs} \tag{4.10}$$

各类大涡模拟模型，就在于将如何将亚格子应力 $\boldsymbol{\tau}_{sgs}$ 进行模化。在对 $\boldsymbol{\tau}_{sgs}$ 进行模化的过程中，可以将其分解为 3 项单独进行模化。但是目前主流的方式是将 $\boldsymbol{\tau}_{sgs}$ 作为单独的一项作为模化，也即上文提到的隐性滤波大涡模拟。最直观的方法就是采用 Bousinessq 方法来处理并假定各向同性。首先将 $\boldsymbol{\tau}_{sgs}$ 写为：

$$\boldsymbol{\tau}_{sgs} = \boldsymbol{\tau}_{sgs} - \frac{1}{3}\mathrm{tr}\left(\boldsymbol{\tau}_{sgs}\right)\mathbf{I} + \frac{1}{3}\mathrm{tr}\left(\boldsymbol{\tau}_{sgs}\right)\mathbf{I} \tag{4.11}$$

定义[3]

$$k_{sgs} = \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\tau}_{sgs}\right) = \frac{1}{2}\mathrm{tr}\left(\overline{\mathbf{U}\mathbf{U}} - \overline{\mathbf{U}}\,\overline{\mathbf{U}}\right) = \frac{1}{2}\left(\overline{\mathbf{U}\cdot\mathbf{U}} - \overline{\mathbf{U}}\cdot\overline{\mathbf{U}}\right) \tag{4.12}$$

有

$$\frac{1}{3}\mathrm{tr}\left(\boldsymbol{\tau}_{sgs}\right)\mathbf{I} = \frac{1}{3}\mathrm{tr}\left(\overline{\mathbf{U}\mathbf{U}} - \overline{\mathbf{U}}\,\overline{\mathbf{U}}\right)\mathbf{I} = \frac{2}{3}k_{sgs}\mathbf{I} \tag{4.13}$$

这样，压力项与湍流应力项可以写为：

$$-\frac{1}{\rho}\nabla \overline{p} - \nabla \cdot \boldsymbol{\tau}_{sgs} = -\frac{1}{\rho}\nabla \overline{p'} - \nabla \cdot \left(\boldsymbol{\tau}_{sgs} - \frac{2}{3}k_{sgs}\mathbf{I}\right) \tag{4.14}$$

其中 $p' = p + 2/3k_{sgs}$。$\boldsymbol{\tau}_{sgs} - \frac{2}{3}k_{sgs}\mathbf{I}$ 表示湍流应力的偏应力部分，其可以写为：

$$\boldsymbol{\tau}_{sgs} - \frac{2}{3}k_{sgs}\mathbf{I} = -\nu_{sgs}\left(\nabla\overline{\mathbf{U}} + \nabla\overline{\mathbf{U}}^{\mathrm{T}} - \frac{2}{3}(\nabla\cdot\overline{\mathbf{U}})\mathbf{I}\right) \tag{4.15}$$

[2]注意 $\boldsymbol{\tau}_{sgs}$ 与 RANS 中的 $\boldsymbol{\tau}_t$ 的区别，除掉密度后 RANS 中的 $\boldsymbol{\tau}_t/\rho = -\overline{\mathbf{U}'\mathbf{U}'}$。同时在 RANS 模型中有 $\overline{\mathbf{U}\,\mathbf{U}} = \overline{\mathbf{U}}\,\overline{\mathbf{U}} + \overline{\mathbf{U}'\mathbf{U}'}$。$\overline{\mathbf{U}'\mathbf{U}'}$ 可以称之为雷诺应力。在隐性大涡模拟中也存在类似的关系，因此有 $\boldsymbol{\tau}_{sgs} = \overline{\mathbf{U}'\mathbf{U}'}$。但对于其他类大涡模拟如果使用其他的滤波函数，则没有这个恒等式，其本质是因为 $\overline{\mathbf{U}'} \neq 0$。因此，为了更加的普适性，大涡模拟中雷诺应力的表达一般不写为 $\overline{\mathbf{U}'\mathbf{U}'}$。

[3]注意 LES 中 k_{sgs} 与 RANS 中 k 的区别。

其中 ν_{sgs} 表示大涡粘度。在 Smagorinsky 湍流模型中，大涡粘度可以表示为

$$\nu_{sgs} = C_{sgs}\overline{\Delta}\sqrt{k_{sgs}} \tag{4.16}$$

其中 k_{sgs} 表示亚格子动能，C_s 表示用户自定义常数。对于各向同性湍流，$C_s \approx 0.2$。然而对于不同的雷诺数以及流形，C_s 倾向于并不是一个常数。例如，在开源 CFD 软件 OpenFOAM 中，C_s 的值被定义为 0.065。在这种情况下，大涡粘度小了一倍之多。在槽道流靠近壁面的区域，C_s 需要更小。这也就是 van Driest 阻尼模型的思想本源。另一种方式是，定义一个随时间与位置变化的 C_s，这就是动态亚格子应力模型的出发点。

在一些 LES 的文章中，方程(4.16)被写为[4]：

$$\nu_{sgs} = (C_s\overline{\Delta})^2\sqrt{2\overline{\mathbf{D}}:\overline{\mathbf{D}}} = (C_s\overline{\Delta})^2|\overline{\mathbf{D}}| \tag{4.17}$$

$$\overline{\mathbf{D}} = \frac{\nabla\overline{\mathbf{U}} + \nabla\overline{\mathbf{U}}^{\mathrm{T}}}{2} - \frac{1}{3}(\nabla\cdot\overline{\mathbf{U}})\mathbf{I} \tag{4.18}$$

方程(4.17)与(4.16)是两种不同的植入方法。但方程(4.16)的植入更加符合物理[5]。在这种情况下，有

$$\boldsymbol{\tau}_{sgs} - \frac{1}{3}\mathrm{tr}\left(\boldsymbol{\tau}_{sgs}\right)\mathbf{I} = \boldsymbol{\tau}_{sgs} - \frac{2}{3}k_{sgs}\mathbf{I} = -2(C_s^2\overline{\Delta}^2|\overline{\mathbf{D}}|)\overline{\mathbf{D}} \tag{4.19}$$

与隐性滤波大涡模拟不同的是显性滤波大涡模拟。在显性滤波大涡模拟中，需要对 NS 方程应用两次滤波操作，第一次滤波操作等同于隐性滤波大涡模拟，这一步的滤波操作通常被称之为网格滤波（grid filter），第二次的滤波操作通常被称之为测试滤波（test filter）。测试滤波的尺度通常要大于网格滤波的尺度，并且会对湍流场产生附加的光顺作用。在第一次滤波方程形成之后，显性滤波大涡模拟再次应用测试滤波，会形成一套完全一样的方程。只不过变量均为测试滤波之后的变量，且亚格子应力发生变化。显性滤波大涡模拟需要显式的指定滤波方法，如盒子滤波或者高斯滤波之类。

最早的显性滤波大涡模拟可追溯到 Bardina et al. 的文章 [15]。在这一篇文章中第一次使用显性滤波的概念。并构建了一种隐性显性滤波混合模型：

$$\boldsymbol{\tau}_{sgs} - \frac{2}{3}k_{sgs}\mathbf{I} = C'\left(\widetilde{\overline{\mathbf{U}}\,\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\,\widetilde{\overline{\mathbf{U}}}\right) - 2(C_s^2\overline{\Delta}^2|\overline{\mathbf{D}}|)\overline{\mathbf{D}} \tag{4.20}$$

其中 C' 为一个可调整的参数。除此之外，在大涡模拟历史上最有影响力的一篇文章出自于 Germano et al. [57]。Germano 等人最早的意识到了 Smagorinsky 模型的模型参数需要依据流态的变化而变化。因此其提出一种动态的模型，其模型参数为一个空间相关可变的参数。一些研究表示（如图4.2所示），显性滤波大涡模拟由于可以将 Δ 固定下来，因此可以达到网格无关解 [23]。隐性滤波大涡模拟之所以被称之为隐性，是因为没有第二次滤波

[4]因为 $\sqrt{2\overline{\mathbf{D}}:\overline{\mathbf{D}}} = \sqrt{2}|\overline{\mathbf{D}}|$，所以 $(C_s\overline{\Delta})^2|\overline{\mathbf{D}}|$ 公式中缺失了 $\sqrt{2}$。目前原因不明。有可能的解释是 $\sqrt{2}$ 的影响被融入了 C_s 参数里面。目前这个算法为动态 Smagorinsky 模型，OpenFOAM 官方并没有进行植入。也没有算例可以参考 C_s 参数的具体的值。注意，在很多讨论 LES 的文章中，直接定义 $|\overline{\mathbf{D}}| = \sqrt{2\overline{\mathbf{D}}:\overline{\mathbf{D}}}$，这里的 $|\overline{\mathbf{D}}|$ 是一种数学符号的定义，并不是张量中的模的意思。

[5]参考相关讨论。

模型的存在。且其第一次滤波操作通过有限体积法的网格来实现[6]。隐性滤波大涡模拟中本身的亚格子模型已经附带了数值耗散。因此建议使用不附加耗散的数值格式，比如中心格式之类[7]。目前大量的大涡模拟方法倾向于使用隐性滤波大涡模拟，显性滤波大涡模拟在学术上则使用较多。

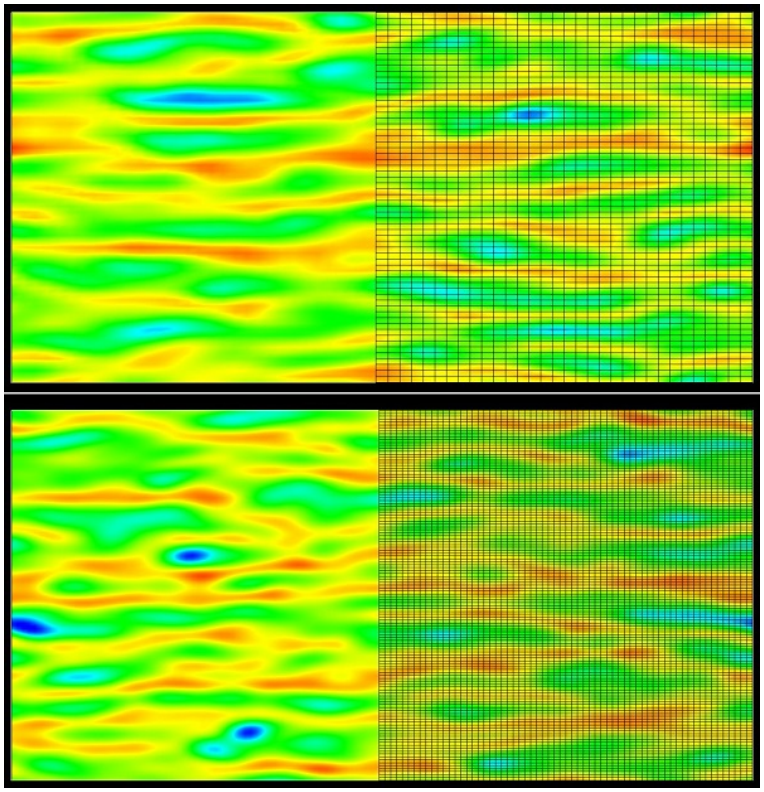


图 4.2: 显性滤波大涡模拟网格无关解 [23]。

在动态类模型中，其中的亚格子应力系数通过计算出来，并不是一个均一值，也即动态的。动态类模型在方程(4.7)的基础之上，再次进行一次滤波（test filter）有 [57]:

$$\frac{\partial \widetilde{\overline{\mathbf{U}}}}{\partial t} + \nabla \cdot (\widetilde{\overline{\mathbf{U}}}\,\widetilde{\overline{\mathbf{U}}}) = -\frac{1}{\rho}\nabla \tilde{\bar{p}} + \nabla \cdot \widetilde{\overline{\boldsymbol{\tau}}} - \nabla \cdot \mathbf{T}_{sgs} \tag{4.21}$$

其中

$$\mathbf{T}_{sgs} = \widetilde{\overline{\mathbf{U}\,\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\,\widetilde{\overline{\mathbf{U}}} \tag{4.22}$$

$\mathbf{T}_{sgs}$ 表示模化后的二次滤波的雷诺应力。对比 $\mathbf{T}_{sgs}$ 与 $\boldsymbol{\tau}_{sgs}$，可以发现 $\mathbf{T}_{sgs}$ 进行了两次不同尺度的滤波，第一次滤波为隐性滤波，第二次滤波为显性滤波，$\boldsymbol{\tau}_{sgs}$ 进行了一次滤波。在这里需要注意的是，在已知一次滤波 $\overline{\phi}$ 的值之后，二次滤波 $\widetilde{\phi}$ 的值可以计算出来（参考第6.14.2节中的 LES 滤波函数）。在这里显性的定义解析的二次滤波的湍流应力（Lernard

[6]有限体积法的空间平均可以看做是 box 滤波。

[7]注意库朗数要小于 0.5。

应力）为:

$$\mathbf{L} = \widetilde{\overline{\mathbf{U}}\,\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\,\widetilde{\overline{\mathbf{U}}} \tag{4.23}$$

可见，$\mathbf{L}$ 表示二次滤波的湍流应力与一次滤波的湍流应力之差。其表示在二次滤波网格与一次滤波网格偏差基础上解析的湍流应力。在获得第一次滤波速度后，$\mathbf{L}$ 是一个可以计算的变量。同时有下述关系：

$$\mathbf{L} = \widetilde{\overline{\mathbf{U}}\,\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\,\widetilde{\overline{\mathbf{U}}} = \mathbf{T}_{sgs} - \widetilde{\boldsymbol{\tau}_{sgs}} \tag{4.24}$$

方程(4.24)也被称之为 Germano 恒等式。其在 1991 年通过对 NS 方程进行二次滤波后被发现 [57]。Germano 分解的独特之处在于，方程(4.24)如果将两边都写成速度的函数，其可以计算且可以用于决定 Smagorinsky 模型的系数。很明显其对于 C_s 固定的 Smagorinsky 模型来说，其 C_s 是可以动态调整的。

现在来看如何通过 Germano 恒等式来得出 C_s 的值。参考方程(4.19)对 $\mathbf{T}_{sgs}-\frac{1}{3}\mathrm{tr}(\mathbf{T}_{sgs})\mathbf{I}$ 以及 $\widetilde{\boldsymbol{\tau}_{sgs} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{sgs})\mathbf{I}}$ 进行模化[8]：

$$\mathbf{T}_{sgs} - \frac{1}{3}\mathrm{tr}(\mathbf{T}_{sgs})\mathbf{I} = -2C_s^2\widetilde{\overline{\Delta}}^2\ |\widetilde{\overline{\mathbf{D}}}|\widetilde{\overline{\mathbf{D}}} \tag{4.25}$$

$$\widetilde{\boldsymbol{\tau}_{sgs}} - \widetilde{\frac{1}{3}\mathrm{tr}(\boldsymbol{\tau}_{sgs})\mathbf{I}} = -2C_s^2\widetilde{\overline{\Delta}}^2\,\widetilde{|\overline{\mathbf{D}}|\overline{\mathbf{D}}} \tag{4.26}$$

将两个式子相减有：

$$\mathbf{L} + \frac{1}{3}\left(\widetilde{\mathrm{tr}(\boldsymbol{\tau}_{sgs})} - \mathrm{tr}(\mathbf{T}_{sgs})\right)\mathbf{I} = -2C_s^2\left(\widetilde{\overline{\Delta}}^2|\widetilde{\overline{\mathbf{D}}}|\widetilde{\overline{\mathbf{D}}} - \overline{\Delta}^2\widetilde{|\overline{\mathbf{D}}|\overline{\mathbf{D}}}\right) \tag{4.27}$$

同时，

$$\widetilde{\mathrm{tr}(\boldsymbol{\tau}_{sgs})} - \mathrm{tr}(\mathbf{T}_{sgs}) = \widetilde{\mathrm{tr}(\overline{\mathbf{U}\mathbf{U}} - \overline{\mathbf{U}}\,\overline{\mathbf{U}})} - \mathrm{tr}(\widetilde{\overline{\mathbf{U}\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\,\widetilde{\overline{\mathbf{U}}}) = -\mathrm{tr}(\widetilde{\overline{\mathbf{U}\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\,\widetilde{\overline{\mathbf{U}}}) = -\mathrm{tr}(\mathbf{L}) \tag{4.28}$$

将方程(4.28)带入到(4.27)有：

$$\mathrm{dev}(\mathbf{L}) = \mathbf{L} - \frac{1}{3}\mathrm{tr}(\mathbf{L})\mathbf{I} = -2C_s^2\left(\widetilde{\overline{\Delta}}^2|\widetilde{\overline{\mathbf{D}}}|\widetilde{\overline{\mathbf{D}}} - \overline{\Delta}^2\widetilde{|\overline{\mathbf{D}}|\overline{\mathbf{D}}}\right) \tag{4.29}$$

方程(4.29)可以简化成关于 $\overline{\mathbf{U}}$ 的关系式，其实际是 5 个方程，但只有唯一的未知数即为 C_s，因此其看起来是可以决定 C_s 的值的。但由于方程(4.29)右侧的值可能为 0，会导致方程病态无法确定 C_s[9]。为了解决这个问题，Lilly 通过最小二乘法的方式可以将 C_s 的误差最小化 [104]。求得后既可以获得随空间时间平均分布的 C_s。如果认为二次滤波尺度是一次滤波尺度的两倍，有：

$$\mathrm{dev}(\mathbf{L}) = 2C_s^2\overline{\Delta}^2\left(\widetilde{|\overline{\mathbf{D}}|\overline{\mathbf{D}}} - 4|\widetilde{\overline{\mathbf{D}}}|\widetilde{\overline{\mathbf{D}}}\right) \tag{4.30}$$

[8] Kim 认为 C_s 也应该进行二次滤波操作 [77]，但只是为了简化计算因此都没有进行二次滤波。这也为其自身的动态模型做了铺垫。

[9] Germano et al. 的文章中通过展向平均来进行类似的操作，其构成最原始的动态 Smagorinsky 模型。但目前很少有人用，大部分以后得工作都是使用 Lilly 的方法。

定义

$$\mathbf{M} = \overline{\Delta}^2 \left(\widetilde{|\overline{\mathbf{D}}|\overline{\mathbf{D}}} - 4|\widetilde{\overline{\mathbf{D}}}|\widetilde{\overline{\mathbf{D}}} \right) \tag{4.31}$$

有 [104]：

$$\mathrm{dev}(\mathbf{L}) : \mathbf{M} = 2C_s^2 \mathbf{M} : \mathbf{M} \tag{4.32}$$

$$C_s^2 = \frac{1}{2} \frac{\mathrm{dev}(\mathbf{L}) : \mathbf{M}}{\mathbf{M} : \mathbf{M}} \tag{4.33}$$

在 Germano 的原始文章中，方程(4.32)左右两边乘以的为 $\overline{\mathbf{D}}$。Lilly 认为这种操作不具有物理意义。综上所述，这种方法即为 Germano 显性大涡模拟方法 [57]，也被称之为动态 Smagorinsky 方法。

显性滤波大涡模拟仅存的自定义参数即为二次滤波以及一次滤波的 Δ 值相对大小。Germano 等建议二者的比值为 2[57]。下面我们证实俩种滤波操作的区别。在一维情况下，网格 P 点的 test 滤波变量可以计算为：

$$\begin{aligned} \widetilde{\overline{\phi}}_P = \frac{1}{2\Delta x} \int_W^E \overline{\phi} \mathrm{d}x = \frac{1}{2\Delta x} \left(\int_W^P \overline{\phi} \mathrm{d}x + \int_P^E \overline{\phi} \mathrm{d}x \right) &= \frac{1}{2\Delta x} \left(\overline{\phi}_w \Delta x + \overline{\phi}_e \Delta x \right) \\ &= \frac{1}{2} \left(\frac{\overline{\phi}_W + \overline{\phi}_P}{2} + \frac{\overline{\phi}_E + \overline{\phi}_P}{2} \right) \end{aligned} \tag{4.34}$$

很明显，$\widetilde{\overline{\phi}}_P \neq \overline{\phi}_P$。在高维的情况下，网格 P 点的 test 滤波变量会涉及到更多的网格面。

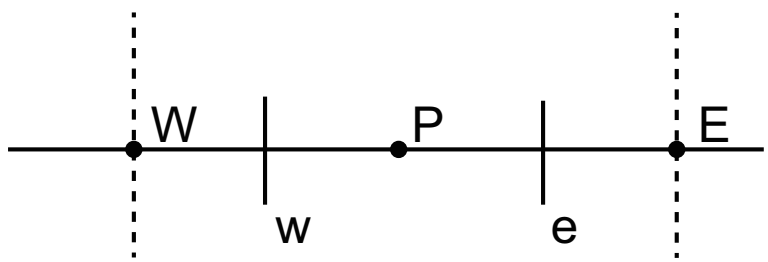


图 4.3: test 滤波与网格滤波。实线表示网格滤波的有限体积，虚线表示 test 滤波的有限体积。

最后在这里强调。在有限体积法中，隐性滤波大涡模拟求解的方程与非稳态 RANS 求解的方程完全相同。不同的仅仅是对于湍流粘度的模化以及用户可选的针对 LES 的特殊的空间时间离散格式。

4.1.3 雷诺平均 RANS

雷诺平均法目前是工程上应用最广泛的方法。例如，很多工程师对一些物体的升阻力、流体的混合程度感兴趣。在这种情况下使用直接模拟和大涡模拟就有些大材小用了。雷诺平均法也存在稳态与瞬态一说。对于稳态算例，所有的非稳态因素都被平均的过程抹掉了。对于瞬态算例，雷诺平均法应捕获相应的瞬态因素。参考图4.4，稳态的雷诺平均通常采用时间平均法，非稳态的雷诺平均通常采用集合平均法。

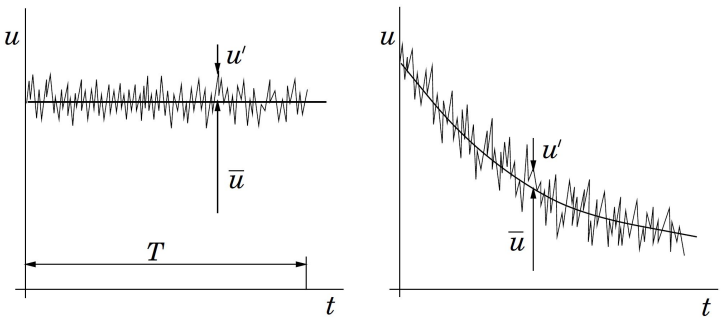


图 4.4: 时间平均和集合平均 [51]。

在进行雷诺平均的过程中，同样需要对模型进行封闭。再一次的，Boussinesq 假定发挥了重要的作用。Boussinesq 假定将未封闭的湍流应力与湍流粘度联合起来，这样，就把未知量转化为了湍流粘度变量。在获得湍流粘度之后，N-S 方程即可进行求解。

目前存在大量的湍流粘度封闭方法。如线性涡粘法、非线性涡粘法等。在线性涡粘法中，湍流粘度可以通过一个代数方程求出。比如，可以将湍流粘度与湍流动能以及湍流动能耗散率联合起来：

$$\nu_t = C_\mu \frac{k^2}{\varepsilon} \tag{4.35}$$

其中 C_μ 为模型自定义常数，k 为湍流动能，ε 为湍流动能耗散率。进一步的，其中的 k 与 ε 可以通过 $k-\varepsilon$ 湍流模型进行求解获得。除此之外，还有大量的其他模型可用于求解湍流粘度，如 Spalart-Allmaras 模型、$k-\omega$ 模型等。依据未知的变量数，这些模型又可以分为一方程模型（如 Spalart-Allmaras 模型具有一个未知变量）、二方程模型（二个未知变量，如 $k-\omega$ 模型）等。

不同于大涡模拟与直接模拟，雷诺平均法对初始条件以及进口边界条件没有那么敏感。实际操作中，使用固定值进口边界条件以及零法向梯度出口边界条件即可。对于壁面，则需要调用壁面函数。在一些情况下，壁面函数对流场结果是高度相关的。同时，还存在一些不需要壁面函数的雷诺平均模型，比如 Lam Bremhorst$k-\varepsilon$ 湍流模型 [84]。这些模型也通常被称之为低雷诺数湍流模型。

大涡模拟与雷诺平均法在使用上具有以下差别：

- 大涡模拟更加适用于分离流、钝体绕流（包含复杂混乱的湍流结构）、以及湍流转捩；
- 雷诺平均法将所有的湍流尺度进行了模化，大涡模拟仅仅对小尺度涡进行模化，对大尺度涡进行解析；

4.1.4 二维 DNS/LES

目前学术界普遍的看法是："无三维涡，则无湍流"。"只有在三维的情况下，流体自身的涡旋才能伸展。"但是大量的文献依旧通过二维大涡模拟、甚至二维直接模拟进行相关研究。例如，Bouris and Bergeles 的文章中表示 [24]

> "针对本文的算例，确实大量的文献表明大涡模拟只能进行三维计算，因为二维模拟缺少相关的涡旋特征。但是，本文的研究同样表明，这样一棒子打死二维大涡模拟也不是那么容易的。我们的结果表示二维大涡模拟相对于任何的雷诺平均法都优越。然而，从物理本质来考虑，除非计算机硬件存在限制，最好还是进行三维大涡模拟。"

在 Roohi et al. 的这篇文章中，作者们表示 [140]：

> "本文使用二维大涡模拟的原因主要是其可以节省计算资源。正如我们的研究在 *3.3* 节中讨论到，针对非稳态空化流动，雷诺平均法获得的流场并不让人满意。因此，二维大涡模拟必然比雷诺平均更为优越。并且，前人也有大量的研究表明，二维大涡模拟可以获得跟实验值非常贴近的结果。因此，对于某些特定的情况，二维大涡模拟是精准的。很明显，我们的模拟结果和实验的符合也验证了这个说法。"

同样，一些研究学者非常激进，比如 Breuer 在研究中表示 [25]：

> "我们的研究表明二维的大涡模拟，甚至直接模拟完全是扯淡的。这是因为物理上流动的物质相互影响是三维的，即使是那些看起来非常二维的计算流域也是如此。"

总体来看，目前学术界，对于二维直接模拟/大涡模拟的观点，尚不统一，仍需探索。

4.1.5 RANS-LES 混合模型、DES 模型

RANS 类模型添加瞬态项可以演变为非稳态模式，也通常被称之为 URANS 模型。然而，即使网格分辨率足够高，时间步长足够小，URANS 同样不能提供精准的湍流能谱。目前学术界普遍认为这是由于 RANS 进行的平均操作所导致。因此，URANS 只能解析那些非常低频的涡以及大尺度流动变化。参考图4.5，对于类似的圆柱绕流，URANS 无论如何都并不能够预测圆柱后的各种频率的涡结构，仅仅能够预测低频的涡脱落，且基本为相同频率。完全达不到真正瞬态模拟的解析程度。Young 等的研究也表明 URANS，即使是三维的，其预测的结果也大体与二维 URANS 类似，并没有显著提高 [198]。

为了克服 URANS 的这个问题，可以调用尺度自适应模拟（Scale-Adaptive Simulation，SAS）。SAS 的概念基于 URANS，区别在于在 URANS 的基础之上，增加一个源项。

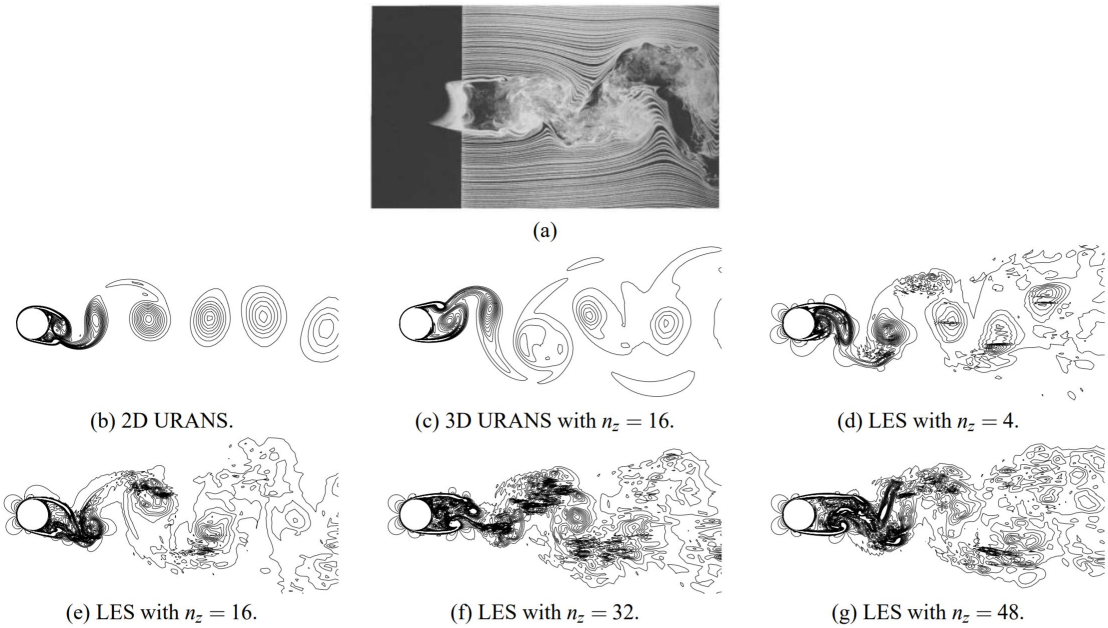


图 4.5: a): 圆柱绕流实验检测流场。b-g): 不同湍流模型预测的涡等高线图（0.5-10）[198]。

这个源项的物理作用将对湍流能谱进行区分，并进一步的识别不同频率的涡。举例：对于 kOmegaSSTSAS 模型，定义 von Karman 尺度 L_{vk}，并且将源项与 von Karman 尺度相关联。如果 ω 方程的源项足够大，会导致 ω 增大，同时湍流粘度降低（$\nu_t = k/\omega$），在这种情况下的粘度阻尼效应降低，高频率涡将可以很好地保持 [158]。

除了 SAS 之外，分离涡模拟（Detached Eddy Simulation，DES）也是非常常用的混合类模型。分离涡模型认为使用大涡模拟进行全场的涡旋捕获是没必要的且不可能的。这是因为壁面处由于湍流尺度通常会变的非常小，如果调用 LES 则需要铺设大量的网格资源。因此分离涡模型在壁面处使用 RANS 进行计算，在主流区采用 LES。以 SpalartAllmaras 模型举例，SpalartAllmaras 模型中的 $\tilde{\nu}$ 存在耗散项 $C_{w1}f_w(\tilde{\nu}/y)^2$，其中 y 表示壁面距离。SpalartAllmaras 模型中的耗散项与产生项的整体作用[10]将 $\tilde{\nu}$ 正比于 Sy^2。类似的，如果将 $\tilde{\nu}$ 与网格尺度 Δ 联系起来，即令 $\tilde{\nu}$ 正比于 $S\Delta^2$。这种思想就是 DES 的本源。在 Spalart 等的文章中，将 y 重新定义为 $\tilde{y}$ [154]：

$$\tilde{y} = \min(y, C_{DES}\Delta) \tag{4.36}$$

其中 C_{DES} 为模型参数。在这种情况下，如果 $y > C_{DES}\Delta$，模型演变为 LES。如果 $y < C_{DES}\Delta$，模型为标准 SpalartAllmaras 模型。考虑整个计算域，尤其是在边界层区域，通常会存在高度各向异性的网格单元。Spalart 等的工作将 Δ 定义为网格三个维度中的最大尺度：

$$\Delta = \max(\Delta_x, \Delta_y, \Delta_z) \tag{4.37}$$

[10] S 的定义参考附录中的 SpalartAllmaras 模型。

因此，在壁面附近，通常 $y \ll \Delta$，即壁面调用标准 SpalartAllmaras 模型。在主流区，通常网格倾向为各向同性，并且主流区的壁面距离通常很大，即 $y \gg \Delta$，这时，则演变为 LES。

然而有些情况下，在层流边界层内，也会导致 DES 演变为 LES，大幅度的降低湍流粘度，导致涡的产生。这种现象被称之为网格引致分离现象（Grid-Induced Seperation，GIS）。在 DES 束手无策的情况下，延迟分离涡模拟（Delayed-DES，DDES）可以很好的解决这个问题。DDES 在 DES 的基础上，附加一个开关函数，强制在边界层内调用 RANS 模拟，在边界层外则为 DES。

不仅仅 SpalartAllmaras 湍流模型可以演变为 DES 模型，其他的模型也可以转变为 DES 模型。在这里需要从湍流尺度的角度来考虑，在大部分的 RANS 模型以及 LES 模型中，湍流粘度的定义分别为：

$$\nu_{RANS} = C_\mu \frac{k^2}{\varepsilon}, \nu_{LES} = C_{LES} \Delta k^{0.5} \tag{4.38}$$

在这里对 k 的写法不做 RANS 以及 LES 的区分。定义湍流尺度 l 为

$$l_{RANS} = C_\mu \frac{k^{1.5}}{\varepsilon}, l_{LES} = C_{LES} \Delta \tag{4.39}$$

则可以将湍流粘度进行整合：

$$\nu_t = \begin{cases} l_{RANS}\, k^{0.5}, & \text{RANS} \\ l_{LES}\, k^{0.5}, & \text{LES} \end{cases} \tag{4.40}$$

可见，在任何情况下，湍流动能 k 都可以表示为：

$$k^{0.5} = \frac{\nu_t}{l} \tag{4.41}$$

考虑不可压缩的 kEpsilon 模型，若认为湍流产生项与耗散项是均衡的，参考第4.11.6节有：

$$G = \varepsilon \tag{4.42}$$

$$2\nu_t |\mathbf{S}|^2 = C_\mu \frac{k^{1.5}}{l} = \frac{C_\mu}{l} \left(\frac{\nu_t}{l}\right)^3 \tag{4.43}$$

也即：

$$\nu_t = \frac{l^2}{C_\mu} 2|\mathbf{S}| \tag{4.44}$$

若将上式的 l 替换为 $l_{LES} = C_{LES}\Delta$，有：

$$\nu_t = \frac{{C_{LES}}^2}{C_\mu} \Delta^2 \left(2|\mathbf{S}|\right) = \frac{{C_{LES}}^2}{C_\mu} \Delta^2 \sqrt{2\mathbf{S} : \mathbf{S}} \tag{4.45}$$

可见，方程(4.45)与 Smagorinsky 模型的定义是一致的。因此，若定义 DES 湍流尺度 l_{DES} 为：

$$l_{DES} = \min(l_{RANS}, l_{LES}) \tag{4.46}$$

则可以在 RANS 模型与 LES 模型中进行切换。同时，在 Smagorinsky 模型中，$\frac{{C_{LES}}^2}{C_\mu}$ 的值在 0.1 至 0.2 之间。因此可以反算出 C_{LES} 的值。

4.2 多相流

多相流是自然界和工程技术中最常见的一种流动。相的概念是从热力学意义上定义的，可以理解为固态、液态和气态。自然界中存在各种各样的多相流动，如雪崩、火山云、雨夹雪等。同时，多相系统也大量的存在于大量的工业过程中，如鼓泡床、流化床、喷雾燃烧、碳烟生成、萃取塔、搅拌反应器等。在这些设备中，离散相粒子可能是自然的存在，也可能是处于某些过程的要求被人为注入。多相流是传热和传质的重要方法。

图4.6中形象的展示了一个气液多相系统和一个气固多相系统。在这些多相系统中，看起来可能并没有联系，但是通常可以用连续相（Continuous phase）和离散相（Disperse phase）对多相系统中的相进行区分。例如鼓泡床中的液体可以看作为连续相，注入的气泡（气体）可以认为是离散相。螺旋桨产生的空泡可以认为是离散相，周围的水可以认为是连续相[11]。引擎中的喷雾可以认为是离散相，周围的空气可以认为是连续相。判定离散相和连续相的关系，主要是离散相往往存在某种属性的分布 [112]。例如鼓泡床中的气泡，存在粒径的分布。流化床中的粒子，存在传输速度的分布。连续相的大部分属性是均一的，比如鼓泡床中的液体以及流化床中的气体并不存在一个粒径分布[12]。

(a) (b)

图 4.6: 多相系统示意图。a): 液滴喷雾。b): 小颗粒中通入空气。图片来源于网络。

离散相的行为对工业过程的影响至关重要。在很多情况下，粒子要么作为一种产物产生，要么作为一种反应物为反应提供原料 [133]。工程师或许迫切想增加离散相的产率，或许想提高离散相和连续相的某种混合特性。对离散相的系统研究，有利于在更深的层次上理解整个过程的流动行为。多相流的研究方向主要有两方面：一是研究离散相的演变（如气泡的聚并和破碎），二是研究多相流体动力学。其中离散相的演变研究内容主要是研究

[11] CFD 中的相不同于物理化学或材料学中的相。比如材料学中的钢可以分为马氏体、奥氏体，在物理化学的铁碳相图中也存在包晶转变。但在 CFD 中，固相并不能够细化到晶型转变的程度。然而一些模拟方法，如离散单元法（Discrete Element Model, DEM），却可以识别固体的形状以及硬度等其他变量。

[12] 在某些情况下连续相和离散相会互相转变。比如表观气速持续增加的鼓泡床中，流型会从泡状流（Bubbly flow）转变为雾状流（Mist flow），这种情况下原本离散的气体变为连续相，原本连续的液体变为离散相。这种现象被称之为相转置。

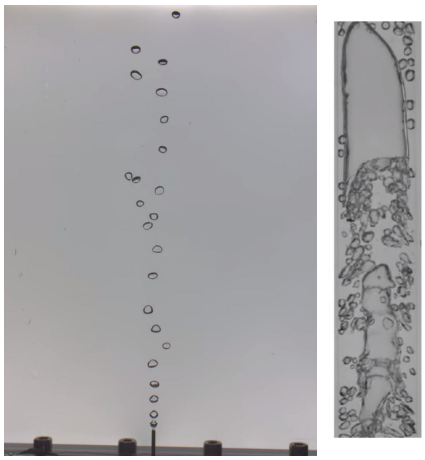
离散相的具体单一粒子的变化过程。例如粒子如何破碎、如何碰撞、碰撞后的运动等。离散相粒子的这些行为主要由群体平衡模型（Population Balance Model，PBM）控制。多相流体力学的研究内容主要关心连续相和离散相的耦合作用，例如粒子运动产生的湍流与涡、粒子的边界层效果、气泡表面张力对气泡的形变等。这些研究需要结合流体力学和群体平衡模型来进行。简单来说，离散相的研究主要关心粒子，多相流体力学主要关心相间作用。近些年来，随着计算机计算能力的提高，计算流体力学（Computational Fluid Dynamic，CFD）依靠其独特的优势，被广泛的应用于多相流研究领域。

在这里，有必要对多相系统进一步进行区分。多相系统通常可以分为单分散系统以及多分散系统 [112]。在单分散系统（mono-disperse）中，离散相的属性为均一的。例如粒子具有相同的大小、温度、以及传输速度。在多分散系统（poly-disperse）中，离散相可以具有不同的属性，例如粒子可以具有不同的大小、温度或者速度。如图4.7所示，左图中的气泡可以近似分为是粒径单一的，因此可近似认为是单分散系统。右图中气泡的粒径差别非常大，可认为是多分散系统。另外一种重要的多分散系统为颗粒的其他属性均相同，但传输速度不同（如图4.8所示），例如大的颗粒移动速度较慢，小的颗粒移动速度较快。这种速度的非均匀分布，对多分散系统数学模型的描述产生了一定的挑战。在实际的工程应用中，单分散系统非常少见。但描述多分散系统的模型通常在数值上难以求解，因此多相流的研究一直是国际上的研究热点问题。

多相流体力学和多相计算流体力学同样也存在区别。多相流体力学侧重研究流动的现象，如微气泡的气泡减阻机理 [110]、颗粒团聚引致湍流的物理描述 [29]、液滴气泡破碎的原因 [173, 64]。多相计算流体力学侧重研究如何求解多相流的数值模型，如双流体模型的双曲特征问题 [126]、界面重构的精准性问题 [138]、群体平衡模型的数值求解问题 [83]、颗粒动力学模化问题 [41]。从研究手段的角度来区分，多相流体力学主要通过实验研究自然界本质物理过程，多相计算流体力学主要通过数学手段和计算机研究如何求解多相流模型并编程植入以及验证。通常，多相流体力学的研究者会提出相应的多相流模型，然后多相计算流体力学研究者会将这些模型数值求解，并进行 CFD 模拟并验证。

4.2.1 微观模型

流体由分子组成。分子做随机热运动，且分子间距比分子尺度大得多。流体力学研究流体的运动，从对流体的解析层级角度来考虑。多相流体力学模型可以分为微观模型（Microscopic model）、介尺度模型（Mesoscopic model）以及宏观模型（Macroscopic model）。如图4.9所示，Sundaresan 等 [162] 将微观模型、介尺度模型以及宏观模型的应用尺度定位为毫米级、厘米级和米级。依据研究关注点的不同，应当使用不同的模型。例如，如果读者想研究小颗粒外部的边界层行为（如图4.10所示），那么必然要使用微观模型。如果读者只是关心宏观的相分数，那么使用宏观模型就可以。



(a) (b)

图 4.7: 近似单分散系统和多分散系统示意图。a): 近似单分散系统，b): 多分散系统。

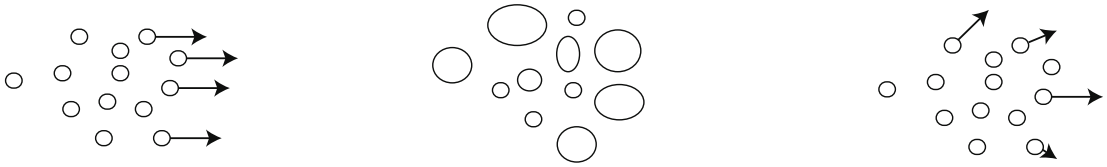


图 4.8: 近似单分散系统和多分散系统示意图。左: 所有粒子具有相同的特性（单分散），中：粒子的直径不同（多分散），右：粒子的移动速度不同（多分散）。

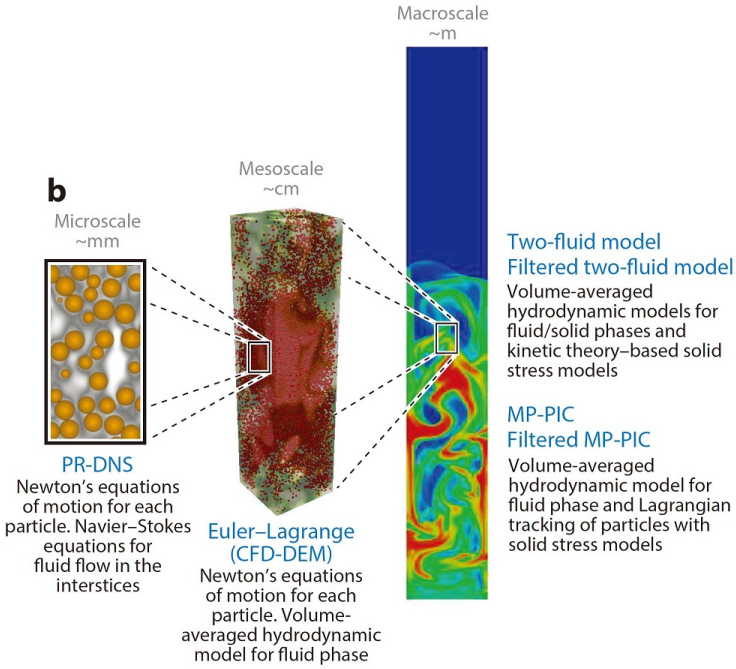


图 4.9: 多相系统的多尺度描述 [162]。

两相流中的微观模型近似为多相流的直接模拟（Direct Numerical Simulation，DNS）。多相流领域的 DNS 和单相流领域的 DNS 略有区分，单相流的 DNS 是指不调用任何模化方法直接采用高阶空间格式以及高阶时间格式的直接模拟[13]，求解方法通常为谱方法以及有限差分法。多相流的 DNS 对求解精度并没有这么高的要求，主要通过求解的空间分辨率进行定义。在多相流的 DNS 中，直接求解 Navier-Stokes（N-S）方程。不需要任何对界面的模化，不需要任何的附加模型。例如，在气固流动中，如果颗粒形状和位置固定，微观模型需要通过网格勾勒出相界面的拓扑结构，进一步的通过求解 N-S 方程获得流场结果，这种方法也被称之为颗粒解析的 DNS（Particle-Resolved DNS，PR-DNS）[165]。作为微观模型，PR-DNS 不需要对颗粒和流体间的作用力进行模化，但只能模拟小规模的系统（大约 10^5 个颗粒 [162]）。主要用于验证介尺度模型和宏观模型中的子模型。

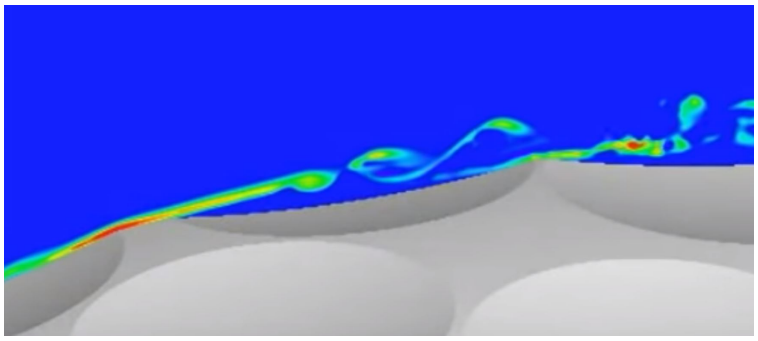


图 4.10: 颗粒表面的边界层行为。

对于其他的流型如气液两相流，由于气液界面是变化的，因此微观模型需要能够对界面自发地进行捕获（如图4.11所示）。后者，这种相界面发生变形的直接模拟，往往比颗粒位置固定的气固流动模拟更为复杂。精准的预测相界面的变形需要对方程中的表面张力、粘度以及相间的密度进行合理的处理。现存的处理方法主要为体方法（Volume method）和面方法（Surface method），前者也被称之为界面捕获法（Surface capturing method），后者也被称之为界面拟合法（Surface fitting method）[174]。在界面捕获法中，界面通过变量重构而来。比较著名的界面捕获法为体分数法（Volume of Fluid，VOF）[69]。在界面拟合法中，界面通过标记点被直接拟合标记。比较著名的界面拟合法有水平集方法（Level-set method）[163] 以及界面跟踪法（Front-tracking method）[175]。

微观模型对网格分辨率和时间解析的要求非常高。例如在对颗粒表面边界层进行微观模型模拟的时候，可能会要求对颗粒表面的涡旋进行解析。解析这些小尺度涡旋需要非常高的计算资源。对于气液体系，微观模型目前只能用于模拟几十几百个数量级的气泡流动。目前，微观模型主要用于小微尺度的流动。虽然微观模型消耗的计算机资源过高，但是下文中即将提到的介尺度模型很大一部分依赖于微观模型的研究工作。例如，对气固体系的

[13] 也有很多文献使用二阶有限体积法做 DNS，在这种情况下没有进行模化，但是精度较低，因此通常被称之为 quasi-DNS。

直接模拟可以预测一些实验很难监控的物理量，如实时的升力、曳力、虚拟质量力等 [18]。因此，多相流的微观模拟，将在未来依然是一个学术上的研究热点问题。

4.2.2 宏观模型

宏观模型研究多相流体的宏观运动。研究的对象不是粒子本身，而是从这些粒子演变出来的物质模型，也即将整个粒子体系看作是连续介质。宏观模型认为粒子无间隙的分布于整个空间中。粒子的变量为空间以及时间的连续函数。宏观模型的变量为考虑大量粒子平均后的统计变量。既然要做平均，那么就涉及到平均的尺度问题。尺度选择过大，宏观变量会受到周围的影响而改变。尺度选择过小，变量会因为分子热运动导致变化。在多相计算流体力学中，宏观模型的尺度通常和粒子的尺度有关，也和粒子的运动尺度有关。粒子的尺度越大，对应的宏观模型的尺度就要增加。粒子的运动尺度很大，宏观模型的尺度也要增加。在错误的尺度上使用宏观模型进行分析，将产生不合理的结果。

讨论宏观模型的非适用性，需要定义克努森数（Knudsen number，Kn）。如图4.12所示，在分子动力学领域，克努森数可以理解为粒子平均自由程和流体特征尺度的比值。在多相流领域，克努森数主要用于描述粒子间的碰撞作用相对于自由传输的重要性。在克努森数较小的时候，粒子间的碰撞起主要作用，因此由于较为频繁的碰撞作用导致粒子的速度分布较为均衡，满足连续性假定。在克努森数大于 0.5 的时候，颗粒间的碰撞可以忽略，连续介质假定不再适用，在这种情况下使用 CFD 的宏观模型（如 N-S 方程）可能会引起错误的结果。在图4.13可以看出，在克努森数较小的情况下，N-S 方程是适用的。在克努森数较大的情况下，N-S 方程将不再适用。类似的，如果将空气分子当做离散的粒子，在稀薄气体领域，由于空气分子近似无碰撞，因此 N-S 方程也将失效，需要调用其他适用的模型。

在宏观模型中，连续相和离散相均在欧拉框架下被阐述。若考虑一个离散相和一个连续相，相应的模型被称之为双流体模型，也即欧拉-欧拉模型 [43]。若考虑多个离散相，相应的模型被称之为多流体模型。不同于下一节讨论的介尺度模型，宏观模型的隐性假定是每一相都可以通过水动力模型（Hydrodynamics model）来模化[14]。

宏观模型的未封闭项至少有两项：1）界面动能/质量传递项：在微观模型中，界面的传递不需要模化，而是通过相边界条件来进行计算。在宏观模型中，相边界不能被解析，因此宏观模型的界面传递通常从微观模型进行修正而来。2）应力张量项：离散相的速度波动并不一定产生于湍流，也有可能产生于源项。这类似于分子模拟的动力学理论-压力张量起源于分子的速度波动。在气-固流动中，宏观模型中的速度波动会导致产生一个附加的变量：颗粒应力张量（Granular stress tensor）。这些附加的压力张量以及颗粒应力张量并不是起源于湍流[15]。这两个未封闭项的处理难易不同。第一种未封闭项可以通过考虑更多的微观

[14] 水动力模型：克努森数非常小的情况下通过 Chapman-Enskog 模型进行展开的低阶矩模型。

[15] 一些文章中指出将这些张量称之为湍流应力张量是不正确的 [52]。

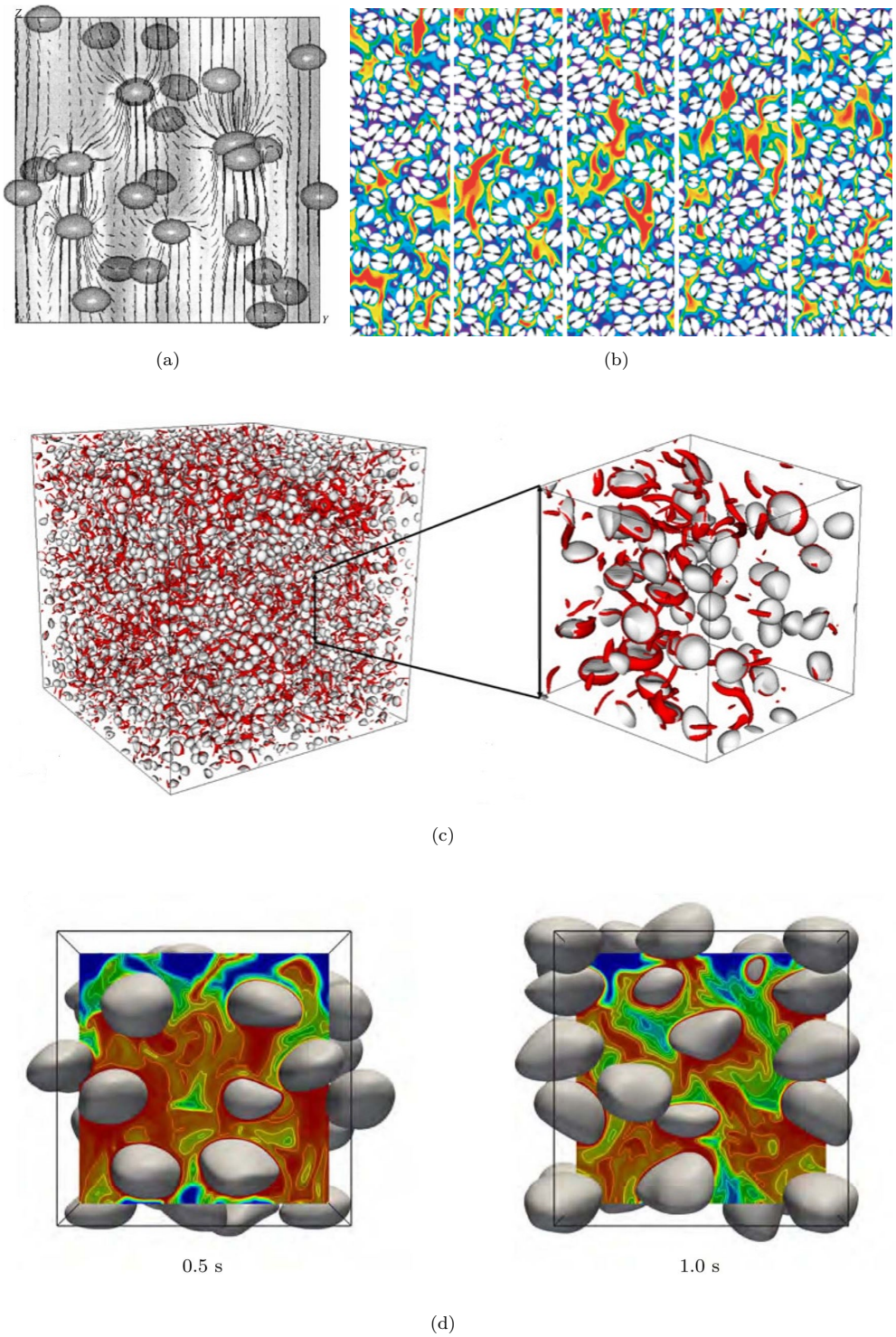


图 4.11: a): 使用 front tracking 直接模拟气泡形状 [172]。b): 直接模拟颗粒间隙的流体流动 [40]。c): 在雷诺数为 83 的情况下使用 VOF 模拟的共 3130 个液滴 [42]。d): 使用 front tracking 直接模拟气泡传质 [139]。

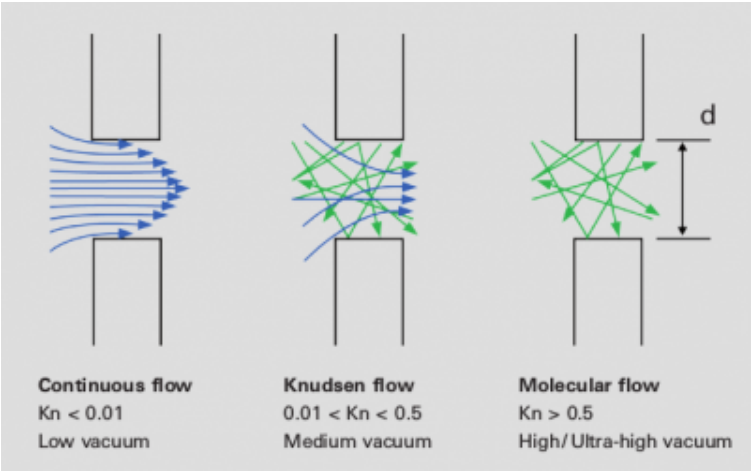


图 4.12: 克努森数的定义与示意图。在克努森数小于 0.01 的情况下，粒子频繁的碰撞近乎均衡，满足连续性介质假定。在克努森数大于 0.5 的情况下，粒子近乎无碰撞，不满足连续性介质假定。在克努森数介于之间的时候，处于一种过渡状态。

因素来提高模型的封闭精度。第二种的封闭并不容易。通常的做法是在模型中引入一个湍流源项，如气泡引致湍流（Bubble Induced Turbulence, BIT）。

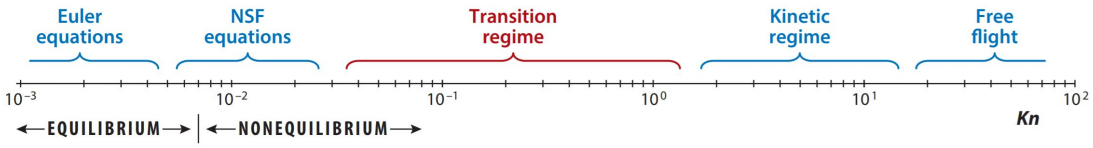


图 4.13: 依据克努森数定义的方程适用性 [169]。

4.2.3 介尺度模型

微观模型通常包含大量的流动信息，这些信息对于大尺度的工业设备并不是必要的。在微观模型和宏观模型之间，还存在介尺度模型。目前科学的研究领域，已经深入小到分子原子，大到行星恒星。但对于介尺度领域的研究还较为空白 [101]。介尺度模型可通过引入介尺度变量来减少这些流动变量的自由度。例如，考虑一个颗粒相，这些颗粒相具有不同的体积和不同的速度波动（即便是相同体积的颗粒也可能具有不同的速度波动）。介尺度模型不需要考虑每个颗粒的速度波动差异，其认为每个相同体积的颗粒具有同样的速度波动大小（一种条件速度分布）。介尺度模型是微观模型与宏观模型之间的一种模型。所有的宏观模型都可以从介尺度模型或微观模型获得。进一步可以区分为两条路径：

- 直接从微观模型推导宏观模型；
- 从微观模型推导介尺度模型，然后从介尺度模型推导出宏观模型；

若直接从微观模型推导宏观模型，则仅仅需要调用体平均（或集合平均技术）来获得矩方程，并通过本构方程来进行直接封闭 [43]。这种方法只涉及到物理方面的建模。在这里需要注意的是，宏观模型中通常将矩具体化为平均动量、平均质量等。若按照第二条路线，在从微观模型推导至介尺度模型的时候同样需要物理方面的建模，在从介尺度模型推导至宏观模型的时候则需要数学方面建模来进行封闭（如封闭介尺度变量的矩）。例如如果从粒子的普适性群体平衡模型出发，0 阶矩表示粒子的数量浓度，1 阶矩表示粒子的平均动能。因此，在第二条路线上，同时考虑了粒子的物理行为以及数学特征。目前在多相流领域，较为受欢迎的介尺度模型为欧拉-拉格朗日模型。同时，作为另一种介尺度模型，通过 CFD 和群体平衡模型的耦合进行求解也越来越引起研究学者的关注 [112, 99, 98, 94, 96, 56, 95, 93, 100]。

不同于宏观模型，欧拉-拉格朗日模型直接跟踪粒子的运动轨迹，并直接求解粒子间的碰撞。若继续考虑颗粒的形状、摩擦、粘连等物理属性，这种直接跟踪粒子的计算方法被称之为离散元法（Discrete Element Method，DEM）。将 CFD 和离散元法进行耦合，即 CFD-DEM。CFD-DEM 被大量的用于研究介尺度下的流动，尤其是气-固流动行为。但 CFD-DEM 对计算资源要求极高，通常仅仅被用于小尺度的模拟。粒子数量一般也小于百万个。

4.2.4 多尺度模型的适用性

各种尺度的模型由于模化的程度不同，具有不同的适用性。在这里通过一些多相体系的例子，讨论应该选用什么样的模型。例如在此处的讨论中，用户需要模拟气泡的聚并和破碎。这是一种介尺度层级的现象，因此需要调用微观模型或介尺度模型。宏观模型并不能够预测类似的行为。在此处的讨论中，用户需要同时模拟气泡以及颗粒的行为，并考虑气泡和颗粒的界面交互作用。用户原本使用微观模型跟踪气泡的界面，同时调用拉格朗日模型处理颗粒行为。但需要注意的是，微观模型需要保证足够网格分辨率才能捕获气泡的界面（如一个三维的气泡内存在 90 个网格），同时介尺度模型需要保证网格充分的大（如一个网格内可包含 10 个颗粒）。在同一个计算域上调用不同尺度的模型，可能会出现问题。

总体来讲，在同一个计算域上，需要优先保证模型的尺度小于需要预测的现象尺度。但需要注意的是，在使用宏观模型的时候，很多工作的网格分辨率均和离散相尺寸同量级。同时，在调用网格依赖类湍流模型时，模型的混用也可能需要适配。例如，如果连续相使用大涡模拟类模型，同时离散相选用拉格朗日模型。一方面要保证网格的分辨率足够高来对大尺度涡进行捕获，一方面还要保证网格单元足够大满足欧拉拉格朗日耦合的数学假定。参考图4.14，Milelli et al. 认为颗粒直径 D_p 和欧拉网格分辨率 Δx 的关系应该满足 $D_p/\Delta x < 0.67$ [121]。Liu et al. 研究了欧拉-欧拉/大涡模拟中网格大小以及气泡直径对结果的影响 [108]。参考图4.15中的研究方法可以看出对于使用混合方法模拟多相体系，其中的网格分

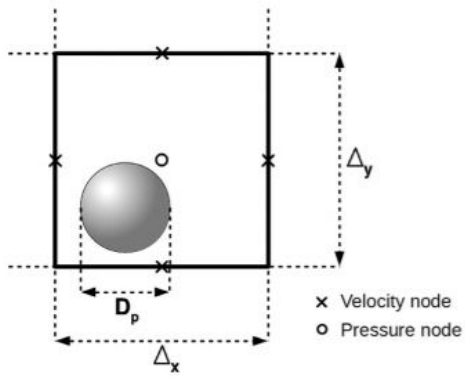


图 4.14: 离散相颗粒直径 D_p 和欧拉网格分辨率 Δx 之间的关系 [121]。

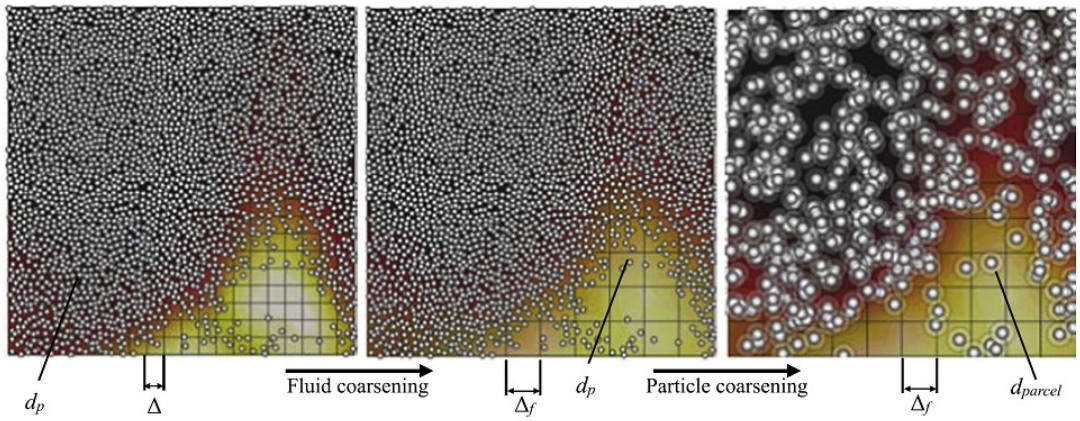


图 4.15: 左：细网格欧拉-拉格朗日模拟。中：糙网格欧拉-拉格朗日模拟示意图。右：MP-PIC 方法 [125]。

辨率标准还需要进行大量研究。感兴趣的读者可参考其他文献资料 [128, 121]。

4.3 气体动理学与矩方法

4.3.1 分子的自由度

气体分子的热力学变量与自由度高度相关。有 [191]：

$$C_v = \frac{f}{2}R, C_p = \frac{f+2}{2}R, \gamma = \frac{C_p}{C_v} \tag{4.47}$$

其中 f 表示分子的总自由度。总自由度为内部自由度与外部自由度的加和。内部自由度进一步的分为旋转自由度以及震动自由度（尤其是高温情况下不能忽略）。外部自由度主要指的是平动自由度。平动自由度用于描述分子在空间的运动方向，也即空间的维度数。一维空间的平动自由度为 1，三维空间的平动自由度为 3。旋转自由度可以理解为与分子形状相关的量，描述分子如何绕轴旋转。其可以空间维度有关。不管什么维度，单原子分子可以认为是一个点，因此没有旋转自由度。双原子分子类似一个哑铃，因此在三维空间中存在 2 个旋转自由度，在二维空间中存在 1 个旋转自由度。一维空间不管什么样的分子，旋转自由度都为 0。

现进行举例，单原子的气体氦气，没有旋转自由度以及震动自由度[16]。因此三维计算域下氦气的 $f = 3$（仅仅存在外部自由度）。对于双原子分子（例如氧气和氮气、其构成空气组成的 99%），在常温常压下内部自由度为 2，其分别为横向旋转和纵向旋转。在高温下，双原子分子会考虑震动自由度，在这种情况下内部自由度为 3。气体分子的自由度主要影响气体的内能以及比热容，例如对于非高温的三维的双原子分子（空气），有 $f = 5$[17]：

$$C_v = \frac{5}{2}R = 718, C_p = \frac{5+2}{2}R = 1005, \gamma = 1.4 \tag{4.48}$$

类似的，如果是二维，则有

$$C_v = \frac{3}{2}R = 430, C_p = \frac{3+2}{2}R = 718, \gamma = 1.668 \tag{4.49}$$

4.3.2 速度分布函数、玻尔兹曼方程

正如第一章中所指出的，N-S 方程并不是描述流动的最底层的形式。更为底层的数学模型需要考虑气体分子的运动。直接对气体分子运动的分布进行模化的方程为玻尔兹曼方程。其在不同的领域有不同的称呼。例如在群体平衡模型研究领域，被称之为普适性群体

[16] 这不是说明气体不是旋转的，而是说明气体的旋转不会影响热力学特性。但是没有震动自由度确实说明单原子是不会发生震动，只有 2 个以上原子才会发生震动。

[17] 空气的 $R = 287\mathrm{m}^2 \cdot \mathrm{s}^{-2} \cdot \mathrm{T}$。

平衡模型，在空气动力学领域被称之为动理学方程，在喷雾燃烧领域被称之为威廉玻尔兹曼方程。三维的玻尔兹曼方程可以写为[18]

$$\frac{\partial f}{\partial t} + \nabla_{\mathbf{x}} \cdot (\mathbf{U} f) + \nabla_{\mathbf{U}} \cdot (\mathbf{A} f) = \mathbb{C}, \tag{4.50}$$

其中 f 即 $f(t, \mathbf{x}, \mathbf{U})$ 的简写，在这里没有考虑气体分子的内部自由度，因此 f 的自由度变量只有 $\mathbf{x}$，否则其可能还需要考虑内部自由度。表示关于气体分子的速度分布函数，$\mathbf{U}$ 表示气体分子的速度，$\mathbf{A}$ 表示受力引起的加速项，$\mathbb{C}$ 表示碰撞项。具体的，如果把 f 考虑成不随时间变化，且空间均一的，则 $f(\mathbf{U})$ 表示气体分子速度在 $\mathbf{U}$ 附近的概率密度。如果把 $\mathbf{U}$ 做成 x 轴，那么可以把 x 轴离散化，每个小格子就是 $\mathbf{U} - 0.5\mathrm{d}\mathbf{U}$ 至 $\mathbf{U} + 0.5\mathrm{d}\mathbf{U}$。$f(\mathbf{U})$ 就表示气体分子处于 $\mathbf{U} - 0.5\mathrm{d}\mathbf{U}$ 至 $\mathbf{U} + 0.5\mathrm{d}\mathbf{U}$ 之间的概率。参考每单位长度、每单位体积相关定义，每单位速度的概率密度的单位为 $\frac{1}{\mathrm{m/s}} = \frac{\mathrm{s}}{\mathrm{m}}$，这既是 f 的单位。例如，某种情况下，麦克斯韦速度分布在 $\mathbf{U} = (1, 2, 3)$ 的情况下的值是 $0.001\frac{\mathrm{s}}{\mathrm{m}}$。这表示速度在 $\mathbf{U} = (1, 2, 3)$ 附近的分子比例是 0.001。比如一共有 1000 个分子，那么速度在 $\mathbf{U} = (1, 2, 3)$ 附近的分子有 1 个。

在接下来的讨论中，都认为 f 不随着空间与时间变化。那么给定任意一个分布，参考统计学理论，其可以定义 n 阶矩。如果不考虑内部自由度的话，气体分子分布的 0 阶矩即为总共的分子数：

$$m_0 = \int \mathbf{U}^0 f \mathrm{d}\mathbf{U} \tag{4.51}$$

m_0 无单位，表示分子的总数。如果 m_0 乘以单一分子的质量 m_p，则变成了所有分子的总质量。如果进一步考虑每立方体积的 m_0 乘以分子的质量，就变成了质量密度 ρ。接下来为了方便讨论，定义速度分布函数 f 为每单位体积每单位速度的分子概率分布。因此 f 的单位变为 $\frac{\mathrm{s}}{\mathrm{m}^4}$。在这种情况下，0 阶矩再乘以分子质量有质量密度：

$$\rho = m_p m_0 = m_p \int \mathbf{U}^0 f \mathrm{d}\mathbf{U} \tag{4.52}$$

其中 m_p 表示每个分子的质量（假设均一）。同理，速度分布概率函数的 1 阶矩乘以分子质量即为动量密度：

$$m_p m_1 = m_p \int \mathbf{U}^1 f \mathrm{d}\mathbf{U} \tag{4.53}$$

速度分布概率函数的 2 阶矩乘以分子质量没有物理意义。但再乘以一个 0.5 即为能量密度：

$$\frac{1}{2} m_p m_2 = m_p \int \frac{1}{2} \mathbf{U}^2 f \mathrm{d}\mathbf{U} \tag{4.54}$$

需要注意的是，动量密度以及能量密度取决于 f 的具体形式。

[18] 其一维形式更好理解，读者应该自行将其展开为一维形式。

下面看 f 的具体形式。麦克斯韦速度分布被认为是气体分子在均衡状态下的速度分布。假定 f 为三维高斯分布 [179][19]：

$$f=\frac{\rho}{(2\pi)^{3/2}\sqrt{|\boldsymbol{\theta}|}}\exp\left(-\frac{1}{2}\begin{bmatrix}u-\bar{u}\\ v-\bar{v}\\ w-\bar{w}\end{bmatrix}^T\cdot\boldsymbol{\theta}^{-1}\cdot\begin{bmatrix}u-\bar{u}\\ v-\bar{v}\\ w-\bar{w}\end{bmatrix}\right) \tag{4.55}$$

其中 $\boldsymbol{\theta}$ 为对称二阶张量：

$$\boldsymbol{\theta}=\begin{pmatrix}\theta_{11} & \theta_{12} & \theta_{13}\\ \theta_{21} & \theta_{22} & \theta_{23}\\ \theta_{31} & \theta_{32} & \theta_{33}\end{pmatrix} \tag{4.56}$$

其中 $\theta_{ij}=\theta_{ji}, i\neq j$。且有压力：

$$p=\rho\frac{\theta_{11}+\theta_{22}+\theta_{33}}{3} \tag{4.57}$$

麦克斯韦分布为高斯分布的各向同性形式。例如，若假定 $\boldsymbol{\theta}$ 的非对角元素为 0，同时认为 $\theta_{11}=\theta_{22}=\theta_{33}=\theta$。高斯分布则演变为麦克斯韦分布。同时，如果考虑一维的情况，高斯分布即麦克斯韦分布。

二维情况下的高斯分布可以写为

$$f=\frac{\rho}{2\pi\sqrt{|\boldsymbol{\theta}|}}\exp\left(-\frac{1}{2}\begin{bmatrix}u-\bar{u}\\ v-\bar{v}\end{bmatrix}^T\cdot\boldsymbol{\theta}^{-1}\cdot\begin{bmatrix}u-\bar{u}\\ v-\bar{v}\end{bmatrix}\right) \tag{4.58}$$

统计学领域更常见的写法是将 $\boldsymbol{\theta}$ 替换为符号 $\boldsymbol{\Sigma}$（描述各个维度之间的相关性）：

$$\boldsymbol{\theta}=\begin{pmatrix}\theta_{11} & \theta_{12}\\ \theta_{21} & \theta_{22}\end{pmatrix}=\boldsymbol{\Sigma}=\begin{pmatrix}\sigma_1^2 & \rho'\sigma_1\sigma_2\\ \rho'\sigma_1\sigma_2 & \sigma_2^2\end{pmatrix} \tag{4.59}$$

在这种情况下，$\boldsymbol{\Sigma}$ 中的相干系数 ρ' 与 $\boldsymbol{\theta}$ 的关系为

$$\rho'=\frac{\theta_{12}}{\sqrt{\theta_{11}\theta_{22}}}, \theta_{11}=\sigma_1^2, \theta_{22}=\sigma_2^2, \tag{4.60}$$

同时，方程(4.58)中 exp 函数内的表达式也可以写为：

$$\begin{bmatrix}u-\bar{u}\\ v-\bar{v}\end{bmatrix}^T\cdot\boldsymbol{\Sigma}^{-1}\cdot\begin{bmatrix}u-\bar{u}\\ v-\bar{v}\end{bmatrix}=\frac{1}{1-\rho'^2}\left(\frac{(u-\bar{u})^2}{\sigma_1^2}+\frac{(v-\bar{v})^2}{\sigma_2^2}-\frac{2\rho'(u-\bar{u})(v-\bar{v})}{\sigma_1\sigma_2}\right) \tag{4.61}$$

(4.58)中的分母为：

$$2\pi\sqrt{|\boldsymbol{\theta}|}=\sigma_1\sigma_2\sqrt{1-\rho'^2} \tag{4.62}$$

[19]之所以出现 $u-\overline{u}$ 的形式，是因为 $\overline{u}$ 是平均后的速度，可能是 1，表示空气分子宏观速度是 1 米每秒。但在这个宏观速度的基础上，其也会波动。这个波动符合高斯分布。

这样，方程(4.58)可以重写为（类似的方程写法出现在 [97, 112]）：

$$f = \frac{\rho}{2\pi\sigma_1\sigma_2\sqrt{1-\rho'^2}} \exp\left(-\frac{1}{2(1-\rho'^2)}\left(\frac{(u-\bar{u})^2}{\sigma_1^2} + \frac{(v-\bar{v})^2}{\sigma_2^2} - \frac{2\rho'(u-\bar{u})(v-\bar{v})}{\sigma_1\sigma_2}\right)\right) \tag{4.63}$$

一维情况下的高斯分布可以写为：

$$f = \frac{1}{(2\pi)^{1/2}\sqrt{|\boldsymbol{\theta}|}} \exp\left(-\frac{1}{2}(u-\bar{u})\cdot\boldsymbol{\theta}^{-1}\cdot(u-\bar{u})\right) \tag{4.64}$$

其中

$$\boldsymbol{\theta} = \theta_{11} = \sigma_1^2 \tag{4.65}$$

因此一维高斯分布可以简写为

$$f = \frac{1}{\sigma_1(2\pi)^{1/2}} \exp\left(-\frac{(u-\bar{u})^2}{2\sigma_1^2}\right) = \frac{1}{\sqrt{2\pi\theta_{11}}} \exp\left(-\frac{(u-\bar{u})^2}{2\theta_{11}}\right) \tag{4.66}$$

定义

$$\theta_{11} = \frac{kT}{m} \tag{4.67}$$

其中 k 玻尔兹曼常数 $k = 1.380649 \times 10^{-23}$，其单位为 $\mathrm{kg{\cdot}m^2{\cdot}s^{-2}{\cdot}K^{-1}}$。一维高斯分布在气体动理学领域有更常见的写法：

$$f = \sqrt{\frac{m}{2\pi kT}} \exp\left(-\frac{m(u-\bar{u})^2}{2kT}\right) \tag{4.68}$$

m 为摩尔分子质量。对于静止的气体，$\bar{u} = 0$，有

$$f = \sqrt{\frac{m}{2\pi kT}} \exp\left(-\frac{mu^2}{2kT}\right) \tag{4.69}$$

如图4.16所示，静止气体的一维高斯分布是一个中心点为 0 的分布。其表明如果气体分子仅仅在一个维度方向移动的话，大部分气体分子是静止的，同时存在正向移动以及负向移动的粒子，二者互相平衡。静止粒子数量的多少，取决于 θ_{11}，也即温度 T。如果温度越高，θ_{11} 越大，分布约平，静止的分子越少。对于 $\bar{u} = -2$ 的情况，即为宏观上气体在向 x 负向移动，则气体分子的分布呈现图4.16绿线的情况，中心点在-2。最后要注意，一维情况下麦克斯韦分布与高斯分布等价。

在考虑气体分子内部自由度的情况下，f 分布内部存在内部自由度的变量。参考方程(4.68)，在一维情况下高斯分布可以写为：

$$f = \left(\frac{m}{2\pi kT}\right)^{\frac{N+1}{2}} \exp\left(-\frac{m}{2kT}\left((u-\bar{u})^2 + \xi_1^2 + \ldots + \xi_N^2\right)\right) \tag{4.70}$$

令

$$\lambda = \frac{m}{2kT} \tag{4.71}$$

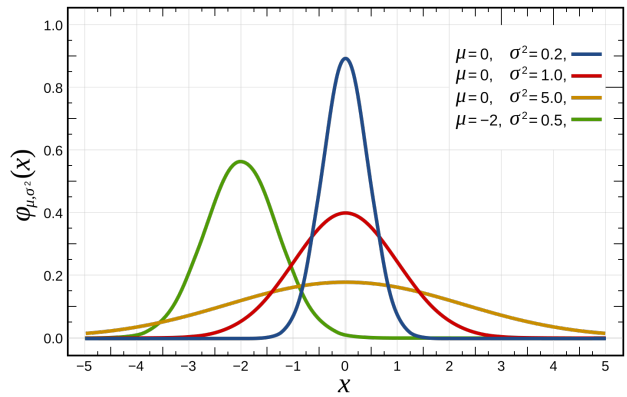


图 4.16: 一维高斯分布示意图（来源维基百科）。

也有

$$f=\left(\frac{\lambda}{2}\right)^{\frac{N+1}{2}}\exp\left(-\frac{\lambda}{2}\left((u-\bar{u})^2+\xi_1^2+...+\xi_N^2\right)\right) \tag{4.72}$$

可见在这种情况下，f 变成了一个多变量的分布，因此在做积分的情况下需要进行多重积分。其实在二维以及三维的情况下，f 也为多变量分布，需要进行多重积分取矩。更详细的介绍请参考 [192]。

4.3.3 矩方程

在下文中考虑内部自由度为 0 的情况。如果对玻尔兹曼方程做积分（忽略碰撞以及加速项），可以有：

$$\begin{aligned}
&\frac{\partial m_0}{\partial t}+\nabla\cdot(\mathbf{U}m_1)=0,\\
&\frac{\partial m_1}{\partial t}+\nabla\cdot(\mathbf{U}m_2)=0,\\
&\frac{\partial m_2}{\partial t}+\nabla\cdot(\mathbf{U}m_3)=0,\\
&\qquad\qquad\qquad\qquad\cdots\\
&\frac{\partial m_n}{\partial t}+\nabla\cdot(\mathbf{U}m_{n+1})=0
\end{aligned} \tag{4.73}$$

会发现方程矩的数量永远大于方程数量存在封闭问题。因此不能进行求解。下面的讨论可以看出，如果 f 选择某种具体的分布，则可以将其进行封闭。

不考虑内部自由度，假定有一维形式的玻尔兹曼方程（其中 f 乘以了分子质量）：

$$\frac{\partial f(u)}{\partial t}+\frac{\partial uf(u)}{\partial x}=0 \tag{4.74}$$

第一种情况（最简单的情况），可以假定 $f(u)$ 的形态为单值函数：

$$f(u)=\overline{\rho}\delta(u-\overline{u}) \tag{4.75}$$

其中 $\bar{u}$ 表示气体分子运动的平均速度，$\overline{\rho}$ 表示密度。其仅具有两个未知量，即 $\overline{\rho}, \bar{u}$。其可以通过 2 个传输方程进行求解。如果对方程(4.74)取矩有

$$\begin{aligned} \partial_t m_0 + \partial_x m_1 = 0 \\ \partial_t m_1 + \partial_x m_2 = 0 \end{aligned} \tag{4.76}$$

其中存在 3 个变量，问题是我们不能将 m_2 写成 m_0, m_1 的函数。依据矩的定义，间断的 $f(u)$ 的 0 阶矩表示为：

$$m_0 = \int f(u)\mathrm{d}u \equiv \overline{\rho} \tag{4.77}$$

1 阶矩表示为：

$$m_1 = \int f(u)u\mathrm{d}u \equiv \overline{\rho}\bar{u} \tag{4.78}$$

2 阶矩表示为：

$$m_2 = \int f(u)u^2\mathrm{d}u \equiv \overline{\rho}\bar{u}\bar{u} \tag{4.79}$$

将矩的原始变量形式代入到方程(4.76)，有

$$\begin{aligned} \partial_t \overline{\rho} + \partial_x \overline{\rho}\bar{u} = 0 \\ \partial_t \overline{\rho}\bar{u} + \partial_x \overline{\rho}\bar{u}\bar{u} = 0 \end{aligned} \tag{4.80}$$

方程(4.80)存在 2 个方程，2 个变量因此封闭。方程(4.80)中第一个方程与第二个方程分别对应连续性方程和无压力无粘性力的动量方程。整个方程系统也被称之为无压力气体动力学方程。在矩方法研究领域也被称之为二阶矩模型 [32]。方程(4.80)呈现弱双曲特征。

不考虑内部自由度，同样可以假定 $f(u)$ 为双值分布函数，有：

$$f(u) = \rho_1\delta(u - u_1) + \rho_2\delta(u - u_2) \tag{4.81}$$

其具有 4 个未知量，即 ρ_1, u_1, ρ_2, u_2。其可以通过 4 个传输方程进行求解。如果对方程(4.74)取矩有

$$\begin{aligned} \partial_t m_0 + \partial_x m_1 = 0 \\ \partial_t m_1 + \partial_x m_2 = 0 \\ \partial_t m_2 + \partial_x m_3 = 0 \\ \partial_t m_3 + \partial_x m_4 = 0 \end{aligned} \tag{4.82}$$

方程(4.82)构成四矩模型。问题依旧是 4 个方程存在 5 个未知变量，其中的 m_4 不能封闭。但如果将 m_k 进行展开，有：

$$m_k = \rho_1 u_1^k + \rho_2 u_2^k \tag{4.83}$$

m_4 可以表示为 ρ_1, ρ_2, u_1, u_2 的函数，其他的矩也采用类似的处理方法。则方程得以封闭。相对于低阶矩模型（如二阶矩模型），高阶矩模型（如四矩模型）可以调用更多的速度节点

（如 u_1, u_2）。在颗粒流领域，只有高阶矩模型才能预测颗粒轨迹交叉。在这里要强调，一般所谓的 NS 方程，只有一个速度分布，即速度为单值的，这是因为其假设颗粒的碰撞非常频繁。因此 NS 方程不能预测颗粒轨迹交叉，也不能预测稀薄流。

在这里看出，f 的不同，m_k 的计算方式也不同。

4.3.4 欧拉方程、五矩方程、十矩方程

上文将 $f(u)$ 假定为离散的狄拉克函数（思想即为积分矩方法）。本节考虑更为现实的 $f(u)$ 形态。不考虑内部自由度，一维的动理学方程可以写为：

$$\frac{\partial f}{\partial t} + \frac{\partial uf}{\partial x} = \frac{1}{\tau}(g - f) \tag{4.84}$$

f 为粒子分布函数，g 表示平衡情况下的分布函数（即麦克斯韦分布），τ 表示碰撞时间尺度。在这里假定是碰撞均衡的（$f = g$）[20]。这样，方程(4.84)演变为

$$\frac{\partial g}{\partial t} + \frac{\partial ug}{\partial x} = 0 \tag{4.85}$$

首先对方程(4.85)取零阶矩有：

$$\frac{\partial m_0}{\partial t} + \frac{\partial m_1}{\partial x} = 0 \tag{4.86}$$

取一阶矩有：

$$\frac{\partial m_1}{\partial t} + \frac{\partial m_2}{\partial x} = 0 \tag{4.87}$$

取二阶矩有：

$$\frac{\partial m_2}{\partial t} + \frac{\partial m_3}{\partial x} = 0 \tag{4.88}$$

方程(4.86)、(4.87)、(4.88)共有三个矩方程，在三维情况下，有五个矩方程。因此，其也被称之为五矩模型。

依据矩的定义，首先看 g 的 0 阶矩，其定义为：

$$m_0 = \int g \mathrm{d}u = \rho \tag{4.89}$$

再次，1 阶矩定义为：

$$m_1 = \int gu \mathrm{d}u = \rho\bar{u} \tag{4.90}$$

2 阶矩定义为[21]：

$$m_2 = \int gu^2 \mathrm{d}u = \rho\bar{u}\bar{u} + \rho\frac{N+1}{2\lambda} \tag{4.91}$$

[20] $f = g$ 本质是 Chapman-Enskog（CE）方法的 0 阶展开。进一步的，假定一个一维的 f 分布，CE 方法的 1 阶展开认为 $f = g - \mathrm{Kn}\left(\frac{\partial g}{\partial t} + u\frac{\partial g}{\partial x}\right)$。在这种情况下，矩方程将演变为 NS 方程。Kn 为一个无量纲数，可以用来衡量稀薄程度，其为分子自由程与特征尺度的比值。在日常生活中，Kn 是一个非常小的数，因此可以忽略高阶项演变成欧拉方程或 NS 方程。在 2 阶、3 阶展开的情况下，矩方程将演变为 Burnett 方程与超 Burnett 方程。二者主要处理转变区与稀薄流。但越高阶的展开越不稳定。因此通常仅仅进行 0 阶（欧拉方程）1 阶（方程）展开。

[21] 注意这里的二阶矩的定义区别于方程(4.79)。因为这里的 g 为麦克斯韦分布，而不是狄拉克函数。

注意，其为 g 的二阶矩的数学定义。其中的 $N+1$ 中的 $N=0$（不考虑自由度），1 是表示一维。其可以进一步展开：

$$m_2 = \int gu^2 \mathrm{d}u = \rho\bar{u}\bar{u} + \rho\frac{1}{2\lambda} = \rho\bar{u}\bar{u} + \frac{\rho kT}{m} = \rho\bar{u}\bar{u} + \rho\theta = \rho\bar{u}\bar{u} + p \tag{4.92}$$

其中 $\rho\theta$ 为一种温度相关量，在颗粒流中表示颗粒温度，在空气动力学中表示压力 p[22]。另一方面，有比总能 E（机械能与比内能之和）为[23]：

$$\rho E = \frac{1}{2}(\rho\bar{u}\bar{u} + p) \tag{4.93}$$

也即：

$$\rho E = \frac{1}{2}m_2 \tag{4.94}$$

$$p = 2\rho E - \rho\bar{u}\bar{u} \tag{4.95}$$

同时也有

$$\rho E = \frac{1}{2}\left(\rho\bar{u}\bar{u} + \frac{\rho}{2\lambda}\right) \tag{4.96}$$

即

$$\lambda = \frac{1}{2}\frac{\rho}{2\rho E - \rho\bar{u}\bar{u}} = \frac{1}{2}\frac{\rho}{p} \tag{4.97}$$

在文献中并没有找到附加自由度的高斯分布的 3 阶矩的直接定义。但是如果忽略自由度，其可以表示为：

$$m_3 = \int gu^3 \mathrm{d}u \equiv \rho\bar{u}\bar{u}\bar{u} + 3\rho\bar{u}\theta \tag{4.98}$$

若对三阶矩乘以 1/2，有另外一种形式为：

$$\frac{1}{2}m_3 = \frac{1}{2}(\rho\bar{u}\bar{u}\bar{u} + 3\rho\theta\bar{u}) = \frac{1}{2}(\rho\bar{u}\bar{u} + p)\bar{u} + p\bar{u} = (\rho E + p)\bar{u} \tag{4.99}$$

综合上述方程，有一维欧拉方程：

$$\begin{aligned} \frac{\partial\rho}{\partial t} + \frac{\partial\rho\bar{u}}{\partial x} &= 0 \\ \frac{\partial\rho\bar{u}}{\partial t} + \frac{\partial\rho\bar{u}\bar{u}}{\partial x} &= -\frac{\partial p}{\partial x} \\ \frac{\partial\rho E}{\partial t} + \frac{\partial\rho E\bar{u}}{\partial x} &= -\frac{\partial p\bar{u}}{\partial x} \end{aligned} \tag{4.100}$$

第一个方程为质量密度方程，第二个方程为动量密度方程，第三个方程为能量密度方程。同理，在三维情况下，可获得五个矩方程（三维欧拉方程），分别对应密度、三个方向的速度方程以及能量方程。在五矩方程中，压力为各向同性（压力二阶张量的非对角线元素为

[22] 由于 $\theta = kT/m$，有 $p = \rho/(2\lambda)$。

[23] 通常的能量方程中比总能 E 为比机械能与比内能之和 $E = K + e = \frac{1}{2}\bar{u}^2 + e$。其中 e 经过推导进一步可以表示为 $e = \frac{p}{\rho(\gamma-1)}$。因此 E 与压力也是存在关系的。进一步的，其中的 γ 跟空间维度以及气体分子原子数以及运动自由度有关，在一维单原子情况下 $\gamma = 3$，$E = \frac{1}{2}\bar{u}^2 + \frac{p}{\rho}$。

0）。除了麦克斯韦分布，$f(u)$ 还可以假定为高斯分布，在这种情况下，三维情况下方程演变为 10 个矩方程，也被称之为十矩方程，十矩方程中压力为各向异性（压力二阶张量的非对角线元素不为 0）。总之，针对动理学方程，若假定不同的 f 分布函数，可以获得不同的矩方程（宏观方程）。同时可以看出，NS 方程仅仅是矩方程在特殊情况下（假定 f 按照 CE 一阶展开）的一个表现形式。

在考虑内部自由度的情况下会更加复杂，例如在进行积分的情况下，需要对速度以及内部自由度变量进行积分，也即变成了多维积分。同时能量密度的积分形式也变成了 $\frac{1}{2}\int(u^2+\xi^2)g\mathrm{d}u\mathrm{d}\xi$。在这里不进行更深层次的拓展。

回看方程(4.68)，其可以从 $-\infty$ 至 0 取矩，也可以从 0 至 ∞ 取矩。从 $-\infty$ 至 0 取 0 阶矩表示向 x 负向运动的所有气体分子的质量密度，取 1 阶矩表示向 x 负向运动的所有气体分子的动量密度。反向依然。相应的矩可以表示为：

$$m_0^{>0} = \frac{1}{2}\mathrm{erfc}(-\sqrt{\lambda}\bar{u}) \tag{4.101}$$

$$m_1^{>0} = \bar{u}m_0^{>0} + \frac{1}{2}\frac{e^{-\lambda\bar{u}^2}}{\sqrt{\pi\lambda}} \tag{4.102}$$

$$m_{k+2}^{>0} = \bar{u}m_{k+1}^{>0} + \frac{k+1}{2\lambda}m_k^{>0} \tag{4.103}$$

$$m_0^{<0} = \frac{1}{2}\mathrm{erfc}(\sqrt{\lambda}\bar{u}) \tag{4.104}$$

$$m_1^{>0} = \bar{u}m_0^{<0} - \frac{1}{2}\frac{e^{-\lambda\bar{u}^2}}{\sqrt{\pi\lambda}} \tag{4.105}$$

$$m_{k+2}^{<0} = \bar{u}m_{k+1}^{<0} + \frac{k+1}{2\lambda}m_k{}^{<0} \tag{4.106}$$

上述矩的定义，在气体动理学格式以及一些动理学通量分裂格式中被大量的使用。在二维或者三维的情况下，将相应的一维的矩进行相乘即可（操作符切分）。

4.4 Von Neumann 稳定性分析与截断误差

4.4.1 变量分离法

对于方程求解，一种方法被称之为变量分离法 [123]。现在用一个特别简单的例子举例说明，考虑下述方程：

$$\frac{\mathrm{d}y}{\mathrm{d}x} = 5xy \tag{4.107}$$

可以进行移项有：

$$\frac{\mathrm{d}y}{y} = 5x\mathrm{d}x \tag{4.108}$$

方程(4.108)可以对方程左右两边同时积分并获得解。这是一个非常简单并且好理解的例子。

下面考虑一个抛物线偏微分方程组：

$$\frac{\partial \phi}{\partial t}=\frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right), t>0, 0<x<1 \tag{4.109}$$

其边界条件与初始条件为：

$$\begin{aligned}\phi(0,t)=\phi(1,t)=0\\ \phi(x,0)=\phi^0(x), 0<x<1\end{aligned} \tag{4.110}$$

若使用分离变量法求解，我们首先假定 ϕ 函数具有下面的形式，即将关于 t 变量的函数与 x 变量的函数分开：

$$\phi(x,t)=f(x)g(t) \tag{4.111}$$

在这种情况下，对 $\phi(x,t)$ 求导数有：

$$\begin{aligned}\frac{\partial \phi}{\partial t}&=f\frac{\partial g}{\partial t}\\ \frac{\partial \phi}{\partial x}&=g\frac{\partial f}{\partial x}\\ \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right)&=g\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right)\end{aligned} \tag{4.112}$$

在这种情况下，方程(4.109)即为：

$$\frac{1}{g}\frac{\partial g}{\partial t}=\frac{1}{f}\frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) \tag{4.113}$$

可以看出，方程(4.113)左侧是一个关于 t 的函数，右侧是关于 x 的一个函数，若想让其对任意 x,t 都满足，则必然其为一个常数。如果我们令这个常数为 $-k^2$，其有解为：

$$g=e^{-k^2t}, f=A\sin kx+B\cos kx \tag{4.114}$$

即

$$\phi(x,t)=e^{-k^2t}\left(A\sin kx+B\cos kx\right) \tag{4.115}$$

其中 A,B 为未知系数。很明显，其中 $\exp(-k^2t)$ 中负号的作用，可以保证 $\phi(x,t)$ 在任何时间 t 下都是有界的。进一步，只要解符合方程(4.115)的形式，都满足条件。因此，下述方程

$$\phi(x,t)=\sum_{m=1}^{\infty}e^{-k_m^2t}\left(A_m\sin k_mx+B_m\cos k_mx\right) \tag{4.116}$$

必然也是解。但其中存在若干的未知系数。这些未知系数可以通过初始条件以及边界条件来求得。

首先依据初始条件 $\phi(0,t)=0$，有：

$$\sum_{m=1}^{\infty}e^{-k_m^2t}B_m=0 \tag{4.117}$$

其解只能是 $B_m = 0$。在这种情况下，方程(4.121)变为：

$$\phi(x,t) = \sum_{m=1}^{\infty} e^{-k_m^2 t} A_m \sin k_m x \tag{4.118}$$

再次，通过 $\phi(1,t) = 0$，有：

$$\sum_{m=1}^{\infty} e^{-k^2 t} A_m \sin k_m = 0 \tag{4.119}$$

其解只能是

$$k_m = m\pi \tag{4.120}$$

其中 m 为整数。这样，方程(4.118)变为：

$$\phi(x,t) = \sum_{m=1}^{\infty} A_m e^{-(m\pi)^2 t} \sin m\pi x \tag{4.121}$$

熟悉傅里叶理论的可以看出方程(4.121)恰巧是傅里叶 sin 级数展开形式。其中的系数 A_m 可以通过初始条件逐级求出。因此有 [5]：

$$A_m = 2\int_0^1 \phi^0(x) \sin m\pi x \mathrm{d}x \tag{4.122}$$

在这种情况下，可获得 $\phi(x,t)$ 函数的精确解。需要注意的是。为了获得 A_m 的值，只有在 $\phi^0(x)$ 非常简单的情况下，方程(4.122)才具有精确解，否则，可能需要进行数值积分来获得数值解。再次，方程(4.121)中存在方程的叠加，只有在进行充分的叠加之后，才可能获得任意的型线。实际上，很难进行无穷次叠加，因此，必然存在误差。可以说，方程(4.121)在完美情况下，即 $\phi^0(x)$ 非常简单，同时进行无穷次叠加的情况下，可以维持精确解的情况。若不能满足，必然会有误差的出现。但是最重要的，获得如此的解是因为使用了变量分离法，但在很多情况下，偏微分方程并不能够使用变量分离法进行精确求解。

还需要注意的是，方程(4.109)还有另外一种解的形式为：

$$\phi(x,t) = e^{-k^2 t} \cdot e^{\mathrm{i}kx} \tag{4.123}$$

其中 i 为虚数单位。对上述方程求导，可以证明其为方程(4.109)的一个精确解。

4.4.2 抛物线方程稳定性分析、无条件稳定

现在对 x 方向的网格进行均分，每个网格单元长度为 Δx，那么，第 j 个网格单元的 x 为 $j\Delta x$。令第 j 个网格单元，第 n 个时间步的 ϕ_j^n 值为 $(t = n, x = j\Delta x)$

$$\phi_j^n = \lambda^n e^{\mathrm{i}kj\Delta x} \tag{4.124}$$

因为 $\phi^{n+1}/\phi_j^n = \lambda$，因此其中的 λ 表示一个增长指数的概念，若方程的解为有界的，必然有

$$|\lambda| = |\phi_j^{n+1}/\phi_j^n| \leqslant 1 \tag{4.125}$$

否则 ϕ 将呈现无穷大。

现在对方程(4.109)的时间项进行一阶离散、空间项进行显性中心离散有：

$$\frac{\phi_j^{n+1} - \phi_j^n}{\Delta t} = \frac{\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n}{(\Delta x)^2} \tag{4.126}$$

参考方程(4.124)，将其代入到方程(4.126)有：

$$\frac{\lambda^{n+1}e^{\mathrm{i}kj\Delta x} - \lambda^n e^{\mathrm{i}kj\Delta x}}{\Delta t} = \frac{\lambda^n e^{\mathrm{i}k(j+1)\Delta x} - 2\lambda^n e^{\mathrm{i}kj\Delta x} + \lambda^n e^{\mathrm{i}k(j-1)\Delta x}}{(\Delta x)^2} \tag{4.127}$$

整理后有：

$$\lambda = 1 + \frac{\Delta t}{(\Delta x)^2}\left(e^{\mathrm{i}k\Delta x} - 2 + e^{-\mathrm{i}k\Delta x}\right) \tag{4.128}$$

依据欧拉三角公式有：

$$\lambda = 1 - 2\frac{\Delta t}{(\Delta x)^2}(1 - \cos k\Delta x) = 1 - 4\frac{\Delta t}{(\Delta x)^2}\sin^2\left(\frac{1}{2}k\Delta x\right) \tag{4.129}$$

可以看出，在

$$\Delta t \leqslant \frac{1}{2}(\Delta x)^2 \tag{4.130}$$

的情况下，可以满足 Von Neumann 稳定性 $|\lambda| \leqslant 1$。

可以看出，方程(4.130)施加的是一个非常严格的条件，需要时间步长非常小才能保证稳定。如果减小 Δx，那么时间步长需要减小的更快。如果更换其他的离散格式，或许会有改变。在上文中使用的是时间显性离散方程。如果对空间项调用隐性中心格式离散，有：

$$\frac{\phi_j^{n+1} - \phi_j^n}{\Delta t} = \frac{\phi_{j+1}^{n+1} - 2\phi_j^{n+1} + \phi_{j-1}^{n+1}}{(\Delta x)^2} \tag{4.131}$$

参考上文的推导，有：

$$\lambda = \frac{1}{1 + 4\frac{\Delta t}{\Delta x^2}\sin^2\frac{1}{2}k\Delta x} \tag{4.132}$$

可以看出有 $0 < \lambda < 1$。因此，隐性格式是无条件稳定的。

4.4.3 截断误差

另外一种方式是采用泰勒展开的方法进行截断误差分析。考虑方程(4.126)，其中的变量 ϕ_j^{n+1} 可以理解为定义为 x_j, t^{n+1} 网格点的变量，那么其可以表示为：

$$\phi_j^{n+1} = \phi(x_j, t^{n+1}) = \phi(x_j, t^n + \Delta t) = \phi_j^n + \frac{\partial \phi}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 \phi}{\partial t^2}\Delta t^2 + \mathcal{O}(\Delta t^2) \tag{4.133}$$

同理，ϕ_{j+1}^n 可以表示为：

$$\phi_{j+1}^n = \phi(x_{j+1}, t^n) = \phi(x_j + \Delta x, t^n) = \phi_j^n + \frac{\partial \phi}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}\Delta x^2 + \frac{1}{6}\frac{\partial^3 \phi}{\partial x^3}\Delta x^3 + \frac{1}{24}\frac{\partial^4 \phi}{\partial x^4}\Delta x^4 + \mathcal{O}(\Delta x^4) \quad (4.134)$$

$$\phi_{j-1}^n = \phi(x_{j-1}, t^n) = \phi(x_j - \Delta x, t^n) = \phi_j^n - \frac{\partial \phi}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \phi}{\partial x^2}\Delta x^2 - \frac{1}{6}\frac{\partial^3 \phi}{\partial x^3}\Delta x^3 + \frac{1}{24}\frac{\partial^4 \phi}{\partial x^4}\Delta x^4 + \mathcal{O}(\Delta x^4) \quad (4.135)$$

将方程(4.134)和(4.135)代入到方程(4.126)中的左右两边有：

$$\begin{aligned} \frac{\phi_j^{n+1} - \phi_j^n}{\Delta t} &= \frac{\frac{\partial \phi}{\partial t}\Delta t + \frac{1}{2}\frac{\partial^2 \phi}{\partial t^2}\Delta t^2 \cancel{+\mathcal{O}(\Delta t^2)}}{\Delta t} = \frac{\partial \phi}{\partial t} + \frac{1}{2}\frac{\partial^2 \phi}{\partial t^2}\Delta t \\ \frac{\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n}{(\Delta x)^2} &= \frac{\frac{\partial^2 \phi}{\partial x^2}\Delta x^2 + \frac{1}{12}\frac{\partial^4 \phi}{\partial x^4}\Delta x^4 \cancel{+\mathcal{O}(\Delta x^4)}}{(\Delta x)^2} = \frac{\partial^2 \phi}{\partial x^2} + \frac{1}{12}\frac{\partial^4 \phi}{\partial x^4}\Delta x^2 \end{aligned} \quad (4.136)$$

即：

$$\frac{\partial \phi}{\partial t} - \frac{\partial^2 \phi}{\partial x^2} = -\frac{1}{2}\frac{\partial^2 \phi}{\partial t^2}\Delta t + \frac{1}{12}\frac{\partial^4 \phi}{\partial x^4}\Delta x^2 \quad (4.137)$$

方程(4.137)等号右边的即为截断误差 T。可以看出，截断误差 T 在 $\Delta t, \Delta x$ 趋向于 0 的情况下同时趋向于 0。同时，可以看出方程(4.126)是时间一阶、空间二阶的格式。同时，可以注意到截断误差第一项为一个正的扩散项，可能对求解产生不稳定性。

4.5 守恒法则与双曲系统

本节主要讨论守恒法则。这里的守恒法则特指 Conservation Law。虽然 NS 方程也满足守恒法则，但 NS 方程的守恒偏向物理的守恒。这里的守恒法则偏向于数学方程的守恒。相应的定义大量的出现在 [168, 92, 113]。本节尤其专注于守恒法则的一个大类别：双曲系统（hyperbolic system）。

双曲系统的偏微分方程求解为应用数学的一大领域。大量的数学问题都可以通过双曲偏微分方程进行求解。从历史的角度来看，CFD 大量的对流格式都起源于对双曲系统的研究，尤其是对可压缩欧拉方程的研究，衍生了大量的离散格式以及求解方法。一些简单的问题如 Burgers 方程、浅水方程都对离散格式的提出起到了重要的推动作用。但是自上世纪五十年代起，研究学者的最终极目标就是提出一个可以应用于欧拉方程的离散格式。这也与 CFD 的起源有关。二战期间，Los Alamos 实验室为了研究原子弹需要对激波进行捕获，CFD 则作为一种数学方法来对其进行预测。二战期间大量的研究工作都是保密的。直至九十年代才被公开。欧拉方程由于其非线性的特点，会在求解过程中发生间断。随之而

然的，会导致产生激波。如何对激波进行精确的捕获则是各种离散格式要解决的问题。大量的前人的工作就是围绕如何提出一个精准的离散格式而进行。

双曲系统的离散格式分为很多不同种形式。由于双曲系统的偏微分方程可以分解为特征变量形式，进一步转化为若干黎曼问题，因此有一大类离散格式要依据波的传输方向来进行求解，也即迎风类格式。另一方面，一些算法在提出的过程中，将黎曼扇区域进行包含，不对积分区域内的黎曼问题进行显性的求解，这种格式通常被称之为中心格式。还有一些方法将非线性的偏微分方程进行线性化，从而可以求出系数矩阵的特征值以及特征向量并进行求解。这也就是 Roe 方法的思想 [137]。Godunov 方法 [59] 则在每个网格面上对黎曼问题进行时间步的推进并进行精准求解，即首先对每个网格面上对变量进行重组获得型线（reconstruct），然后将变量进行 Δt 时间步推进在每个网格面上求解黎曼问题（evolve），最后将变量在网格上进行平均获得体场（average），整个流程也被称之为 REA 过程。类似的还有 HLL 格式、Osher 格式、AUSM 格式等，每个方法都源自于对双曲系统的特征传输特性的不同的理解。

仅仅讨论不可压缩流动的离散格式，忽略双曲系统的各种奇妙的求解方法是不完美的。OpenFOAM 中也对中心类格式进行了植入。因此本节主要讨论有限体积法下的双曲系统的各种数值问题，同时顺其自然的引出各种离散格式以及求解方法。

4.5.1 守恒法则

考虑下述方程：

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} &= 0 \\ \frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2 + P(\rho)}{\partial x} &= 0 \end{aligned} \tag{4.138}$$

注意，这里求解的为作为一个整体的守恒变量 $\rho, \rho u$，而非原始变量 ρ, u。之所以其为耦合在一起的，是因为变量 $\rho, \rho u$ 在每个方程中都存在，二者互相影响。且其为一套非线性方程组。在这里我们来推导特征变量的控制方程。

首先用 $\boldsymbol{\mathcal{U}}$ 矢量表示

$$\boldsymbol{\mathcal{U}} = \begin{cases} \rho \\ \rho u \end{cases} \tag{4.139}$$

也即

$$\frac{\partial \boldsymbol{\mathcal{U}}}{\partial t} = \begin{cases} \frac{\partial \rho}{\partial t} \\ \frac{\partial \rho u}{\partial t} \end{cases} = \begin{cases} \frac{\partial \mathcal{U}_1}{\partial t} \\ \frac{\partial \mathcal{U}_2}{\partial t} \end{cases} \tag{4.140}$$

$$\frac{\partial f(\boldsymbol{\mathcal{U}})}{\partial x} = \begin{cases} \frac{\partial \rho u}{\partial x} \\ \frac{\partial \rho u^2 + P(\rho)}{\partial x} \end{cases} = \begin{cases} \frac{\partial \mathcal{U}_2}{\partial x} \\ \frac{\partial \mathcal{U}_2^2/\mathcal{U}_1 + P(\mathcal{U}_1)}{\partial x} \end{cases} \tag{4.141}$$

因此，方程(4.138)可以写为下述系统：

$$\frac{\partial \boldsymbol{\mathcal{U}}}{\partial t}+\frac{\partial f(\boldsymbol{\mathcal{U}})}{\partial x}=0 \tag{4.142}$$

其中：

$$f(\boldsymbol{\mathcal{U}})=\begin{cases}\mathcal{U}_2\\ \mathcal{U}_2^2/\mathcal{U}_1+P(\mathcal{U}_1)\end{cases} \tag{4.143}$$

方程(4.142)在双曲系统研究领域通常被称之为守恒法则，这是因为 $f(\mathcal{U})$ 可以放在导数内部。需要注意的是，这种方程形式中的 $\boldsymbol{\mathcal{U}}$ 被假定为是可微分的，也就是说变量被假定为是光滑的。既然变量假定为光滑的，则进一步调用链条法则：

$$\frac{\partial \boldsymbol{\mathcal{U}}}{\partial t}+\frac{\partial f(\boldsymbol{\mathcal{U}})}{\partial \boldsymbol{\mathcal{U}}}\frac{\partial \boldsymbol{\mathcal{U}}}{\partial x}=0 \tag{4.144}$$

其中 $\frac{\partial f(\boldsymbol{\mathcal{U}})}{\partial \boldsymbol{\mathcal{U}}}$ 即为 Jacobian 矩阵 $\mathcal{A}$（参考4.11.12节）。其可以写为：

$$\mathcal{A}=\frac{\partial f(\boldsymbol{\mathcal{U}})}{\partial \boldsymbol{\mathcal{U}}}=\begin{bmatrix}0, & 1\\ -\mathcal{U}_2^2/\mathcal{U}_1^2+P'(\mathcal{U}_1), & 2\mathcal{U}_2/\mathcal{U}_1\end{bmatrix} \tag{4.145}$$

针对 $\mathcal{A}$，在这里要注意一个问题：

- 若 $\mathcal{A}$ 为常量，则构成线性双曲系统。
- 若 $\mathcal{A}$ 为依赖 x 的变量，则构成可变系数双曲系统。
- 若 $\mathcal{A}$ 为依赖 $\boldsymbol{\mathcal{U}}$ 的一阶函数，则构成准线性双曲系统。
- 若 $\mathcal{A}$ 为依赖 $\boldsymbol{\mathcal{U}}$ 的高阶函数，则构成非线性双曲系统。

4.5.2 线性双曲系统

针对雅克比矩阵 $\mathcal{A}$，考虑其特征向量，如果其特征值全部为实数且各不相同，在这种情况下一定存在互不相关的特征向量，该系统构成严格的双曲系统。如果特征值存在重根（在这种情况下特征向量可能互不相关，也可能相关），且都是实数，则构成弱双曲系统。如果特征值全部都是虚数，则构成椭圆系统。因此，一维欧拉方程为严格的双曲系统（三个不同的特征值），三维欧拉方程为弱双曲系统（五个特征值，三个重根）。本节主要讨论双曲系统。

现考虑严格的线性双曲系统，在这个情况下 $\mathcal{A}$ 可以被对角化，且有：

$$\mathcal{A}=\mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^{-1},\mathbf{R}^{-1}\mathcal{A}=\boldsymbol{\Lambda}\mathbf{R}^{-1} \tag{4.146}$$

其中 $\boldsymbol{\Lambda}$ 为一个对角阵，且对角线系数 λ_1,λ_2 各不相同。$\mathbf{R}$ 表示右侧特征向量 $\mathbf{r}^i$ 组建的矩阵：

$$\mathbf{R}=\left[\mathbf{r}^1|\mathbf{r}^2\right] \tag{4.147}$$

需要注意的是，由于这里考虑的是一个线性系统。因此相关量 $\mathcal{A}, \mathbf{R}$ 等都为不变量。

将方程(4.144)左右两边同时乘以 $\mathbf{R}^{-1}$，有：

$$\mathbf{R}^{-1}\frac{\partial \boldsymbol{\mathcal{U}}}{\partial t} + \boldsymbol{\Lambda}\mathbf{R}^{-1}\frac{\partial \boldsymbol{\mathcal{U}}}{\partial x} = 0 \tag{4.148}$$

令：

$$\boldsymbol{\mathcal{W}} = \mathbf{R}^{-1}\boldsymbol{\mathcal{U}} \tag{4.149}$$

有：

$$\frac{\partial \boldsymbol{\mathcal{W}}}{\partial t} + \boldsymbol{\Lambda}\frac{\partial \boldsymbol{\mathcal{W}}}{\partial x} = 0 \tag{4.150}$$

也即：

$$\begin{aligned} \frac{\partial \mathcal{W}_1}{\partial t} + \frac{\partial \lambda_1 \mathcal{W}_1}{\partial x} = 0 \\ \frac{\partial \mathcal{W}_2}{\partial t} + \frac{\partial \lambda_2 \mathcal{W}_2}{\partial x} = 0 \end{aligned} \tag{4.151}$$

其中 $\boldsymbol{\mathcal{W}}$ 被称之为特征变量，$\mathcal{W}_i$ 为 $\boldsymbol{\mathcal{W}}$ 的元素。同时可以注意到，方程(4.151)为解耦的。求解时，其可以被看做为两个单独的黎曼问题，具有不同的传输速度 λ_1, λ_2。同时可以看出，若可以获得 $\boldsymbol{\mathcal{W}}$ 的解，$\boldsymbol{\mathcal{U}}$ 也可以求出：

$$\boldsymbol{\mathcal{U}} = \mathbf{R}\boldsymbol{\mathcal{W}} = \left[\mathbf{r}^1|\mathbf{r}^2\right]\begin{bmatrix}\mathcal{W}_1\\ \mathcal{W}_2\end{bmatrix} = \mathcal{W}_1\mathbf{r}^1 + \mathcal{W}_2\mathbf{r}^2 = \sum \mathcal{W}_i\mathbf{r}^i \tag{4.152}$$

可以看出，$\boldsymbol{\mathcal{U}}$ 即为特征向量的叠加。

4.5.3 黎曼问题

黎曼问题通常指的是求解一个具有特殊初值的双曲方程。这个初值存在一个不连续点。图4.17所示为求解具有单一特征值的双曲方程 $\phi_t + u\phi_x = 0$，其中 u 为常数。假定 $\phi^t(x)$ 为初始分布，那其解为 $\phi^{t+\Delta t}(x) = \phi^t(x - u\Delta t)$。

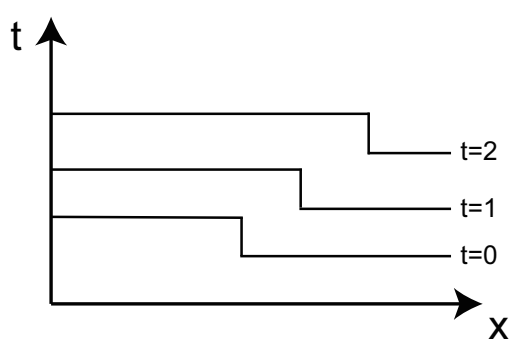


图 4.17: 经典黎曼问题不同时间步下的场分布。

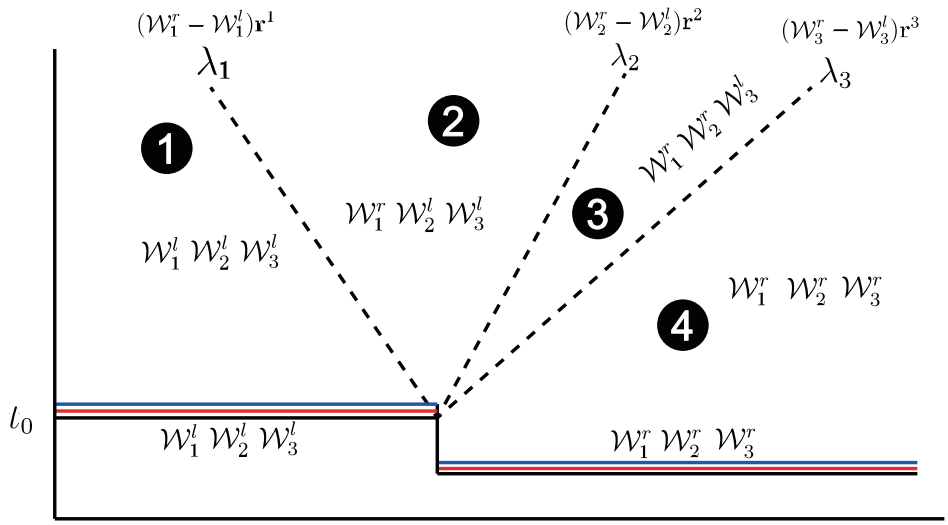


图 4.18: 三特征值黎曼问题解的区域。三个特征线分别对应三个 Rankine-Hugoniot 间断。

进一步考虑图4.18的黎曼问题，该系统存在 3 个传输方程。在 t_0 时刻，左侧 $\mathcal{W}$ 的初始值分别为 $\mathcal{W}_i^l, i = 1,2,3$，右侧 $\mathcal{W}$ 的初始值分别为 $\mathcal{W}_i^r, i = 1,2,3$。现在看图4.18的解。在 t_0 时刻，间断左侧的初始值分别为 $\mathcal{W}_1^l, \mathcal{W}_2^l, \mathcal{W}_3^l$，间断右侧的初始值分别为 $\mathcal{W}_1^r, \mathcal{W}_2^r, \mathcal{W}_3^r$。随着时间步的推进，针对区域 1，所有特征变量均没有跨域 λ_1 特征线，因此区域 1 的值保持不变。因此，若认为黎曼问题左侧的初始值为：

$$\boldsymbol{\mathcal{U}}_l = \mathbf{R}\begin{bmatrix}\mathcal{W}_1^l\\ \mathcal{W}_2^l\\ \mathcal{W}_3^l\end{bmatrix} = \mathbf{R}\boldsymbol{\mathcal{W}}_l = \mathcal{W}_1^l\mathbf{r}^1 + \mathcal{W}_2^l\mathbf{r}^2 + \mathcal{W}_3^l\mathbf{r}^3 \tag{4.153}$$

则区域 1 的解也为 $\boldsymbol{\mathcal{U}}_l$。其中 $\mathcal{W}$ 相关量是给定的初始值，$\mathbf{r}$ 相关量是依据雅克比矩阵求出来的特征向量，由于其为一个线性系统，其为一个定值。

针对区域 2，$\mathcal{W}_1$ 特征变量的值为 $\mathcal{W}_1^r$，其他的特征变量保持不变为 $\mathcal{W}_2^l, \mathcal{W}_3^l$。因此 $\boldsymbol{\mathcal{U}}$ 的解可以写为：

$$\boldsymbol{\mathcal{U}} = \mathbf{R}\begin{bmatrix}\mathcal{W}_1^r\\ \mathcal{W}_2^l\\ \mathcal{W}_3^l\end{bmatrix} = \mathcal{W}_1^r\mathbf{r}^1 + \mathcal{W}_2^l\mathbf{r}^2 + \mathcal{W}_3^l\mathbf{r}^3 \tag{4.154}$$

另一方面，其还可以写为：

$$\boldsymbol{\mathcal{U}} = \mathcal{W}_1^l\mathbf{r}^1 + \mathcal{W}_2^l\mathbf{r}^2 + \mathcal{W}_3^l\mathbf{r}^3 + (\mathcal{W}_1^r\mathbf{r}^1 - \mathcal{W}_1^l\mathbf{r}^1) = \mathbf{R}\boldsymbol{\mathcal{W}}_l + (\mathcal{W}_1^r\mathbf{r}^1 - \mathcal{W}_1^l\mathbf{r}^1) \tag{4.155}$$

其中的 $(\mathcal{W}_1^r - \mathcal{W}_1^l)\mathbf{r}^1$ 可以看做是区域 2 相对于区域 1 的有一个跃阶。在这里定义跃阶 $\boldsymbol{\alpha}$

为：

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} \mathcal{W}_1^r - \mathcal{W}_1^l \\ \mathcal{W}_2^r - \mathcal{W}_2^l \\ \mathcal{W}_3^r - \mathcal{W}_3^l \end{bmatrix} \tag{4.156}$$

则区域 2 的解为

$$\boldsymbol{\mathcal{U}} = \mathbf{R}\boldsymbol{\mathcal{W}}_l + \alpha_1 \mathbf{r}^1 \tag{4.157}$$

同理，从跃阶的角度来考虑，区域 3 的解可以写为：

$$\boldsymbol{\mathcal{U}} = \mathbf{R}\boldsymbol{\mathcal{W}}_l + \alpha_1 \mathbf{r}^1 + (\mathcal{W}_2^r - \mathcal{W}_2^l)\mathbf{r}^2 = \mathbf{R}\boldsymbol{\mathcal{W}}_l + \alpha_1 \mathbf{r}^1 + \alpha_2 \mathbf{r}^2 \tag{4.158}$$

区域 4 的解可以写为：

$$\boldsymbol{\mathcal{U}} = \mathbf{R}\boldsymbol{\mathcal{W}}_l + \alpha_1 \mathbf{r}^1 + \alpha_2 \mathbf{r}^2 + (\mathcal{W}_3^r - \mathcal{W}_3^l)\mathbf{r}^3 = \mathbf{R}\boldsymbol{\mathcal{W}}_l + \alpha_1 \mathbf{r}^1 + \alpha_2 \mathbf{r}^2 + \alpha_3 \mathbf{r}^3 \tag{4.159}$$

另外，区域 4 的解也为 $\mathbf{R}\boldsymbol{\mathcal{W}}_r$，也即 $\boldsymbol{\mathcal{U}}_r$。因此也有

$$\mathbf{R}\boldsymbol{\mathcal{W}}_l + \alpha_1 \mathbf{r}^1 + \alpha_2 \mathbf{r}^2 + \alpha_3 \mathbf{r}^3 = \mathbf{R}\boldsymbol{\mathcal{W}}_r = \boldsymbol{\mathcal{U}}_r \tag{4.160}$$

另外也可以看出，跨越整个黎曼扇的跃阶为：

$$\boldsymbol{\mathcal{U}}_r - \boldsymbol{\mathcal{U}}_l = \mathbf{R}\boldsymbol{\mathcal{W}}_r - \mathbf{R}\boldsymbol{\mathcal{W}}_l = \alpha_1 \mathbf{r}^1 + \alpha_2 \mathbf{r}^2 + \alpha_3 \mathbf{r}^3 = \mathbf{R}\boldsymbol{\alpha} \tag{4.161}$$

其也可以写为

$$\boldsymbol{\mathcal{U}}_r - \boldsymbol{\mathcal{U}}_l = \mathbf{R}\boldsymbol{\mathcal{W}}_r - \mathbf{R}\boldsymbol{\mathcal{W}}_l = \sum \alpha_i \mathbf{r}^i \tag{4.162}$$

同时有

$$\mathbf{R}^{-1}(\boldsymbol{\mathcal{U}}_r - \boldsymbol{\mathcal{U}}_l) = \boldsymbol{\alpha} \tag{4.163}$$

在这里需要强调的是，上述 4 个不同区域求解出来的解是精准的。

同时考虑图4.18，里面若存在若干个特征线，则可以采用一种普适性的方法来获得区域 2 个解。之所以区域 2 的解是重要的，这是因为计算网格面正好位于区域 2 内。假定每一个特征值的正负是未知的，有：

$$\boldsymbol{\mathcal{U}} = \mathbf{R}\boldsymbol{\mathcal{W}}_l + \sum_{\text{if } \lambda_i < 0} \alpha_i \mathbf{r}^i \tag{4.164}$$

$$\boldsymbol{\mathcal{U}} = \mathbf{R}\boldsymbol{\mathcal{W}}_r - \sum_{\text{if } \lambda_i > 0} \alpha_i \mathbf{r}^i \tag{4.165}$$

二者加和有：

$$\boldsymbol{\mathcal{U}} = \frac{1}{2}\left(\mathbf{R}\boldsymbol{\mathcal{W}}_r + \mathbf{R}\boldsymbol{\mathcal{W}}_l\right) + \frac{1}{2}\left(\sum_{\text{if } \lambda_i < 0} \alpha_i \mathbf{r}^i - \sum_{\text{if } \lambda_i > 0} \alpha_i \mathbf{r}^i\right) \tag{4.166}$$

上式为 Roe 格式的基本思想。最根本的，本节讨论的内容都是线性双曲系统。对于实际情况下的非线性双曲系统，特征向量特征值等都是变化的量会导致求解非常复杂。Roe 格式作为近似黎曼求解器，就是把非线性系统转变为线性系统后，依据上文讨论的方法进行求解。在 Roe 方法中，原本变化的 $\alpha_i, \mathbf{r}_i, \lambda_i$ 都会转变成不取决于网格左右变量的固定值后，通过方程(4.166)来进行推进。

4.5.4 积分形式与守恒方程

前述章节推导了各种积分形式的控制方程。在双曲系统研究领域，由于间断存在的可能性，主要是用积分形式的控制方程，如方程(3.23)。现对其进行回顾。

将方程(3.23)移项，并整理为方程(4.142)的形式：

$$\int_V \boldsymbol{\mathcal{U}}(t+\Delta t)\mathrm{d}V = \int_V \boldsymbol{\mathcal{U}}(t)\mathrm{d}V - \int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t'))\mathrm{d}t' \tag{4.167}$$

为方便讨论问题，一维形式可以写为：

$$\int_{\Delta x} \boldsymbol{\mathcal{U}}(t+\Delta t)\mathrm{d}x = \int_{\Delta x} \boldsymbol{\mathcal{U}}(t)\mathrm{d}x - \left(\int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t')_{i+\frac{1}{2}})\mathrm{d}t' - \int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t')_{i-\frac{1}{2}})\mathrm{d}t'\right) \tag{4.168}$$

两边同时除以 Δx 有：

$$\begin{aligned}\frac{1}{\Delta x}\int_{\Delta x} \boldsymbol{\mathcal{U}}(t+\Delta t)\mathrm{d}x = &\frac{1}{\Delta x}\int_{\Delta x} \boldsymbol{\mathcal{U}}(t)\mathrm{d}x \\ &- \frac{1}{\Delta x}\left(\int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t')_{i+\frac{1}{2}})\mathrm{d}t' - \int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t')_{i-\frac{1}{2}})\mathrm{d}t'\right)\end{aligned} \tag{4.169}$$

其中 $\frac{1}{\Delta x}\int_{\Delta x} \boldsymbol{\mathcal{U}}(t+\Delta t)\mathrm{d}x$ 表示 $\boldsymbol{\mathcal{U}}$ 在 $t+\Delta t$ 时间步下，网格区间内的体平均值，用 $\overline{\boldsymbol{\mathcal{U}}}_i^{t+\Delta t}$ 来表示。同理，当前时间步的体平均值用 $\overline{\boldsymbol{\mathcal{U}}}_i^{t}$ 来表示。方程可以继续写为：

$$\overline{\boldsymbol{\mathcal{U}}}_i^{t+\Delta t} = \overline{\boldsymbol{\mathcal{U}}}_i^{t} - \frac{\Delta t}{\Delta x}\left(\int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t')_{i+\frac{1}{2}})\mathrm{d}t' - \int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t')_{i-\frac{1}{2}})\mathrm{d}t'\right) \tag{4.170}$$

方程(4.170)的意义很明确。控制体网格下变量的下一个时间步的值，等于当前时间步的值，以及网格通量的贡献之和。$\int_t^{t+\Delta t} f(\boldsymbol{\mathcal{U}}(t')_{i+\frac{1}{2}})\mathrm{d}t'$ 表示定义在网格面 $i+\frac{1}{2}$，时间步长 Δt 内通量的影响。其可以继续简写为：

$$\overline{\boldsymbol{\mathcal{U}}}_i^{t+\Delta t} = \overline{\boldsymbol{\mathcal{U}}}_i^{t} - \frac{\Delta t}{\Delta x}\left(\mathcal{F}_{i+\frac{1}{2}} - \mathcal{F}_{i-\frac{1}{2}}\right) \tag{4.171}$$

同时需要注意的是，方程(4.171)是精准的。但其尚未封闭。因此，各种格式的意义，即对通量进行封闭。

4.5.5 封闭与模板

封闭的根本是通过当前时间步的值，来对通量 $\mathcal{F}$ 进行模化。考虑最简单的情况，我们认为网格 $i+\frac{1}{2}$ 处的通量，仅仅跟相邻的网格点的变量有关。即：

$$\mathcal{F}_{i+\frac{1}{2}} = \mathcal{F}(\overline{\boldsymbol{U}}_i^t, \overline{\boldsymbol{U}}_{i+1}^t) \tag{4.172}$$

$$\overline{\boldsymbol{U}}_i^{t+\Delta t} = \overline{\boldsymbol{U}}_i^t - \frac{\Delta t}{\Delta x}\left(\mathcal{F}(\overline{\boldsymbol{U}}_i^t, \overline{\boldsymbol{U}}_{i+1}^t) - \mathcal{F}(\overline{\boldsymbol{U}}_{i-1}^t, \overline{\boldsymbol{U}}_i^t)\right) \tag{4.173}$$

具体 $\mathcal{F}(\overline{\boldsymbol{U}}_i^t, \overline{\boldsymbol{U}}_{i+1}^t)$ 采用什么样的形式，与离散格式有关。不管什么样的格式，方程(4.173)仅仅与 $i, i-1, i+1$ 网格点的变量有关。因此，本方法采用的是三点模板（three-point stencil）。

4.5.6 中心格式

方程(4.172)最简单的形式，即网格面的通量为相邻网格单元的值的一种算术平均：

$$\mathcal{F}_{i+\frac{1}{2}} = \mathcal{F}(\overline{\boldsymbol{U}}_i^t, \overline{\boldsymbol{U}}_{i+1}^t) = \frac{1}{2}\left(f(\boldsymbol{U}_i^t) + f(\boldsymbol{U}_{i+1}^t)\right) \tag{4.174}$$

将方程(4.174)代入到(4.173)有：

$$\overline{\boldsymbol{U}}_i^{t+\Delta t} = \overline{\boldsymbol{U}}_i^t - \frac{\Delta t}{2\Delta x}\left(f(\boldsymbol{U}_{i+1}^t) - f(\boldsymbol{U}_{i-1}^t)\right) \tag{4.175}$$

需要注意的是，中心格式是无条件不稳定的。

4.5.7 Lax-Friedrichs 格式

参考中心格式，Lax-Friedrichs 格式的通量定义为：

$$\mathcal{F}_{i+\frac{1}{2}} = \mathcal{F}(\overline{\boldsymbol{U}}_i^t, \overline{\boldsymbol{U}}_{i+1}^t) = \frac{1}{2}\left(f(\boldsymbol{U}_i^t) + f(\boldsymbol{U}_{i+1}^t)\right) - \frac{\Delta x}{2\Delta t}\Delta y \Delta z(\boldsymbol{U}_{i+1}^t - \boldsymbol{U}_i^t) \tag{4.176}$$

从方程形式来看，Lax-Friedrichs 格式像是在中心格式上添加了一定的贡献来获得稳定性。在这里回忆一下扩散项，扩散项的形式为 $-\frac{\partial}{\partial x}\left(\Gamma\frac{\partial \boldsymbol{U}}{\partial x}\right)$，将扩散项在 $i+\frac{1}{2}$ 处离散有

$$-\Gamma_{1+\frac{1}{2}}\frac{\boldsymbol{U}_{i+1}^t - \boldsymbol{U}_i^t}{\Delta x^2} \tag{4.177}$$

对比方程(4.176)，可以发现 Lax-Friedrichs 格式即为在中心格式上添加了一个数值扩散项，系数为

$$\Gamma = \frac{\Delta x^2}{2\Delta t}\Delta x \Delta y \Delta z = \frac{\Delta x^2}{2\Delta t}\Delta V \tag{4.178}$$

正是这个数值扩散项的存在，使得 Lax-Friedrichs 格式可以稳定。

需要注意的是，当前讨论的 Lax-Friedrichs 格式有时也被称之为全域 Lax-Friedrichs 格式。这是因为附加的扩散项的系数 $\Delta x/\Delta t$ 固定。这个扩散系数还可以通过一个局部的值来替换，也即局部 Lax-Friedrichs 格式。

4.5.8 Kurganov-Tadmor 格式

参考 Lax-Friedrichs 格式，Kurganov-Tadmor 格式的通量定义为：

$$\mathcal{F}_{i+\frac{1}{2}} = \mathcal{F}(\overline{\boldsymbol{U}}_i^t, \overline{\boldsymbol{U}}_{i+1}^t) = \frac{1}{2}\left(f(\boldsymbol{U}_i^t) + f(\boldsymbol{U}_{i+1}^t)\right) - \frac{1}{2}a_{i+\frac{1}{2}}\Delta y \Delta z(\boldsymbol{U}_{i+1}^t - \boldsymbol{U}_i^t) \tag{4.179}$$

其中 $a_{i+\frac{1}{2}}$ 表示黎曼扇区域的最大波速。Kurganov-Tadmor 格式认为 Lax-Friedrichs 格式的数值耗散系数过大，因此在 $a_{i+\frac{1}{2}} < \Delta x/\Delta t$ 的情况下，Kurganov-Tadmor 格式的数值耗散更小。理论上来讲，$a_{i+\frac{1}{2}} < \Delta x/\Delta t$ 表示当前时间步 Δt 下，黎曼扇区域在两个网格中心点之间。

可以看出 Kurganov-Tadmor 格式与局部 Lax-Friedrichs 格式本质上是一致的。在一些教材中，二者称呼混用 [62, 92]。在一些教材中也被称之为 Rusanov 格式 [62, 168]。

4.5.9 再看 Roe 格式

Roe 格式对于常系数 PDE 以及非线性 PDE 的处理方式略有不同，若考虑常系数 PDE（传输速度为常数 A）。Roe 格式可以定义为：

$$\mathcal{F}_{i+\frac{1}{2}} = \mathcal{F}(\overline{\boldsymbol{U}}_i^t, \overline{\boldsymbol{U}}_{i+1}^t) = \frac{1}{2}\left(f(\boldsymbol{U}_i^t) + f(\boldsymbol{U}_{i+1}^t)\right) - \frac{1}{2}|A|\Delta y \Delta z(\boldsymbol{U}_{i+1}^t - \boldsymbol{U}_i^t) \tag{4.180}$$

可见，在极端的情况下，Roe 格式与 Kurganov-Tadmor 格式相同。

4.5.10 Godunov 格式

Godunov 格式的通量计算是精准的，其采用解析解来进行，并不包含任何假设，因此 Godunov 格式的通量仅仅用下式来进行表达：

$$\mathcal{F}_{i+\frac{1}{2}} = \mathcal{F}^g_{i+\frac{1}{2}} \tag{4.181}$$

但这并不证明 Godunov 求解的结果是精准的，因为其还涉及到网格点到网格面的变量重构算法（图4.19所示的即为一阶重构）。且通量为时间积分形式。从历史上来讲，在 Godunov 格式开发之前，主要是用有限差分来求解线性问题。但有限差分由于处理间断能力不足，将其拓展并处理非光滑问题难上加难。因此 Godunov 格式可以认为是有限体积法的起源。Godunov 在开发这种有限体积类算法的过程中，引入了三种主要流程（思想），这虽然在现在看来是顺其自然的。但是在上世纪 50 年代，只有 Godunov 格式开发出来之后，后人才相继开发出更多的高阶格式。

如图4.19所示，有限差分法在每个网格上认为存在一点，并且这个点代表了网格的值。Godunov 思想的第一步是认为在整个网格内部变量存在一定的分布，图4.19所示的为分段常数分布（piecewise constant distribution），其为一种一阶假定。这一步也被称之为重构

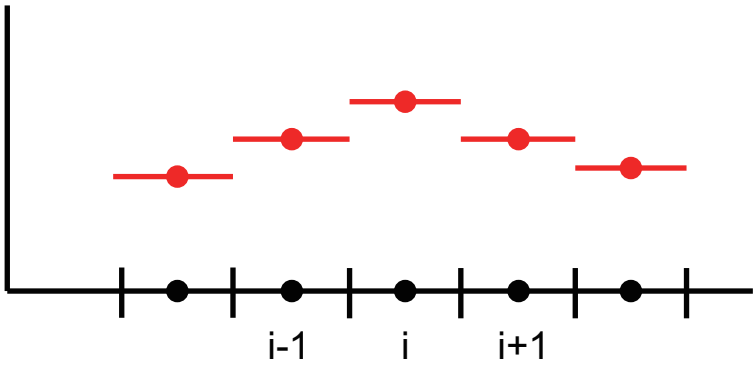


图 4.19: 图中红点为有限差分假定的变量值。图中红线为有限体积法假定的变量分布。

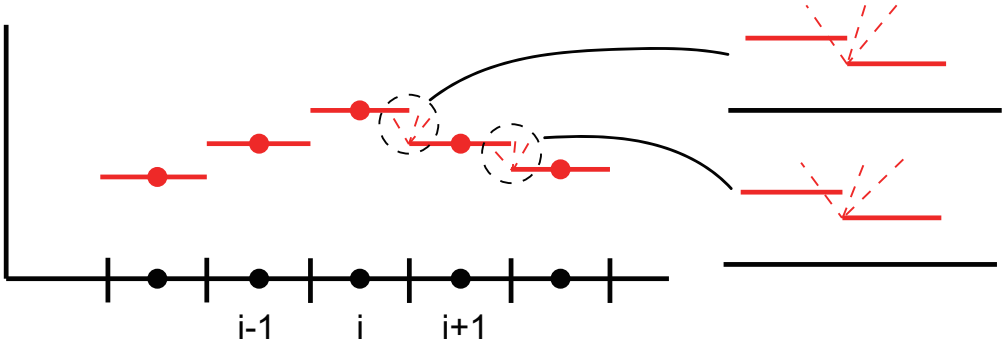


图 4.20: Godunov 第二步在每个网格交界面求解黎曼问题的精确解。

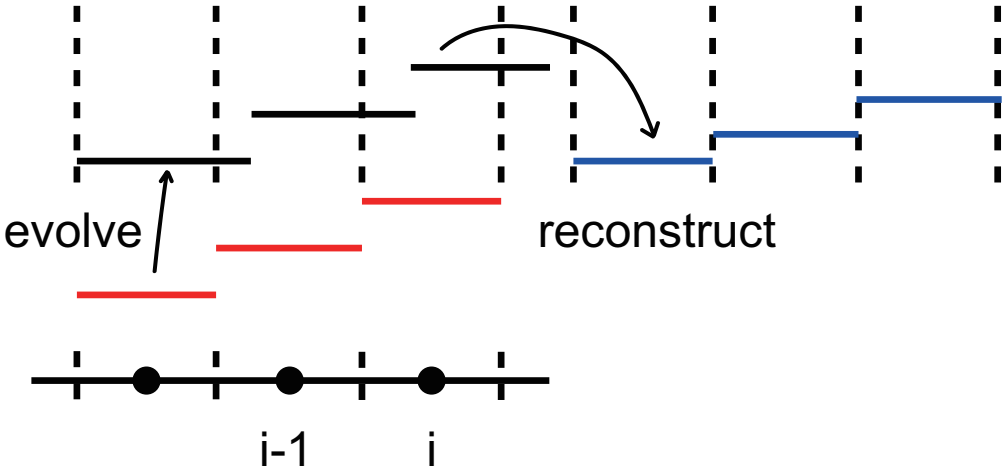


图 4.21: Godunov 第三步在每个网格将变量进行重组。

（reconstruct）步。如图4.20所示，Godunov 格式的第二步在每个网格交界面求解黎曼问题的精确解，也即演变（evolve）步骤。在求解每个网格交界面的黎曼问题之后，如图4.21所示，Godunov 格式的第三步需要将变量分布进行重组（reconstruct）。然后进行下一个时间步的计算。

在实施的过程中，在每个网格交界面求解黎曼问题是不可能的。因此，衍生出一些近似黎曼求解器，如上文提到的 Lax-Friedrichs 格式。近似黎曼格式并不需要在每个网格面求解黎曼问题，例如 Kurganov-Tadmor 格式将黎曼扇进行包含，黎曼扇内部的变量演变并不关心。

4.6 化学反应、ODE 求解器

4.6.1 复杂反应、基元反应、可逆反应

化学反应按照反应过程可以分为复杂反应与基元反应。基元反应一步就可以转化为反应物。复杂反应由多个基元反应构成。微观上看所有化学反应过程都是经过一个或多个简单的反应步骤（即基元反应）才转化为产物分子。考虑下述的复杂反应：

$$2H_2 + 2NO \longrightarrow N_2 + 2H_2O \tag{4.182}$$

其本质上可以分解为三个基元反应：

$$\begin{aligned} &2NO \rightleftharpoons N_2O_2 \quad \text{(fast equilibrium)} \\ &N_2O_2 + H_2 \longrightarrow N_2O + H_2O \quad \text{(slow)} \\ &N_2O + H_2 \longrightarrow N_2 + H_2O \quad \text{(fast)} \end{aligned} \tag{4.183}$$

其中第一个为可逆反应。许多化学反应都是可逆的，正向反应和逆向反应可以同时发生。当正向反应的速率等于逆向反应的速率时，我们称之为动态平衡。平衡状态是指正向反应的速率等于反向反应的速率的状态。所有反应物和产物浓度在平衡状态下都是恒定的。在很快的达到平衡状态后，反应物的浓度满足下述关系：

$$[N_2O_2] = K_1[NO]^2 \tag{4.184}$$

其中的 K_1 表示平衡常数[24]。

[24] 对于某个反应物 X，其摩尔浓度可以表示为 $[X]$，单位为 $\mathrm{mol/m^3}$。K_1 其可以改写为

$$K_1 = \frac{[N_2O_2]}{[NO]^2}$$

如果平衡常数非常大，例如大于 1000，则产物居多。如果非常小，则反应物居多。

4.6.2 反应速率 RR、质量作用定律

在化学反应中，可以使用反应速率来表示反应进行的快慢程度。反应速率也表示反应物或产物浓度随时间的变化率。化学反应速率的单位通常为 $\mathrm{mol/(m^3{\cdot}s)}$。表示每单位时间（秒）、每单位体积（立方米）的反应物（摩尔）的变化。其中 $\mathrm{mol/m^3}$ 也可以理解为反应物的摩尔浓度。因此化学反应速率也可以理解为每单位时间反应物浓度的变化。反应速率可以通过反应物来标识，也可以使用生成物来标识。例如针对方程(4.182)的反应，可以用 $\frac{\mathrm{d[N_2]}}{\mathrm{d}t}$ 表示反应速率，也可以用 $\frac{\mathrm{d[H_2O]}}{\mathrm{d}t}$ 表示反应速率。同时，也可以用反应物来标识反应速率：$-\frac{\mathrm{d[H_2]}}{\mathrm{d}t}$ 或 $-\frac{\mathrm{d[NO_2]}}{\mathrm{d}t}$。使用反应物表示反应速率的时候，需要使用负号。

$\frac{\mathrm{d[N_2]}}{\mathrm{d}t}, \frac{\mathrm{d[H_2O]}}{\mathrm{d}t}, -\frac{\mathrm{d[H_2]}}{\mathrm{d}t}, -\frac{\mathrm{d[NO_2]}}{\mathrm{d}t}$ 表示的反应速率并不相同。然而在体积不变的情况下，有：

$$\frac{\mathrm{d[N_2]}}{\mathrm{d}t} = \frac{1}{2}\frac{\mathrm{d[H_2O]}}{\mathrm{d}t} = -\frac{1}{2}\frac{\mathrm{d[H_2]}}{\mathrm{d}t} = -\frac{1}{2}\frac{\mathrm{d[NO_2]}}{\mathrm{d}t} \tag{4.185}$$

因此当我们提及某反应的反应速率的时候，如果没有指定某反应物，则表示我们采用的是方程(4.185)所示的反应速率。

基元反应的反应速率与各反应物浓度因子的计量系数次方的连乘积成正比。也即质量作用定律。其只适用于基元反应。不适用于复杂反应（复杂反应的速率需要通过实验测定）。但对构成复杂反应的各个基元反应都适用。例如下述基元反应：

$$a\mathrm{A} + b\mathrm{B} \longrightarrow c\mathrm{C} + d\mathrm{D} \tag{4.186}$$

其反应速率（Reaction Rate）可以表示为

$$\mathrm{RR} = k[A]^a[B]^b = -\frac{1}{a}\frac{\mathrm{d}[A]}{\mathrm{d}t} = -\frac{1}{b}\frac{\mathrm{d}[B]}{\mathrm{d}t} = \frac{1}{c}\frac{\mathrm{d}[C]}{\mathrm{d}t} = \frac{1}{d}\frac{\mathrm{d}[D]}{\mathrm{d}t} \tag{4.187}$$

其中的 k 表示速率常数。其表示反应速率与反应物的摩尔浓度以及化学计量数有关。速率常数 k 的单位与反应级数有关（因为反应速率的单位是固定的），值与温度之间的关系满足阿瑞尼斯方程：

$$k = AT^{\beta}\exp\left(-\frac{T_a}{T}\right), T_a = \frac{E}{R} \tag{4.188}$$

其中 A 表示指前因子，β 表示温度指数，E 表示反应的活化能。每个反应，均需要给定 A, β, T_a。

方程(4.186)若为一个复杂反应，其反应速率可以写为[25]：

$$\mathrm{RR} = k[A]^{\alpha}[B]^{\beta} \tag{4.190}$$

[25] 复杂反应的反应速率一般取决于其中最慢的反应，且通常由实验测定，这是反应动力学的研究内容。例如，对于方程(4.182)中的反应，其通过实验测量的反应速率或许为

$$\mathrm{RR} = k[\mathrm{H_2}][\mathrm{NO}] \tag{4.189}$$

可以看出其并不是 $\mathrm{RR} = k[\mathrm{H_2}]^2[\mathrm{NO}]^2$。这是因为其不是一个基元反应。在某些情况下，复杂反应的反应速率可能与某些反应物没有关系（例如固体反应、溶液反应之类）。

在这里可以定义反应级数。从反应速率可以看出，反应的总级数为：

$$n = \alpha + \beta \tag{4.191}$$

若 $n = 1$，表示一级反应。$n = 2$，表示二级反应。n 可以为整数、分数、负数以及 0。需要强调的是，复杂反应的 α, β 的具体数值是通过实验测量出来的，与反应计量数 a, b 无关。在某些情况下，化学反应不能归结为类似方程(4.190)的形式，在这种情况下被称之为无级数反应[26]。对于反应速率 $\mathrm{RR} = k[A]^{\alpha}[B]^{\beta}$，其具有以下特性：

- k 与浓度无关，但是与温度、活化能、催化剂有很大的关系。例如其满足方程(4.188)；
- k 的单位与具体的方程形式有关，其要保证 RR 的单位表示摩尔浓度的时间变化；

大部分情况下，可以把某一个基元反应写成加和的形式：

$$\sum v_i' M_i \rightarrow \sum v_i'' M_i \tag{4.192}$$

其中 i 表示第 i 个物质；v_i' 表示第 i 个物质的反应计量数，v_i'' 表示第 i 个物质的产出计量数。举例，考虑下述反应：

$$2\mathrm{H_2} + \mathrm{O_2} \rightarrow 2\mathrm{H_2O} \tag{4.193}$$

其总组分为 3 个，分别为 $M_1 = \mathrm{H_2}, M_2 = \mathrm{O_2}, M_3 = \mathrm{H_2O}$。其也可以写为：

$$2\mathrm{H_2} + \mathrm{O_2} + \cancel{0\mathrm{H_2O}} \rightarrow \cancel{0\mathrm{H_2}} + \cancel{0\mathrm{O_2}} + 2\mathrm{H_2O} \tag{4.194}$$

很明显，$v_1' = 2, v_2' = 1, v_3' = 0$，$v_1'' = 0, v_2'' = 0, v_3'' = 2$。在这种情况下，反应速率可以写为：

$$\mathrm{RR} = k'[\mathrm{H_2}]^2[\mathrm{O}][\mathrm{H_2O}]^0 = k'[\mathrm{H_2}]^2[\mathrm{O}] \tag{4.195}$$

其可以简写为：

$$\mathrm{RR} = k' \prod_{k=1}^{N} [M_i]^{v_i'} \tag{4.196}$$

另外，由于反应一般由很多的基元反应构成，将所有的基元反应加和构成整套反应系统：

$$\sum v_{ji}' M_{ji} \rightarrow \sum v_{ji}'' M_{ji} \tag{4.197}$$

上述方程表示存在 j 个基元反应构成的总反应。

上述讨论并没有考虑可逆的情况，然而在大多数情况下，基元反应是可逆的：

$$\sum v_i' M_i \rightleftharpoons \sum v_i'' M_i \tag{4.198}$$

因此对于物质 M_i，会存在一个平衡状态。在考虑反应速率的时候，要同时考虑反应和生成，也即方程左右两边的状态。在这种情况下的净反应速率可以表示为

$$\mathrm{RR} = k' \prod_{k=1}^{N} [M_i]^{v_i'} - k'' \prod_{k=1}^{N} [M_i]^{v_i''} \tag{4.199}$$

[26] 例如氢气与溴的反应速率为 $v = \frac{k[\square][\square]^{1/2}}{1+k'\frac{[\square]}{[\square]}}$ 为无级数反应。

4.6.3 化学反应实例

化学反应中的各个元素通常会形成耦合的方程组。考虑下述经典的 Robertson 反应：

$$\begin{aligned} A &\xrightarrow{k_1} B \\ 2B &\xrightarrow{k_2} B + C \\ B + C &\xrightarrow{k_3} A + C \end{aligned} \tag{4.200}$$

上述三个反应均为基元反应，因此有反应速率：

$$\begin{aligned} \mathrm{RR}_1 &= k_1[A] \\ \mathrm{RR}_2 &= k_2[B]^2 \\ \mathrm{RR}_3 &= k_3[B][C] \end{aligned} \tag{4.201}$$

其中 $k_1 = 0.04, k_2 = 3 \times 10^7, k_3 = 10^4$。同时，有各个组分的摩尔浓度变化：

$$\begin{aligned} \frac{\mathrm{d}[A]}{\mathrm{d}t} &= -\mathrm{RR}_1 + \mathrm{RR}_3 = -k_1[A] + k_3[B][C] \\ \frac{\mathrm{d}[B]}{\mathrm{d}t} &= \mathrm{RR}_1 - \mathrm{RR}_2 - \mathrm{RR}_3 = k_1[A] - k_2[B]^2 - k_3[B][C] \\ \frac{\mathrm{d}[C]}{\mathrm{d}t} &= \mathrm{RR}_2 = k_2[B]^2 \end{aligned} \tag{4.202}$$

方程(4.202)都成 3 个常微分方程系统，其可以进行求解。

下面继续看另外一套基元反应，其可以写为：

$$\begin{aligned} A + B &\xrightarrow{k_1} 2B \\ B + C &\xrightarrow{k_2} 2C \\ C &\xrightarrow{k_3} D \end{aligned} \tag{4.203}$$

有反应速率：

$$\begin{aligned} \mathrm{RR}_1 &= k_1[A][B] \\ \mathrm{RR}_2 &= k_2[B][C] \\ \mathrm{RR}_3 &= k_3[C] \end{aligned} \tag{4.204}$$

同时，有各个组分的摩尔浓度变化：

$$\begin{aligned} \frac{\mathrm{d}[A]}{\mathrm{d}t} &= -k_1[A][B] \\ \frac{\mathrm{d}[B]}{\mathrm{d}t} &= k_1[A][B] - k_2[B][C] \\ \frac{\mathrm{d}[C]}{\mathrm{d}t} &= k_2[B][C] - k_3[C] \\ \frac{\mathrm{d}[D]}{\mathrm{d}t} &= k_3[C] \end{aligned} \tag{4.205}$$

其构成 4 个常微分方程系统，可以通过数值方法进行求解。在真实的 CFD 计算中，可能会存在上百个化学反应，且大概率为一套刚性方程组。因此 CFD 代码中会存在若干的数值方法来求解刚性方程组。

4.6.4 刚性方程组、ODE 求解器

在化学反应中，每个反应的时间尺度并不相同。由于反应速率之间差异较大，因此数值求解器将微分方程视为刚性方程。对于刚性微分方程，有些数值求解器无法收敛得出一个解，除非步长极小。如果步长极小，计算时间可能长得让人难以接受。Robertson 反应的化学反应时间尺度差异巨大，是一个经典的刚性方程组。下面我们介绍几种数值方法对其求解，分别为欧拉显性、欧拉隐性、以及欧拉半隐性方法。首先假定方程组可以写为矢量形式：

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = -\mathbf{C}\cdot\mathbf{y} \tag{4.206}$$

其中 $\mathbf{y}$ 表示一个 n 阶矢量，$\mathbf{C}$ 表示一个常数，其为一个 n 阶矩阵。

欧拉显性方法认为：

$$\mathbf{y}^{t+\Delta t} = \mathbf{y}^t + \Delta t \left.\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}\right|_t = \mathbf{y}^t + \Delta t(-\mathbf{C}\cdot\mathbf{y}^t) \tag{4.207}$$

$$\mathbf{y}^{t+\Delta t} = (\mathbf{I} - \mathbf{C}\Delta t)\cdot\mathbf{y}^t \tag{4.208}$$

其中 $\mathbf{I}$ 表示单位矩阵。

欧拉隐性方法认为：

$$\mathbf{y}^{t+\Delta t} = \mathbf{y}^t + \Delta t \left.\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t}\right|_{t+\Delta t} = \mathbf{y}^t + \Delta t(-\mathbf{C}\cdot\mathbf{y}^{t+\Delta t}) \tag{4.209}$$

$$\mathbf{y}^{t+\Delta t} = (\mathbf{I} + \mathbf{C}\Delta t)^{-1}\cdot\mathbf{y}^t \tag{4.210}$$

在很多情况下，$\mathbf{C}$ 并不是一个常数，而与 $\mathbf{y}$ 有关。考虑下述方程：

$$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}t} = \mathbf{f}(\mathbf{y}) \tag{4.211}$$

欧拉隐性方法认为[27]

$$\mathbf{y}^{t+\Delta t} = \mathbf{y}^t + \Delta t\mathbf{f}(\mathbf{y}^{t+\Delta t}) \tag{4.212}$$

其中 $\mathbf{f}(\mathbf{y}^{t+\Delta t})$ 与 $\mathbf{y}^{t+\Delta t}$ 的值有关，因此构成非线性方程组。在这里可以将它线性化：

$$\mathbf{f}(\mathbf{y}^{t+\Delta t}) = \mathbf{f}(\mathbf{y}^t) + \left.\frac{\partial\mathbf{f}}{\partial\mathbf{y}}\right|_{\mathbf{y}^t}\cdot(\mathbf{y}^{t+\Delta t} - \mathbf{y}^t) \tag{4.213}$$

[27] 通过移项更好理解：$(\mathbf{y}^{t+\Delta t} - \mathbf{y}^t)/\Delta t = \mathbf{f}(\mathbf{y}^{t+\Delta t})$

参考第4.11.12节，其中 $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ 为 Jacobian 矩阵。将方程(4.213)代入到(4.212)有：

$$\mathbf{y}^{t+\Delta t} = \mathbf{y}^t + \Delta t \left(\mathbf{f}(\mathbf{y}^t) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^t} \cdot (\mathbf{y}^{t+\Delta t} - \mathbf{y}^t) \right) \tag{4.214}$$

$$\left(\mathbf{I} - \Delta t \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^t} \right) \mathbf{y}^{t+\Delta t} = \mathbf{y}^t + \Delta t \mathbf{f}(\mathbf{y}^t) - \Delta t \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^t} \cdot \mathbf{y}^t \tag{4.215}$$

$$\left(\mathbf{I} - \Delta t \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^t} \right) \mathbf{y}^{t+\Delta t} = \left(\mathbf{I} - \Delta t \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^t} \right) \cdot \mathbf{y}^t + \Delta t \mathbf{f}(\mathbf{y}^t) \tag{4.216}$$

$$\mathbf{y}^{t+\Delta t} = \mathbf{y}^t + \Delta t \left(\mathbf{I} - \Delta t \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^t} \right)^{-1} \cdot \mathbf{f}(\mathbf{y}^t) \tag{4.217}$$

方程(4.217)也通常被称之为半隐性欧拉方法（将隐性的非线性系统线性化）[28]，也被称之为线性化隐性欧拉方法。可以看出，半隐性欧拉方法需要计算函数 $\mathbf{f}(\mathbf{y})$ 的雅可比矩阵，同时还要对矩阵做逆运算。给定矩阵，其逆的计算通常采用 LU 分解的形式来进行（因为雅克比为密集型矩阵）。在计算雅克比矩阵的时候，如果没有解析解（大部分情况下），同样需要数值的方法进行计算。

如上所述，现在来看一个雅克比矩阵求解的实例。首先，(4.202)中的反应也可以(4.211)的形式，其中：

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}, \mathbf{f}(\mathbf{y}) = \begin{pmatrix} k_2 y_2 y_3 - k_1 y_1 \\ k_1 y_1 - k_2 y_2 y_3 - k_3 y_2^2 \\ k_3 y_2^2 \end{pmatrix} \tag{4.218}$$

则有雅克比矩阵：

$$\frac{\partial \mathbf{f}}{\partial \mathbf{y}} = \begin{pmatrix} -k_1 & k_2 y_3 & k_2 y_2 \\ k_1 & -k_2 y_3 - 2k_3 y_2 & -k_2 y_2 \\ 0 & 2k_3 y_2 & 0 \end{pmatrix} \tag{4.219}$$

同时结合方程(4.217)，此化学反应各个组分随着时间的变化可解。

下面进行另外一个 ODE 方程求解的演示。假定某 ODE 形式为（Rayleigh-Plesset 方程）

$$r \frac{\partial^2 r}{\partial t^2} + 1.5 \frac{\partial r}{\partial t} = S \tag{4.220}$$

其中 S 表示与 r 无关的源项。在使用 ODE 进行求解的时候，首先需要进行预处理，有：

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \mathbf{f}(\mathbf{y}) = \begin{pmatrix} y_2 \\ \frac{S - 1.5 y_2}{y_1} \end{pmatrix} \tag{4.221}$$

[28] EulerSI.C

有雅克比矩阵：

$$\frac{\partial \mathbf{f}}{\partial \mathbf{y}} = \begin{pmatrix} 0 & 1 \\ -(S-1.5y_2)y_1^{-2} & -\frac{1.5}{y_1} \end{pmatrix} \tag{4.222}$$

结合方程(4.217)后可解。在 OpenFOAM 中，需要把雅克比矩阵 $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$，导数 $\mathbf{f}(\mathbf{y})$，以及未知数数量输入到`ODESystem`中。

4.7 大气边界层与大气稳定度

大气边界层也被称之为行星边界层。整个大气层可以看做是一种流体，那么在靠近地面的部分，由于地表摩擦力的存在，以及各种树木、房屋、建筑等的存在，空气的流动受阻并减慢，形成 CFD 中的所谓的边界层的现象。大气边界层具有最大的速度梯度，其占据边界层高度的 5%-10%。另一方面，由于地表的形状不规则，也会产生不规则旋涡，变为湍流。在计算风工程领域，大气边界层的处理是否合理，会严重影响计算结果。

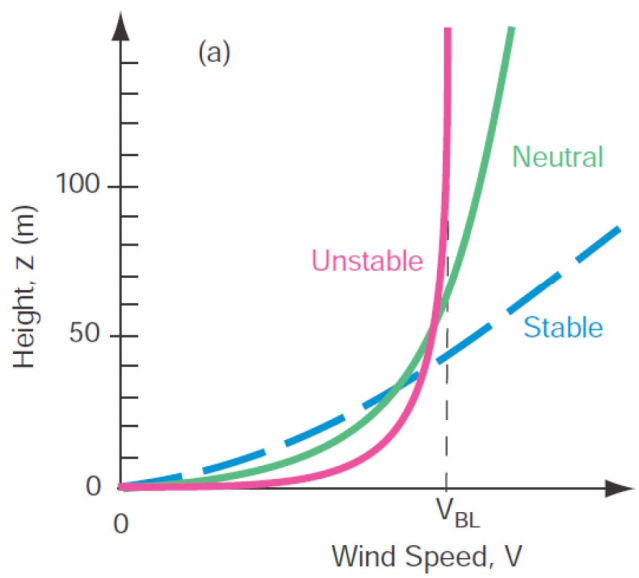


图 4.22: 不同大气稳定度条件下的典型速度型线分布 [78]。

大气稳定度用来描述空气气团的运动程度。考虑大气中的一团空气，如果给他初始的一个向上的速度，其会向上上升。在上升的过程中，由于气压的下降，该气团会发生膨胀并降温。在上升的过程中，如果气团的温度大于环境温度，气团会进一步的上升。这种情况被称之为不稳定大气。如果气团的温度小于环境温度，气团会下降，也就是上升的气团上升到一定高度之后，下降到原来的高度。这种情况的大气被认为是稳定的。如果气团的温度一直跟环境的温度相当。那么气团上升到一定高度之后会停止。这种大气环境被认为是中性平衡状态。大气稳定性主要取决于实际的气温、湿度变化、热的传导等因素。不同大

气稳定度条件下的速度型线可以参考图4.22。在不稳定大气条件下，地表湍流效应比较明显，会导致比较强的动量交换，因此会呈现比较陡峭的速度型线。在稳定大气条件下，地表湍流效应较弱，混合效应较差，因此速度型线分布较为平缓。除此二者之外，中性大气环境处于二者直接，较为折中。

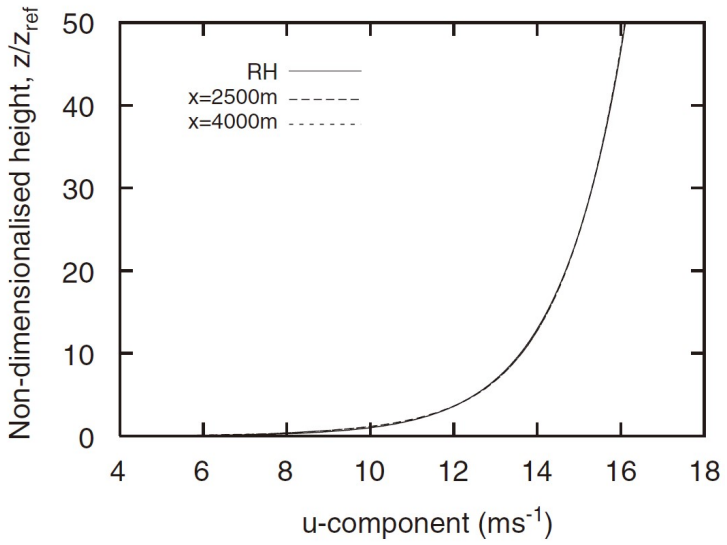


图 4.23: Hargreaves and Wright 针对中性大气稳定度模拟的不同位置的速度型线 [65]。

4.7.1 湍流动能自保持

大气边界层领域 CFD 计算模拟主要的研究点在于如何将湍流动能保持住。在一个空的计算域内进行 RANS 模拟，未考虑中性大气稳定度的计算结果中的湍流动能会随着流向的发展越来越弱。在 CFD 中，大量的工作已经用于研究如何施加中性均衡大气稳定性条件 [195, 65]，以及如何使得变量可以沿着流线方向进行自保持。这是因为对于中性均衡的大气稳定性条件，流线 x 方向上的速度以及湍流动能等变量可以相对保持恒定。也就是说在一个典型的规则计算域，进口速度与出口速度会保持相同的型线。中性大气稳定度的 CFD 计算模拟，通常通过特定的进口边界条件，附加特定的壁面函数来进行。CFD 可以进行稳态计算来获得稳态解。图4.23所示的即为中性大气稳定度条件下不同 x 位置的速度型线 [195]。可以看出其呈现一种均一的分布。同时在 Hargreaves and Wright 的文章中，作者明确表明，单一的施加进口边界条件，不足以维系相关变量的型线，还需要考虑特定的壁面函数类型。CFD 中变量的进口边界条件，通常为垂直高度的 y 相关函数。由于湍流的作用，即使最简单的中性稳定条件，垂直的 y 方向的型线也很难获得（但通常呈现一种 log 幂率分布）。为了湍流动能能够自保持，本节讨论几个易于实施的方法 [65]。

第一个方法在 1993 年被提出，Richards and Hoxey 从标准 $k-\varepsilon$ 模型出发，认为进口

条件处的湍流变量应该符合下述关系 [136]：

$$u = \frac{u_*}{\kappa}\ln\left(\frac{z+z_0}{z_0}\right) \tag{4.223}$$

$$k = \frac{u_*^2}{\sqrt{C_\mu}} \tag{4.224}$$

$$\varepsilon = \frac{u_*^3}{\kappa(z+z_0)} \tag{4.225}$$

其中 u_* 表示摩擦速度，z_0 表示表面粗糙度高度。此模型简称 RH93 进口边界条件。后来经过证实 [195]，RH93 进口边界条件在符合下面条件的情况下：

$$\sigma_\varepsilon = \frac{\kappa^2}{(C_2-C_1)\sqrt{C_\mu}} \tag{4.226}$$

可以看做是标准 $k-\varepsilon$ 模型的一维解析解（若 $C_\mu = 0.09, \sigma_\varepsilon = 1.11$）。可以看出，RH93 进口边界条件下的 σ_ε 为一个变量。另外需要注意的事，在使用 RH93 进口边界条件的时候，壁面函数也需要考虑粗糙度的影响。

另一方面，如果要使得 RH93 进口边界条件下 σ_ε 为一个常量，可以通过在湍流动能耗散率传输方程中添加一个源项来进行：

$$S_\varepsilon = \frac{\rho u_*^4}{(z+z_0)^2}\left(\frac{(C_2-C_1)\sqrt{C_\mu}}{\kappa^2} - \frac{1}{\sigma_\varepsilon}\right) \tag{4.227}$$

在这种情况下，RH93 进口边界条件同样是标准 $k-\varepsilon$ 模型的一维解析解。

第二个方法于 2009 年被提出。很明显，RH93 进口模型预测的湍流动能随高度是一个常数，并不符合实际。且在使用 RH93 的时候，计算域顶部需要施加剪切应力条件与之相配合，在实际情况中难以实施。因此，Yang 等认为进口的边界条件组合可以写为 [195, 196]：

$$u = \frac{u_*}{\kappa}\ln\left(\frac{z+z_0}{z_0}\right) \tag{4.228}$$

$$k = \frac{u_*^2}{\sqrt{C_\mu}}\sqrt{C_1\ln\left(\frac{z+z_0}{z_0}\right)+C_2} \tag{4.229}$$

$$\varepsilon = \frac{u_*^3}{\kappa(z+z_0)}\sqrt{C_1\ln\left(\frac{z+z_0}{z_0}\right)+C_2} \tag{4.230}$$

$$\omega = \frac{u_*}{\kappa\sqrt{C_\mu}}\frac{1}{z+z_0} \tag{4.231}$$

其中 z_0 表示参考高度，z 表示距离地表的高度，其他均为用户自定义常数：$\kappa = 0.42, C_\mu = 0.028, C_1 = -0.17, C_2 = 1.62$。在下文中称之为 Yang2009 方法。使用 Yang2009 方法，$k-\varepsilon$ 标准模型里面的参数要发生变化：$C_1 = 1.5, C_2 = 1.92, C_\mu = 0.028, \sigma_k = 1.67, \sigma_\epsilon = 2.51$。同

时，一些文章里面 u_* 可以通过自定义参数来实现 [195]，一些文章里面的 u_* 可以通过下式进行计算 [136]：

$$u_* = \frac{u_{ref}\kappa}{\ln\left(\frac{z_{ref}+z_0}{z_0}\right)} \tag{4.232}$$

Yang2009 方法的实施，不需要在计算域顶部施加剪切应力边界条件，但是需要使用粗糙壁面函数边界条件。

第三个方法于 2007 年被提出。Hargreaves 和 Wright 基于 RH93 条件，进行了一些改动。在下文中我们简化为 HW07 方法 [65]。HW07 方法与 Yang2009 方法非常类似，其中的速度进口、湍流动能进口、湍流动能耗散率进口均与 Yang2009 方法相同。不同的是其中的模型参数。HW07 方法的模型参数为：$\kappa = 0.4, \sigma_\varepsilon = 1.11, C_1 = 0, C_2 = 1$。且其中的摩擦速度 u_* 通过方程(4.232)来计算。其余的参数均与标准 $k-\varepsilon$ 参数相同。HW07 方法在实施的时候，要更改湍流粘度壁面函数，参考方程(6.308)，HW07 方法通过下式进行计算：

$$\nu_t = \left(\frac{y^+\kappa}{\ln((y+z_0)/z_0)} - 1\right)\nu \tag{4.233}$$

HW07 方法的实施同样需要在计算域顶部施加剪切应力边界条件。

第四个方法于 2011 年被提出。Parente 等人也提出了一种湍流模型处理方式，来使得湍动能剖面可以自保持 [127]。该方法在下文中简称为 Parente 方法。Parente 方法通过下列修正来进行实现。首先需要在湍流耗散率方程上施加源项，即方程(4.236)。同时在湍流动能方程上施加如下源项：

$$S_k(z) = \frac{\rho u^*\kappa}{\sigma_k}\frac{\partial}{\partial z}\left((z+z_0)\frac{\partial k}{\partial z}\right) \tag{4.234}$$

且湍流模型系数 C_μ 随高度进行变化：

$$C_\mu(z) = \frac{u^{*4}}{k(z)^2} \tag{4.235}$$

Parente 方法的速度进口条件可以表示为方程(4.223)。湍流动能进口可以使用任意的型线。ε 通过下式进行指定：

$$\varepsilon(z) = \sqrt{C_\mu}k\frac{du}{dz} = \frac{u_*^3}{\kappa(z+z_0)} \tag{4.236}$$

Parente 方法的使用还需要配合壁面函数来进行。

4.7.2 地转风与科氏力

另外需要注意的是，对于大尺度的计算模拟，还需要考虑科氏力对结果的影响。在这种情况下需要在动量方程中添加源项的作用。同时，科氏力与压力梯度的里会互相抗衡，形成地转风。地球表面由于天气的影响，不同的区域压力并不相同。将压力相同的区域连线

会形成等压线。等压线之前会存在压力梯度。地转风是一种科氏力和气压梯度力巧妙平衡下产生的一种理论上的风。这个状况称为“地转平衡”。地转风的方向和等压线平行。这种理论上的平衡其实很少见，但可以看做一种完美的数学模型，并且可以用于 CFD 模拟。

如图4.24所示，在最开始的时刻，假定流体不存在运动，因此科氏力的大小为零。由于压力梯度的存在，导致大气开始向上移动，同时科氏力的大小逐渐增加。由于科氏力的存在，流动向右偏移。由于科氏力与流动的方向垂直，因此科氏力的方向进行改变。最终，科氏力与大气梯度互相平衡，呈现 180 度的交叉。且大气的流动方向与等压线平衡。

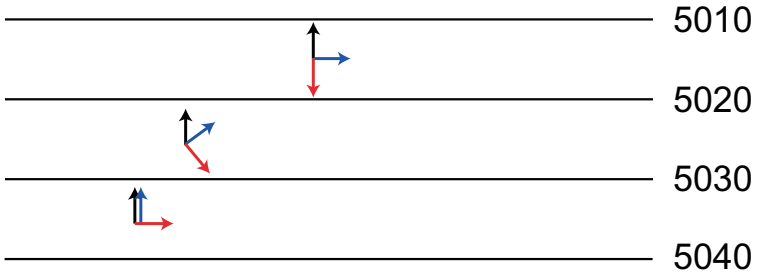


图 4.24: 由于压力梯度与科氏力平衡引起的地转风。其中黑色箭头表示压力梯度的方向。红色箭头表示科氏力方向。绿色箭头表示大气流动方向。

图4.25中也形象的展示了由于地球自转科氏力产生的影响 [4]。在地球不旋转的时候，从北极向下运动的物体，会笔直的走一条直线。如果地球存在自转，从北极向下走的物体，会向右发生偏移。

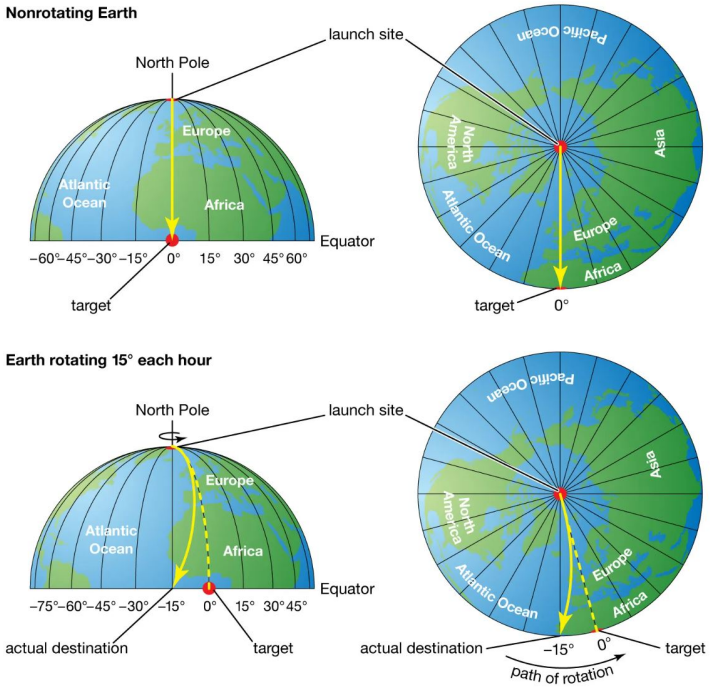


图 4.25: 由于地球自转，科氏力对大气流动产生的影响。

科氏力一般可以表示为 $\mathbf{f}\times\mathbf{U}=2\Omega\sin\lambda\times\mathbf{U}$，$\Omega$ 表示地球旋转速率，单位为 [弧度/s]=1/s，λ 表示维度 [无单位]。因此 f 的单位为 [1/s]。$\mathbf{f}$ 表示科氏力参数，一些文献会直接给出 $\mathbf{f}$ 的值。如果绕 z 旋转，那么 $\mathbf{f}$ 在 z 方向存在值。在这种情况下，$\mathbf{f}\times\mathbf{U}$ 的值为：

$$\mathbf{f}\times\mathbf{U}=\begin{bmatrix} f_2u_3-f_3u_2 \\ f_3u_1-f_1u_3 \\ f_1u_2-f_2u_1 \end{bmatrix}=\begin{bmatrix} -f_3u_2 \\ f_3u_1 \\ 0 \end{bmatrix} \tag{4.237}$$

如果进一步考虑速度方向是 x 方向的，那么变为：

$$\mathbf{f}\times\mathbf{U}=\begin{bmatrix} 0 \\ f_3u_1 \\ 0 \end{bmatrix} \tag{4.238}$$

可以看出如果地球绕 z 旋转，对于 x 方向的速度，科氏力导致 y 方向的偏离。

对于 CFD 中常见的周期性边界模拟。这种科氏力导致 y 方向的偏离会是持续的，有时候会导致流场持续的发生打转行为。因此在模拟中，如果给定一个压力梯度，如图4.25所示，这样压力梯度会与科氏力发生平衡，形成地转风。这个压力梯度导致的速度大小，即 f_3u_1，且为 y 方向。

最后归纳一下常见的粗糙度高度设置 [111]：

下垫面类型	z_0 (m)
光滑地面、冰、泥	0.00001
平静的开放水面	0.0002
存在风浪的水面	0.0005
雪面	0.003
草地	0.008
牧场	0.01
田地	0.03
庄家	0.05
比较少的树	0.1
很多树，不多的建筑	0.25
森林	0.5
郊区地形	1.5
存在高楼的市中心	3

4.7.3 中性、稳定以及不稳定大气环境

大气环境的模拟由于尺度非常大，且涉及到科氏力、浮力等各种复杂的条件，因此常规实验室规模的标准 kEpsilon 湍流模型并不适用。Apsley and Castro 证明了标准 kEpsilon 湍

流模型预测的湍流混合长与地面距离直接线性相关 [14]，因此其会预测一个过厚的大气边界层。他们提出了一个适用于中性以及稳定大气环境的对湍流尺度进行限制的湍流模型。其在 $k-\varepsilon$ 模型的基础上，添加了一定的源项以及沉降项来对大气的稳定性来进行解释。Apsley and Castro 的湍流模型不需要调用温度方程，既可以用于中性以及稳定的大气环境条件。van der Laan 等人随后在 Apsley and Castro 的湍流模型的基础上，将其包含浮力的影响，进而可以用于预测非稳定大气环境 [176]。相关模型可以写为 [14, 176]：

$$\frac{\partial k}{\partial t}+\nabla\cdot(\mathbf{U}k)-\nabla\cdot(D_k\nabla k)\ =G-\varepsilon+B \tag{4.239}$$

$$\frac{\partial \varepsilon}{\partial t}+\nabla\cdot(\mathbf{U}\varepsilon)-\nabla\cdot(D_\varepsilon\nabla \varepsilon)=\left(C_1\frac{\varepsilon}{k}+C_1^*\right)G-C_2\frac{\varepsilon^2}{k}+C_3\frac{\varepsilon}{k}B+D \tag{4.240}$$

其中：

- 湍流混合长修正源项：C_1^*G 用来在标准 $k-\varepsilon$ 模型上对大气环境的混合长进行修正；C_1^* 可以表示为：

$$C_1^*=(C_2-C_1)\frac{L}{L_{max}}\frac{\varepsilon}{k}=(C_2-C_1)\frac{L}{L_{max}}\frac{C_\mu k}{\nu_t} \tag{4.241}$$

  其中 L 表示湍流混合长，有 $L=C_\mu^{0.75}\frac{k^{1.5}}{\varepsilon}$；$L_{max}$ 需要用户给定。Apsley and Castro 认为 [14]，对于中性大气环境，L_{max} 约等于边界层高度的三分之一。对于稳定大气环境，L_{max} 约等于 MOST 尺度的 0.08 倍。Sogachev 等认为在给定科氏力以及地转风的情况下，L_{max} 的计算公式为 $L_{max}=0.00027G/f$[149]，其中 G 表示地转风速，f 表示科氏力参数。另外需要注意的是，其中湍流模型的参数 $\sigma_\varepsilon=1.11$[14]，其他模型参数与标准 kEpsilon 湍流模型参数相同。需要注意的是，在 Sogachev 等的工作中采用了另外一套模型参数 [149]。van der Laan 的模型中不需要温度变量，在这种情况下 B 可以写为 [176]：

- 浮力修正源项：$C_3\frac{\varepsilon}{k}B$ 也为一个附加的源项，其用来解释浮力对湍流的影响；若存在温度方程，则 B 可以写为：

$$B=\beta\alpha_t(\nabla T\cdot\mathbf{g}) \tag{4.242}$$

  在不稳定大气环境下，B 为正数，会导致湍流动能的增强。在稳定大气以及中性大气环境下，B 为负数，会对湍流产生抑制作用。同时 C_3 需要进一步进行模化：

$$C_3=(C_1-C_2)\alpha_B+1; \tag{4.243}$$

$$\alpha_B=\begin{cases}1-\frac{L}{L_{max}}, & R>0\\ 1-\left(1+\frac{C_2-1}{C_2-C_1}\right)\frac{L}{L_{max}}, & R<0\end{cases} \tag{4.244}$$

$$L=\frac{C_\mu^{0.75}k^{1.5}}{\varepsilon},R=-\frac{B}{G} \tag{4.245}$$

van der Laan 的湍流模型中不需要附加的温度变量，在这种情况下 B 可以写为 [176]：

$$B = -\nu_t \left(\frac{\mathrm{d}u}{\mathrm{d}z} + \frac{\mathrm{d}v}{\mathrm{d}z} \right) \frac{z}{L} \tag{4.246}$$

其中 z 表示地面垂直的坐标分量，u, v 表示其他两个坐标方向上的速度，L 表示 MOST 尺度。在这种情况下，C_3 定义为：

$$C_3 = 1 + C_{\varepsilon,1} - C_{\varepsilon,2} + (2C_{\varepsilon,2} - C_{\varepsilon,1} - 1)\frac{L}{L_{max}} \tag{4.247}$$

- D 是一个附加的源项，在部分情况下可忽略[29]；

4.7.4 下垫面

下垫面指与大气下层直接接触的地球表面。它包括陆地、海洋、湿地、沙漠、不同植被区、地形区等。在 CFD 中，大量的工作被应用于研究城市冠层、树木冠层等对大气边界层的影响。下垫面在数值上，主要通过源项来影响速度、温度（热通量）、湍流动能以及湍流动能耗散率。在进行模化的时候，存在不同的方法。在这里介绍一些比较受欢迎的方法。

下垫面对于速度的影响主要通过添加阻力源项来进行。其速度源项可以写为 [150, 124]：

$$\mathbf{S_U} = -C_d A|\mathbf{U}|\mathbf{U} \tag{4.248}$$

其中 C_d 表示阻力系数，A 表示每单位体积的树叶面积（单位为 1/m）。这两个系数需要用户指定。阻力系数可以取值在 0.15 至 0.37 之间。

下垫面对湍流的影响，在一些模型中，需要在湍流动能以及湍流动能耗散率方程中添加源项，如 [36]：

$$\begin{aligned} S_k &= \rho C_z \left(\beta_p |\mathbf{U}|^3 - \beta_d |\mathbf{U}| k \right) \\ S_\varepsilon &= \rho C_z \left(C_{\varepsilon 4} \beta_p \frac{\varepsilon}{k} |\mathbf{U}|^3 - C_{\varepsilon 5} \beta_d |\mathbf{U}| \varepsilon \right) \end{aligned} \tag{4.249}$$

其中的用户自定义参数在不同的工作中有不同的值。在 Sogachev 的工作中，其从另外一种角度进行推导，其认为仅仅需要对湍流动能耗散率添加源项即可 [150]：

$$\begin{aligned} S_\varepsilon &= -(C_1 - C_2) S_d \frac{\varepsilon}{k}, \\ S_d &= 12 C_\mu^{0.5} C_d A |\mathbf{U}| k \end{aligned} \tag{4.250}$$

方程(4.250)看起来更易于植入，因为其只需要考虑添加一个源项即可。

[29] 混合长源项 D 的存在是因为，Sogachev 等人发现 $k-\omega$ 系列模型预测的结果要比 $k-\varepsilon$ 等预测的结果要好 [149]，同时这两个湍流模型可以互相转化，因此在 $k-\varepsilon$ 模型的基础上，添加了互相转化而存在的源项 D：

$$D = C_\mu \left(\frac{1}{\sigma_k} - \frac{1}{\sigma_\varepsilon} \right) k \nabla \cdot (\nabla k) - C_\mu \left(\frac{1}{\sigma_k} + \frac{1}{\sigma_\varepsilon} \right) \frac{k}{\varepsilon} \nabla \varepsilon : \nabla k + C_\mu \frac{2}{\sigma_k} \nabla k : \nabla k$$

4.8 多孔介质模型

海绵、填料等在某些情况下会存在于计算工况中。对于这一类物质，CFD 通常使用多孔介质来完成。CFD 中的多孔介质模型的存在，大大简化了多孔区域的网格处理。对于复杂的多空区域，如果要做全解析，需要把这部分区域的网格生成出来，会导致网格量成倍的增加，计算时间成倍的增长。多孔介质模型通过数学模型，将多孔介质区域的物理过程进行描述，主需要在 CFD 计算中指定一个多孔介质区域，而不需要细化网格即可进行计算。

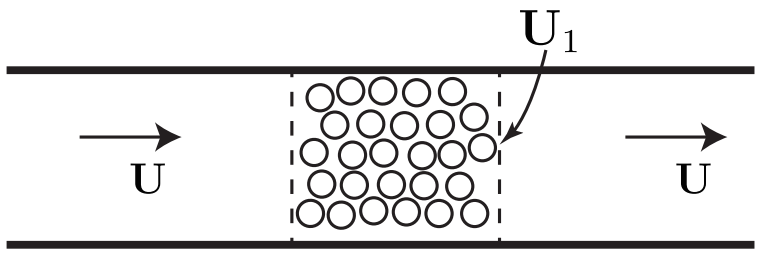


图 4.26: 管道中的多孔介质示意图。进口速度为 $\mathbf{U}_1$，出口速度同样为 $\mathbf{U}_1$（质量守恒）。在多孔介质区域内，由于管道的截面积发生变化，速度通常会增加至 $\mathbf{U}_2$。

4.8.1 多孔介质平均技术

参考图4.27，整个网格的体积为 V_{cell}，白色固体填充物的体积为 V_s，剩余的流体体积为 V_f，有 $V_{cell} = V_s + V_f$。第一种平均技术被称之为空间平均（spatial average），对于任意的变量 ψ，其定义为 [183]：

$$\langle \psi \rangle = \frac{1}{V_{cell}} \int_{V_{cell}} \psi \mathrm{d}V \tag{4.251}$$

第二种平均技术被称之为 intrinsic 平均（intrinsic average 或 intrinsic phase average）[30]，其定义为 [183]：

$$\hat{\psi} = \frac{1}{V_f} \int_{V_f} \psi \mathrm{d}V \tag{4.252}$$

第三种平均技术被称之为表观相平均（phase average 或 superficial phase average），其定义为 [183]：

$$\overline{\psi} = \frac{1}{V_{cell}} \int_{V_f} \psi \mathrm{d}V \tag{4.253}$$

结合孔隙率的定义：

$$\epsilon = \frac{V_f}{V_{cell}} \tag{4.254}$$

[30] intrinsic average 并没有找到明确的中文翻译。

有 $\hat{\psi}$ 与 $\overline{\psi}$ 的关系为：

$$\overline{\psi} = \epsilon\hat{\psi} \tag{4.255}$$

如图4.26所示，在管道中若存在多孔介质，由于管道横截面积的不连续变化，会导致速度的不连续。在多孔介质区域，速度会比较大。因此，多孔介质模型通常会定义两种速度，一个是表观速度 $\overline{\mathbf{U}}$，一个是真实的物理速度 $\mathbf{U}$。多孔介质领域的表观速度，采用的是表观相平均的方法：

$$\overline{\mathbf{U}} = \frac{1}{V_{cell}}\int_{V_f}\mathbf{U}\mathrm{d}V \tag{4.256}$$

二者可以通过孔隙率 ϵ 来建立联系：

$$\overline{\mathbf{U}} = \epsilon\mathbf{U} \tag{4.257}$$

在多相流中，由于相分数的存在，大多使用表观速度，因为不管流态如何变化，表观速度为守恒且连续的。在多孔介质领域，表观速度同样是连续的。因此计算域中若孔隙率的分布呈现不连续分布，真实的物理速度必然存在不连续。同时，在实验观测中，一般手段测量出来的速度同样为表观速度。

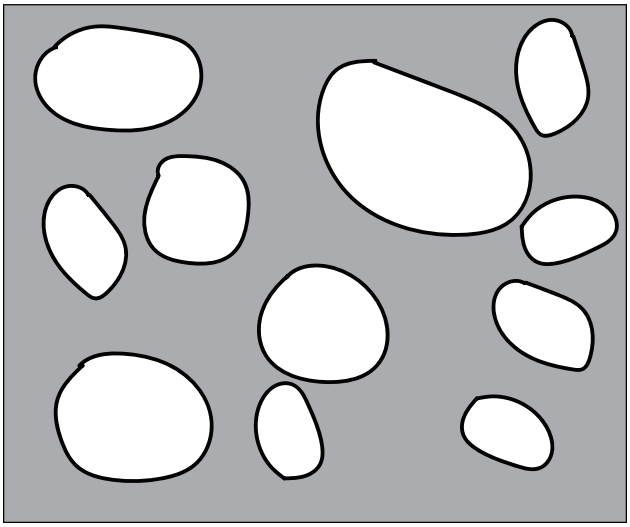


图 4.27: 一个代表性的计算网格。其中白色的表示固体颗粒物填充，灰色的标识真实的计算网格体积。整个网格的体积为 V，白色填充物的体积为 V_s，剩余的体积为 V_f。

4.8.2 Darcy 定律

在雷诺数小于 1 的情况下，依据实验观测以及数学模型均可以达到一个普适性方程。这个方程就是 Darcy 定律。考虑最简单的多孔介质模型（各向同性、渗透性均一），Darcy 定律可以表示为：

$$\overline{\mathbf{U}} = -\frac{K}{\mu}\nabla p \tag{4.258}$$

其中 K 表示渗透率（permeability），单位为 m^2。其用于表征流体穿过多孔介质的能力。Darcy 定律的适用情况为多孔介质区域的雷诺数小于 1，在这种情况下，附加连续性方程，流体的控制方程如下：

$$\begin{aligned}\overline{\mathbf{U}} &= -\frac{K}{\mu}\nabla p \\ \nabla\cdot\overline{\mathbf{U}} &= 0\end{aligned} \tag{4.259}$$

若进一步考虑一个随着空间分布的渗透性 K，方程(4.259)依旧适用。

一些情况下需要考虑各向异性多孔介质。在这种情况下，流动阻力在某个方向上比较大，某个方向上比较小。各向异性多孔介质的渗透率为一个二阶张量 $\mathbf{K}$。在这种情况下，Darcy 定律写成分量的不同影响会更容易理解：

$$\begin{aligned}\overline{u} &= -K_{11}\frac{\partial p}{\partial x} - K_{12}\frac{\partial p}{\partial y} - K_{13}\frac{\partial p}{\partial z} \\ \overline{v} &= -K_{21}\frac{\partial p}{\partial x} - K_{22}\frac{\partial p}{\partial y} - K_{23}\frac{\partial p}{\partial z} \\ \overline{w} &= -K_{31}\frac{\partial p}{\partial x} - K_{32}\frac{\partial p}{\partial y} - K_{33}\frac{\partial p}{\partial z}\end{aligned} \tag{4.260}$$

其也可以写为紧凑形式：

$$\overline{\mathbf{U}} = -\frac{\mathbf{K}}{\mu}\cdot\nabla p \tag{4.261}$$

对于均质的多孔介质，二阶张量 $\mathbf{K}$ 为对称的，因此减少到为 6 个分量。大部分情况下，$\mathbf{K}$ 的非对角线系数也为 0。因此 $\mathbf{K}$ 可以简化为一个非对角线系数为 0 的二阶张量。

Darcy 定律也可以通过在动量方程的基础上，忽略时间项、对流项、扩散项而进行推导而来。在此基础上，Brinkman 定律认为时间项、对流项可以忽略，但是扩散项不能忽略。

4.8.3 Forschheimer-Darcy 定律

上文中的 Darcy 定律也可以看做是表示压力梯度与速度的线性关系。这在雷诺数比较低的情况下是适用的。若雷诺数稍微增加，压力梯度与速度不再呈现线性关系，而呈现非线性发展：

$$\nabla p = C_1\overline{\mathbf{U}} + C_2|\overline{\mathbf{U}}|\overline{\mathbf{U}} \tag{4.262}$$

在这种情况下，方程(4.259)描述的 Darcy 定律不再适用。这是因为雷诺数在增加的过程中，粘性力的贡献占比逐渐减小，多孔区域内的固体导致的阻力贡献占比逐渐增加。回看方程(4.262)，其中右侧的第一项可以表示粘性损失，第二项可以表示惯性损失。其中 C_1 可以参考 Darcy 定律，C_2 通常可以写为：

$$\nabla p = -\frac{\mu}{K}\overline{\mathbf{U}} - \frac{1}{2}F|\overline{\mathbf{U}}|\overline{\mathbf{U}} \tag{4.263}$$

其中 F 表示 Forchheimer 系数。若 Forchheimer 系数以及渗透率均为二阶张量，定义 $\mathbf{D} = \mathbf{K}^{-1}$，有：

$$\nabla p = -\mu \mathbf{D}\overline{\mathbf{U}} - \frac{1}{2}\mathbf{F}|\overline{\mathbf{U}}|\overline{\mathbf{U}} \tag{4.264}$$

4.8.4 一个错误

回到方程(4.257)。这是目前主流商业 CFD 软件普遍采用的方法。更严谨来说，可能存在一定的错误。瑞典隆德大学曾在 Youtube 发表视频，题目为“Where text books go wrong about porous media”。在这里进行简单归纳。参考图4.28，俩种多孔介质模型具有不同的孔隙率。很明显右侧的孔隙率会更小一些。因此，按照方程(4.257)来计算的话，右侧的真实速度会比左侧的更大。但是如果对多孔介质区域做解析，会发现通过多孔管道的速度（假定右侧弯弯曲曲的多孔管道的横截面积相同），左右二者相同。早在 1989 年，Norman 就曾经发过文章讨论相关的问题 [48]。作者建议采用对方程(4.257)采用一个修正，演变为：

$$\overline{\mathbf{U}} = \tau\epsilon\mathbf{U} \tag{4.265}$$

其中 τ 取决于多孔管道的行程与当量长度。

图 4.28: 两种多孔介质模型，左右具有不同孔隙率。

4.8.5 多相多孔介质、IMPES 算法

如果多孔介质里面存在多相流，整个系统会变得更加复杂。其中最重要的就是附加了一个饱和度（saturation）的概念。考虑图4.27所示的网格单元，其中灰色的填充区域可能会存在两种流体。如果用 i 来表示第 i 种流体，V_i 表示第 i 种流体在网格单元内占据的体积，则有第 i 种流体的饱和度：

$$S_i = \frac{V_i}{V_f} \tag{4.266}$$

且有 $V_f = \sum V_i$，$\sum S_i = 1$。饱和度 S_i 是一个未知的变量。在无源项的情况下，其具有传输方程：

$$\epsilon\frac{\partial S_i}{\partial t} + \nabla \cdot \mathbf{U}_i^s = 0 \tag{4.267}$$

其中 $\mathbf{U}_i^s$ 表示第 i 项的表观速度。参考方程(4.261)，因为是多相流，则需要考虑重力的影响：

$$\mathbf{U}_i^s = -\frac{\mathbf{K}_i}{\mu_i} \cdot (\nabla p_i - \rho_i \mathbf{g}) \tag{4.268}$$

p_i 表示每一相的压力。注意其中的 $\mathbf{K}_i$，其取决于 i 项的饱和度，以及多孔介质本身的普适性的用于衡量传输流体能力的 $\mathbf{K}$。因此可以将 $\mathbf{K}_i$ 写为：

$$\mathbf{K}_i = \mathbf{K} k(S_i) \tag{4.269}$$

其中 $k(S_i)$ 表示第 i 相的相对渗透率，其取决于第 i 相的饱和率。

若考虑简单的两相流，则有：

$$\begin{aligned} \epsilon \frac{\partial S_a}{\partial t} + \nabla \cdot \mathbf{U}_a^s = 0 \\ \epsilon \frac{\partial S_b}{\partial t} + \nabla \cdot \mathbf{U}_b^s = 0 \end{aligned} \tag{4.270}$$

将两个式子相加，有：

$$\nabla \cdot (\mathbf{U}_a^s + \mathbf{U}_b^s) = 0 \tag{4.271}$$

即

$$\nabla \cdot \left(-\frac{\mathbf{K}_a}{\mu_a} \cdot (\nabla p_a - \rho_a \mathbf{g}) - \frac{\mathbf{K}_b}{\mu_b} \cdot (\nabla p_b - \rho_b \mathbf{g}) \right) = 0 \tag{4.272}$$

因为表面张力的存在，定义毛细张力 p_c 与两相的压力关系为：

$$p_c = p_a - p_b \tag{4.273}$$

将方程(4.273)带入到(4.272)中有：

$$\nabla \cdot \left(-\frac{\mathbf{K}_a}{\mu_a} \cdot (\nabla p_a - \rho_a \mathbf{g}) - \frac{\mathbf{K}_b}{\mu_b} \cdot (\nabla p_a - \rho_b \mathbf{g} - \nabla p_c) \right) = 0 \tag{4.274}$$

移项有：

$$-\nabla \cdot \left(\frac{\mathbf{K}_a}{\mu_a} + \frac{\mathbf{K}_b}{\mu_b} \right) \cdot \nabla p_a = -\nabla \cdot \left(\frac{\rho_a \mathbf{K}_a}{\mu_a} + \frac{\rho_b \mathbf{K}_b}{\mu_b} \right) \cdot \mathbf{g} - \nabla \cdot \left(\frac{\mathbf{K}_b}{\mu_b} \cdot \nabla p_c \right) \tag{4.275}$$

依据链条法则 $\nabla p_c = \frac{\partial p_c}{\partial S_b} \nabla S_b$，有：

$$\nabla \cdot \left(\frac{\mathbf{K}_a}{\mu_a} + \frac{\mathbf{K}_b}{\mu_b} \right) \cdot \nabla p_a = \nabla \cdot \left(\frac{\rho_a \mathbf{K}_a}{\mu_a} + \frac{\rho_b \mathbf{K}_b}{\mu_b} \right) \cdot \mathbf{g} + \nabla \cdot \left(\frac{\mathbf{K}_b}{\mu_b} \cdot \frac{\partial p_c}{\partial S_b} \nabla S_b \right) \tag{4.276}$$

同时将速度与压力的关系带入到饱和度连续性方程有：

$$\epsilon \frac{\partial S_b}{\partial t} - \nabla \cdot \left(\frac{\mathbf{K}_b}{\mu_b} \cdot \left(\nabla p_a - \rho_b \mathbf{g} - \frac{\mathbf{K}_b}{\mu_b} \cdot \frac{\partial p_c}{\partial S_b} \nabla S_b \right) \right) = 0 \tag{4.277}$$

方程(4.276)与(4.277)一共具有两个方程，未知数为 S_b 以及 p_a，若进一步将 $\frac{\partial p_c}{\partial S_b}$ 进行封闭[31]，则方程(4.276)与(4.277)既可以进行求解。其中方程(4.276)可以隐性离散来计算压力，方程(4.277)显性离散来更新饱和度。这种算法也被称之为隐性压力显性饱和度方法（IMPES，IMplicit Pressure Explicit Saturation）[70]。

4.8.6 Darcy-Brinkman-Stokes 方程、固相连续介质模型

常规的多孔介质方法，是在 NS 方程中添加一个多孔介质源项来实现。另外一种方法是使用拓展的 Darcy-Brinkman-Stokes（DBS）方法来实现。在 DBS 方法中，需要附加孔隙率 ϵ 的概念，若全部为流体，$\epsilon_f = 1$。拓展 Darcy-Brinkman-Stokes（DBS）方法的控制方程可以写为 [151]：

$$\epsilon_s + \epsilon_f = 1 \tag{4.278}$$

$$\frac{\partial \overline{\mathbf{U}}_f}{\partial t} + \nabla \cdot \left(\frac{1}{\epsilon_f} \overline{\mathbf{U}}_f \overline{\mathbf{U}}_f \right) = -\epsilon_f \nabla \hat{p}_f + \epsilon_f \mathbf{g} + \nabla \cdot (\mu_f \nabla \overline{\mathbf{U}}_f) - \epsilon_f \nu_f \frac{1}{k} \overline{\mathbf{U}}_f \tag{4.279}$$

$$\frac{1}{k} = \frac{(1-\epsilon_f)^2}{k_0 \epsilon_f^3} \tag{4.280}$$

当 ϵ_f 趋向于 1 时（k^{-1} 趋向于 0），方程(4.279)趋向于 NS 方程。在其他情况下，所有的项均存在，但是 Darcy 项占主导，也即演变为 Darcy 方程。一些文献采用的为 DBS 方法对多孔介质模型进行模拟 [60, 151]。相对于通过在 NS 方程中添加源项的方法（其可以被当做一种固定分区孔隙率的方法），DBS 方法可适用性更广。例如其同时可以用于一些孔隙率可变的工况（如溶解与析出）。

例如，若考虑溶解行为，也即流体的包含某些反应物，会对多孔介质进行化学侵蚀，在这种情况下有质量守恒方程 [151]：

$$\frac{\partial \epsilon_f}{\partial t} + \nabla \cdot \overline{\mathbf{U}}_f = \frac{\dot{m}_f}{\rho_f} \tag{4.281}$$

如果 $\dot{m}_f > 0$，则表示发生溶解行为，反之则会发成析出。同时有：

$$\frac{\partial \epsilon_s}{\partial t} = -\frac{\dot{m}_f}{\rho_s} \tag{4.282}$$

$\dot{m}_f$ 仅仅在固体流体界面存在值。

对于给定反应物 C，其在 DBS 框架下的传输方程可以写为：

$$\frac{\partial \epsilon_f C}{\partial t} + \nabla \cdot \left(\overline{\mathbf{U}}_f C \right) - \nabla \cdot (\epsilon_f D_C \nabla C) = 0 \tag{4.283}$$

若写成质量分数的形式，可以写为 [151]：

$$\frac{\partial \epsilon_f Y}{\partial t} + \nabla \cdot \left(\overline{\mathbf{U}}_f Y \right) - \nabla \cdot (\epsilon_f D_C \nabla Y) = \dot{m}_f \tag{4.284}$$

[31] 可以参考文献 [70]，调用 Van Genuchten 模型进行封闭。

4.9 结构力学、可变形的固体

大部分的固体应力分析都使用的是有限元方法。然而使用有限体积法计算应力应变以及位移也是可行的。同时有限体积法具有一些优良的优点：如守恒特性。在将力离散到网格单元面的时候，所有的力均被抵消仅仅在边界上存在受力。这与与流体的质量守恒类似。因此使用有限体积法做结构分析也未尝不可。同样的，流体与固体也有很大的相似性。流体被认为是可以“流动”的，固体被认为是“可以变形的”。虽然流体力学与固体力学基本是完全不同的研究领域，且研究人员基本不重叠，但是流体与固体的控制方程均来自于牛顿第二定律。流体与固体的主要区别在与内部的应力的不同处理。固体之所以可以被保持完整不动的，也是因为内部存在流体没有的应力：弹性应力。固体同样存在屈服应力，只有受力在屈服应力之上，才会发生变形。在流体与固体的中间区域，还存在一些近似流体近似固体的流体，这些流体被称之为特定的非牛顿流体（如宾汉流体）。

对固体进行分析时，区分为不同的材料，如弹性固体以及塑性固体。塑性固体的控制方程类似于流体的控制方程。只不过其中的应力模型有很大的区别。弹性固体还进一步细分为线弹性、超弹性等。固体的弹性可以理解为固体在经受一定的应力之后，恢复到原始状态的行为。如果应力持续增加，固体会超过弹性极限，导致不可恢复的形变。在这种情况下，固体表现出一种塑性的行为，即一种可变形的状态。如果应力持续增大，固体会达到断裂点发生断裂。

对于一个线弹性材料，其位移矢量的控制方程为：

$$\frac{\partial^2 \mathbf{D}}{\partial t^2} - \nabla \cdot \left(\mu(\nabla \mathbf{D} + \nabla \mathbf{D}^T) + \lambda(\nabla \cdot \mathbf{D})\mathbf{I}\right) = 0 \tag{4.285}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{4.286}$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{4.287}$$

其中 $\mathbf{D}$ 表示位移矢量，E 表示杨氏模量，λ 表示拉梅系数的第 1 个参数，μ 表示拉梅系数的第 2 个参数（在流体力学中理解为粘度，在固体力学中理解为剪切模量）。

对于粘塑性材料，一些文献表示可以将粘塑性材料采用近似动量方程的方程来描述[30]：

$$\frac{\partial \mathbf{U}_s}{\partial t} + \nabla \cdot (\mathbf{U}_s \mathbf{U}_s) - \nabla \cdot \left(\nu_{eff}(\nabla \mathbf{U}_s + \nabla \mathbf{U}_s^T) - \frac{2}{3}(\nabla \cdot \mathbf{U}_s)\mathbf{I}\right) = \nabla \cdot \boldsymbol{\tau} + \mathbf{g} \tag{4.288}$$

其中 $\mathbf{U}_s$ 表示固体的移动速度，$\boldsymbol{\tau}_s$ 表示 Terzaghi 应力等相关项，$\mathbf{g}$ 表示重力加速度。

4.9.1 两相固液多孔介质模型

将方程(4.288)与上一节讨论的 DBS 方法互相结合，可以构建适用于两相的固液多孔介质模型。在这个模型中，固体区域可以移动，且液体在流经固体区域时可以将固体看做

多孔介质区域。为了实现这种物理过程，需要定义相分数 ϵ。在 $\epsilon = 1$ 的时候表示纯流体区域。在 $\epsilon = 0$ 表示纯多孔介质区域。ϵ 通过固相速度 $\mathbf{U}_s$ 进行传输。因此固相多孔介质区域可移动。因此，流体区域需采用 DBS 方法来处理。在两相情况下，固液存在在动量交换，因此存在动量交换项。综上所述，在不考虑重力以及不可压缩的情况下，两相固液多孔介质模型中的流体相的控制方程为：

$$\epsilon_s + \epsilon_f = 1 \tag{4.289}$$

$$\frac{\partial \epsilon_f}{\partial t} + \nabla \cdot \overline{\mathbf{U}}_f = 0 \tag{4.290}$$

$$\frac{\partial \overline{\mathbf{U}}_f}{\partial t} + \nabla \cdot \left(\frac{1}{\epsilon_f} \overline{\mathbf{U}}_f \overline{\mathbf{U}}_f \right) = -\epsilon_f \nabla \hat{p}_f + \nabla \cdot \overline{\boldsymbol{\tau}}_f - \frac{\epsilon_f \nu_f}{k} \left(\overline{\mathbf{U}}_f - \overline{\mathbf{U}}_s \right) \tag{4.291}$$

若参考方程(4.291)构建固相方程，有 [30]：

$$\frac{\partial \overline{\mathbf{U}}_s}{\partial t} + \nabla \cdot \left(\frac{1}{\epsilon_s} \overline{\mathbf{U}}_s \overline{\mathbf{U}}_s \right) = -\epsilon_s \nabla \hat{p}_f + \nabla \cdot \overline{\boldsymbol{\tau}}_s + \frac{\epsilon_f \nu_f}{k} \left(\overline{\mathbf{U}}_f - \overline{\mathbf{U}}_s \right) \tag{4.292}$$

其中对流项中需要除掉 ϵ_s，在很多情况下多孔介质仅仅存在计算域内的一部分因此大部分区域 $\epsilon_s = 0$，这会导致分母为 0 的情况引起稳定性问题。因此针对固相可以采用 intrinsic 形式的速度方程[32][30]：

$$\frac{\partial \epsilon_s \hat{\mathbf{U}}_s}{\partial t} + \nabla \cdot \left(\epsilon_s \hat{\mathbf{U}}_s \hat{\mathbf{U}}_s \right) = -\epsilon_s \nabla \hat{p}_f + \nabla \cdot \left(\epsilon_s \hat{\boldsymbol{\tau}}_s \right) + \frac{\epsilon_f \nu_f}{k} \left(\overline{\mathbf{U}}_f - \epsilon_s \hat{\mathbf{U}}_s \right) \tag{4.293}$$

同时有固相相分数传输方程：

$$\frac{\partial \epsilon_s}{\partial t} + \nabla \cdot \left(\epsilon_s \hat{\mathbf{U}}_s \right) = 0 \tag{4.294}$$

方程(4.294)可以用来传输固相分数，(4.293)可以用来更新固相速度，(4.291)与(4.290)可以用来更新流体速度与压力。

4.10 稀疏线性系统求解器

CFD 方程在离散之后会形成下面的求解系统：

$$\mathcal{A}\mathbf{x} = \mathbf{b} \tag{4.295}$$

其中 $\mathcal{A}$ 表示一个矩阵系统，$\mathbf{x}$ 为待求的未知量，$\mathbf{b}$ 表示源项。对于真实的 CFD 问题，$\mathbf{x}$ 的未知量与网格数量相当。因此，如果使用大家非常熟悉的直接解法求解方程(4.295)是基本上不可行的。这是因为直接解法需要进行的操作次数是非常庞大的。例如，对于三维网格，每个方向存在 100 个网格点，那么会形成 100 万阶矩阵。对这个 100 万阶矩阵进行直接求解，需要使用 100! 次操作。

[32] 其在相分数趋向于 0 的时候存在方程奇异问题，其可以通过类似双流体模型的做法来进行处理。

在这种情况下，通常需要采用迭代的方法进行计算。迭代的方法看起来天生就是为了求解偏微分方程组而生。这是因为大量的偏微分方程组在离散的过程中，在调用格式的时候，通常仅仅调用相邻网格点的变量，即使调用高阶格式（三阶四阶），在网格数量很大的时候，离散形成的线性系统也是一个非常稀疏的求解系统。这里的稀疏是指变量 $\mathcal{A}$ 大部分的元素都是 0。如果使用中心格式，对于一维问题，$\mathcal{A}$ 的每一行只有 3 个元素，形成一个对角阵。对于二维问题，$\mathcal{A}$ 的每一行只有 5 个元素。会形成一个带状（Banded）系统[33]。很明显，$\mathcal{A}$ 为一个稀疏矩阵。另外一个规律是，对于很多网格编号中规中矩的范例（如没有对网格编号进行打乱排布的结构网格），$\mathcal{A}$ 的形态也是非常规则的。典型的迭代方法有高斯赛德尔迭代法与雅克比迭代法。

4.10.1 基本迭代求解器

本书中将雅克比迭代求解器、高斯迭代求解器以及相关衍生分支分类为基本迭代求解器。雅克比迭代法的思想非常简单。对于矩阵 $\mathcal{A}$，其可以分解为上三角矩阵 U、下三角矩阵 L 以及对角阵 D[34]：

$$\mathcal{A} = D - U - L \tag{4.296}$$

在这种情况下，那么第 $k+1$ 次迭代的解 $\mathbf{x}^{k+1}$ 可以这样获得：

$$\mathbf{x}^{k+1} = D^{-1}[\mathbf{b} + (L+U)\mathbf{x}^k] \tag{4.297}$$

类似的，高斯赛德尔迭代法可以这样求解：

$$\mathbf{x}^{k+1} = (I - D^{-1}L)^{-1}[D^{-1}\mathbf{b} + D^{-1}U\mathbf{x}^k] \tag{4.298}$$

对于各种不同的迭代法，可以定义普适性方程：

$$\mathbf{x}^{k+1} = Gx^k + c \tag{4.299}$$

不同的迭代法，对应不同的 G 与 c。同时，真实解 $\mathbf{x}$ 满足方程：

$$(I - G)\mathbf{x} = c \tag{4.300}$$

定义 k 次迭代的误差为 $\mathbf{e}^k$，有：

$$\mathbf{e}^k = \mathbf{x} - \mathbf{x}^k \tag{4.301}$$

例如，第 0 次（初始条件）以及第 1 次的误差为：

$$\mathbf{e}^0 = \mathbf{x} - \mathbf{x}^0 \tag{4.302}$$

[33] 一维问题如果调用高阶格式也会形成带状系统。同时，多维问题带状系统的带宽往往要比一维问题的大。

[34] 在这里将 L 和 D 写成负号的形式，是因为对于 PDE 离散系统，对角系数与非对角系数通常为反号。例如我们来看一个简单的一维方程：$-\frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) = 0$。在这里需要注意的是这个方程中的扩散项系数为负值。虽然其正负不影响计算结果，但这个括号会影响对角线系数的正负。只有为负值的时候，对角线系数为正。且只有为负值的时候，其表示一个扩散方程。

$$\mathbf{e}^1 = \mathbf{x} - \mathbf{x}^1 \tag{4.303}$$

如果将方程(4.300)、(4.299)代入到方程(4.303)有：

$$\mathbf{e}^1 = \mathbf{x} - \mathbf{x}^1 = G\mathbf{x} - (\mathbf{x}^1 - c) = G\mathbf{x} - G\mathbf{x}^0 = G(\mathbf{x} - \mathbf{x}^0) = G\mathbf{e}^0 \tag{4.304}$$

同理，有：

$$\mathbf{e}^{k+1} = G\mathbf{e}^k = G^2\mathbf{e}^{k-1} = ... = G^{k+1}\mathbf{e}^0 \tag{4.305}$$

可以看出 G 的作用类似一种缩放因子，为了使得 $\mathbf{e}^{k+1}$ 趋向于无穷小，其谱半径需要小于 1，即：

$$\rho(G) < 1 \tag{4.306}$$

其中 $\rho(G)$ 表示 G 的谱半径。

4.11 CFD 小问题

4.11.1 物质导数

CFD 中存在一个概念，叫物质导数。物质导数就是数学中的全导数。理解物质导数需要借助上文中提及的流动模型。现选取随流线运动的无穷小微团上的温度 $T(x, y, z, t)$ 进行分析。需要注意的是，在上文中指出无穷小微团上的变量可以微分算符 d 来表示，这里的温度并没有表示为 $\mathrm{d}T$，其原因在于温度并不是一个随物质的量变化而变化的量。不管存在多少的分子，温度均为 T。类似的这种量，在 CFD 被称之为强度量。反之，质量、体积等为广度量。

在 t_1 时间点，流体微团的位置在 (x_1, y_1, z_1)，温度为 $T_1(x_1, y_1, z_1, t_1)$。随着微团的移动，在 t_2 时刻，位置变为 (x_2, y_2, z_2)，同时温度为 $T_2(x_2, y_2, z_2, t_2)$。既然温度为时间和位置的函数，则可参考方程(3.4)对温度进行泰勒展开：

$$T_2 = T_1 + \left.\frac{\partial T}{\partial x}\right|_{x=x_1}(x_2 - x_1) + \left.\frac{\partial T}{\partial y}\right|_{y=y_1}(y_2 - y_1) + \left.\frac{\partial T}{\partial z}\right|_{z=z_1}(z_2 - z_1) + \left.\frac{\partial T}{\partial t}\right|_{t=t_1}(t_2 - t_1) \tag{4.307}$$

对方程 (4.307) 左右两边除以 $t_2 - t_1$ 有：

$$\frac{T_2 - T_1}{t_2 - t_1} = \left.\frac{\partial T}{\partial x}\right|_{x=x_1}\frac{x_2 - x_1}{t_2 - t_1} + \left.\frac{\partial T}{\partial y}\right|_{y=y_1}\frac{y_2 - y_1}{t_2 - t_1} + \left.\frac{\partial T}{\partial z}\right|_{z=z_1}\frac{z_2 - z_1}{t_2 - t_1} + \left.\frac{\partial T}{\partial t}\right|_{t=t_1} \tag{4.308}$$

接下来定义温度 T 在 (x_1, y_1, z_1) 点的物质导数为移动的无穷小微团通过 (x_1, y_1, z_1) 点的时候，无穷小微团针对时间的温度瞬时变化率，即：

$$\left.\frac{\mathrm{D}T}{\mathrm{D}t}\right|_{x=x_1, y=y_1, z=z_1} = \lim_{t_2 \to t_1}\frac{T_2 - T_1}{t_2 - t_1} \tag{4.309}$$

依据速度的定义，有 $\lim_{t_2\to t_1}\frac{x_2-x_1}{t_2-t_1}=u$，$\lim_{t_2\to t_1}\frac{y_2-y_1}{t_2-t_1}=v$，$\lim_{t_2\to t_1}\frac{z_2-z_1}{t_2-t_1}=w$，代入到方程 (4.308) 有 (x_1,y_1,z_1) 点的物质导数：

$$\left.\frac{\mathrm{D}T}{\mathrm{D}t}\right|_{x=x_1,y=y_1,z=z_1}=\left.\frac{\partial T}{\partial x}\right|_{x=x_1}u+\left.\frac{\partial T}{\partial y}\right|_{y=y_1}v+\left.\frac{\partial T}{\partial z}\right|_{z=z_1}w+\left.\frac{\partial T}{\partial t}\right|_{t=t_1} \tag{4.310}$$

对于任意一点，有

$$\frac{\mathrm{D}T}{\mathrm{D}t}=\frac{\partial T}{\partial x}u+\frac{\partial T}{\partial y}v+\frac{\partial T}{\partial z}w+\frac{\partial T}{\partial t} \tag{4.311}$$

方程 (4.311) 即为温度 T 在笛卡尔坐标系下的物质导数的定义。可见，物质导数即为数学上物理量对时间的全导数。

参考内积的数学定义，方程 (4.311) 可写为[35]：

$$\frac{\mathrm{D}T}{\mathrm{D}t}=\frac{\partial T}{\partial t}+\mathbf{U}\cdot\nabla T \tag{4.312}$$

上述以温度 T 举例说明物质导数的由来。类似的，有速度 $\mathbf{U}$ 的物质导数：

$$\frac{\mathrm{D}\mathbf{U}}{\mathrm{D}t}=\frac{\partial \mathbf{U}}{\partial t}+\mathbf{U}\cdot\nabla\mathbf{U} \tag{4.313}$$

速度的物质导数表示流体微团的加速度。除此之外，任何的流场数据都可以用物质导数来表示，如：

$$\frac{\mathrm{D}()}{\mathrm{D}t}=\frac{\partial()}{\partial t}+u\frac{\partial()}{\partial x}+v\frac{\partial()}{\partial y}+w\frac{\partial()}{\partial z}=\frac{\partial()}{\partial t}+\mathbf{U}\cdot\nabla() \tag{4.314}$$

方程 (4.314) 中的 $\frac{\partial()}{\partial t}$ 又称为局部导数，第二项 $\mathbf{U}\cdot\nabla()$ 又称之为对流导数。需要注意的是，如果求解器不调用拉格朗日粒子，方程(4.312)左边的物质导数项并不能够直接求出，其只能通过方程右边计算而来。

方程(4.312)可以理解为：

$$\underbrace{\frac{\mathrm{D}}{\mathrm{D}t}}_{\text{Lagrangian}}=\underbrace{\frac{\partial}{\partial t}+\mathbf{U}\cdot\nabla}_{\text{Euler}} \tag{4.315}$$

其中 $\frac{\partial}{\partial t}$ 表示对时间的变化率，$\mathbf{U}\cdot\nabla$ 表示对流的变化率。在这里增加一个对欧拉以及拉格朗日的解释，如图4.29所示，欧拉方法描述一个固定点的变量的时间变化值，如图中方框内的变量。拉格朗日方法描述一个移动的粒子的变量的时间变化（如图中红色粒子的移动）

[35] 在这里要注意，∇ 算子一般也可以写为：

$$\nabla=\left(\frac{\partial}{\partial x},\frac{\partial}{\partial y},\frac{\partial}{\partial z}\right)^T$$

因此可以定义操作符 $\mathbf{U}\cdot\nabla$ 为 $\mathbf{U}$ 与 ∇ 算子的内积：

$$\mathbf{U}\cdot\nabla=u\frac{\partial}{\partial x}+v\frac{\partial}{\partial y}+w\frac{\partial}{\partial z}$$

这样，$\mathbf{U}\cdot\nabla T$ 也可以写成 $(\mathbf{U}\cdot\nabla)T$。同理：

$$(\mathbf{U}\cdot\nabla)\mathbf{U}=\begin{pmatrix} u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z} \\ u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z} \\ u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\end{pmatrix}$$

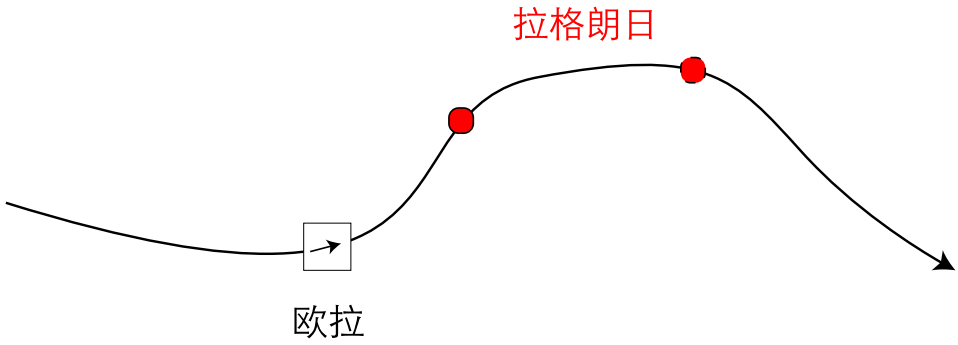


图 4.29: 欧拉与拉格朗日的描述。

[36]。以温度举例，欧拉方法描述方框内温度随着时间的变化，拉格朗日方法描述移动的粒子的温度在不同的位置处随时间产生的变化。另外，拉格朗日方法的自变量通常为 t，欧拉方法的自变量通常为 (x, y, z, t)。例如，有粒子随时间变化的位置矢量定义为：

$$\mathbf{S} = (x, y, z) = (t^2, 2t, 0) \tag{4.316}$$

则有

$$x = t^2, y = 2t, z = 0 \tag{4.317}$$

那么拉格朗日框架下的粒子的速度矢量通常表示为：

$$\mathbf{U}(t) = (2t, 2, 0) \tag{4.318}$$

欧拉框架下的速度矢量通常表示为：

$$\mathbf{U}(x, y, z, t) = (2t, 2, 0) = (2\sqrt{x}, 2, 0) \tag{4.319}$$

另外从方程(4.319)可以看出，针对本示例在固定的网格位置处的速度不变。从物理意义来讲，即粒子均以同样的速度通过某固定网格点。

4.11.2 结构网格与非结构网格

结构网格与非结构网格的区别主要在于网格节点的编号形式。结构网格的网格节点通常可以按照次序排列，对于三维网格，通常按照 i, j, k 的形式排列。例如 y 方向上游与下游的网格序号可以表示为 $i, j-1, k$，$i, j+1, k$ 等，上游的上游节点可以表示为 $i, j-2, k$。由于结构网格寻址方便，因此在结构网格上很容易拓展高阶格式。非结构网格则不限定于网格编号形式。其通过复杂的网格连接性来定义网格编号，例如需定义网格节点序号、定义网格节点与面相连接的序号等。结构网格思想更偏向于有限差分。有限体积法则不限定于任何网格类型，更适用于非结构网格。

[36]在守恒法则研究领域，物质导数也表示沿着特征线的变量时间变化率。

由于有限差分法诞生较早，有限体积法诞生较晚。有限差分主要使用规则的结构网格来进行。同时，对于方程离散的各个项，有限差分的方式更易于理解。因此，CFD 教材中大量的采用结构网格与有限差分结合的方式进行介绍。现存商业/开源 CFD 软件为了追求普适性，很少使用结构网格的方式进行寻址。在这里强调一些常见的误区：

- 结构网格单元不一定是四边形或六面体。结构网格单元也可以是三角形等。如图4.30所示，其中的三角形网格也可以用 i, j, k 进行标识。
- 四边形、六面体网格也可能是非结构网格。如图4.31所示，是一种六面体非结构网格，由于几何中心部位进行了糙化处理，无法通过 i, j, k 进行寻址，这是一种多块六面体非结构网格。

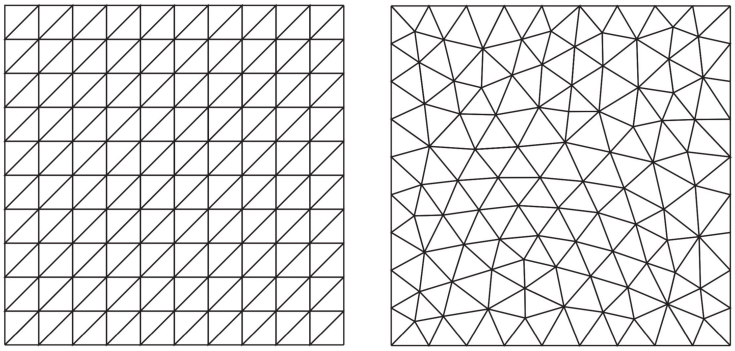


图 4.30: 三角形结构网格（左）与非结构网格（右）。

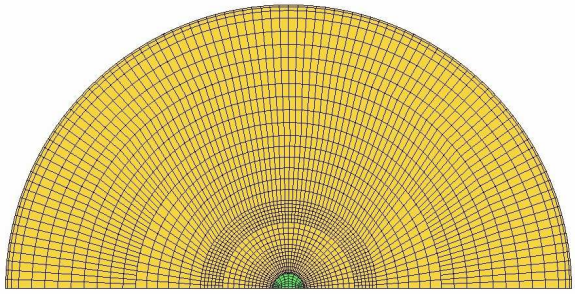


图 4.31: 六面体非结构网格 [56]。

4.11.3 体膨胀系数 $-\frac{2}{3}\mu(\nabla\cdot\mathbf{U})\mathbf{I}$

在方程(3.69)中，可压缩流体的 τ 中需要考虑 $-\frac{2}{3}\mu(\nabla\cdot\mathbf{U})\mathbf{I}$。其中的 $-\frac{2}{3}\mu$ 也经常被称之为体膨胀/压缩粘度。这是因为 $-\frac{2}{3}\mu(\nabla\cdot\mathbf{U})\mathbf{I}$ 可以转换为密度相关变量：

$$-\frac{2}{3}\mu(\nabla\cdot\mathbf{U})\mathbf{I} = -\frac{2}{3}\mu\left(-\frac{1}{\rho}\frac{\mathrm{D}\rho}{\mathrm{D}t}\right)\mathbf{I} \tag{4.320}$$

可见，$\frac{1}{\rho}\frac{\mathrm{D}\rho}{\mathrm{D}t}$ 为密度的物质导数项，从第4.11.1节可知，其表示拉格朗日框架下密度小微团的变化率。因此，即跟体膨胀相关。

4.11.4 二阶张量不变量 Invariants

给定一个二阶张量，可以定义其各种不变量。给定一个矩阵 $\mathbf{A}$，其最基本的三个不变量分别是 I_1, I_2, I_3：

$$\begin{aligned} I_1 &= \mathrm{tr}(\mathbf{A}) = \lambda_1 + \lambda_2 + \lambda_3 \\ I_2 &= \frac{1}{2}\left((\mathrm{tr}(\mathbf{A}))^2 - \mathrm{tr}\left(\mathbf{A}^2\right)\right) = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 \\ I_3 &= \det(\mathbf{A}) = \lambda_1\lambda_2\lambda_3 \end{aligned} \tag{4.321}$$

其中 λ 为 $\mathbf{A}$ 的特征值。其中 I_1 被称之为第一不变量，I_2 被称之为第二不变量，I_3 被称之为第三不变量。不变量也可以通过 $\mathbf{A}$ 的元素来进行表示：

$$\begin{aligned} &I_1 = A_{11} + A_{22} + A_{33} \\ &I_2 = A_{11}A_{22} + A_{22}A_{33} + A_{11}A_{33} - A_{12}A_{21} - A_{23}A_{32} - A_{13}A_{31} \\ &I_3 = \\ &-A_{13}A_{22}A_{31} + A_{12}A_{23}A_{31} + A_{13}A_{21}A_{32} - A_{11}A_{23}A_{32} - A_{12}A_{21}A_{33} + A_{11}A_{22}A_{33} \end{aligned} \tag{4.322}$$

4.11.5 应力的 Deviatoric 与 Hydrostatic

在 CFD 中，通常要将二阶张量 σ 的 deviatoric 部分和 hydrostatic 部分做区分，如：

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_{dev} + \boldsymbol{\sigma}_{hyd} \tag{4.323}$$

用下面这个例子可以表明 $\boldsymbol{\sigma}_{dev}$ 和 $\boldsymbol{\sigma}_{hyd}$ 的计算关系。如果

$$\boldsymbol{\sigma} = \begin{pmatrix} 50 & 30 & 20 \\ 30 & -20 & -10 \\ 20 & -10 & 10 \end{pmatrix} \tag{4.324}$$

则正应力 hydrostatic 部分可以这样计算：

$$\sigma_{hyd} = \frac{50 - 20 + 10}{3} = 13.3 \tag{4.325}$$

$$\boldsymbol{\sigma}_{hyd} = \begin{pmatrix} 13.3 & 0 & 0 \\ 0 & 13.3 & 0 \\ 0 & 0 & 13.3 \end{pmatrix} \tag{4.326}$$

偏应力 deviatoric 部分可以这样计算：

$$\boldsymbol{\sigma}_{dev} = \boldsymbol{\sigma} - \boldsymbol{\sigma}_{hyd} = \begin{pmatrix} 36.7 & 30 & 20 \\ 30 & -33.3 & -10 \\ 20 & -10 & -3.3 \end{pmatrix} \tag{4.327}$$

可以看出，$\boldsymbol{\sigma}_{dev}$ 的迹是 0，也就是说，对于一个二阶张量的偏应力部分，其正应力部分为 0。

二阶张量 $\boldsymbol{\sigma}$ 的正应力与偏应力部分也可以这样推导：

$$\boldsymbol{\sigma}_{hyd} = \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I} \tag{4.328}$$

$$\boldsymbol{\sigma}_{dev} = \boldsymbol{\sigma} - \boldsymbol{\sigma}_{hyd} = \boldsymbol{\sigma} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I} \tag{4.329}$$

为什么要将二阶张量分为正应力部分与偏应力部分，这是因为压力 p 可以看做一个正应力部分，剪切应力 $\boldsymbol{\tau}$ 可以看做是偏应力部分，二者可以构成一个整体的二阶张量 $\boldsymbol{\sigma}$：

$$\boldsymbol{\sigma} = \boldsymbol{\sigma} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I} + \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I} \tag{4.330}$$

令

$$-p\mathbf{I} = \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I} \tag{4.331}$$

有：

$$\boldsymbol{\sigma} = -p\mathbf{I} + \left(\boldsymbol{\sigma} - \frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\mathbf{I}\right) = \boldsymbol{\sigma}_{hyd} + \boldsymbol{\sigma}_{dev} \tag{4.332}$$

在这种情况下，动量方程右侧可以理解为：

$$-\nabla p + \nabla \cdot \boldsymbol{\tau} = \nabla \cdot \boldsymbol{\sigma} \tag{4.333}$$

一些教材偏向使用 $\nabla \cdot \boldsymbol{\sigma}$ 来进行讲解，其中 $\boldsymbol{\sigma}$ 被称之为柯西应力张量。本笔记则采用 $-\nabla p + \nabla \cdot \boldsymbol{\tau}$，因为其更有易于理解。

4.11.6 湍流的涡粘近似、$\frac{2}{3}\rho k\mathbf{I}$，$P = \tau_{ij}\frac{\partial u_i}{\partial x_j}$，$\sqrt{2S_{ij}S_{ij}}$

将动量方程左侧的对流项进行雷诺平均后，会出现雷诺应力项：

$$\overline{\nabla \cdot (\rho \mathbf{U}\mathbf{U})} = \nabla \cdot (\overline{\rho}\overline{\mathbf{U}\mathbf{U}}) + \nabla \cdot (\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}) \tag{4.334}$$

$\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}$ 这一项若放在动量方程右侧会出现一个负号，即方程右侧为 $= \ldots - \overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}$。我们把 $-\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}$ 称之为雷诺应力 $\boldsymbol{\tau}_t$[37]，其为一个二阶对称张量。$\overline{\mathbf{U}'\mathbf{U}'}$ 可以写为：

$$\overline{\mathbf{U}'\mathbf{U}'} = \begin{pmatrix} \overline{\mathbf{U}_x'^{\,2}} & \ldots & \ldots \\ \ldots & \overline{\mathbf{U}_y'^{\,2}} & \ldots \\ \ldots & \ldots & \overline{\mathbf{U}_w'^{\,2}} \end{pmatrix} \tag{4.335}$$

[37]注意，雷诺应力项在方程右侧为负号：$\ldots = \ldots - \nabla \cdot (\rho\overline{\mathbf{U}'\mathbf{U}'})$

另一方面，湍流动能可以写为

$$k=\frac{1}{2}\mathrm{tr}\left(\overline{\mathbf{U}'\mathbf{U}'}\right)=\frac{1}{2}\left(\overline{\mathbf{U}_x'^{\,2}}+\overline{\mathbf{U}_y'^{\,2}}+\overline{\mathbf{U}_w'^{\,2}}\right) \tag{4.336}$$

因此有：

$$\overline{\mathbf{U}'\mathbf{U}'}-\frac{2}{3}k\mathbf{I}=$$
$$\begin{pmatrix} \mathbf{U}_x'^{\,2}-\frac{1}{3}\left(\overline{\mathbf{U}_x'^{\,2}}+\overline{\mathbf{U}_y'^{\,2}}+\overline{\mathbf{U}_w'^{\,2}}\right) & \cdots & \cdots \\ \cdots & \mathbf{U}_y'^{\,2}-\frac{1}{3}\left(\overline{\mathbf{U}_x'^{\,2}}+\overline{\mathbf{U}_y'^{\,2}}+\overline{\mathbf{U}_w'^{\,2}}\right) & \cdots \\ \cdots & \cdots & \mathbf{U}_w'^{\,2}-\frac{1}{3}\left(\overline{\mathbf{U}_x'^{\,2}}+\overline{\mathbf{U}_y'^{\,2}}+\overline{\mathbf{U}_w'^{\,2}}\right) \end{pmatrix} \tag{4.337}$$

很明显

$$\mathrm{tr}\left(\overline{\mathbf{U}'\mathbf{U}'}-\frac{2}{3}k\mathbf{I}\right)=0 \tag{4.338}$$

$$\mathrm{tr}\left(\overline{\mathbf{U}'\mathbf{U}'}\right)=\mathrm{tr}(\frac{2}{3}k\mathbf{I}) \tag{4.339}$$

从数学公式的角度来看，$\overline{\mathbf{U}'\mathbf{U}'}$ 的偏应力 $\mathrm{dev}\left(\overline{\mathbf{U}'\mathbf{U}'}\right)$ 为：

$$\mathrm{dev}\left(\overline{\mathbf{U}'\mathbf{U}'}\right)=\overline{\mathbf{U}'\mathbf{U}'}-\frac{1}{3}\mathrm{tr}(\overline{\mathbf{U}'\mathbf{U}'})\mathbf{I} \tag{4.340}$$

因此，对于 $\overline{\mathbf{U}'\mathbf{U}'}$，其偏应力部分还可以写为：

$$\mathrm{dev}\left(\overline{\mathbf{U}'\mathbf{U}'}\right)=\overline{\mathbf{U}'\mathbf{U}'}-\frac{2}{3}k\mathbf{I} \tag{4.341}$$

现在将雷诺应力项通常被分为正应力部分与偏应力部分：

$$-\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}=-\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}-\frac{1}{3}\mathrm{tr}\left(-\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}\right)\mathbf{I}+\frac{1}{3}\mathrm{tr}\left(-\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}\right)\mathbf{I} \tag{4.342}$$

将方程(4.336)代入到(4.342)有：

$$-\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}=-\left(\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}-\frac{2}{3}\overline{\rho}k\mathbf{I}\right)-\frac{2}{3}\overline{\rho}k\mathbf{I} \tag{4.343}$$

最后的 $-\frac{2}{3}\overline{\rho}k\mathbf{I}$ 通常与动量方程中的压力梯度项 $\nabla\overline{p}$ 组合到一起：

$$-\nabla\tilde{\overline{p}}=-\nabla\overline{p}-\nabla\cdot\left(\frac{2}{3}\overline{\rho}k\mathbf{I}\right)=-\nabla\left(\overline{p}+\frac{2}{3}\overline{\rho}k\right) \tag{4.344}$$

另外一种说法是，对于非超音速流动，$-\frac{2}{3}\overline{\rho}k\mathbf{I}$ 可以忽略[38]。

方程(4.343)中的 $-(\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}-\frac{2}{3}\overline{\rho}k\mathbf{I})$ 可以继续通过 Bousinessq 近似附加湍流粘度进行模化，Boussinesq 认为其可以表示为形变率的无迹部分的线性关系，即：

$$-\left(\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}-\frac{2}{3}\overline{\rho}k\mathbf{I}\right)=\mu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right) \tag{4.345}$$

[38]参考 NASA 的介绍。

其中 $\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}$ 表示形变率的无迹部分。重整上述方程，会有另外一种形式：

$$-\overline{\rho}\overline{\mathbf{U}'\mathbf{U}'}=\boldsymbol{\tau}_t=\mu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right)-\frac{2}{3}\overline{\rho}k\mathbf{I} \tag{4.346}$$

结合形变率的定义：

$$\mathbf{S}=\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^{\mathrm{T}}}{2} \tag{4.347}$$

有

$$\boldsymbol{\tau}_t=2\mu_t\left(\mathbf{S}-\frac{1}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right)-\frac{2}{3}\overline{\rho}k\mathbf{I} \tag{4.348}$$

其另外一种写法即为：

$$\tau_{ij}=2\mu_t\left(S_{ij}-\frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)-\frac{2}{3}\overline{\rho}k\delta_{ij} \tag{4.349}$$

在很多湍流模型中都存在 $\sqrt{2S_{ij}S_{ij}}$ 这一项。在这里对其进行解释。对于二阶张量，有：

$$\sqrt{2S_{ij}S_{ij}}=\sqrt{2}\sqrt{\mathbf{S}:\mathbf{S}}=\sqrt{2}|\mathbf{S}|=\sqrt{2|\mathbf{S}|^2} \tag{4.350}$$

也有：

$$2S_{ij}S_{ij}=2|\mathbf{S}|^2 \tag{4.351}$$

事实上，这一项与湍流动能的产生项高度相关。在 OpenFOAM 大部分的湍流模型中，湍流动能方程中的 G 被定义为：

$$G=\mu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right):\nabla\mathbf{U} \tag{4.352}$$

在不可压缩的情况下，将其展开为[39]：

$$G=2\mu_t\left(\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2}\right):\left(\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2}+\frac{\nabla\mathbf{U}-\nabla\mathbf{U}^T}{2}\right)=\nu_t 2\mathbf{S}:\mathbf{S}=2\nu_t|\mathbf{S}|^2 \tag{4.353}$$

同时在不可压缩的情况下，G 有另外一种写法：

$$G=\nu_t S^2, S=(\sqrt{2S_{ij}S_{ij}})^2 \tag{4.354}$$

现在看一些文献里面常用的湍流产生项的表示方法：

$$P=\tau_{ij}\frac{\partial u_i}{\partial x_j} \tag{4.355}$$

将方程(4.349)带入到上述方程中有：

$$P=\left(2\mu_t\left(S_{ij}-\frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)-\frac{2}{3}\overline{\rho}k\delta_{ij}\right)\frac{\partial u_i}{\partial x_j}=$$
$$2\mu_t\left(S_{ij}-\frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)\frac{\partial u_i}{\partial x_j}-\frac{2}{3}\overline{\rho}k\delta_{ij}\frac{\partial u_i}{\partial x_j} \tag{4.356}$$

[39] $\left(\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2}\right):\left(\frac{\nabla\mathbf{U}-\nabla\mathbf{U}^T}{2}\right)=0$

其中

$$2\mu_t\left(S_{ij}-\frac{1}{3}\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)\frac{\partial u_i}{\partial x_j}=\mu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right):\nabla\mathbf{U}=G \tag{4.357}$$

$$-\frac{2}{3}\overline{\rho}k\delta_{ij}\frac{\partial u_i}{\partial x_j}=-\frac{2}{3}\overline{\rho}(\nabla\cdot\mathbf{U})k \tag{4.358}$$

因此有

$$P=G-\frac{2}{3}\overline{\rho}(\nabla\cdot\mathbf{U})k \tag{4.359}$$

上述方程中的 P 项大量的出现在湍流方程中。

4.11.7 变量有界性

某些传输的变量具备有界的特征，比如体积分数 α 严格的介于 0 和 1 之间，密度则严格的大于 0。在离散的过程中，高阶格式会产生一些越界的解，这会带来收敛问题。考虑波传输方程

$$\frac{\partial\alpha}{\partial t}+\mathbf{U}\cdot\nabla\alpha=0 \tag{4.360}$$

其中 $\mathbf{U}$ 表示传输速度。其解 α 为严格有界的。因此在求解传输方程的时候，方程(4.360)优先选择。但方程(4.360)为非守恒形式并且不存在通量函数[40]。因此在对方程(4.360)离散求解时候，其通常改写为

$$\frac{\partial\alpha}{\partial t}+\nabla\cdot(\mathbf{U}\alpha)-\alpha\nabla\cdot\mathbf{U}=0 \tag{4.361}$$

由于存在对速度的散度，因此解决了通量函数不存在的问题。同时，虽然连续性方程 $\nabla\cdot\mathbf{U}=0$ 意味着第三项可以忽略，即求解

$$\frac{\partial\alpha}{\partial t}+\nabla\cdot(\mathbf{U}\alpha)=0 \tag{4.362}$$

但在稳态算法的迭代步中，$\nabla\cdot\mathbf{U}\neq 0$。因此，只有求解方程(4.361)才能保证变量的有界。另外，瞬态算法可以保证每一个时间步都收敛（$\nabla\cdot\mathbf{U}-0$ 降低至压力残差以下），因此可以直接求解方程(4.362)。但对于需要有界的变量来讲，只有求解方程(4.361)，同时调用有界的离散格式（如 vanLeer 格式）才能保证变量的有界。

上述理论可以通过开源 CFD 软件 OpenFOAM 进行测试和验证，在 OpenFOAM 中的稳态算例中，对流项的离散格式需要添加`bounded`关键词，例如湍流动能对流项指定为[41]

```
div(phi,k)      bounded Gauss upwind;
```

这意味着求解方程(4.361)的形式。一些瞬态算例，某些变量同样指定为[42]

[40] 考虑在 i 网格点对 $\mathbf{U}\cdot\nabla\alpha$ 进行离散，在离散时，$\mathbf{U}$ 在 $i+\frac{1}{2},i-\frac{1}{2}$ 可以具有不同的值。同时 $i+\frac{1}{2},i-\frac{1}{2}$ 上的面积应该也具有不同的值。比如一个进口面积较大，出口面积较小的计算域，必然进口速度小，出口速度大。但若对方程(4.360)直接离散不能体现面积的影响。

[41] tutorials/incompressible/simpleFoam/motoBike

[42] tutorials/incompressible/pisoFoam/LES/pitzDaily

```
div(phi,s)       bounded Gauss limitedLinear 1;
```

这也意味着求解方程(4.361)的形式。某些变量（如速度）则指定为

```
div(phi,U)       Gauss LUST grad(U);
```

这意味着求解方程(4.362)的形式。其表示并没有对速度进行严格有界处理。对于湍流动能也通过求解方程(4.362)的形式来处理，其有界性通过bound函数来保证。

在实际测试中，相比方程(4.362)，方程(4.361)虽然有很大改善，但依然不能严格的保证变量的有界。在这种情况下，则需要一些更高级的处理方式来进行，如反扩散高阶格式。

4.11.8 守恒与非守恒、守恒变量与原始变量

间断解对于求解的控制方程与算法来说是一种挑战。在3.4.3节描述了守恒与非守恒方程的转化关系。在3.4.5节讨论了积分形式的控制方程。但为什么他们被称之为守恒形式和非守恒形式？从物理与数学形式上看他们有什么意义？方程为什么要区分为积分形式与微分形式？原始变量与守恒变量在求解上哪一个更优？

从数学角度来讲，如果考虑 x 方向动量的导数，网格如图4.32所示，字母表示网格单元编号，数字表示面编号。对于 B 网格点，对于动量 ρu 这个守恒变量进行有限体积法离散为：

$$\int_1^2 \frac{\partial \rho u}{\partial x}\mathrm{d}x = (\rho u)_2 - (\rho u)_1 \tag{4.363}$$

对 A，B，C 网格点进行离散并加和有：

$$\begin{aligned}\int_0^1 \frac{\partial \rho u}{\partial x}\mathrm{d}x + \int_1^2 \frac{\partial \rho u}{\partial x}\mathrm{d}x + \int_2^3 \frac{\partial \rho u}{\partial x}\mathrm{d}x &= (\rho u)_1 - (\rho u)_0 + (\rho u)_2 - (\rho u)_1 + (\rho u)_3 - (\rho u)_2 \\ &= (\rho u)_3 - (\rho u)_0 = \int_0^3 \frac{\partial \rho u}{\partial x}\mathrm{d}x \end{aligned} \tag{4.364}$$

也即不管如何加和，最后只有边界通量的影响，内部点互相约去，即通量永远是守恒的，进来多少，出去多少。相反，对于动量 ρu 这个守恒变量，将其拆分为原始变量 ρ, u 同时进

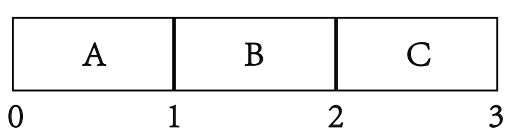


图 4.32: 一维有限体积网格。

行有限体积法离散为：

$$\int_1^2 \frac{\partial \rho u}{\partial x}\mathrm{d}x = \int_1^2 \left(u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x}\right)\mathrm{d}x = \tilde{u}(\rho_2 - \rho_1) + \tilde{\rho}(u_2 - u_1) \tag{4.365}$$

其中的面上的 $\tilde{u},\tilde{\rho}$ 需要计算而来，不同的格式对应不同的结果。如果定义为网格的平均值，同时对 A，B，C 网格点进行离散并加和有：

$$\int_0^1\left(u\frac{\partial\rho}{\partial x}+\rho\frac{\partial u}{\partial x}\right)\mathrm{d}x+\int_1^2\left(u\frac{\partial\rho}{\partial x}+\rho\frac{\partial u}{\partial x}\right)\mathrm{d}x+\int_2^3\left(u\frac{\partial\rho}{\partial x}+\rho\frac{\partial u}{\partial x}\right)\mathrm{d}x=$$
$$u_A(\rho_1-\rho_0)+\rho_A(u_1-u_0)+u_B(\rho_2-\rho_1)+\rho_B(u_2-u_1)+u_C(\rho_3-\rho_2)+\rho_C(u_3-u_2) \tag{4.366}$$

很明显，方程(4.366)的右边并不能够互相约去，也即为非守恒。

从物理意义来讲，在守恒法则下，守恒变量方程表示的是“真正”守恒的方程，原始变量方程表示的是“虚假”守恒的方程。例如考虑一维欧拉方程，守恒变量方程可以表示为：

$$\begin{aligned}\frac{\partial\rho}{\partial t}+\frac{\partial\rho u}{\partial x}&=0\\ \frac{\partial\rho u}{\partial t}+\frac{\partial\rho uu}{\partial x}&=0\end{aligned} \tag{4.367}$$

将动量方程展开：

$$\begin{aligned}\frac{\partial\rho}{\partial t}+u\frac{\partial\rho}{\partial x}+\rho\frac{\partial u}{\partial x}&=0\\ \frac{\partial\rho u}{\partial t}+\frac{\partial\rho uu}{\partial x}=u\frac{\partial\rho}{\partial t}+\rho\frac{\partial u}{\partial t}+u\frac{\partial\rho u}{\partial x}+\rho u\frac{\partial u}{\partial x}&=0\end{aligned} \tag{4.368}$$

结合连续型方程，其可以写为原始变量方程：

$$\begin{aligned}\frac{\partial\rho}{\partial t}+\frac{\partial\rho u}{\partial x}&=0\\ \frac{\partial u}{\partial t}+\frac{\partial\frac{1}{2}uu}{\partial x}&=0\end{aligned} \tag{4.369}$$

对比方程(4.367)与(4.369)，可以看出二者都表示守恒法则，不过前者表示质量与动量的守恒，后者表示质量与“速度”的守恒。由于速度并不是守恒的变量，因此方程(4.369)表示的是一种“虚假”守恒的状态。但数学过程是严谨的，那么方程(4.369)是否可以求解呢？答案是可以的，只要解是光滑的。若存在间断，“虚假”守恒的方程会预测不同的结果。

4.11.9 分离式求解器与耦合式求解器

在求解 CFD 变量的情况下，可以使用分离式求解器以及耦合式求解器。这是因为 CFD 方程组通常存在若干的变量，比如 p，$\mathbf{U}$。在使用分离式求解器的时候，针对每个变量进行次序求解。例如对于 p，$\mathbf{U}$，分离式求解器将求解 p 方程，u 方程，v 方程，w 方程。如果要求解 p 方程，那其中的除了 p 变量之外的所有变量都只能通过已知量来替换。因此分离式求解器可以看做是一种解耦。耦合式求解器则同时求解相关变量。例如在形成矩阵系统

时，p 方程，u 方程，v 方程，w 方程将组成一个大的矩阵。在进行求解的时候，p、u、v、w 一次性被求出。例如，下面的方程即为 4 网格点分离式求解器组建的压力方程以及 x 方向速度方程，其为两套矩阵系统：

$$\begin{bmatrix} \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \\ \star \end{bmatrix}, \begin{bmatrix} \star & \star & 0 & 0 \\ \star & \star & \star & 0 \\ 0 & \star & \star & \star \\ 0 & 0 & \star & \star \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \\ \star \end{bmatrix} \tag{4.370}$$

下面是一个耦合式求解器的范例：

$$\begin{bmatrix} \star & \star & 0 & 0 & \star & \star & 0 & 0 \\ \star & \star & \star & 0 & \star & \star & \star & 0 \\ 0 & \star & \star & \star & 0 & \star & \star & \star \\ 0 & 0 & \star & \star & 0 & 0 & \star & \star \\ \star & \star & 0 & 0 & \star & \star & 0 & 0 \\ \star & \star & \star & 0 & \star & \star & \star & 0 \\ 0 & \star & \star & \star & 0 & \star & \star & \star \\ 0 & 0 & \star & \star & 0 & 0 & \star & \star \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ \star \\ \star \\ \star \\ \star \\ \star \\ \star \end{bmatrix} \tag{4.371}$$

很明显，耦合式求解器一次性的求解出相关变量，并且为一个 block 矩阵，并且变量通过一个矩阵系统耦合起来，耦合性更强。

4.11.10 方程奇异

在很多情况下，CFD 求解的方程会出现奇异问题。考虑下述非常简单的方程：

$$\frac{\partial \alpha \mathbf{U}}{\partial t} - \nabla \cdot (\alpha D \nabla \mathbf{U}) = 0 \tag{4.372}$$

在一个四正交网格点上进行离散，边界条件均给定为固定值边界条件。则离散的矩阵系数为：

$$\begin{bmatrix} 3 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 3 & 0 \\ 0 & 0 & -1 & 3 \end{bmatrix} \tag{4.373}$$

其为一个可逆矩阵。如果网格第一个网格点的 $\alpha = 0$，且 $\frac{\partial \alpha}{\partial t} = 0$（$\alpha$ 不变），则矩阵变为：

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 3 & 0 \\ 0 & 0 & -1 & 3 \end{bmatrix} \tag{4.374}$$

其为一个奇异矩阵，因此在这种情况下方程不可解。CFD 里面类似的问题均需要通过数值的方法进行处理。例如，如果将方程(4.372)展开，且左右两边同时除以 α 有：

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\nabla \mathbf{U}) + \frac{\mathbf{U}}{\alpha}\frac{\partial \alpha}{\partial t} + \frac{1}{\alpha}\nabla \mathbf{U} \cdot \nabla \alpha = 0 \tag{4.375}$$

在 $\alpha = 0$ 的时候，依然存在这个问题。CFD 里面可以在分母上添加一个非常小的数来处理：

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\nabla \mathbf{U}) + \frac{\mathbf{U}}{\alpha + 1 \times 10^{-6}}\frac{\partial \alpha}{\partial t} + \frac{1}{\alpha + 1 \times 10^{-6}}\nabla \mathbf{U} \cdot \nabla \alpha = 0 \tag{4.376}$$

在这种情况下，方程(4.376)的左边第三项与第四项可以通过显性离散来计算，因此解决了方程奇异的问题。

4.11.11 显性离散、隐性离散

在离散 CFD 方程的时候，可以对变量进行显性离散以及隐性离散。二者的区别主要体现对离散后的矩阵系数的影响上。考虑速度 $\mathbf{U}$ 方程：

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) - \nabla \cdot (\nu \nabla \mathbf{U}) = -\nabla \frac{p}{\rho}, \tag{4.377}$$

其中的每一项在离散的过程中，都可以区分为显性离散以及隐性离散。其中的时间项，隐性离散可以写为：

$$\frac{\partial \mathbf{U}}{\partial t} = \frac{\mathbf{U}^{t+\Delta t} - \mathbf{U}^{t}}{\Delta t} \tag{4.378}$$

其中的 $\mathbf{U}^{t+\Delta t}$ 为未知变量，进入到矩阵系统成为待求变量。$1/\Delta t$ 成为了矩阵系统的对角线系数。$\mathbf{U}^{t}/\Delta t$ 作为已知量，进入到矩阵系统的源项部分。如果时间项显性离散，则有：

$$\frac{\partial \mathbf{U}}{\partial t} = \frac{\mathbf{U}^{t} - \mathbf{U}^{t-\Delta t}}{\Delta t} \tag{4.379}$$

方程(4.379)的右侧全部进入矩阵源项部分。不对矩阵系数产生影响。

4.11.12 Jacobian 矩阵

给定自变量 $\mathbf{x} = [x_1, x_2, ..., x_n]$ 以及待求函数 $\mathbf{y} = [y_1, y_2, ..., y_n]$，假定函数关系为：

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{4.380}$$

也即：

$$\begin{aligned} y_1 &= f_1(x_1, x_2, ..., x_n), \\ y_2 &= f_2(x_1, x_2, ..., x_n), \\ &..., \\ y_n &= f_3(x_1, x_2, ..., x_n), \end{aligned} \tag{4.381}$$

对于上述的方程组，可以从数学上定义其 Jacobian 矩阵为：

$$\boldsymbol{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{4.382}$$

Jacobian 在流体力学中的应用主要涉及到体积微元以及坐标转换。考虑原始坐标系 (α, β, γ)，在将体积 V_0 投影至新坐标系 (x, y, z) 之后，其体积将会发生变化。Jacobian 矩阵被定义为：

$$\boldsymbol{J} = \frac{\partial(x, y, z)}{\partial(\alpha, \beta, \gamma)} = \begin{pmatrix} \frac{\partial x}{\partial \alpha} & \frac{\partial x}{\partial \beta} & \frac{\partial x}{\partial \gamma} \\ \frac{\partial y}{\partial \alpha} & \frac{\partial y}{\partial \beta} & \frac{\partial y}{\partial \gamma} \\ \frac{\partial z}{\partial \alpha} & \frac{\partial z}{\partial \beta} & \frac{\partial z}{\partial \gamma} \end{pmatrix} \tag{4.383}$$

Jacobian 被定义为 Jacobian 矩阵的行列式：

$$J = |\boldsymbol{J}| \tag{4.384}$$

这样，体积变化可以表示为：

$$\mathrm{d}V = \frac{\partial(x, y, z)}{\partial(\alpha, \beta, \gamma)} \mathrm{d}\alpha \mathrm{d}\beta \mathrm{d}\gamma = J \mathrm{d}V_0 \tag{4.385}$$

同时参考方程(3.36)以及速度散度的定义，有：

$$\nabla \cdot \mathbf{U} = \frac{1}{J} \frac{\mathrm{D}J}{\mathrm{D}t} \tag{4.386}$$

可参考其他资料获取更详细的讨论 [3].

4.11.13 非牛顿流体

上文的讨论均假定流体为牛顿流体，即流体的粘度为常数（剪切应力和形变率的比值为定值），剪切应力越大，形变的速度呈现同比例的变化。也即方程(3.69)中的 μ 为定值。在非牛顿流体中，μ 表示为一个跟时空有关的变量。非牛顿流体在日常生活中非常常见，如果酱、洗洁精、血液、泥浆、蛋清、等。举例，淀粉溶液在轻轻搅动的时候较易，快速搅动的时候较难。这是因为淀粉溶液的粘度和剪切应力呈现非线性的关系。剪切应力越大，粘度越高，形变率越慢，越难以搅动。非牛顿流体可以按照以下标准进行区分，剪切变稀流体的粘度随着剪切应力增加而降低，剪切增稠流体的粘度随着剪切应力增加而增加，宾汉流体在剪切力小于一定值的情况下，呈现固体的特性，粘弹性流体的粘度具备粘性，同时具备固体类似的弹性。图4.33为开源 CFD 软件 OpenFOAM 模拟的粘弹性两相流流动。

在 CFD 中，牛顿流体的动量方程中的剪切应力 τ 可以通过方程(3.69)转变为 $\mathbf{U}$ 项进而封闭。但由于粘度为一个随时空变化的量，因此动量方程中会存在一个附加项。现对此

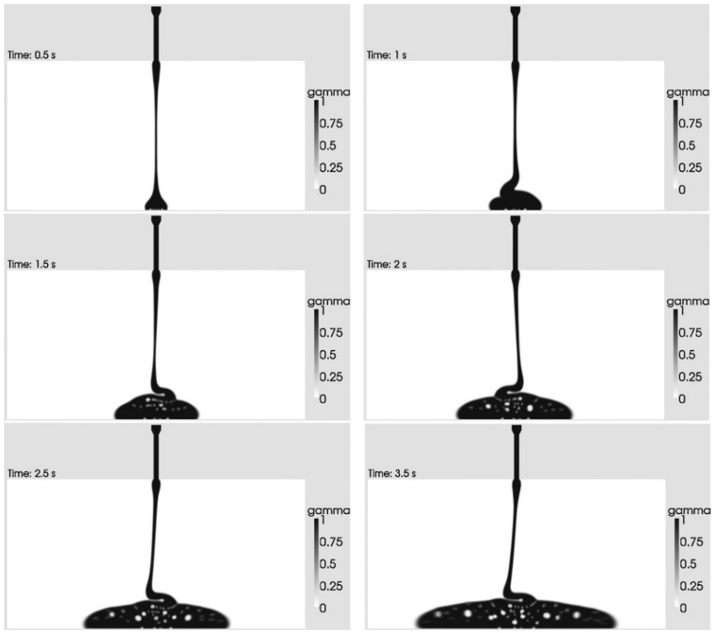


图 4.33: 通过开源 CFD 软件 OpenFOAM 模拟的粘弹性流体两相流 [49]。

进行推导。考虑粘性项 $\nabla\cdot(\mu(\nabla\mathbf{U}+\nabla\mathbf{U}^T))$，有：

$$\nabla\cdot(\mu(\nabla\mathbf{U}+\nabla\mathbf{U}^T))=\nabla\cdot(\mu\nabla\mathbf{U})+\nabla\cdot(\mu\nabla\mathbf{U}^T) \tag{4.387}$$

现重点关注 $\nabla\cdot(\mu\nabla\mathbf{U}^T)$，将其展开（考虑速度仅有 2 个分量）：

$$\begin{aligned}\nabla\cdot(\mu\nabla\mathbf{U}^T)=\nabla\cdot\begin{bmatrix}\mu\frac{\partial u_1}{\partial x} & \mu\frac{\partial u_1}{\partial y}\\ \mu\frac{\partial u_2}{\partial x} & \mu\frac{\partial u_2}{\partial y}\end{bmatrix}&=\begin{bmatrix}\frac{\partial}{\partial x}\left(\mu\frac{\partial u_1}{\partial x}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u_2}{\partial x}\right)\\ \frac{\partial}{\partial x}\left(\mu\frac{\partial u_1}{\partial y}\right)+\frac{\partial}{\partial y}\left(\mu\frac{\partial u_2}{\partial y}\right)\end{bmatrix}\\ &=\begin{bmatrix}\frac{\partial\mu}{\partial x}\frac{\partial u_1}{\partial x}+\mu\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial x}\right)+\frac{\partial\mu}{\partial y}\frac{\partial u_2}{\partial x}+\mu\frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial x}\right)\\ \frac{\partial\mu}{\partial x}\frac{\partial u_1}{\partial y}+\mu\frac{\partial}{\partial x}\left(\frac{\partial u_1}{\partial y}\right)+\frac{\partial\mu}{\partial y}\frac{\partial u_2}{\partial y}+\mu\frac{\partial}{\partial y}\left(\frac{\partial u_2}{\partial y}\right)\end{bmatrix}=\begin{bmatrix}\frac{\partial\mu}{\partial x}\frac{\partial u_1}{\partial x}+\frac{\partial\mu}{\partial y}\frac{\partial u_2}{\partial x}\\ \frac{\partial\mu}{\partial x}\frac{\partial u_1}{\partial y}+\frac{\partial\mu}{\partial y}\frac{\partial u_2}{\partial y}\end{bmatrix}\\ &=\begin{bmatrix}\frac{\partial u_1}{\partial x},\frac{\partial u_2}{\partial x}\\ \frac{\partial u_1}{\partial y},\frac{\partial u_2}{\partial y}\end{bmatrix}\cdot\begin{bmatrix}\frac{\partial\mu}{\partial x}\\ \frac{\partial\mu}{\partial y}\end{bmatrix}=\nabla\mathbf{U}\cdot\nabla\mu\end{aligned} \tag{4.388}$$

因此，非牛顿流体动量方程中的粘性项的最终形式为：

$$\nabla\cdot(\mu(\nabla\mathbf{U}+\nabla\mathbf{U}^T))=\nabla\cdot(\mu\nabla\mathbf{U})+\nabla\mathbf{U}\cdot\nabla\mu \tag{4.389}$$

其中 $\nabla\mathbf{U}\cdot\nabla\mu$ 在代码植入的过程中要体现出来。另一方面，需要植入与速度相关的粘度变化，也即本构方程。在一些较为简单的非牛顿流体中（如宾汉流体），可以通过将 μ 表示为关于时空的函数进行计算植入。在一些复杂的非牛顿流体中（如粘弹性流体的 Giesekus 模型），则需要植入 τ 的传输方程，然后对其求解散度项中代入到动量方程中再进行显性离散计算。

对于各类不可压缩流体，有剪切应力与形变率的关系：

$$\boldsymbol{\tau} = \mu \mathbf{S} \tag{4.390}$$

若考虑牛顿流体，则 μ 为一个常数。若考虑非牛顿流体，μ 为一个变量且通常将其称之为表观粘度。各种非牛顿流体模型即建立表观粘度与速度等变量的相关关系。目前在 OpenFOAM 中植入的经典的非牛顿流体粘度计算模型主要有 powerLaw、HerschelBulkley、CrossPowerLaw、Casson、BirdCarreau 模型。在计算表观粘度之前，定义形变率标量 γ：

$$\gamma = \sqrt{2\mathbf{S} : \mathbf{S}} \tag{4.391}$$

其中 $\mathbf{S}$ 在方程(3.67)中已定义。Bird-Carreau 模型定义表观粘度为：

$$\nu = \nu_\infty + (\nu_0 - \nu_\infty)\left[1 + (k\gamma)^a\right]^{(n-1)/a} \tag{4.392}$$

其中系数 a 默认值为 2。其他的参数需要用户自行给定。CrossPowerLaw 模型定义表观粘度为：

$$\nu = \nu_\infty + \frac{\nu_0 - \nu_\infty}{1 + (m\gamma)^n} \tag{4.393}$$

其中所有参数需要用户自行给定。Herschel-Bulkley 模型的表观粘度为：

$$\nu = \min\left(\nu_0, \frac{\tau_0}{\gamma} + k\gamma^{n-1}\right) \tag{4.394}$$

4.11.14 Stokes 流动

Stokes 流动也被称之为蠕流。在 Stokes 流动中，雷诺数非常小，通常忽略惯性项（对流项）。Stokes 流动的流速非常小，一般在一些特殊的工况中会发现，如地下水渗透、精子的流动行为等。无体积力的稳态的 Stokes 方程可以用于描述 Stokes 流动：

$$0 = -\nabla p + \nabla \cdot \boldsymbol{\tau} \tag{4.395}$$

4.11.15 算子分裂法

CFD 算法中，经常会讨论到算子分裂法。算子分裂的英文解释为 Operator Splitting。对于 CFD 中的方程，有些情况下，按项去求解会比较简单。考虑下面的无量纲方程：

$$\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} = \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) \tag{4.396}$$

在求解过程中，可以按项来处理，先求

$$\frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x} = 0 \tag{4.397}$$

在获得解 ϕ^* 后，再求

$$\frac{\partial \phi}{\partial t} = \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) \tag{4.398}$$

获得解 ϕ^{**}。这样求解有一个非常特殊的好处。那就是可以在不同的阶段调用不同的格式。例如在方程(4.397)中，其为一个纯对流方程，可以调用显性迎风高阶格式。在方程(4.398)中，其为一个含有扩散项的时间推进方程，可调用较大时间步的隐性格式。

4.11.16 总温、驻点温度

对于流动的流体，可以定义几种不同的温度。例如，静温用来衡量无流动的流体的温度。静温的大小主要取决于分子动力学，其取决于分子的热运动。对于流动的流体，可以定义动温，其与流动的动能有关。同时，也可以定义总温，即静温与动温之和。流体的总温取决于马赫数。如果在计算域内突然出现静止点（驻点，局部速度为 0 或者近似于 0 的区域）。驻点处的温度也被称之为驻点温度，其与总温几乎相同。这是因为在驻点处，可以使流体以绝热过程完全静止时，那么它的动能将全部转化为内能，这时，动温为 0，总温即驻点温度。流体的总温与静温的关系为：

$$T_0 = \left(1 + \frac{\gamma - 1}{2} Ma^2\right) T \tag{4.399}$$

其中 T_0 表示总温，T 表示静温，γ 表示热容比，Ma 表示马赫数。总温在经过激波区域，其值不变。

4.11.17 Sutherland 法则

对于纯的无反应气体，其粘度仅仅与温度有关。Sutherland 法则用于描述粘度与温度的关系。在 OpenFOAM 中，Sutherland 法则可以表示为：

$$\mu = \frac{A_s T^{3/2}}{T + T_s} \tag{4.400}$$

对于空气来说，$A_s = 1.458 \times 10^{-6}$，$T_s = 110.4$。在超音速以及一些壁面传热流动中，温度范围可能会很大并进一步的影响空气的粘度。在这些情况下，可以使用 Sutherland 法则来计算粘度。

4.11.18 波的传输与边界条件

同样的 PDE，对于超音速以及亚音速的会具有不同的特征。如图4.34所示，对于亚音速流动，其中的 $u - c < 0$，存在向左侧传输的变量，也即左侧为流动的下游。同时其他四个变量均为向右侧传输，则右侧为下游。因此左右两侧均有可能为上下游。对于超音速流动，由于 $u - c > 0$，因此仅仅右侧为下游。因此对于计算域内的扰动，对于超音速工

况，仅仅向右侧传输。如果将控制方程分解为波方程，则每个变量的 inlet 都可以看作为进口，可以给定固定值边界条件，outlet 出口给定零法向梯度。也即速度、温度、压力在进口处均为固定值边界条件，出口给定零法向梯度。这样的边界条件与数学特征是一致的。对

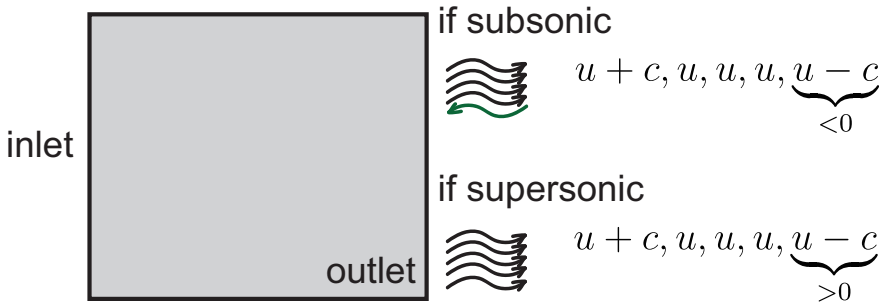


图 4.34: 常见进出口计算域针对亚音速以及超音速导致不同的特征。

于亚音速流动情况略有变化。左右两侧均为上下游，因此如果将 PDE 写成波方程的情况，必然有一个变量在 outlet 处需要为固定值边界条件。只有这样，其才能按照 $u-c$ 的速度向左传输。压力方程，可以理解为声波的传输。在亚音速的情况下，压力会向不同的方向传输，且很有可能是流场 **U** 的上游。因此对于亚音速算例，压力在出口可以给定固定值边界条件。压力进口可以给定零梯度边界条件。这也可以被理解为声波是可以向上游传递的。其他速度以及温度在出口给定零梯度边界条件。

但有些情况下，可能进口是亚音速，出口是超音速。这需要压力在进口是零梯度。压力在出口同样也是零梯度。这回产生压力相容性问题（参考第6.268节）。这个问题最好的处理办法是更改计算域，使得出口扩大为亚音速，进而出口压力可以为固定值边界条件 [61]。

4.11.19 计算气动声学

计算气动声学（Computational Aero Acoustic，CAA）与 CFD 存在交叉。目前通过计算机的方法来计算气动噪声主要分为两种方法。第一种方法为直接法。第二种方法为混合法。直接法对可压缩 NS 方程进行直接求解。但因为流体与声波的特征尺度差别非常大，为了满足气动噪声的精确模拟，需要调用大量的网格。因此目前直接法不能用于工程计算。在混合法中，采用两步来进行计算。第一步首先求解流场的流动，第二步来求解波方程。声音的传输对流场不存在影响。在混合法中，Lighthill 方法被认为是最早可用于实际应用的计算方法 [103]。Lighthill 方法将流场中的变量分解为波动场与场常量：

$$\rho' = \rho - \rho_0 \tag{4.401}$$

其中 ρ_0 表示常密度，ρ' 表示密度波动。其他的变量如压力、速度等也可以以此类推。经过分解后，Lighthill 方法中的波方程可以表示为 [103]：

$$\frac{\partial^2 \rho'}{\partial t^2} - c^2 \nabla \cdot (\nabla \rho') = c^2 \frac{\partial^2 T_{ij}}{\partial x_i \partial x_j} \tag{4.402}$$

上述方程中的 T_{ij} 被称之为 Lighthill 张量。c 表示音速。在忽略粘性效应以及 Lighthill 张量原始形式的可压缩部分后其可以封闭为：

$$T_{ij} \approx \rho_0 \mathbf{U}_i \mathbf{U}_j \tag{4.403}$$

随后有

$$\frac{\partial^2 \rho'}{\partial t^2} - c^2 \nabla \cdot (\nabla \rho') = c^2 \rho_0 \nabla \cdot (\nabla \cdot (\mathbf{U}\mathbf{U})) \tag{4.404}$$

依据压力与速度的关系 $p' = c^2 \rho'$，有

$$\frac{1}{c^2}\frac{\partial^2 p'}{\partial t^2} - \nabla \cdot (\nabla p') = c^2 \rho_0 \nabla \cdot (\nabla \cdot (\mathbf{U}\mathbf{U})) \tag{4.405}$$

方程(4.405)右侧的源项可以被认为是噪音发生的源项。由于方程(4.405)存在了一些假定（如忽略可压缩项），因此在马赫数比较小的情况下是比较适用的。其误差随着马赫数的增加也越来越明显。

在可压缩行比较强的情况下，例如对于高马赫数流动，可以采用 NS 方程线性化的方法来进行计算。若完全的忽略粘性，NS 方程可以简化为线性欧拉方程（Linearized Euler Equations，LEE）。在推导的过程中，主要分为两个步骤：第一步是消除粘性项。第二步是消除非线性项（将变量进行分解）。消除非线性项的过程可以参考 Lighthill 方法。例如对于连续性方程，其可以写为：

$$\frac{\partial \rho_0 + \rho'}{\partial t} + \nabla \cdot ((\rho_0 + \rho')(\mathbf{U}_0 + \mathbf{U}')) = 0 \tag{4.406}$$

其可以分解为

$$\frac{\partial \rho'}{\partial t} + \frac{\partial \rho_0}{\partial t} + \nabla \cdot (\rho_0 \mathbf{U}_0) + \nabla \cdot (\rho_0 \mathbf{U}') + \nabla \cdot (\rho' \mathbf{U}_0) + \nabla \cdot (\rho' \mathbf{U}') = 0 \tag{4.407}$$

其中 $\nabla \cdot (\rho' \mathbf{U}')$ 可以忽略。$\frac{\partial \rho_0}{\partial t} + \nabla \cdot (\rho_0 \mathbf{U}_0) = 0$，因此其可以继续写为：

$$\frac{\partial \rho'}{\partial t} + \nabla \cdot (\rho' \mathbf{U}_0) + \nabla \cdot (\rho_0 \mathbf{U}') = 0 \tag{4.408}$$

方程(4.408)即为线性欧拉方程的波动密度方程。类似的，有线性欧拉方程的波动速度方程：

$$\frac{\partial \mathbf{U}'}{\partial t} + \nabla \cdot (\mathbf{U}_0 \mathbf{U}') + \frac{\rho'}{\rho_0}\mathbf{U}_0 \cdot \nabla \mathbf{U}_0 = -\frac{1}{\rho_0}\nabla p' \tag{4.409}$$

以及能量方程：

$$\frac{\partial p'}{\partial t} + \mathbf{U}' \cdot \nabla p_0 + p' \gamma \nabla \cdot \mathbf{U}_0 + \mathbf{U}_0 \cdot \nabla p' + \gamma p_0 \nabla \cdot \mathbf{U}' = 0 \tag{4.410}$$

4.11.20 Partial Elimination 算法

在多相流以及涉及到多域耦合求解问题的情况下，不同的场存在耦合。例如对于 ϕ_1 与 ϕ_2 场，假定离散后的方程为：

$$\begin{aligned} a_1^P \phi_1^P &= \sum a_1^N \phi_1^N + K(\phi_2^P - \phi_1^P) \\ a_2^P \phi_2^P &= \sum a_2^N \phi_2^N + K(\phi_1^P - \phi_2^P) \end{aligned} \tag{4.411}$$

其中 $K(\phi_1^P-\phi_2^P)$ 以及 $K(\phi_2^P-\phi_1^P)$ 耦合系数。方程(4.411)可以继续化简为

$$\begin{aligned}(a_1^P+K)\phi_1^P&=\sum a_1^N\phi_1^N+K\phi_2^P\\(a_2^P+K)\phi_2^P&=\sum a_2^N\phi_2^N+K\phi_1^P\end{aligned}\tag{4.412}$$

方程(4.412)在求解的过程中可能会产生问题。在求解过程中，方程(4.412)采用顺序解耦求解的方法来推进。首先求解 ϕ_1，然后使用 ϕ_1 的值来更新 ϕ_2。反复进行进而将两场耦合起来直至收敛。当系数 K 比较小的情况下，可以看出二者的耦合效应非常小，这不会对求解产生若干问题。但在系数 K 比较大的情况下，方程(4.412)会趋向于：

$$\begin{aligned}K\phi_1^P&\approx K\phi_2^P\\K\phi_2^P&\approx K\phi_1^P\end{aligned}\tag{4.413}$$

方程(4.413)完全取决于 ϕ_2 的初值并且 ϕ_1 ϕ_2 会直接停止在 ϕ_2 的初值不会继续变化。Partial Elimination 算法将方程改写为：

$$a_1^P\phi_1^P=\sum a_1^N\phi_1^N-K\left(\phi_1^P-\frac{\sum a_2^N\phi_2^N+K\phi_1^P}{a_2^P+K}\right)\tag{4.414}$$

重整后有：

$$\left(a_1^P+\frac{a_2^PK}{a_2^P+K}\right)\phi_1^P=\sum a_1^N\phi_1^N+\frac{K}{a_2^P+K}\sum a_2^N\phi_2^N\tag{4.415}$$

类似的，对于 ϕ_2 有：

$$\left(a_2^P+\frac{a_1^PK}{a_1^P+K}\right)\phi_2^P=\sum a_2^N\phi_2^N+\frac{K}{a_1^P+K}\sum a_1^N\phi_1^N\tag{4.416}$$

方程(4.415)与(4.416)可以用来对 ϕ_1,ϕ_2 进行更新。在耦合系数 K 趋向于小值的情况下，方程(4.415)与(4.416)回归与原始方程(4.412)。当耦合系数 K 比较大的时候，方程(4.415)与(4.416)趋向于

$$\begin{aligned}\left(a_1^P+a_2^P\right)\phi_1^P&=\sum a_1^N\phi_1^N+\sum a_2^N\phi_2^N\\\left(a_2^P+a_1^P\right)\phi_2^P&=\sum a_2^N\phi_2^N+\sum a_1^N\phi_1^N\end{aligned}\tag{4.417}$$

方程(4.417)表示 ϕ_1,ϕ_2 的值趋向于相同，也即耦合系数 K 比较大的情况。且不存在收敛停止的问题。

4.11.21 阻力、切向力、法向力、阻力系数

当某个物体在流体中运动的时候，其会遭受流体施加给运动物体的力。这个力可以区分为切向力以及法向力。贡献主要来自于压力以及壁面剪切力。壁面剪切力为一种切向力，其与流体的粘度有关。压力为一种法向力。切向力与法向力之和为物体的总受力。

针对一个物体，其表面被划分为不同的网格面，其压力产生的力的计算公式为：

$$\mathbf{F}_{pressure} = \sum p_f \mathbf{S}_f \tag{4.418}$$

其剪切应力产生的力的计算公式为：

$$\mathbf{F}_{shear} = \sum \mathbf{S}_f \cdot \boldsymbol{\tau}_f = \sum \mathbf{S}_f \cdot \left(-\nu_{eff} \left(\nabla \mathbf{U} + \nabla \mathbf{U}^T - \frac{2}{3}(\nabla \cdot \mathbf{U})\mathbf{I} \right) \right)_f \tag{4.419}$$

注意其中 $\left(-\nu_{eff}\left(\nabla \mathbf{U} + \nabla \mathbf{U}^T - \frac{2}{3}(\nabla \cdot \mathbf{U})\mathbf{I}\right)\right)_f$ 取边界面上定义的值。在有了总压力以及总剪切应力之后，可以计算总受力：

$$\mathbf{F}_{total} = \mathbf{F}_{pressure} + \mathbf{F}_{shear} \tag{4.420}$$

获得总受力后，可以计算物体收到的阻力。在计算之前，需要给定施加阻力的方向 $\mathbf{dir}_{drag}$，则有阻力为：

$$F_{drag} = \mathbf{F}_{total} \cdot \mathbf{dir}_{drag} \tag{4.421}$$

类似的，升力的计算公式为

$$F_{lift} = \mathbf{F}_{total} \cdot \mathbf{dir}_{lift} \tag{4.422}$$

在阻力计算后，可以计算阻力系数以及升力系数：

$$Cd = \frac{F_{drag}}{(\sum |\mathbf{S}_f|)\frac{1}{2}\rho|\mathbf{U}|^2} \tag{4.423}$$

$$Cl = \frac{F_{lift}}{(\sum |\mathbf{S}_f|)\frac{1}{2}\rho|\mathbf{U}|^2} \tag{4.424}$$

4.11.22 三维下的库朗数计算

OpenFOAM 中的库朗数主要依据可压缩以及不可压缩有不同的计算方式。进一步的，某一些特殊的求解器可能还存在特殊的库朗数计算方法，但大体思想一致。在不可压缩流中，OpenFOAM 中通过下式定义库朗数[43]：

$$Co = \frac{1}{2}\frac{\sum |\phi_f|}{\Delta V}\Delta t \tag{4.425}$$

其中 ϕ_f 为体积通量。其中要除以 2 是因为对于一个网格单元，流入的通量与流出的通量必然相等，在做 $\sum |\phi_f|$ 操作的时候，其流量为流入的通量或流出的通量的二倍。对于可压缩流动，ϕ_f 为质量通量，因此库朗数的定义略有区别：

$$Co = \frac{1}{2}\frac{\sum |\phi_f|}{\rho \Delta V}\Delta t \tag{4.426}$$

[43]每个网格单元都存在一个库朗数。

其中 ρ 表示当前网格单元的密度。在双流体模型中，还存在相对库朗数的概念，其定义为：

$$Co = \frac{1}{2}\frac{\sum|\phi_1 - \phi_2|}{\Delta V}\Delta t \tag{4.427}$$

其中 ϕ_1、ϕ_2 表示多相流中不同的相通量。在 VOF 模型中，还存在界面库朗数的概念，其定义为：

$$Co = \frac{1}{2}\frac{\sum|\phi|}{\Delta V}\Delta t, 0.01 < \alpha < 0.99 \tag{4.428}$$

可见，界面库朗数仅仅在 $0.01 < \alpha < 0.99$（也即界面处）存在值。其可以用于判定界面处速度的大小。

第五章　数据驱动 CFD

什么是大模型？机器自己能进行学习？神经网络与人脑有无联系？为什么所有机器学习的论文都用图5.1这种的示意图？训练、标签、损失、奖励、神经元、深度学习、激活函数等各类术语都是什么意思[1]？本节从 CFD 从业者的角度，浅谈机器学习以及 CFD 与机器学习潜在的交叉应用。机器学习本身是一个研究方向，有很多专门指向机器学习的巨著。非常具有特色的是，本书中讨论的内容紧密贴合 CFD 相关应用，重点讨论数据驱动 CFD，而非普适性机器学习算法与理论。

在一些其他领域，机器学习要相对好理解。比如常用的语音识别，用户在微信输入一段语音，微信可以把他转换成文字。这可以理解为一个函数 $f(x)$，其中的 x 是用户输入的语音，$f(x)$ 输出的是文字。这个函数会非常非常的复杂。那么如何去找到这个函数。计算机可以通过一些特定的算法，把这个函数找出来。在计算机在寻找这个函数的过程，就可以理解为**机器学习**。这个函数 $f(x)$ 在机器学习领域被称之为**模型**（在机器学习领域经常会听说大家讨论大模型）。这里需要人去指挥计算机去找函数，计算机当然不能自己开机自己去找。人去设定一段程序，告诉计算机说："去运行代码，去找这个特定函数吧！"这个过程就叫**训练**。**训练模型**，就是去找函数的这个过程。

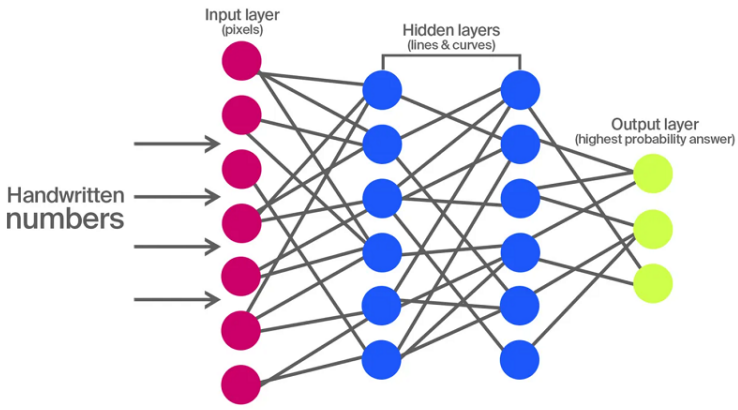


图 5.1: 神经网络示意图

[1] 机器学习领域有很多新的名词，附录中这些新名词都通过黑体来标识。

5.1 监督学习

监督学习可以看做是最简单的可以理解的机器学习的一种方法。在纯的机器学习领域，监督学习还会更加的细分为分类问题等。然而其与 CFD 的联系较小。本节主要讨论与 CFD 应用比较大的方法。在这个方法中（机器学习领域称之为线性回归），**监督学习**的过程与数据拟合的过程非常类似。如果去参考传统高等数学的教材，比如文献 [207]，其编写在 21 世纪初，那个时候机器学习并没有跟 PDE 求解有太大的关系。文献 [207] 中详细的讨论了函数拟合方法以及函数逼近方法。这两个方法都是用户给定样本点，去通过纯数学的方法去寻找最优的函数去与样本点尽可能的去贴合[2]。计算机监督学习的好与坏，重要的是评估这个函数的新数据预测能力。数据拟合更倾向于寻找一个函数，来使拟合函数对现有的数据具有最小的差异。带入新数据，数据拟合方法找出来的函数，对于新数据预测的好坏。这不属于数据拟合函数的评估内容。它可能预测的较好，也可能预测的不好。但是不管好与坏，只要数据拟合函数对于现存样本点贴合较好，数据拟合函数就完成了他的工作。同时，监督学习找函数的过程（也即训练模型的过程），与数据拟合找函数的过程，具有本质的差异。数据拟合找函数的过程，可以参考传统高等数学相关教材，如 [207]。本节介绍监督学习背后的数学算法。

现在举一个简单的例子，化工领域的鼓泡反应器通过在底部注入空气，将泡泡持续的注入水中。实现水与气泡的混合。在这种情况下，存在气含率的概念。即空气的体积除以水汽的总体积。不同的进气速度，对应不同的气含率。可想而知的是，注入气体的速度越大，气含率越大。假定通过 CFD 的方法模拟了 5 种气速与气含率的对应状态，如表5.1。这种实验测量出来的原始数据气含率，在机器学习领域被称之为**标签**。在这里区别一下特征与标签的区别。训练数据指的整套的这 5 组特征与标签，其存在 5 个（气速）**特征**，每个特征对应一个气含率（标签）。在图像识别领域标签看起来更形象。比如现存一个图像识别的模型，用户输入一个图片，这个图片就是特征，这些“狗”“猫”就是标签。在这里我们拍脑袋说，气含率 $f(x)$ 与气速 x 的函数，就是这样的关系：

$$f(x) = ax + b \tag{5.1}$$

但是行内的专家可能会说，他们拍脑袋想这个函数可能要这样写：

$$f(x) = ax^2 + b \tag{5.2}$$

可以看出不同的人对不同的这个潜在的拟合函数有不同的理解。这个在机器学习领域叫做**领域知识**。领域知识很重要，如果写出不符合领域知识的函数，将会是完全不可理解的。我们假定使用最简单的方程(5.1)。方程(5.1)在机器学习领域叫做**模型**。我们经常听说的**训练模型**，就是针对方程(5.1)去寻找其中的未知参数 a 和 b 的过程。在机器学习领域，a 一

[2] 如最小二乘法拟合、傅里叶多项式拟合等。

	气体注入速度 m/s	测量气含率（标签）
第一组	1	0.02
第二组	2	0.02
第三组	3	0.04
第四组	4	0.04
第五组	5	0.06

般写成 w，b 凑巧就是 b。其中 w 是 weights 的简写，被称之为**权重**。b 是 bias 的简写，在机器学习领域被称之为**偏置**。现在简单的把方程(5.1)改写为：

$$f(x) = wx + b \tag{5.3}$$

在这里还不知道如何训练模型去找未知参数，同时假定也不会通过传统高等数学的方法去处理，最简单的就是拍脑袋说，方程(5.3)中的 $w = 0.01$，$b = 0.01$，在这种情况下，预测的气含率有：很明显预测的气含率与测量的气含率存在误差：这个误差在 CFD 领域就是误

	气体注入速度 m/s	预测气含率
第一组	1	0.02
第二组	2	0.03
第三组	3	0.04
第四组	4	0.05
第五组	5	0.06

	气体注入速度 m/s	误差
第一组	1	0
第二组	2	0.01
第三组	3	0
第四组	4	0.01
第五组	5	0

差，在机器学习领域被称作为**损失**。很明显，损失跟某个气速有关。比如第一次数据，损失为 0。因此损失是一个跟标签相关的函数，因为标签跟自变量有关系，因此损失也是个跟自变量有关的函数。在机器学习领域，被称作为**损失函数**。目前存在一些主要的损失函数，比如均方误差（Mean Squared Error，MSE）：

$$MSE = \frac{1}{n}\sum(f(x_i) - y_i)^2 \tag{5.4}$$

很明显，损失 MSE 为一个与权重 w 以及偏置 b 相关的函数。MSE 的自变量有 2 个。参考图5.2，不同的权重 w 以及偏置 b 分别对应一个 MSE，形成一个二维的误差表面。其中的最低点对应的权重 w 以及偏置 b，就是最优的值。

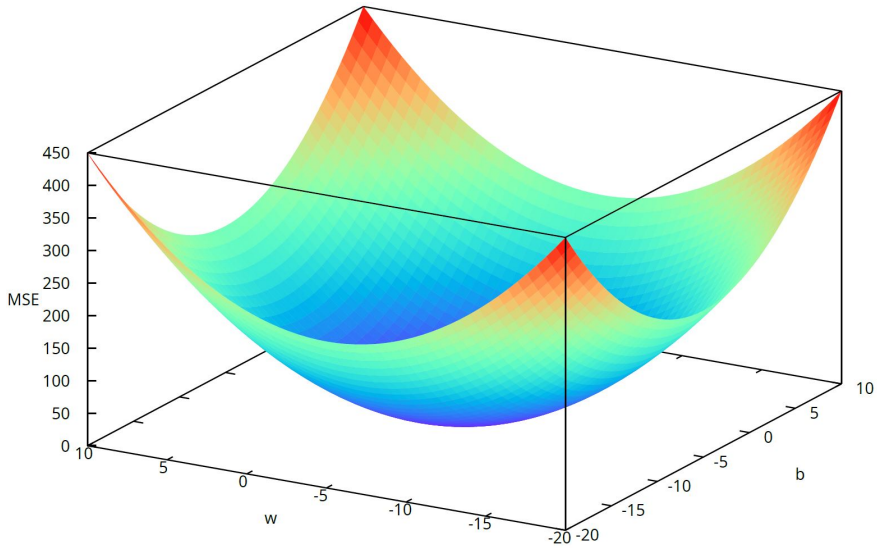


图 5.2: 误差表面示意图。

那么如何去寻找这个最小的点。在机器学习领域存在不同的方法，比如梯度下降法、随机优化方法等。梯度下降法看起来是比较好理解的方法。在高等数学领域，梯度下降法有特定的公式。在机器学习领域，梯度下降法有很大的区别。共同点都是要寻找一个最小的损失。在实操过程中，这部分内容一般被封装起来。就类似有限体积法，用户需要写一个求解器，并不需要知道矩阵迭代求解器是怎么迭代的（OpenFOAM 已经封装好了），用户只需要在顶层定义要离散一个什么样的方程就可以了。在机器学习领域，例如在 PINN 中，用户只需要定义要求解的方程，其中梯度下降算法在 libtorch 中已经封装好了不需要改动。

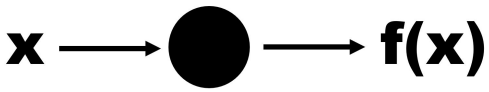


图 5.3: $f(x) = wx + b$ 示意图。其中的圆球就可以理解为是这个线性函数。注意，输入到模型的是可以是一个 x，也可以是多个 x，也即为一个批量（后续会介绍批量的概念）。如果输入多个 x，损失计算也是这一系列 x 对应的均方误差，来使得其最小化。

现在假定，经过这个梯度下降的黑盒子之后，机器经过算法（例如梯度下降法）寻找出了最优的 w 以及 b，那么输入任意一个 x 的值，就可以计算出 $f(x)$ 的值。图5.3中表示的是 $f(x) = wx + b$ 函数对数据的处理过程。图5.4中可以看出，对于某一些的数据点，$f(x) = wx + b$ 函数可以完美的满足任务进行预测。同时也可以看出，对于图5.4的情况，$f(x) = wx + b$ 函数就类似一个拟合函数。现在看图5.5，如果气速与气含率的关系比较复杂，很明显单一的 $f(x) = wx+b$ 模型并不能够精准的进行预测。然而分段的 $f(x) = wx+b$ 函数模型则对结果有很大的提升。因此现在需要的是一个分段的线性函数。

要实现这个分段的功能，可以通过设定不同的 x 的区间来实现，这是最好理解的办法。

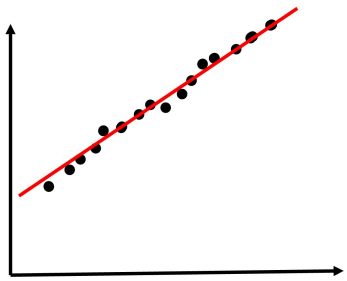


图 5.4: 气含率与 $f(x) = wx + b$ 示意图。黑点：测量气含率，红线：$f(x) = wx + b$。

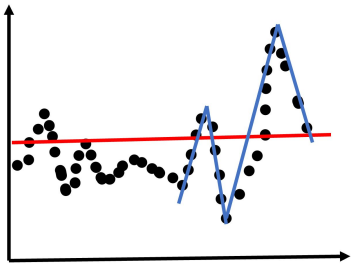


图 5.5: 气含率与 $f(x) = wx + b$ 示意图。黑点：测量气含率，红线：$f(x) = wx + b$。蓝线：分段 $f(x) = wx + b$ 函数。

但是数学上存在另外一种处理方法，一个复杂的曲线，可以通过很多基本的函数来叠加来实现。基本函数的数量越多，叠加出来的曲线越复杂。在机器学习领域，存在几个重要的基本数学函数，比如 Sigmoid 函数以及 ReLU 函数。Sigmoid **函数**可以表示为：

$$f(x) = \sigma(wx + b), \sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.5}$$

ReLU **函数**更加简单：

$$f(x) = \max(0, x) \tag{5.6}$$

针对一个复杂的曲线，可以写成多个基本函数的组合来构成。这些基本函数在机器学习领域被称之为**激活函数**。假定新的复杂的函数关系可以分为 2 个 ReLU 激活函数的叠加：

$$f(x) = v_1 \max(0, w_1 x + b_1) + v_2 \max(0, w_2 x + b_2) + b \tag{5.7}$$

如果使用大于 1 个神经元，神经元前方存在权重 w，神经元后方也存在权重 v。如果考虑更多的 ReLU 激活函数，其也可以写为：

$$f(x) = \sum v_i \max(0, w_i x + b_i) + b \tag{5.8}$$

图5.6表示了调用 5 个 ReLU 函数组成的 $f(x)$ 函数，用于预测气含率的示意图。在图5.6中，这些加和操作与激活函数构成了一个个的**神经元**[3]。很多的神经元连在一起就是**神经网络**。

[3]一些资料里面为了更明确的表示神经元的概念，将神经元分成两个部分，左半部分表示加和操作，右半部分表示激活函数。在本资料中，加和函数通过连线进行表示。

具体使用几个激活函数，这个数量需要人为定义，这种参数在 CFD 里面叫做自定义参数，在机器学习里面叫做**超参数**。同时很明显的可以看出，方程(5.8)是一个连续函数，但是其可以实现分段的数学特征。

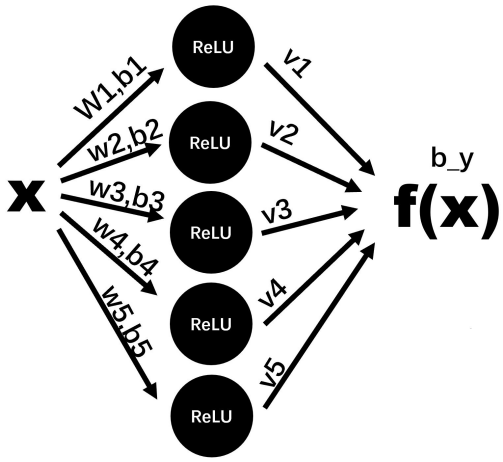


图 5.6: 5 个神经元组成的神经网络示意图。同样要注意，这里输入的可能是一个值，也可能是一系列的 x 的值（多个批次）。比如要找出 $x_1, ...x_5$ 与 $y_1^t, ...y_5^t$ 的对应关系，则模型需要输入 $x_1, ...x_5$，对其进行预测，力求预测的 $y_1, ...y_5$ 与 $y_1^t, ...y_5^t$ 的损失最小。

接下来考虑更复杂的情况，日常生活中，液体的粘度、温度也会影响气含率。真实的测量数据可能是这样的： 参考方程5.8，这种情况下，气含率 α 的模型可以写为：

	注气速度 x m/s	液体粘度 y m^2/s	温度 z	测量气含率 α
第一组	1	0.001	300	0.02
第二组	2	0.002	298	0.02
第三组	3	0.003	301	0.04
第四组	4	0.002	310	0.04
第五组	5	0.001	305	0.06

$$\alpha(x, y, z) = v_i \sum \max(0, w_{ii}x + w_{ij}y + w_{ik}z + b_i) + b \tag{5.9}$$

例如，针对第一组数据，考虑 5 个激活函数，有气含率：

$$\begin{aligned}\alpha(x_1, y_1, z_1) = {} & v_1 \max(0, w_{11}x_1 + w_{12}y_1 + w_{13}z_1 + b_1) \\ & + v_2 \max(0, w_{21}x_1 + w_{22}y_1 + w_{23}z_1 + b_2) \\ & + v_3 \max(0, w_{31}x_1 + w_{32}y_1 + w_{33}z_1 + b_3) \\ & + v_4 \max(0, w_{41}x_1 + w_{42}y_1 + w_{43}z_1 + b_4) \\ & + v_5 \max(0, w_{51}x_1 + w_{52}y_1 + w_{53}z_1 + b_5) + b\end{aligned} \tag{5.10}$$

在这种情况下，神经网络可以表示为如图5.7。5.7中左侧需要输入 x, y, z 的值，这一层也被称之为**输入层**。其中的 5 个神经元由 ReLU 函数组成，这一层被称之为**隐藏层**。最终的

α，被称之为**输出层**。很明显方程(5.10)中存在了大量的未知参数，训练模型就是要把这些

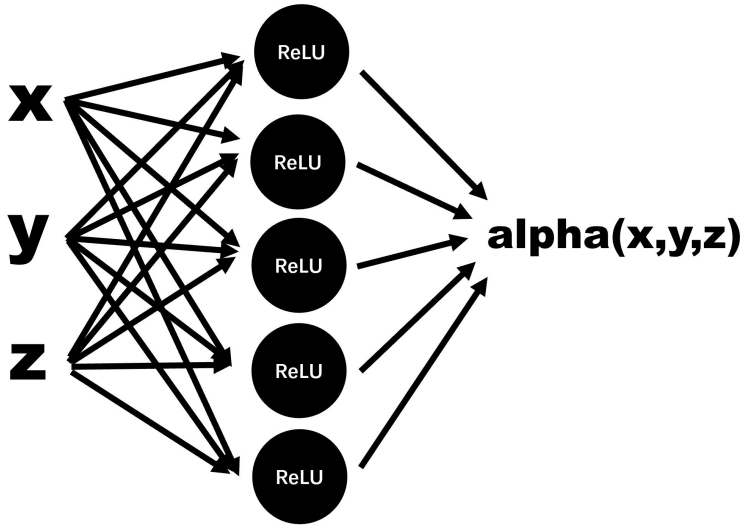


图 5.7: 5 个 ReLU 函数多自变量组成的神经网络示意图。这里输入的是一系列的 x, y, z 的值。

参数全部找出来使得损失最小。在 chatGPT 中有 1750 亿个未知参数（这个是问 chatGPT 他们自己回答的）。通常把这些未知参数都写成单独一列的向量，如：

$$\boldsymbol{\theta} = [w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, ..., b]^T \tag{5.11}$$

向量 $\boldsymbol{\theta}$ 被称之为**参数向量**。训练模型之后会得到参数向量。下一步就可以使用这个**模型**了。在获得这个模型之后，从模型左侧输入层输入相应的特征，模型会给出输出。这个过程叫做**前向传播**。另外一个相对应的是**反向传播**，反向传播是模型计算权重与偏置的方法，真实的计算流程通常是通过前向传播来计算预测值，然后计算损失，然后通过反向传播来更新权重与偏置，然后继续迭代进行。反向传播要结合算法更好理解。这部分内容请参考其他内容[4]。如果来看单独的神经元的话，所谓训练的过程，就是针对每个神经元寻找其权重与偏置的过程。

理论上，图5.7中的单隐藏层的神经网络（即**浅层神经网络**）也可以逼近任何函数[5]，但可能需要非常多的神经元，例如图5.5看起来需要 10 个分段函数，因此如果只有一层隐藏层，则需要至少 10 个神经元。对于一个更复杂的例子，肯定需要更多的神经元，这在实际中会导致计算资源的巨大消耗和训练难度的增加。而多层神经网络可以用较少的神经元和参数逼近同样复杂的函数。考虑图5.8中的神经元，在输入层的数据进来做加和之后，第一层的隐藏层会进行简单的进行 ReLU 激活函数的操作。定义这 3 个神经元的输出分别为

[4] http://dyfluid.com/backpropagation.html 在这个链接里面，后一个首选反向传播的例子。

[5] 确实存在这种神经网络，其被称之为极限学习机。

a_1, a_2, a_3：

$$
\begin{aligned}
a_1 &= w_{11}x_1 + w_{12}y_1 + w_{13}z_1 + b_1, \\
a_2 &= w_{21}x_1 + w_{22}y_1 + w_{23}z_1 + b_1, \\
a_3 &= w_{31}x_1 + w_{32}y_1 + w_{33}z_1 + b_1
\end{aligned}
\tag{5.12}
$$

进一步的可以把 a_1, a_2, a_3 看做输入，加和后可以在第二个隐藏层中继续做 ReLU 激活，同时每个隐藏层中的神经元数量可以不同。例如，第二个隐藏层的神经元可以输出：$a_1^{'}, a_2^{'}, a_3^{'}, a_4^{'}$：

$$
\begin{aligned}
a_1^{'} &= w_{11}^{'}a_1 + w_{12}^{'}a_2 + w_{13}^{'}a_3 + b^{'}, \\
a_2^{'} &= w_{21}^{'}a_1 + w_{22}^{'}a_2 + w_{23}^{'}a_3 + b^{'}, \\
a_3^{'} &= w_{31}^{'}a_1 + w_{32}^{'}a_2 + w_{33}^{'}a_3 + b^{'}, \\
a_4^{'} &= w_{41}^{'}a_1 + w_{42}^{'}a_2 + w_{43}^{'}a_3 + b^{'}
\end{aligned}
\tag{5.13}
$$

之所以有四个 $w_{1i}, w_{2i}, w_{3i}, w_{4i}$，是因为第二层存在 4 个神经元，因此是 4 个函数的叠加。具体的，$w_{1i}, w_{2i}, w_{3i}, w_{4i}$ 里面的 i 存在 3 个，是因为这一层前一层的神经元有 3 个。图5.8里面存在 3 个隐藏层，在更多的情况下，存在很多的几十个隐藏层，很多隐藏层组成的神经网络叫做**深度神经网络**。依据深度神经网络进行的训练，叫做**深度学习**。图5.8中每个神经元与下一层的每个神经元都进行了链接，这种神经网络也被称之为**全连接神经网络**。在一些情况下也被称之为**多层感知器**或者**前馈神经网络**。同时要注意，一般情况下如果说使用的是 N 层神经网络，指的是其中存在 $N-1$ 层隐藏层。这种情况下也可以说使用的是 $N-1$ 个隐藏层的神经网络。参数向量中未知参数的个数，针对全连接网络，连线的数量就是未知权重的个数，神经元数量就是未知偏置的个数。

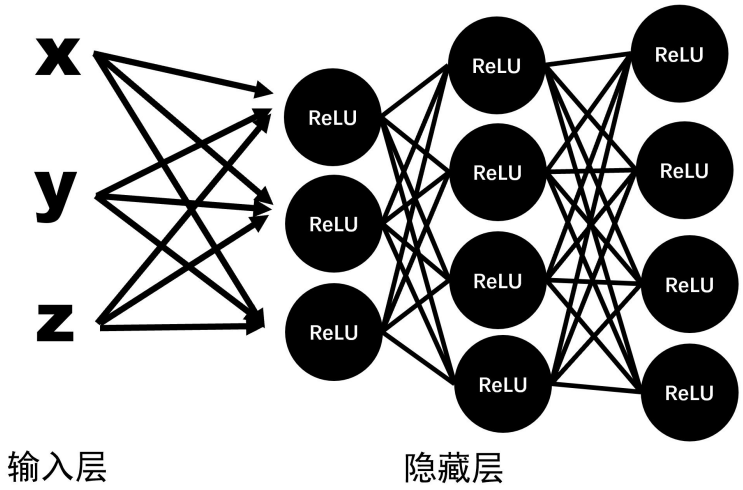


图 5.8: 三隐藏层 11 个神经元组成的神经网络示意图。

隐藏层内部的所有参数都需要通过特定的算法来计算出来（比如梯度下降法，在 libtorch 里已经封装好）。也就是模型通过训练之后，所有的参数都寻找出来之后，用户

在输入层输入 x, y, z 参数，神经网络的输出层就会预测出一个具体的气含率的值。也就达成了训练的目的。模型里面的参数越多越复杂，含有特别多参数的模型被称之为**大模型**。

现在看一个具体的实例。现在通过二维的固定网格，改变进口速度的工况，演示一下神经网络全流场预测的步骤。考虑最简单的情况，假定存在 1 个网格单元，针对不同的进口速度，存在一个对应的流场速度，这是一个一对一的关系。模型的目的就是找出这个一对一关系的函数，这与前文讨论的气速与气含率的关系是完全一样的流程。假定存在 10 个网格点，每个网格点的速度，都与进口速度存在一个一对一关系。那么模型可以首先找第 1 个网格点速度与进口速度的一对一关系，再找第 2 个网格点速度与进口速度的一对一关系，直到第 10 个网格点速度与进口速度的一对一关系。即使存在 10000 个网格点。流程也是相同的。现归纳如下[6]：

1. 首先通过 CFD 来模拟原始数据。比如针对一套固定的网格（10000 个网格点），调整 100 个进口速度，来模拟 100 个算例直至收敛状态。这 100 个收敛的流场数据，可以当做 100 套标签；每一套标签存储 30000 个标量值（流场变量数乘以网格点数，即 3×10000，其中 3 表示速度的 2 个速度分量与 1 个压力）。100 套标签共有 300 万个标量值；

2. 在训练模型的时候，模型需要读取标签来计算损失。一次性读取 100 套标签共 300 万的标量值是不太可能的。因此在这里以 10 套数据为一个批次。代码需要读取 10 套收敛的流场标签，共 30 万个标量值；同时，模型需要读取特征，即 10 个速度标量值；

3. 将 10 个速度标量值创建为一个张量命名为 `x_train`，通过前向传播预测 10 套定义在 10000 个网格点上的预测数据，其共存在 30 万个预测值（伪代码 `model->forward(x_train)`，预测的值可以称之为 `y_train`）；

4. 计算 30 万个预测值 `y_train` 与 30 万个标签的损失；

5. 对模型进行训练，训练后进行下一个批次的训练，共训练 10 个批次，也即一个回合。实操过程需要训练多个回合；

6. 训练后的网络，输入一个进口速度，可以预测 10000 个网格点的流场数据（网格点不能发生变化，也即网格不能发生变化）；

5.2 流场重组、超分辨率、扩散模型

数据驱动 CFD 很火的一个方向是通过**卷积神经网络** [89] 进行的流场重组，其与图像识别相似度高达 95%[7]。上文中对于网格固定不变的算例，相对来说比较好理解。很多情况

[6] 在下一节会介绍批次与回合的概念。

[7] 在这里推荐一个网站 https://poloclub.github.io/cnn-explainer/ ，虽然属于图像识别领域（卷积神经网络的全流场预测与图片识别实在太相似），但其详细的讲解了卷积神经网络的每一步的重组流程可以有助于理解。

下，用户想知道风洞里面任意一个汽车外形的流场数据。这种情况要比网格固定不变的算例复杂的多。在工业界这种应用也非常常见，一般被命名为 AI-CFD。也即通过神经网络的方法来进行流场重组。

具体来讲，这一类流场重组可以采用卷积神经网络来进行处理。监督学习的一种方法就是调用卷积神经网络来进行图像识别。在没有讨论卷积神经网络的原理之前，介绍卷积神经网络的含义是非常困难的。更详细的有关 CFD 与卷积神经网络相结合的基本理论请参考此资料[8]。这里可以初步把卷积神经网络理解为糙化后的全连接神经网络。这类似 CFD 中的多重网格求解器（在细网格上求解效率很低，因此需要进行网格糙化来增加低频分量迭代速度）。传统卷积神经网络主要干这个事：一张照片由很多像素点构成。每个像素点存在一定的数值表示颜色的深浅。把这些像素点构成的数据输入给计算机，模型会告诉大家这个图里面画的是什么。1989 年 LeCun 等的文章最开始提出了卷积神经网络的概念，就是用来识别手写的邮政编码 [88]。当然了，全连接神经网络也能做这个事。就是重组的不太准以及参数向量太大。在流体领域，如果使用卷积神经网络来重组流动结果，其精神就是把 CFD 或者实验获取到的每个网格点上的流动信息输入给计算机，并且告知计算机这些网格点的最终的收敛的结果作为标签。计算机经过学习后，就可以生成不同几何外形的流场数据。因此卷积神经网络进行的流场重组，跟图像识别非常相似。

发表于 2016 年的文献 [63] 被认为最早使用卷积神经网络来进行流场重组的研究。作为一种监督学习，通过卷积神经网络进行流场重组需要提供大量的学习数据。当然了，这种方法也不需要求解 NS 方程。文献 [142] 中针对 110 个翼形通过 OpenFOAM 模拟了 5280 个算例作为训练数据，文献 [19] 中针对 3 个翼形计算了 252 次 CFD 计算模拟。同时需要注意的是，类似 CFD 中的网格的概念，因为卷积神经网络模型需要重组整个流场。因此其输出的参数与输入的参数必然会特别多（对于固定网格改变速度的算例，输入的参数就少的很多）。举例来说，对于一个 10 万网格的传统的 CFD 流场来说，模型的输入为 10 万个网格点的数据以及 30 万个标签（考虑只有二维的速度以及压力一共 3 个变量）。模型的输出为是 30 万个数值，这还是仅仅一个算例。如果针对 100 套网格进行学习，输入的数据标签量为 100 个 30 万，即 3000 万个标量。这是非常庞大的数据。在传统图像识别领域，为了能够识别出 10 种物体，CIFAR-10 数据库包含了 60000 张图片需要进行训练。在 CFD 领域，如果进行 60000 次计算模拟，输入的标签数量将达到 6 万乘以 30 万。这是完全不可能的。当然了，实际过程中科研人员并没有对 60000 套算例进行训练。在相关工作中，文献 [142] 中的输入标签为 $60 \times 20 \times 10^6$（其中的 20 表示 20 套翼形数据，60×10^6 表示网格数据，因此其仅仅使用了 20 套训练数据），输出参数为 $60 \times 3 \times 10^6$（其中的 3 表示速度的两个二维分量与压力）。如果进一步的考虑神经元的数量，假如第一层使用了 5 个神经元，那么第一层参数向量数量则为 $60 \times 5 \times 10^6$。这是非常庞大的数字，训练起来异常耗时并且可能还会出现不准的情况。在机器学习领域，训练数据上看起来不错，但是对

[8] http://dyfluid.com/cnn.html

于新的数据却重组不准的情况称之为**过拟合**。在使用全连接网络进行流场重组或者图片识别的时候，因为模型参数实在太多，很多的细节与噪声都会被捕捉到，相反反而会重组的不准。这是一种典型的过拟合。过拟合的一种情况就是神经网络的层数的增加，结果重组能力并没有更好。

因为输入的特征实在是太多了，在神经网络中，用户不会一次性把训练所有的训练数据。假定一个网格，训练模型需要训练 100 套数据。将 100 套数据进行训练是不可能的。因此通常要把他们分成**批量**来处理。比如一次性输入 5 套数据，则 100 个流场数据会形成 20 个批量。每个批量里面存在 5 个流场数据。**批量大小**即为 5。神经网络会针对第一个批量，来更新一套参数。随后用户输入第二个批量，神经网络会在第一个批量的基础上，继续更新参数。直至所有批量更新完成，也即完成一个**回合**。卷积神经网络相对于普通的神经网络在与多个卷积层可以把输入的特征最小化。其中通常需要使用卷积核对输入特征进行扫描，在这一层操作会将输入的特征减小一点，随后进行 ReLU 操作（输入特征大小不变），随后再次进行卷积核的扫描，输入的特征再次减小。随后继续进行 ReLU（输入特征大小不变）。随后进行最大池化操作（输入特征大大减小）。随后继续进行卷积核扫描，ReLU 操作，最大池化等操作。最终跟随的是全连接的神经网络。在文献 [63] 中，输入特征最开始为一个 256×128 的向量，共有 32678 个值。经过几次卷积层后，与全连接网络相连的一层只有 1024 个值。有关卷积核、最大池化等操作原理，一些资料通过 excel 表格来进行介绍非常简洁，读者可以参考相关的工作[9]。

卷积神经网络还可以进行流场超分辨率重组。如图5.9所示，流场超分辨率重组的本质是给模型一个糙网格的流场结果，模型可以重组一个细网格下的流场结果。这其实类似解码器。也确实如此，在 Fukami 等人在 2019 年刊发在 JFM 的一篇文章中 [53]，首次使用解码器以及残差网络来进行流场超分辨率重组。Fukami 等研究了平均池化、最大池化以及跳跃连接对流场超分辨率的影响。图5.9所示的是平均池化以及最大池化进行超分辨率重组之后的结果。可以看出最大池化进行重组的流场，看起来略尖锐。流场超分辨率重组不仅仅可以用于 CFD，也可以用于实验中。kong 等的文章中 [80]，通过卷积神经网络来对实验监测的超音速流动进行超分辨率重组。其原理大同小异。

然而在使用数据驱动的方法进行流场重组的时候，很容易忽视的问题就是守恒特性。通过有限体积法进行的计算，因为其基于守恒定律因此不存在这个问题。Lee 和 You 在 2019 年在进行流场重组的同时，将流场的质量守恒、动量守恒也通过损失项进行表达，重组的结果相对较好 [90]。另外一个可能的技术点在于卷积网络的输入都是正方形的，然而 CFD 的计算域很少有正方形的情况，这就需要进行一个数据处理。简单的方法就是对空白区域进行填充空元素。另外如图5.11所示，一些文章则采用投影的方法来进行 [159]。

[9] https://poloclub.github.io/cnn-explainer/ 这个链接用图示讲解卷积层、池化层对图像处理的过程，理解很直观。https://towardsdatascience.com/understanding-convolutions-using-excel-886ca0a964b7 这个连接中通过 excel 讲解卷积层的数学操作，理解起来非常简单。

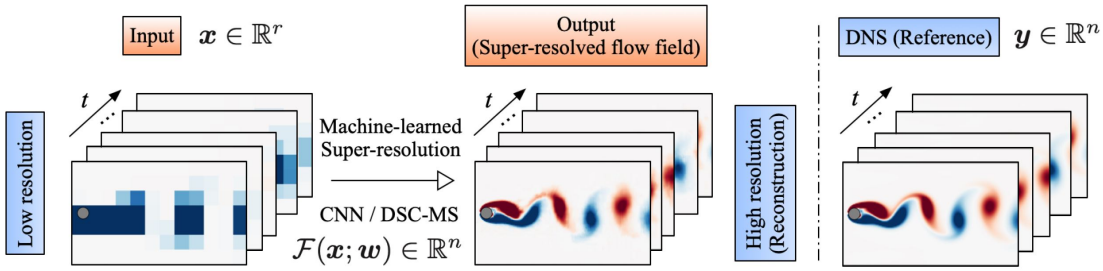


图 5.9: 流场超分辨率重组 [53]。

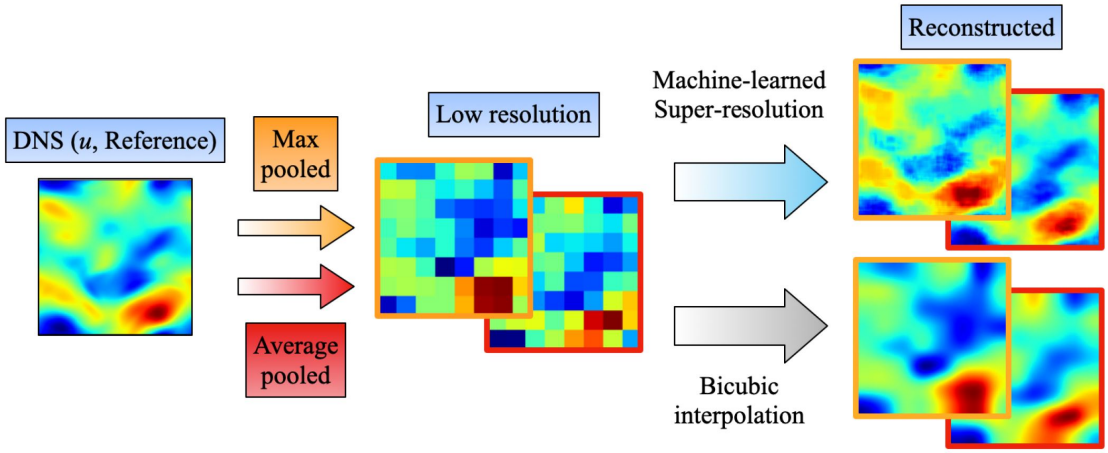
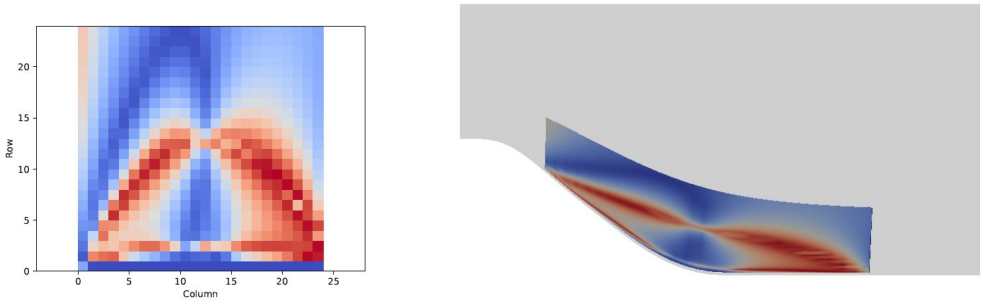


图 5.10: 平均池化以及最大池化进行流场长分辨率重组后的结果 [53]。



(a) Region as CNN input of shape 24×24.

(b) Region in the physical domain.

图 5.11: 将 OpenFOAM 计算的流场结果映射到卷积网络 [159]。

另一方面，数据驱动 CFD 通常需要大量的训练集。例如在 Guo 等的文章中，其针对每个特别简单的外形结构如三角形等，每个外形都需要进行 20000 次 CFD 计算来生成训练集 [63]。在 Sekar 等的研究中 [142]，其针对翼形模型，一共用 OpenFOAM 生成了 5280 个 CFD 算例来训练。同样是翼形训练，Thuerey 等人则通过 OpenFOAM 生成了 12800 个 CFD 算例来作为训练样本 [166]。可以预见的是，在不能通过实验来获取训练集的情况下，数据驱动算法只能通过经典 CFD 的方法来生成大量样本来进行训练。

另外一种流场重组方法可以结合生成模型来使用。生成式对抗网络（GAN）以及扩散模型是目前比较常见的生成模型。如图5.12所示，2023 年，Shu 等采用去噪概率扩散模型（DDPM）以及去噪隐性扩散模型（DDIM）来针对一个糙流场来重组一个精细解 [146]。其中还融入了物理偏微分方程的约束。其原理很简单。在 DDPM 中通过 x_t, t 来预测 ϵ_t^θ，在物理约束 DDPM 中，通过 x_t, t, c 来预测 ϵ_t^θ。其中的 c 表示源项为 0 的偏微分方程[10]。2024 年相继有一些使用扩散模型来进行流场重组的工作，在这里不一一赘述。目前来看通过扩散模型进行流场重组属于相对小众且非常新的工作。

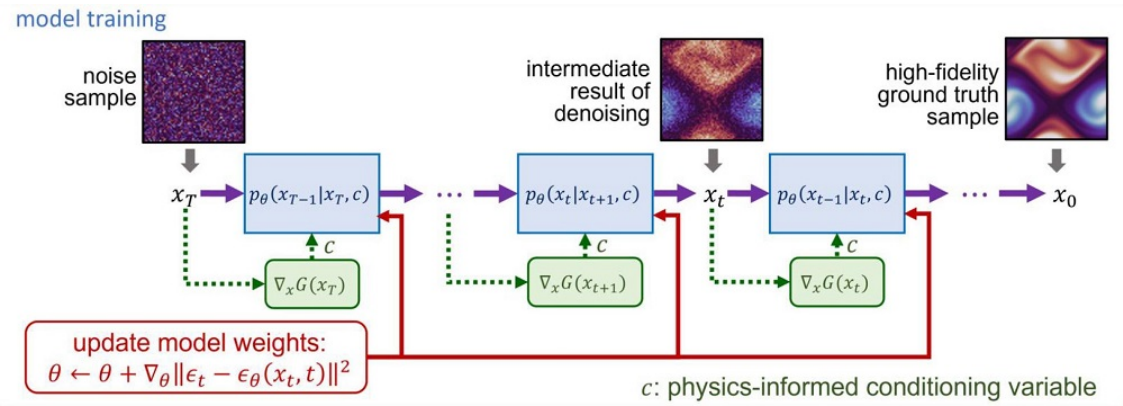


图 5.12: 通过物理嵌入扩散模型进行流场重组的流程图 [146]。

5.3 数据驱动加速求解

另外的卷积神经网络与 CFD 方向的结合是加速求解技术。Kochkov 等人成功的第一次的将卷积神经网络加速技术应用于 NS 方程并进行了 DNS 模拟 [79]。在此之前，他们将加速技术应用于标量传输方程小试牛刀 [204]。卷积神经网络 CFD 加速神奇的地方在于：它可以在粗糙的网格上获得与经典 CFD 算法在高分辨率网格下相同精度的解。图5.13显示的是文献 [79] 在四倍糙化的网格上进行了 LES 与 DNS 模拟，在缩放 4 倍空间分辨率以及 4 倍时间分辨率的情况下，CFD 结合卷积神经网络进行的重组与高分辨率 CFD 进行的计算结果相同。考虑传统的标量传输方程，其中的对流项 $\nabla \cdot (\mathbf{U}c)$ 在离散过程中需要

[10] 有关扩散模型请参考 http://dyfluid.com/diff.html

进行体网格到面网格的插值。在文献 [204] 中，作者们将卷积神经网络用来实现一种对流项的插值格式。目的是在低分辨率网格下，通过卷积神经网络重组的面插值结果与高分辨率网格的结果相同。并且在整个求解流程中，仅仅通过卷积神经网络来进行面插值的重组，其他部分与经典 CFD 求解技术相同。这存在一些优点，比如传统的 CFD 求解技术可以保证是守恒的。如果类似前述通过卷积神经网络处理整个流程，会存在不满足守恒的特性。Kochkov 等人后续将标量传输方程拓展到了 NS 方程 [79]。这种使用卷积神经网络的目的更倾向是构建一种数据驱动的数值格式。

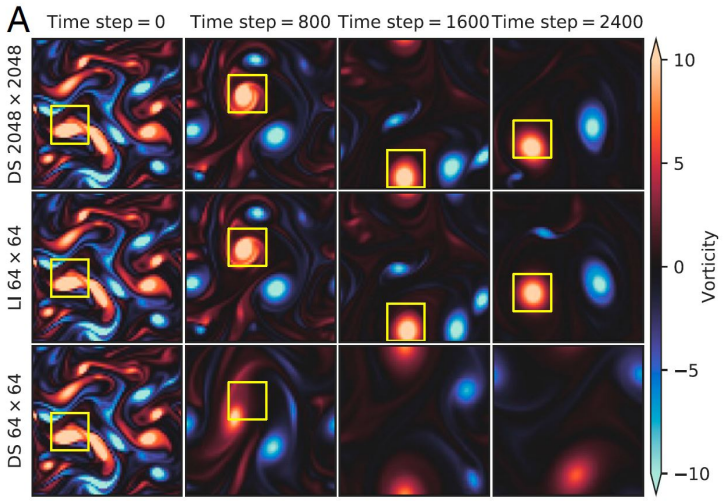


图 5.13: 卷积神经网络 CFD 加速技术示意图 [79]。纵列表示时间序列。横列表示分辨率（顶行：CFD 在细网格上进行的直接模拟，中行：CFD 结合卷积神经网络进行的模拟，底行：CFD 在糙网格下进行的直接模拟）。

另外的流场加速方法诞生于计算图形学领域。在电子游戏以及计算图形领域，需要生成烟雾、水流等渲染技术。尤其在电子游戏中，需要快速的进行烟雾渲染。在这种应用领域，求解的准确性并不是主要的。在 2016 年 Yang 等发表的文章中 [194]，看起来是最早的实现了一种数据驱动的投影法求解器。经典 CFD 领域的投影法每个时间步的最终的速度可以写为：

$$\mathbf{U}^{t+\Delta t} = \mathbf{U}^{*} - \frac{1}{\Delta t}\nabla p^{t+\Delta t} \tag{5.14}$$

其中 $p^{t+\Delta t}$ 通过下面的压力泊松方程求出：

$$-\nabla \cdot \left(\frac{1}{\Delta t}\mathbf{U}^{*}\right) = -\nabla \cdot (\nabla p^{t+\Delta t}) \tag{5.15}$$

其中 $\mathbf{U}^{*}$ 为通过动量方程预测的速度。求解方程(5.15)可以使用非线性系统求解器来进行求解，比如预条件共轭矩阵求解器（PCG）。在 PCG 中，初始迭代值可以选用已知的压力场 p^{t}。因此，在使用 PCG 求解方程(5.15)的时候，需要调用的为已知的压力场 p^{t}，动量预测速度的散度 $\nabla \cdot \mathbf{U}^{*}$。Yang 等的数据驱动替代模型，就是给定一系列的输入参数，替换掉使用 PCG 求解方程(5.15)的过程。在 Yang 等模型的输入层，需要输入当前网格以及相

邻网格的 p^t。也要输入当当前网格的以及相邻网格的散度 $\nabla \cdot \mathbf{U}^*$[11]。同时输入一个非 0 即 1 的位置函数（在壁面为 1，其他位置为 0）。输入层一共具有 19 个参数，输出层即为当前网格点的 $p^{t+\Delta t}$。模型训练好之后，可以替代通过 PCG 来求解方程(5.15)的过程。Ajuria Illarramendi 等以及 Tompson 在后续进行了类似的研究 [12, 167]。不过他们并没有将损失写成机器学习计算的压力与经典 CFD 预测的压力的函数，而是考虑预测的压力后形成的速度的散度是否为 0，从其来计算损失。这样的话形成了一种无监督的学习，且可以更容易的控制长效损失。在这一类方法推出之后，也陆续出现了一些改进方法 [189]，思想基本一致。

5.4 数据驱动湍流模型

不管是 RANS 方程还是 LES 方程，之所以大家认为他们存在不准确的地方在于湍流应力的模化。目前普遍认为不管是 Boussinesq 近似，还是各种非线性的湍流模型，计算的 $\boldsymbol{\tau}$ 与真实的 $\boldsymbol{\tau}$ 都存在差距。这就就成机器学习与湍流模型结合的一个渗入点。最早的这方面可以追溯到 2013-2014 年斯坦福大学湍流研究中心的工作 [170, 45]（然而 2013 年 Tracey 等的工作貌似没有掀起波澜）[12]，Duraisamy 和 Durbin 尝试使用机器学习的方法来优化一个湍流模型。在当时的工作里，神经网络的代码还是自己写的。并且通篇没有讨论到隐藏层神经元等概念（但是明确的使用了神经网络以及优化算法）。这说明在当时将机器学习用于提升 RANS 模型还处于一个初期阶段，或者说在当时在流体领域讨论神经元可能看起来是相当前卫的。2015 年，Tracey 等刊发的文章继续完善了 Duraisamy 和 Durbin 的工作 [171]（Duraisamy 为第二作者）。然而在这一篇工作中，使用了大量的篇幅讨论了神经网络、隐藏层、神经元、损失函数等各种概念。这一篇工作主要的目的就是建立流体变量与 Spalart-Allmaras 湍流模型中源项的关系。在 Spalart-Allmaras 湍流模型中，方程左侧的含义是非常明确的（时间项、对流项、扩散项），然而方程右侧是通过实验构造的一个数学关系。Tracey 等建立了一个模型，这个模型将流体相应的变量与 Spalart-Allmaras 湍流模型的源项建立了一个关系。因此在 CFD 求解过程中，不需要去复杂的计算 Spalart-Allmaras 湍流模型的源项。这个模型就可以输出源项。结果表明了该模型在二维平板、三维亚音速翼形等工况下都可以预测比较准的结果。

最具有影响力的工作应该是在 2016 年刊发在 J. Fluid Mech. 上的文章 [105]。传统的线性涡粘模型一直被认为是有缺陷的，尤其是其对各向异性湍流能力预测的缺失。类似的非线性涡粘模型虽然有一定的改进，但是由于收敛性问题也并没有大范围的代替线性涡粘模型。如图5.14所示，Ling 等在传统神经网络的基础上，提出了一套新的网络结构 TBNN（Tensor Basis Neural Network）[105]。传统的神经网络把剪切形变率张量以及旋转应变率

[11] Yang 等输入的是速度的倒数场，其可以理解为 $\nabla \cdot \mathbf{U}$ 的展开形式。

[12] 更早的出现在 2002 年 [120]，但是已经不属于机器学习与湍流模型的结合。

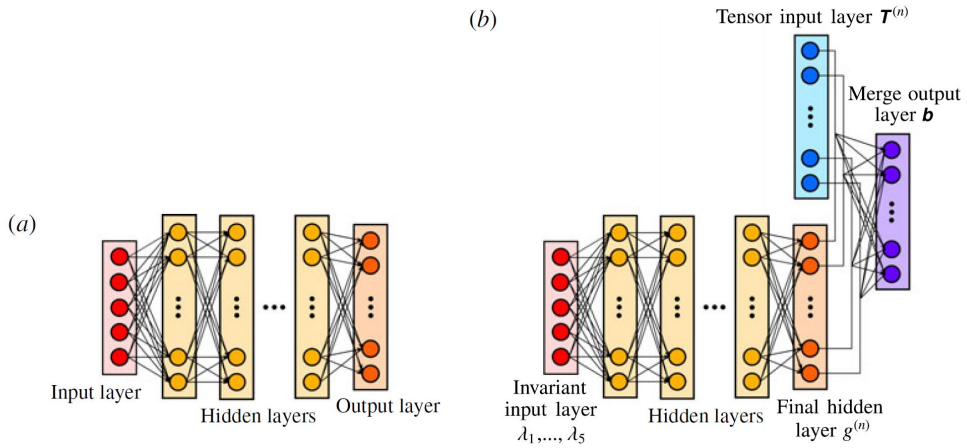


图 5.14: 传统神经网络与 TBNN 示意图 [105]。左侧表示传统神经网络，右侧表示 TBNN。

张量当做输入，输出层为各向异性的雷诺应力。在 TBNN 中，输入的为五个系数 $\lambda_1, \ldots, \lambda_5$（其为剪切形变率张量以及旋转应变率张量的函数），输出的为 g 参数，这个参数与 10 个二阶张量进行代数计算，可以获得各向异性雷诺应力。如图5.15所示，TBNN 模型可以更好的预测流场结果。

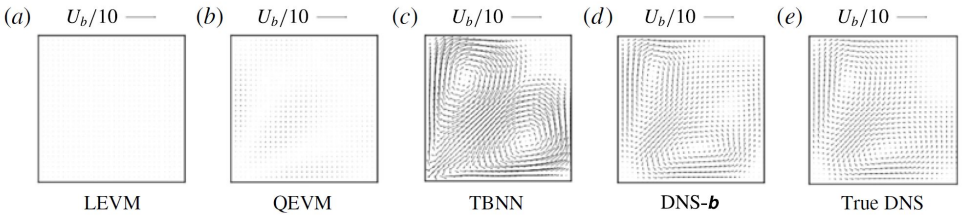


图 5.15: 不同计算方法计算的速度矢量图 [105]。LEVM：线性涡粘模型，QEVM：非线性涡粘模型，TBNN：TBNN 模型，DNS-b：把各向异性的值硬植入到 RANS 模型中，DNS：直接模拟。

然而在一些其他的工作中指出，另外一种提升湍流应力的方法不需要借助 TBNN。比如在 Xiao 等的工作中 [188, 182]，假定真实的雷诺应力与计算的雷诺应力的误差为 $\Delta\boldsymbol{\tau}$，其认为流场的一系列变量 $\mathbf{q}$ 与 $\Delta\boldsymbol{\tau}$ 存在一定的关系。这个关系一定是非线性且难以寻找的。因此需要通过神经网络把 $\mathbf{q}$ 与 $\Delta\boldsymbol{\tau}$ 的关系寻找出来。$\Delta\boldsymbol{\tau}$ 可以用于对现有的 RANS 模型进行优化，进而提升计算效果。Wang 等的工作表示该方法建立的模型，通过建立 $\mathbf{q}$ 与 $\Delta\boldsymbol{\tau}$ 的关系有一定的普适性 [182]。如图5.17显示，Wang 等在进行训练的时候使用了不同几何的工况来寻找 $\mathbf{q}$ 与 $\Delta\boldsymbol{\tau}$ 的关系。在这种情况下依然能够预测比原始 RANS 更精准的结果。Wang 等与 Ling 等的工作有一个最重要的区别，就是 Wang 等通过模型预测的是一个雷诺应力的误差量，Ling 等是直接预测一个各向异性雷诺应力进而不需要调用任何湍流模型。

同期还存在一些更简单的方法，例如 Zhu 等将网格点的速度、压力等变量与湍流粘度建立一个关系，这个关系通过神经网络给出 [203]。在训练后，这个模型就可以用于各种不

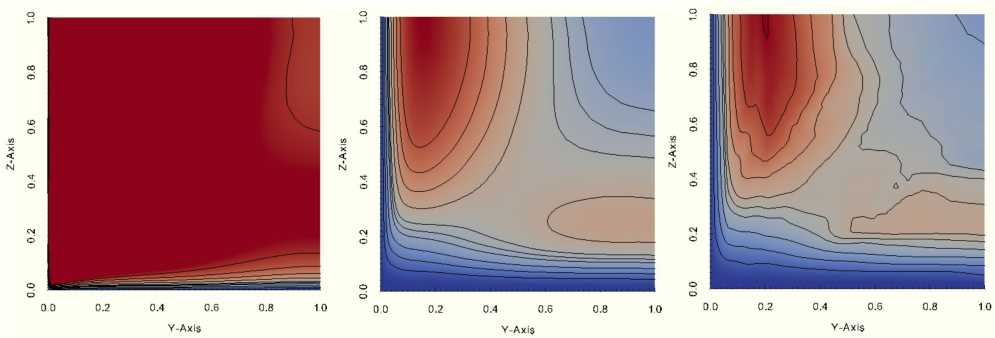


图 5.16: 采用不同方法计算的 τ_{zz}[188]。左侧：普通 RANS，中间：直接模拟，右侧：机器学习。

同的流场结构。在这个算例中，甚至不需要调用湍流模型，因为湍流模型是通过神经网络直接预测出来的。2021 年刊发在 Computers and Fluids 的一篇文章中 [115]，作者们也采用了基本一样的策略，并没有看出有太大的区别。

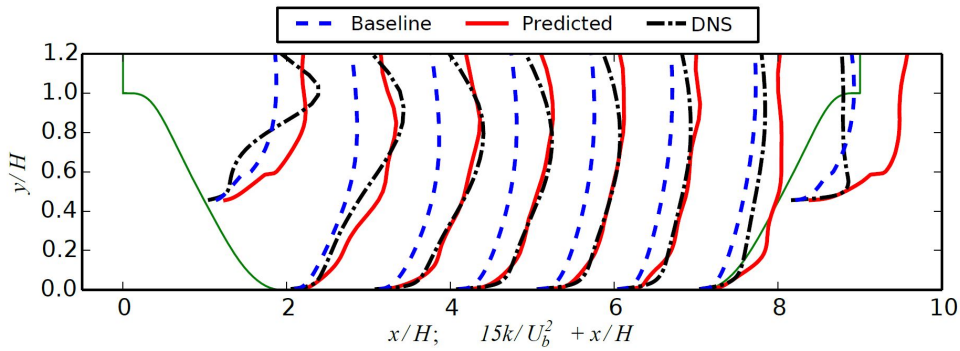


图 5.17: 采用不同方法计算的速度 [182]。蓝线：原生 RANS 模型，红线：数据驱动模型，黑线：DNS。

机器学习在 LES 领域的结合略有不同。据作者所知，Sarghini 等在 2003 年最早的进行了数据驱动的 LES 模型的开发 [141]。在他们的工作中，通过 Bardina 湍流模型来生成 LES 湍流模型的数据并将其作为标签。输入层为 15 个流动变量，分别是 3 个速度分量在 3 个方向上的导数以及六个脉动速度乘积分量。输出层为 1 个 Smagorinsky 的模型系数 C_s。因为预测的 C_s 是一个局部的值，因此这是一个动态的 LES 模型。该文章发表后的 10 年只有 16 次引用。然而在 2017 年之后距今 5 年时间，文章被引 160 次。单单 23 年一年就被引 70 次。Gamahara 和 Hattori 也进行了类似的工作 [55]。从算法层面唯一的不同在于 Gamahara 和 Hattori 使用的是 DNS 的数据，而 Sarghini 等使用的是 LES 的数据。图5.18是 Gamahara 和 Hattori 使用神经网络预测的雷诺应力分量与 DNS 预测的结果的对比。Xie 等的工作也是基于同样的思路，只不过输入层为速度分量的一阶导数，输出层为湍流应力的导数 [190]。在 Maulik 等发表的文章中 [114]，通过全连接网络，在结构网格中，将某个网格单元以及周围的 8 个网格单元（一共 9 个网格单元）的旋度以及流函数变量，附加剪切应力的模以及旋度梯度的模，一共 20 个变量作为输入层，输出层为一个粘

度相关量。作者们还测试了忽略掉剪切应力的模以及旋度梯度的模作为输入层对结果的影响。从训练的角度来看，损失的下降基本无差异，但是从能谱的角度来看，20 个输入变量看起来要更好一些。作者们还尝试了 5 个网格单元作为输入层，在这种情况下如果增加网格的层数，结果与 9 网格点作为输入层结果一致。

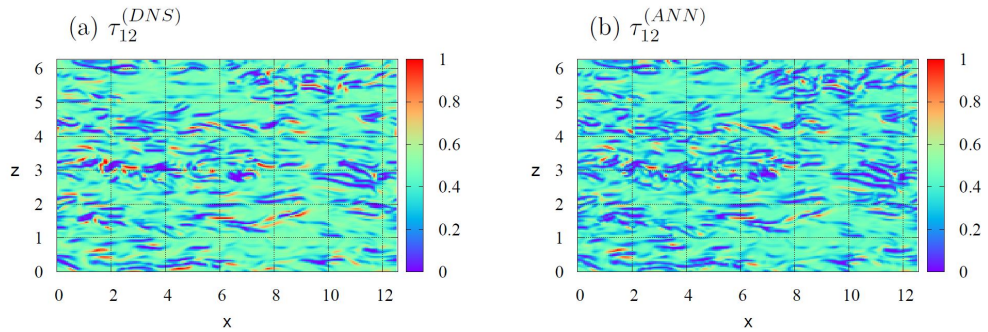


图 5.18: 使用神经网络预测的雷诺应力分量与 DNS 预测的结果的对比 [55]。左侧：DNS 结果，右侧：神经网络结果。

Beck 等也通过神经网络进行了大涡模拟 [17]。他们认为他们数据驱动的 LES 模型可以被认为是一种 perfect LES 并且在文章中大量的对其进行解释“为什么我们将其称为是 perfect 的”。Beck 等的模型可以在糙网格上进行 LES 模拟并且精度与 DNS 相当。回头看方程(4.8)与(4.9)，传统的 LES 模型通过 Boussinesq 对湍流应力 $\boldsymbol{\tau}$ 进行封闭。Beck 等的“完美 LES”模型通过机器学习，并借助 DNS 的数据，将糙网格的湍流应力 $\boldsymbol{\tau}$ 与细网格 DNS 模拟的湍流应力建立模型。因此该模型在训练过程中，需要读取糙网格的输入数据同时读取细网格的标签。为了解决参数比较大的问题，研究采用了残差神经网络（一种卷积神经网络）。图5.19为“完美”LES 预测的湍流粘度与 Smagorinsky 模型预测的湍流粘度对比。可以看出 Smagorinsky 模型预测的湍流粘度要大于前者（比较耗散的）。

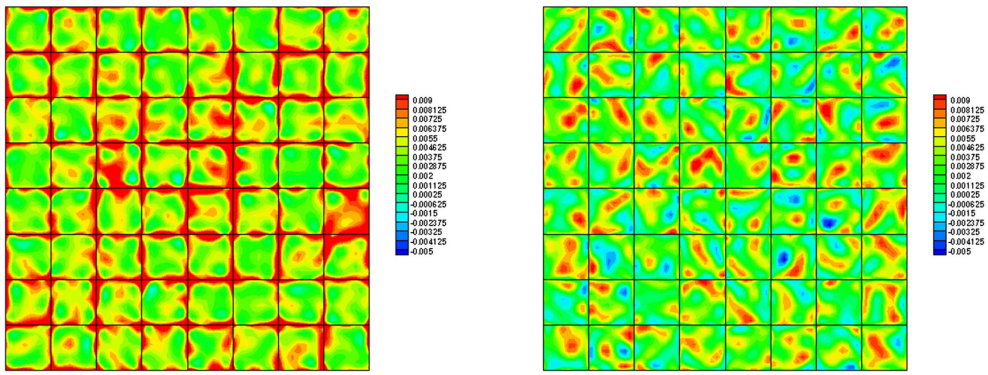


图 5.19: “完美”LES 预测的湍流粘度与 Smagorinsky 模型预测的湍流粘度对比 [17]。左侧：Smagorinsky 模型，右侧：“完美”LES。

在数据驱动湍流模型如火如荼的发展中，也存在不一样的声音。其中尤其以 Spalart 为重。Spalart 在上世纪 90 年代提出了著名的 Spalart Allmaras 湍流模型 [153]。Spalart 认

为当前使用数据驱动来进行湍流建模不具有最终的工业应用意义。2023 年 Spalart 发表文章硬怼数据驱动湍流建模 [152]。现对其观点进行总结如下：

- 数据驱动湍流建模远远没有成功。许多文章提出了一些不实际的方法，这些方法在数学上或者物理上都是错误的。或者存在严重的过拟合缺点。Spalart 认为湍流文化需要多年去学习。如果没有相当长的时间去学习，这很难领会到真正的湍流文化。

- 湍流模型的几个关键属性是通过偏微分方程的复杂特性确定的，这些特性并不容易推导出来。这意味着，纯粹依赖数据驱动的湍流模型无法有效捕捉这些复杂的数学特性，其超出了简单数据模式识别的能力范围。

- 一些文章的标题表明了作者低估了湍流问题的复杂性和难度，没有认识到这个领域已经变得多么艰难和棘手。

- 只有很少的机器学习论文产生的模型是可以编写代码复现的。大多数情况下，作者们都在自娱自乐，缺乏实用性。

- 机器学习充其量是人脑的工具。在 CFD 里面，最简单的就是去优化，比如湍流模型里面的 C_μ 这个系数。另外就是 Spalart Allmaras 模型里面的 f_w 系数。然而在 1992 年，相关文献就已经使用 DNS 数据来提供更精准的 f_w 分布。机器学习领域则把训练出来的 f_{v1} 分布声称是一个更好的 Spalart Allmaras 模型。同时，非常时髦的起了个新的名字，叫数据驱动湍流模型。

- Spalart Allmaras 模型里为了尽可能拟合 f_w 分布，需要采用各种各样的人为手段去施加限定（比如防止 0 除以 0，再比如 f_w 和 r 的分布并不是一对一的关系）。机器学习如果不施加人类的干预就很完美是完全不可能的。对模型的调试需要大量的经验。

- 数据驱动湍流模型普适性太差。在机器学习领域，他们一般仅仅用于一个特定的算例，比如一个交叉射流。为了尽可能的保证模型的普适性，数据驱动类湍流模型也能仅仅如此。

- 很多湍流理论的细枝末叶在机器学习里面并没有遵守，因此很多机器学习 CFD 的文章存在很多错误并且持续反复的发生。这在经典 CFD 领域也时有发生。有的时候由于审稿人放水的原因，一些有错误的文章被成功发表。很多审稿人也不会去花几个小时的时间去推导稿件里面的公式。一旦错误的算法被权威的期刊发表，这个错误就会繁衍开来。

- Spalart 认为在湍流文化里，应该存在一些软限制以及硬限制（比如应该使用速度而不是速度的模等）。可能有一些人认为是 Spalart 过于老套且不敢面对新技术的挑战。但是在 2022 年的 NASA 研讨会上，一些人当面指出了数据驱动湍流模型忽略了很多关

键性技术问题。有一些受邀请的汇报人直接违背湍流文化的软限制以及硬限制。一些工作还发表在了非常好的期刊上。

- 某些人曾经认为可以通过简单地将大数据和机器学习结合起来，以一种强烈的“数据驱动”方式创造出有用的新模型。然而，这种想法不切实际。因为湍流模型里面的软硬限制都只能通过人类去分析 PDE 的数学特性。

5.5 数据驱动多相计算流体力学

多相流与机器学习的交叉研究。多相实验流体力学与机器学习也有很多的交叉（比如泡泡识别、流场重组等）。在这里我们讨论机器学习与计算流体力学的结合。多相计算流体力学比较好理解的就是曳力的替代/代理模型（Surrogate Modeling）。工业尺度的多相流模拟通常使用欧拉欧拉模型或欧拉拉格朗日模型。在这两种模型里，需要指定模型的曳力项。一些文章直接对这个曳力建立替代模型。在 He 与 Tafti 的文章中 [67]，作者们首先通过颗粒解析的多相流直接模拟，求得每个颗粒上的曳力。随后将这些数据作为特征与标签，通过神经网络建立映射最终得到每个颗粒的曳力。需要处理的地方在于在 CFD-DEM 方法中，一个网格通常存在多个颗粒，针对每个网格，现存的是相分数与颗粒数量。然而颗粒解析的直接模拟中，网格要小的多。为了解决这个看起来不兼容的问题，作者们在颗粒解析的多相流直接模拟中，建立了一个视窗，在这个视窗内包含了多个粒子（5 个或者 15 个），这个视窗就可以看做是 CFD-DEM 中的网格。进一步的，将这个视窗的体积分数、每个颗粒的位置坐标，以及雷诺数当做神经网络的输入层。输出层则为针对这个体积分数的每个颗粒的曳力。实现了替代模型的建立。图5.20所示的为高亮颗粒与其他颗粒距离导致的曳力关系，在左侧的图中，高亮颗粒的曳力最大。在右侧的图中，高亮颗粒的曳力最小。作者的意图就在于表示在颗粒流中，某一个颗粒的曳力与相邻颗粒存在高度的非线性关系。这个非线性关系可以建立一个替代模型。在一些文章中，替代模型主要用来处理曳力系数。传统方法的曳力系数中通常表示为雷诺数的函数。这个曳力系数与雷诺数的一对一关系是通过实验拟合而来。在替代模型中，这个拟合关系可以通过机器学习模型来进行。例如在 Yang 等的工作中，EMMS 的模型系数 H_d 通过替代模型与一系列的参数（如密度、粘度、气体速度等）建立映射 [197]。图5.21中显示的是不同的神经网络结构预测的 H 与解析解的对比，很明显更多的神经元可以预测更精准的结果。在另外的一些研究中，作者们更偏向对比构建不同的神经网络来测试哪一种网络结构预测的结果更好 [202]。

在多相流直接模拟领域，机器学习也用来建立相应的替代模型。例如对于传统的 VOF 方法，需要在动量方程源项中求解界面的曲率。在有限体积方法中，大量的文章通过 CSF 模型来计算，这也是 OpenFOAM 采用的标准方法。不过这个曲率的计算方法目前依然是国际上的研究热点。早在 2002 年，Meier 等就通过最小二乘法的方法，建立了相分数与曲

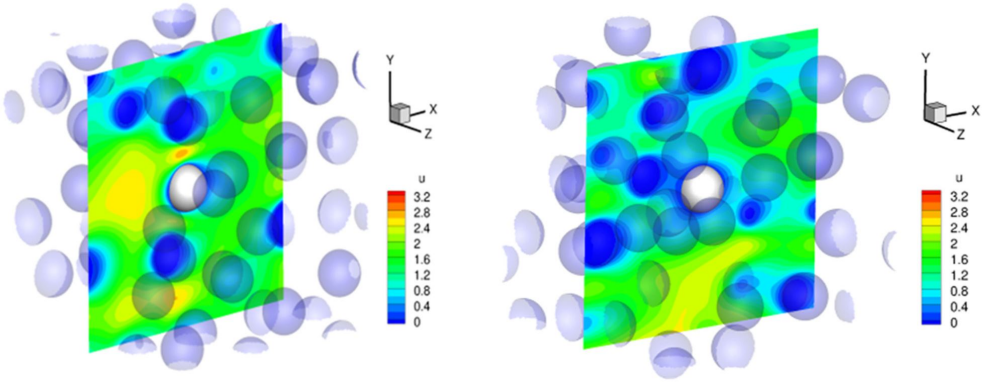


图 5.20: 高亮颗粒与其他颗粒距离导致的曳力关系 [67]。左侧：高亮颗粒曳力最大。右侧：高亮颗粒曳力最小。

率的直接关系 [116]。2019 年，Qi 等的这一篇文章中 [130]，通过机器学习的方法将最小二乘法替代，通过建立相分数与曲率的替代模型来求解曲率。

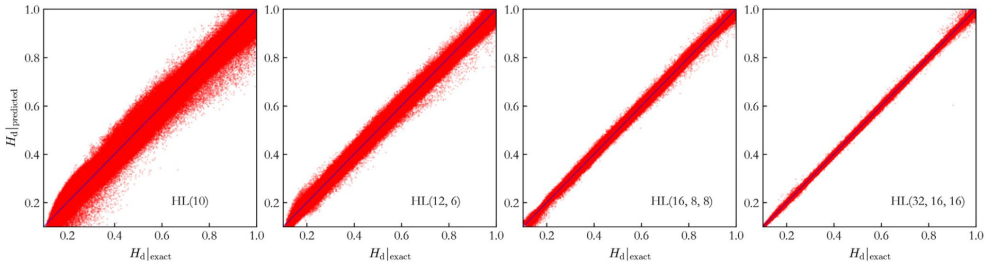


图 5.21: 不同神经网络机构预测的不同的结果 [197]。

5.6 循环神经网络与瞬态流场预测

循环神经网络主要用于处理文字以及语音。例如人类输入一段文字之后，机器要可以判断出可能得下一个文字是什么。比如人类说："我是中国人，我说 xx"。机器要能够自动填充为"我是中国人，我说汉语"。在 CFD 领域，一些研究将流场信息考虑为时间相关的，然后通过循环神经网络对下一个时间步的结果进行预测。这尤其适用于处理瞬态流场。这也是很好理解的，比如前序 50 个时间步里面的某个网格的 x 方向的速度结果，一直呈现了类似 sin 函数的数据。那很容易的可以判断出后续呈现 sin 函数的结果的可能性相当大。因此使用循环神经网络来预测圆柱扰流的瞬态流场时间序列看起来是理所当然的。因为圆柱扰流在相当长的一个周期内都呈现了重复的波动。Srinivasan 等看起来最先使用了长短时记忆网络（一种循环神经网络）进行的瞬态流场预测 [157]。温凯杰等通过长短时记忆网络在 OpenFOAM 下通过与 libtorch 耦合进行了气固多相流的瞬态流场预测 [208]。其思想主要是通过 OpenFOAM/CFD 来计算 1-10 帧的流场数据。在获得流场时间序列之后，将其输入到 libtorch 模型中进行下 11-20 帧的流场数据预测。随后再通过 OpenFOAM/CFD

来计算 10 帧数据，并反复进行。图5.22显示的是输入 10 帧数据后，长短时记忆网络预测的最后一帧的结果与真实的 CFD 计算的结果的对比。

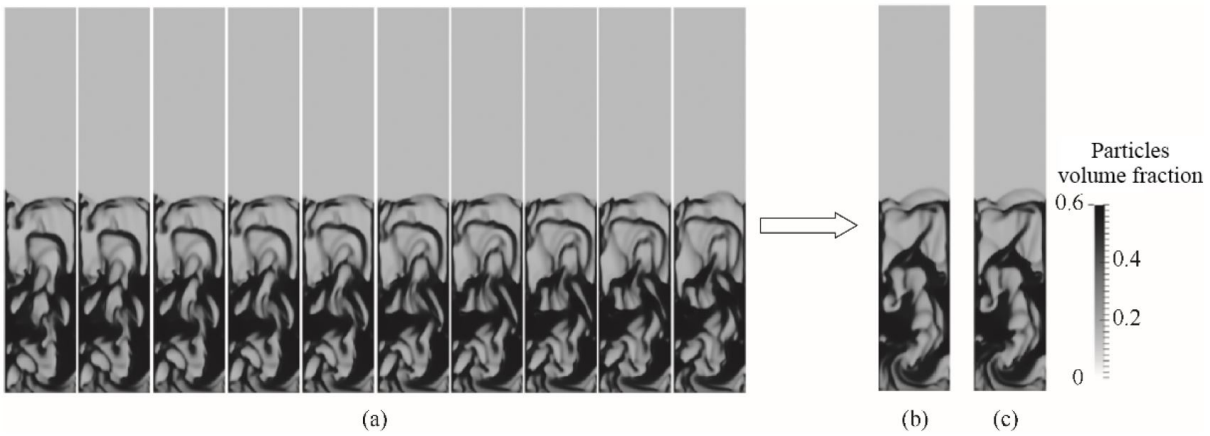


图 5.22: 长短时网络预测的输入时间序列流场、预测流场以及 CFD 计算的流场 [208]。

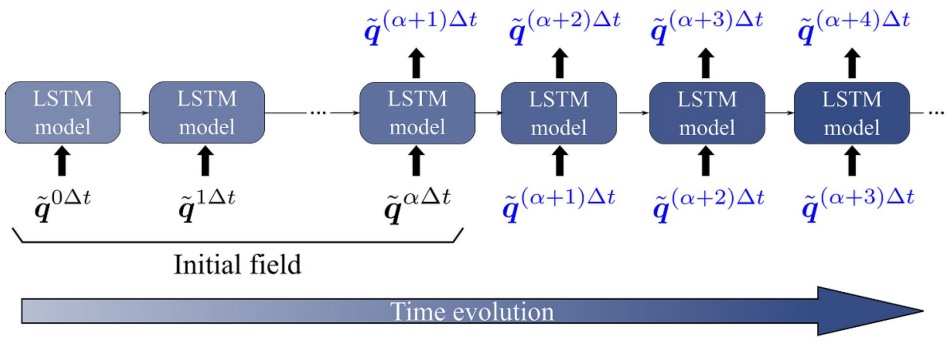


图 5.23: 长短时记忆网络瞬态流场预测原理 [66]。

Hasegawa 等将卷积神经网络与长短时记忆网络进行结合 [66]，首先通过卷积神经网络作为编码器将 DNS 数据进行编码，随后将编码后的时间序列信息输入到长短时记忆网络并进行后续的流场信息预测，最后用卷积网络的解码器还原成为 DNS 的高精度数据。图5.23形象的介绍了 Hasegawa 等工作中的原理。其将 DNS 流场的 $0-\alpha\Delta t$ 时间序列的流场结果输入到长短时记忆网络，随后其可以预测后 4 个时间步的流场结果。图5.23只是 Hasegawa 等工作中关于长短时记忆网络的一部分。图5.24是 Hasegawa 等工作的全流程。DNS 的数据首先通过编码器进行编码后，

在 CFD 相关领域，长短时记忆网络也大量的被用于溯源研究 [147, 34]。因为其与 CFD 关联较小在此不做详细介绍，但在这些研究中通常使用 CFD 来生成数据集。

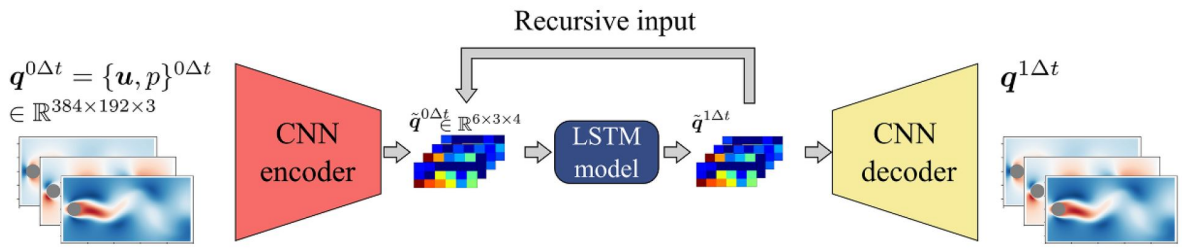


图 5.24: 长短时记忆网络与卷积网络结合瞬态流场预测原理 [66]。

5.7 正向逆向 PINN、隐藏流体力学

Raissi 等在 2019 年于 Journal of Computational Physics 首发 PINN（物理信息神经网络）的研究论文之后 [131]，吸引了学术界的大量关注。在 PINN 中，被求解的 PDE 以及相应的边界条件，成为了损失函数。机器学习就是要对模型进行训练，使得相应的数据点上的损失将为最低，也就是满足待求解的 PDE。这些 PDE 是符合物理模型的（例如 NS 方程等）。因此 PINN 的解要遵守相应的物理定律。类似于有限体积法、谱方法，PINN 是一种求解 PDE 的方法，且不基于任何求解方法。使用 PINN 求解 NS 方程，与传统有限体积法毫无关系，也不需要有限体积法的任何知识。因此可以看做是一个全新的技术。PINN 求解 PDE 可以看做监督学习以及无监督学习的混合。对于一些比较复杂的 PDE 系统，用户需要输入一些训练点来帮助 PINN 提升训练效率。对于一些特殊的边界条件，比如固定值边界条件，PINN 要求在边界处的预测值满足给定值，也可以看做是一种监督学习。另一方面，对于 PINN 要满足的方程约束，以及一些类似零法向梯度的边界条件约束，更像是一种无监督学习。PINN 求解 PDE 的关键技术在于自动微分。传统 FDM 在求解 PDE 的时候，使用数值导数的概念来计算（在这里不做介绍）。PINN 求解 PDE 需要使用自动微分的方法来计算导数[13]。自动微分的参数都建立在神经网络之上，因此 PINN 的训练过程，就是寻找神经网络的正确参数，在使用这些参数计算的自动微分的解，满足相应的物理约束（PDE 以及边界条件）以及相应的训练点。

下面以一维 ODE 方程为例，其方程可以写为：

$$\frac{\mathrm{d}T}{\mathrm{d}x} = 0 \tag{5.16}$$

方程(5.16)中存在一个导数。如果考虑 CFD 的网格，假定存在一个从左到右的 5 个网格点。在求解的时候需要给定边界条件以及初始条件。假定左边是固定值边界条件，右边是零法向梯度边界条件。很简单，如果右侧边界的 T_{bc} 给定，那么这个方程的解就是：

$$T_{bc} = T_1 = T_2 = T_3 = T_4 = T_5 \tag{5.17}$$

其解并不取决于初始条件。

[13]请参考 http://dyfluid.com/backpropagation.html

如果使用 PINN 来求解方程(5.16)，同样需要定义配点（其可以近似理解为 CFD 中的网格点，英文为 collocation points）。假定同样存在 5 个配点。这些数据点是算例的初始条件，其被作为了输入层的数据。输出层就是温度变量。要注意，在网格设计过程中是一对一的，而不是五对五。也就是输模型可以输入一个 x 的值，输出一个 T，而不是必须输入 5 个 x 的值，输出 5 个 T。在计算的过程中，计算机会寻找一个特定的参数，使得满足两个条件：

- 满足方程(5.16)。不满足方程(5.16)的程度可以看做方程损失。因此方程(5.16)的损失就可以看做是在所有 x 点上 $\frac{\mathrm{d}T}{\mathrm{d}x}$ 的导数与 0 的差值的加和的函数（例如均方误差）；
- 满足边界条件。如果是固定值边界条件，那么在边界处预测的 T 与给定的固定值的差值的函数（例如均方误差）就可以看做边界损失；
- 附加训练点/监督点/标签的情况，可以按照监督学习的概念求得损失最小化；

将上面三个损失加和，即为总损失；PINN 训练的过程追求总损失最小化。一旦训练的模型总损失最小，很明显求解的解同时满足 ODE，同时满足边界条件，同时满足配点。也即方程的解。

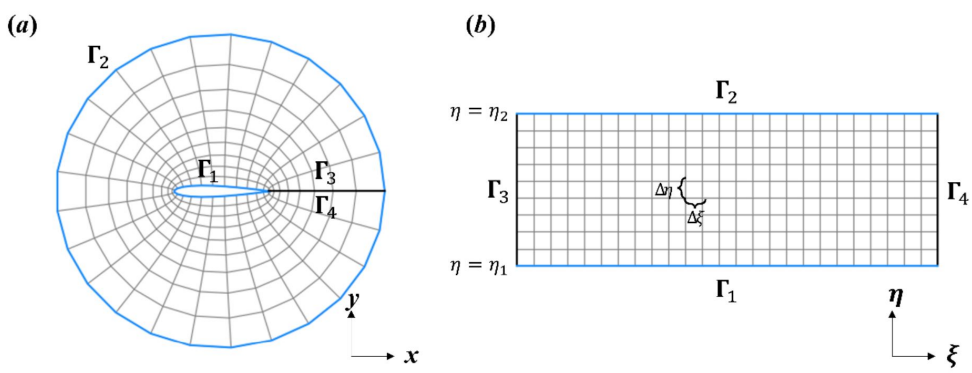


图 5.25: 网格坐标转换 [28]。

目前国际上也有一些通过 PINN 来求解 CFD 方程的研究。在最开始 Raissi 等的研究中，PINN 就被应用于求解 Burgers 方程以及圆柱绕流 [131]。PINN 目前求解 PDE 主要的问题是在梯度比较大以及方程比较刚性的情况，PINN 训练比较困难。Ren 等建立了一个简短区域与损失项的函数关系 [135]，例如在激波附近的速度梯度比较大，因此将其梯度的倒数（比较小）作为一个权重乘在了方程的损失项中。这样可以在梯度大的区域减少方程损失使其更好的训练。如图5.25所示，Cao 等在使用 PINN 计算翼形的时候，将贴体网格转换为特定网格坐标系并进行类似有限差分方法的坐标转换 [28]。图5.26显示了在将网格转换后，相应变量的转变区更加的光滑，这有利于 PINN 的训练。

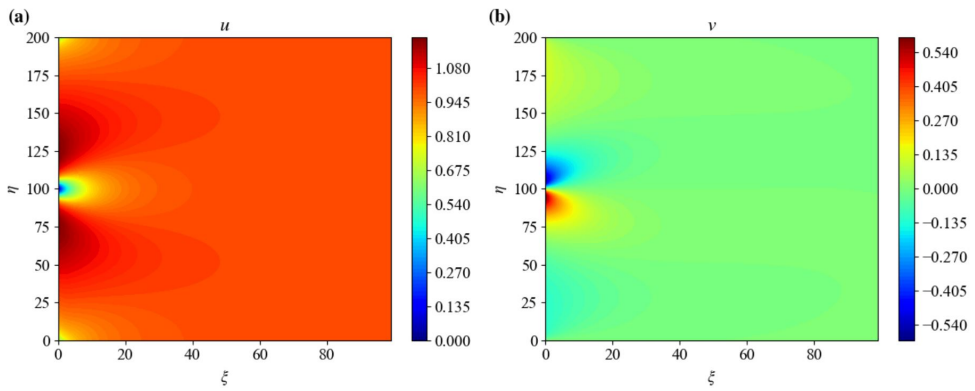


图 5.26: 网格转换后的相应变量的云图 [28]。

在其他领域，群体平衡模型作为一系列的 PDE 方程，一直以来比较难求解。类方法可以将这一系列未封闭的 PDE 方程进行封闭并通过常规数学方法求解。然而 Chen 等通过 PINN 方法求解了类方法得到了比较好的效果 [33]。同样的，化学反应通常涉及到常微分方程求解，且很多情况下是刚性的，Ji 等针对化学反应 ODE，尝试用 PINN 进行训练失败后，将化学反应 ODE 改成了准稳态的方程后 PINN 训练成功 [74]。类似使用 PINN 训练刚性 ODE 的文章也可以参考 [16, 16]。在 Baty 等的文章中测试了很多的 ODE[16]，一些基本的测试表明对于 ODE，PINN 需要施加训练点以及一些附加的限制。图5.27可以看出，施加 2 个训练点与能量守恒的情况下就比施加 1 个训练点要大幅改善，且表明附加能量守恒的限制是至关重要的。

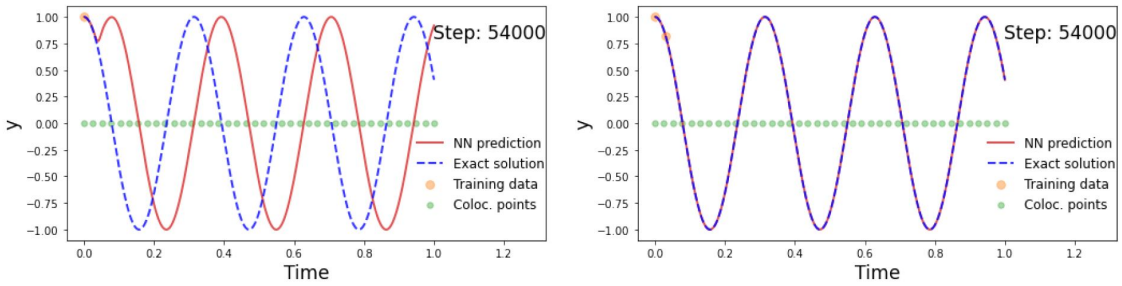


图 5.27: PINN 训练 ODE 与数值解的对比。左图：施加单训练点，右图：施加两个训练点与能量守恒 [16, 16]。

实测在 OpenFOAM 环境下挂载 libtorch 进行了若干 PINN 的计算。最重要的一个体验就是 PINN 训练过程特别慢。例如在图5.28中采用了 1 万个配点（可以理解为网格点）的通过 PINN 来求解的顶盖驱动流（粘度为 0.1）。其采用了 4 隐藏层共 80 个神经元。在采用 CPU 训练的过程中大约需要 2 小时。同样的 10000 网格点的 CFD 算例，采用单核计算大约需要 3 分钟。当粘度进一步降低时，则必须要调整学习率，还需要增加神经元数量，否则训练失败。结果如图5.29所示，但依旧存在可提升的空间。PINN 原始的提出者也

认为，原生的 PINN 在雷诺数较大的情况下，训练困难，甚至有可能出现多个合理解 [75]。在文献 [184] 中也存在类似的结果，如图5.30所示，PINN 求解顶盖驱动流在雷诺数增加的时候无法训练。文献 [184] 建议逐步的降低雷诺数来进行训练。该方法也被称之为课程训练方法（Curriculum Training）。

在 PINN 领域还存在一些其他的研究点。之前也曾有研究直接放弃深层网络，而采用极限学习机来与 PINN 结合进行计算。感兴趣的请关注 [47]。还有的算法将 PINN 定义的损失，定义到了经典有限体积法的网格面上，新的 PINN 方法被称之为深度有限体积法 [31]。在这种方法中除了配点只要定义之外，还需要定义包围配点的网格面。另外一些工作研究了 PINN 采用的激活函数，从文献的工作来看，Swish 函数以及 Tanh 函数是比较受欢迎的 [13, 37, 44]。ReLU 以及其他类似的分段线性函数在 PINN 中用的比较少，因为其二阶导数为 0 且不连续。文献 [180] 中表示：在深度学习领域，ReLU 激活函数看起来是默认的激活函数。但是在 PINN 领域不存在一个默认的激活函数。ReLU 在 PINN 中的问题主要就是其二阶导数为 0。同时，由于偏微分方程存在各种不同的数值特征，PINN 对激活函数看起来比较敏感。图5.31显示的是不同的激活函数预测的流场结果 [44]，很明显 ReLU 类激活函数预测的结果并不合理。因此参考这些工作，在 PINN 领域使用 ReLU 函数看起来并不是主流。

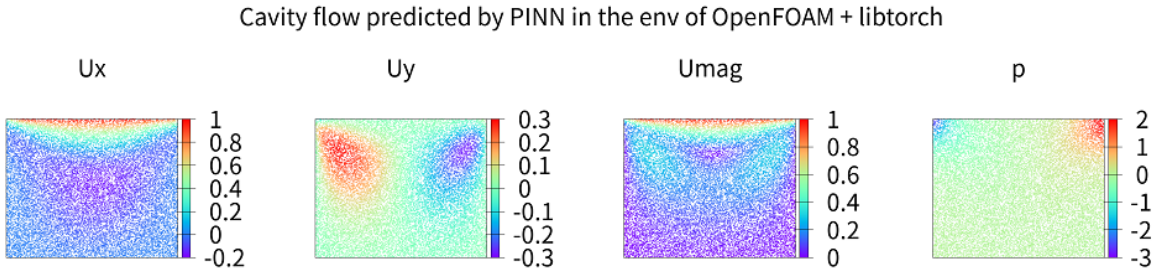


图 5.28: 在 OpenFOAM 环境下挂载 libtorch 进行的顶盖驱动流计算，$\nu = 0.1$。

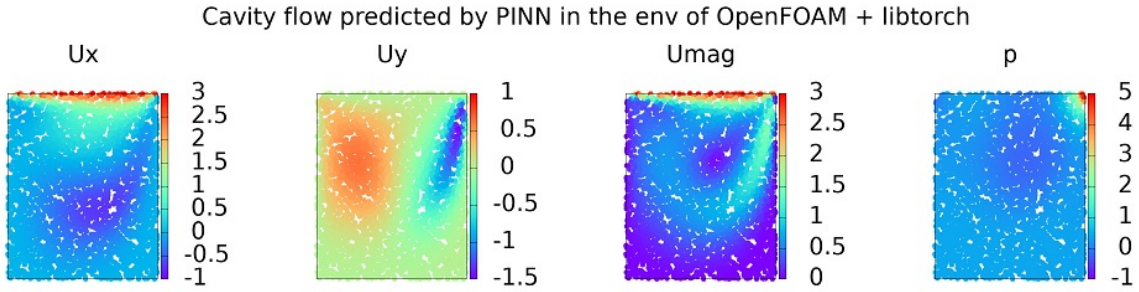


图 5.29: 在 OpenFOAM 环境下挂载 libtorch 进行的顶盖驱动流计算，$\nu = 0.005$。

因此，使用 PINN 来求解经典 CFD 能够计算的问题，这并不是 PINN 的强项。说直白点，比如圆柱扰流、后向台阶流这种明确定义的物理问题，还是需要使用经典 CFD。那

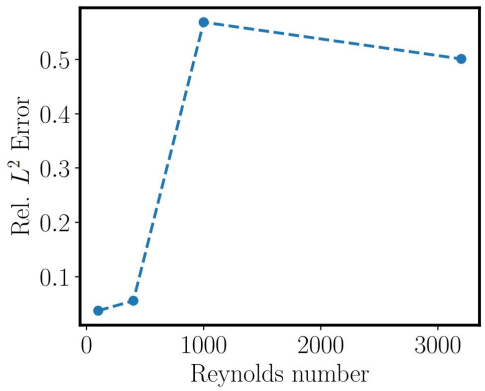


图 5.30: 通过 PINN 进行顶盖驱动流训练的时候，误差与雷诺数的关系 [180]。

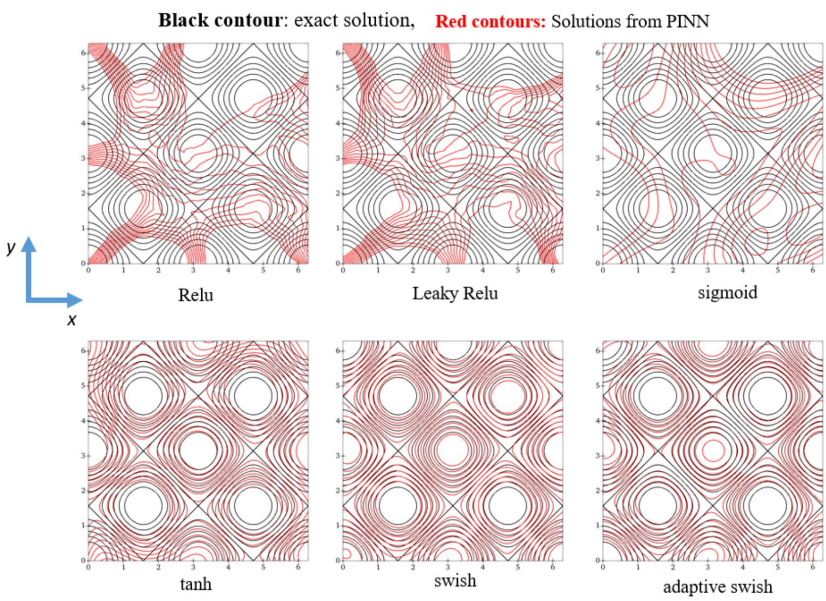


图 5.31: 通过 PINN 进行顶盖驱动流训练的时候，不同激活函数预测的结果 [44]。

么 PINN 在哪些方面可能会发挥更大的作用？PINN 主要的强项在于去发现，去处理边界条件不明确的流动问题。这些问题经典 CFD 是无法处理的。随着大数据的发展，如果能从大数据中学习出一些规律，如果是一些流动的规律。那么可能会发现一些新奇的东西。利用大数据来发现相关规律正是机器学习擅长的事情。例如目前一些公司已经通过机器学习来识别一些数据的突变（例如天气异常变化、人流的异常流动或者流场的异常间断等）。传统数值方法与数据驱动方法擅长的事情是俩种极端。在物理方程非常明确，数据量非常少的时候（例如边界条件），传统数值方法进行求解是非常有效的。然而一些情况下，存在了大量的数据，但相应的物理参数却大量缺失。例如在进行散热问题模拟的时候，可能并不明确热通量的边界条件，但却可以通过传感器来检测流场内部某些位置的温度信息。对于边界条件不明确的偏微分方程对其求解是病态的。常规数值方法也不能够利用监控点的温度作为边界条件。但是在这种情况下，PINN 可以将其处理为数据损失来进行求解，这也即为逆向 PINN 的原理。逆向 PINN 主要用来从数据来获得未知的流动参数。

2020 年，Raissi 等在 Science 发表研究观点类文章，标题为：“隐藏的流体力学，从流动视觉进行流场推理”[132]。从一定程度来讲可以看做一种全新的流场预测技术。与逆向 PINN 相同，隐藏流体力学的本质是给定一些流场数据，PINN 可以利用这些数据，来推测其他的流场。例如如果给定一系列电场的数据，同时借用麦克斯韦方程，那么就可以推理出磁场。如图5.32所示，Raissi 等从达芬奇的名画中吸取思想，首先生成一些满足 NS 方程的浓度场结果，随后通过隐藏流体力学，可以推理出相关的速度场、压力场以及浓度场。图5.33所示的是通过 PINN 来预测流场以及压力场 [76]。因为无法获取到准确的边界条件信息，这个问题通过经典 CFD 方法是无法处理的。通过将检测到的温度场时间序列输入到 PINN，其可以预测相应的流场以及压力场。

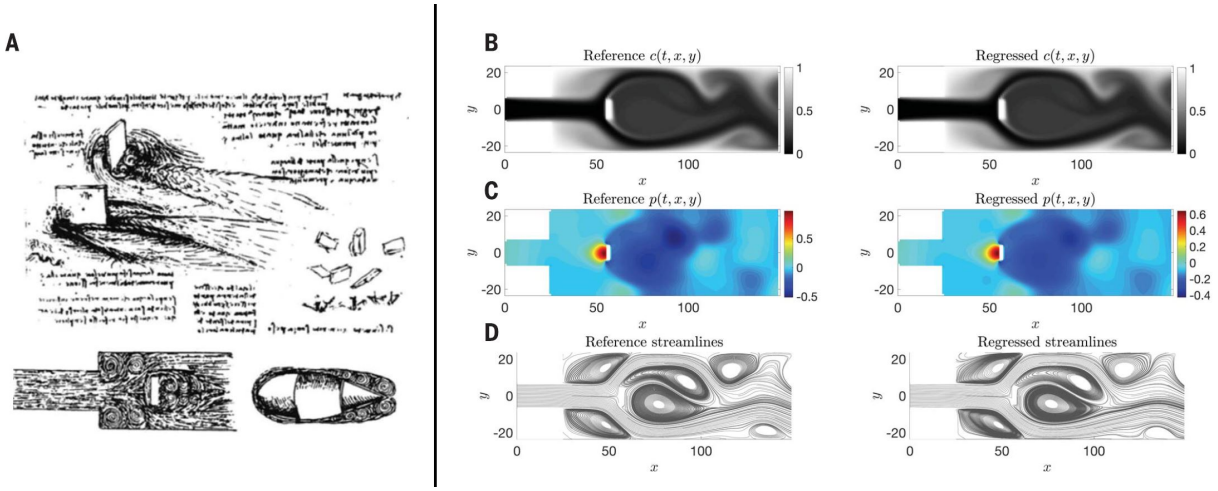


图 5.32: 隐藏流体力学预测的流场结果 [132]。

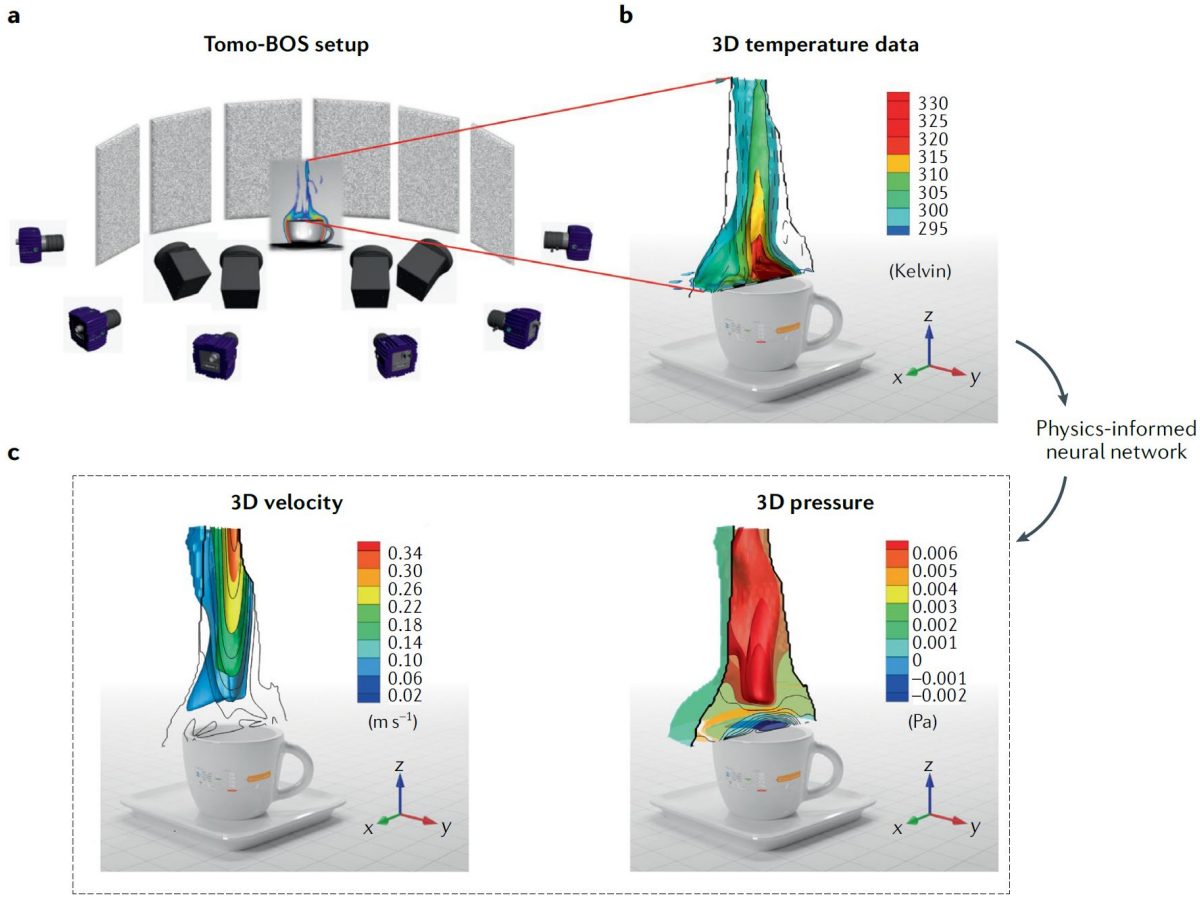


图 5.33: 隐藏流体力学预测的流场结果 [76]。

5.8 POD、DMD、降阶类方法

POD 全称 Proper Orthogonal Decomposition，也即本征正交分解方法。在一些情况下被称之为主成分分析。其可以理解为一种降阶类方法。在 OpenFOAM 中已经存在了一些降阶类方法[14]。经典 CFD 在计算流场结果的时候需要对偏微分方程进行求解。降阶类方法不需要考虑方程，从数据出发来进行流场重组。因此降阶类方法可以认为是一种数据驱动的方法。但区别于机器学习，这一类方法有一大部分并不需要神经网络。POD 方法是最早的一种降阶类方法，最早被用于湍流模式分析 [109]。在 POD 方法中，可以理解为将流场结果全部输入到程序，形成一个大矩阵。然后寻找这个大矩阵里面最重要的数据。其他的贡献比较小的就忽略掉。在这里通过数学的方法来解释会更加简单。相关算法主要来自于 [27]。

假定要对 2 个网格点的 u_a^t, u_b^t 做 POD（其中 u 表示波动速度，省略了 $'$ 符号）。同时假定存在 3 个时间点的数据，首先组建一个场，记为

$$\mathcal{U} = \begin{pmatrix} u_a^1 & u_b^1 \\ u_a^2 & u_b^2 \\ u_a^3 & u_b^3 \end{pmatrix} \tag{5.18}$$

针对 $\mathcal{U}$ 被称之为快照（snapshot）矩阵。快照矩阵也可以写成其转置的形式。其用来构建其协方差矩阵[15]：

$$\mathcal{C} = \frac{1}{3-1}\mathcal{U}^T \cdot \mathcal{U} = 0.5 \begin{pmatrix} (u_a^1)^2 + (u_a^2)^2 + (u_a^3)^2 & u_a^1 u_b^1 + u_a^2 u_b^2 + u_a^3 u_b^3 \\ u_a^1 u_b^1 + u_a^2 u_b^2 + u_a^3 u_b^3 & (u_b^1)^2 + (u_b^2)^2 + (u_b^3)^2 \end{pmatrix} \tag{5.19}$$

可见 $\mathcal{C}$ 为一个对称矩阵。$\mathcal{C}$ 的阶数为 2，与空间的点坐标数量相等（如果写成 $\mathcal{U}$ 的转置，那么阶数为 3）。因此 $\mathcal{C}$ 也可以理解为将快照矩阵进行了时间的平均（如果写成 $\mathcal{U}$ 的转置，那么可以理解为空间的平均）。$\mathcal{C}$ 的非对角线系数表示 u_a, u_b 两个网格点的波动速度的相关性。如果非对角线系数为 0，则表示完全不相关。由于 $\mathcal{C}$ 为一个对称矩阵，因此其一定可以对角化，且可以表示为：

$$\mathcal{C} = \Phi \Lambda \Phi^{-1} \tag{5.20}$$

取 Φ 的列向量，其即为 $\mathcal{C}$ 的特征向量，由于 $\mathcal{C}$ 为一个对称矩阵，因此特征向量是正交的。有：

$$\Phi^{-1} = \Phi^T \tag{5.21}$$

在有了 Φ 之后，现在定义另外一个矩阵 $\mathcal{A}$：

$$\mathcal{A} = \mathcal{U} \cdot \Phi \tag{5.22}$$

[14] 参考 ESI-OpenCFD 团队版本的 incompressible/pimpleFoam/laminar/cylinder2D 算例

[15] 其中的 3-1 中的 3 表示一共 3 个时间序列。也是一种时间的平均。

借助于方程(5.21)，有

$$\mathcal{U} = \mathcal{A} \cdot \Phi^{-1} = \mathcal{A} \cdot \Phi^{T} \tag{5.23}$$

将其展开有：

$$\mathcal{U} = \begin{pmatrix} \mathcal{A}_a^1 & \mathcal{A}_b^1 \\ \mathcal{A}_a^2 & \mathcal{A}_b^2 \\ \mathcal{A}_a^3 & \mathcal{A}_b^3 \end{pmatrix} \cdot \begin{pmatrix} \Phi_{11} & \Phi_{21} \\ \Phi_{12} & \Phi_{22} \end{pmatrix} = \begin{pmatrix} \mathcal{A}_a^1\Phi_{11} & \mathcal{A}_b^1\Phi_{12} \\ \mathcal{A}_a^2\Phi_{11} & \mathcal{A}_b^2\Phi_{12} \\ \mathcal{A}_a^3\Phi_{11} & \mathcal{A}_b^3\Phi_{12} \end{pmatrix} + \begin{pmatrix} \mathcal{A}_a^1\Phi_{21} & \mathcal{A}_b^1\Phi_{22} \\ \mathcal{A}_a^2\Phi_{21} & \mathcal{A}_b^2\Phi_{22} \\ \mathcal{A}_a^3\Phi_{21} & \mathcal{A}_b^3\Phi_{22} \end{pmatrix}$$

$$\begin{pmatrix} \mathcal{A}_a^1 \\ \mathcal{A}_a^2 \\ \mathcal{A}_a^3 \end{pmatrix} \cdot \begin{pmatrix} \Phi_{11} & \Phi_{21} \end{pmatrix} + \begin{pmatrix} \mathcal{A}_b^1 \\ \mathcal{A}_b^2 \\ \mathcal{A}_b^3 \end{pmatrix} \cdot \begin{pmatrix} \Phi_{12} & \Phi_{22} \end{pmatrix} \tag{5.24}$$

将方程(5.24)改写为：

$$\mathcal{U} = \mathcal{U}_{\mathcal{A}}^1 + \mathcal{U}_{\mathcal{A}}^2 = \begin{pmatrix} \mathcal{A}_a^1 \\ \mathcal{A}_a^2 \\ \mathcal{A}_a^3 \end{pmatrix} \cdot \begin{pmatrix} \Phi_{11} & \Phi_{21} \end{pmatrix} + \begin{pmatrix} \mathcal{A}_b^1 \\ \mathcal{A}_b^2 \\ \mathcal{A}_b^3 \end{pmatrix} \cdot \begin{pmatrix} \Phi_{12} & \Phi_{22} \end{pmatrix} \tag{5.25}$$

方程(5.25)可以分解为 2 个矩阵 $\mathcal{U}_{\mathcal{A}}^1$ 和 $\mathcal{U}_{\mathcal{A}}^2$ 的加和。这两个矩阵也即 POD 的模态。如果进一步的发现 $\mathcal{U}_{\mathcal{A}}^1$ 的贡献远远大于 $\mathcal{U}_{\mathcal{A}}^2$ 的话，那么 $\mathcal{U}_{\mathcal{A}}^2$ 就可以被忽略。即实现一种维度上的减少。在实际使用中，很少针对 2 个网格点来进行分析而是全场。假若存在 100 个网格点，那么最后 POD 分解的 $\mathcal{U}$ 可以存在 100 个维度。在 POD 中一般仅仅取其中的三五个维度即可。在这种情况下维度的降低是非常有效的。另外，之所以方程(5.25)可以分解为 2 个矩阵而不是 3 个矩阵，这是因为 $\mathcal{C}$ 进行的是时间的平均，最终留下来的是空间维度，因此 $\mathcal{U}_{\mathcal{A}}^1$ 和 $\mathcal{U}_{\mathcal{A}}^2$ 表示的是 2 个空间模态。方程(5.25)右侧的 $\mathcal{A}_a^1$ 等系数也可以理解为时间系数。

上述操作的过程分解出来的 POD 的模式也可以理解为不同空间点的贡献大小。如果取第一个模态，那么说明 a 点的贡献较大。如果在最开始，将快照矩阵写成转置的形式如：

$$\mathcal{U} = \begin{pmatrix} u_a^1 & u_a^2 & u_a^3 \\ u_b^1 & u_b^2 & u_b^3 \end{pmatrix} \tag{5.26}$$

有协方差矩阵：

$$\mathcal{C} = \frac{1}{2-1}\mathcal{U}^T \cdot \mathcal{U} \tag{5.27}$$

可以发现在这种情况下 $\mathcal{C}$ 是一个 3 阶矩阵。最后经过计算后，快照矩阵可以分分解为 3 个分量：

$$\mathcal{U} = \mathcal{U}_{\mathcal{A}}^1 + \mathcal{U}_{\mathcal{A}}^2 + \mathcal{U}_{\mathcal{A}}^3 \tag{5.28}$$

在这种情况下，POD 进行的是空间的平均，$\mathcal{A}$ 表示的是时间系数。在一般使用过程中，取决于网格点与时间序列的大小。在 CFD 计算中，通常网格点的数量要大于时间序列的数

量，因此一般采用第二个方法。在实验中，一般采样点的数量要小于时间序列的数量，因此采用第一种方法。图5.34显示的使用 POD 方法针对圆柱扰流计算的前 6 个模态[16]。另外，POD 方法也可以不定义协方差矩阵而直接采用奇异值分解（SVD）来进行。在这里不做介绍。

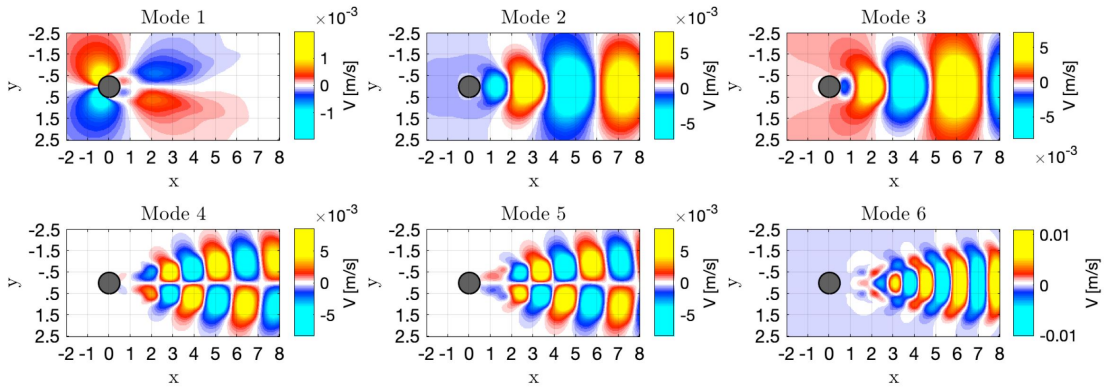


图 5.34: POD 方法针对圆柱扰流计算的前 6 个模态。

DMD 的全称是 Dynamic Mode Decomposition，也即动态模态分解技术。类似 POD 也可以用于降维。DMD 相对 POD 更倾向于处理时间序列数据。如图5.35所示，DMD 需要收集一系列的时间数据 x_t，然后构建矩阵来生成相应的模态。DMD 存在若干个不同的实施方法，目前来看通过 SVD 算法来实施是比较主流的。下面来讨论基于 SVD 算法的 DMD 实施过程。

如图5.35所示，首先基于 $1-t$ 的全场的流场数据，将其变成列数据，并将 $x_1, x_2, ..., x_t$ 组合成矩阵 $\mathbf{X}$：

$$\mathbf{X} = [x_1, x_2, ..., x_t] = \begin{pmatrix} u_a^1 & u_a^2 & u_a^3 \\ u_b^1 & u_b^2 & u_b^3 \end{pmatrix} \tag{5.29}$$

这一步与 POD 的构建快照矩阵的思想其实是一样的。然后进一步的通过全场的瞬态数据，构建另外一个矩阵 $\mathbf{X}'$：

$$\mathbf{X}' = [x_2, x_3, ..., x_{t+1}] = \begin{pmatrix} u_a^2 & u_a^3 & u_a^4 \\ u_b^2 & u_b^3 & u_b^4 \end{pmatrix} \tag{5.30}$$

在这里要注意，按照这样排布的快照矩阵是一个非常瘦高的矩阵，也就是行数远远大于列数。DMD 的思想是寻找一个 $\mathcal{A}$，有：

$$\mathbf{X}' = \mathcal{A}\mathbf{X} \tag{5.31}$$

然而这个 $\mathcal{A}$ 一般特别大。所以要采取一种迂回的战术。首先将 $\mathbf{X}'$ 进行奇异值分解：

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5.32}$$

[16]图片来自于 sehenaoga.github.io/projects

其中 $\mathbf{U},\mathbf{V}$ 都是正交矩阵，有：

$$\mathbf{U}\mathbf{U}^T=\mathbf{I},\mathbf{V}\mathbf{V}^T=\mathbf{I} \tag{5.33}$$

假定网格点是 m，在网格点很多的情况下，$\mathbf{U}$ 是一个非常大 $m\times m$ 的矩阵。假定时间序列共有 n 个，则 $\mathbf{\Sigma}$ 为一个 $m\times n$ 的对角阵，$\mathbf{V}$ 为一个 $n\times n$ 的矩阵。$\mathbf{V}$ 的元素数量要小于 $\mathbf{U}$[17]。得益于 SVD 分解的特性，有：

$$\mathbf{X}^{'}=\mathcal{A}\mathbf{X}=\mathcal{A}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{5.35}$$

$$\mathbf{X}^{'}\mathbf{V}=\mathcal{A}\mathbf{U}\mathbf{\Sigma} \tag{5.36}$$

$$\mathbf{X}^{'}\mathbf{V}\mathbf{\Sigma}^{-1}=\mathcal{A}\mathbf{U} \tag{5.37}$$

$$\mathbf{U}^T\mathbf{X}^{'}\mathbf{V}\mathbf{\Sigma}^{-1}=\mathbf{U}^T\mathcal{A}\mathbf{U} \tag{5.38}$$

定义 $\tilde{\mathcal{A}}$：

$$\tilde{\mathcal{A}}=\mathbf{U}^T\mathcal{A}\mathbf{U}=\mathbf{U}^{-1}\mathcal{A}\mathbf{U} \tag{5.39}$$

可见 $\tilde{\mathcal{A}}$ 和 $\mathcal{A}$ 二者为相似矩阵。二者具有相同的特征值。在这里尚未进行任何的简化，因为 $\mathbf{U}$ 比较大，各种矩阵操作还是相对复杂。在 DMD 方法中，对于 $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ 只保留其中的 r 阶，其中 r 表示 $\mathbf{\Sigma}$ 的阶数，也即 $\mathbf{U}$ 进行了截断[18]。在这种情况下，有：

$$\tilde{\mathcal{A}}=\left(\mathbf{U}^{-1}\right)_{m\times m}\left(\mathcal{A}\right)_{m\times m}\left(\mathbf{U}\right)_{m\times m}\rightarrow\left(\mathbf{U}^{-1}\right)_{r\times m}\left(\mathcal{A}\right)_{m\times m}\left(\mathbf{U}\right)_{m\times r} \tag{5.41}$$

因此 $\tilde{\mathcal{A}}$ 也是一个 $r\times r$ 矩阵，其相对 $\mathcal{A}$ 要小很多。如果在分析中，发现只保留更小的列数，例如只保留 5 行的话，那么 $\tilde{\mathcal{A}}$ 会非常小。

然后对 $\tilde{\mathcal{A}}$ 特征分解：

$$\tilde{\mathcal{A}}\mathbf{W}=\mathbf{W}\boldsymbol{\lambda} \tag{5.42}$$

其中 $\boldsymbol{\lambda}$ 表示 DMD 的特征值。相当于 POD 中的 $\mathcal{A}$。定义 $\mathbf{\Phi}$：

$$\mathbf{\Phi}=\mathbf{X}^{'}\mathbf{V}\mathbf{\Sigma}\mathbf{W} \tag{5.43}$$

$\mathbf{\Phi}$ 为一个 $r\times r$ 甚至更小的矩阵。其即为 DMD 的模态。在有了这些变量后，即可预测某 $k\Delta t$ 时间步下的流场：

$$x^{k\Delta t}=\mathbf{\Phi}\boldsymbol{\lambda}^k\mathbf{b}_0 \tag{5.44}$$

[17] 下面是一个真实的奇异值分解范例，主要用来演示 3 个矩阵的形状：

$$\begin{bmatrix}2&4\\1&3\\0&0\\0&0\end{bmatrix}\approx\begin{bmatrix}-0.82&-0.58&0&0\\-0.58&0.82&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}5.46&0\\0&0.37\\0&0\\0&0\end{bmatrix}\begin{bmatrix}-0.40&-0.91\\-0.91&0.40\end{bmatrix} \tag{5.34}$$

[18]

$$\mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}\left(\mathbf{V}^T\right)_{n\times n}\rightarrow\mathbf{U}_{m\times r}\mathbf{\Sigma}_{r\times r}\left(\mathbf{V}^T\right)_{r\times n} \tag{5.40}$$

$$\mathbf{b}_0 = \mathbf{\Phi}^* x_1 \tag{5.45}$$

其中 $\mathbf{\Phi}^*$ 为 $\mathbf{\Phi}$ 的伪逆矩阵。最后总结 DMD 的计算方法：

1. 通过方程(5.32)进行奇异值分解；
2. 对 $\mathbf{U}$ 进行截断，通过方程(5.39)求解 $\tilde{\mathcal{A}}$；
3. 通过方程(5.42)求解 $\tilde{\mathcal{A}}$ 的特征值与特征向量；
4. 通过方程(5.43)求解 $\mathbf{\Phi}$；
5. 通过方程(5.44)进行时间步推进进行流场预测；

可见，POD 以及 DMD 都不需要对偏微分方程进行求解，既可以进行流场预测。因此这两类方法是数据驱动类方法。另一方面，常规的这一类方法并不需要神经网络。在植入过程中，原生的求解器既可以进行，并不需要外挂机器学习相关程序库。

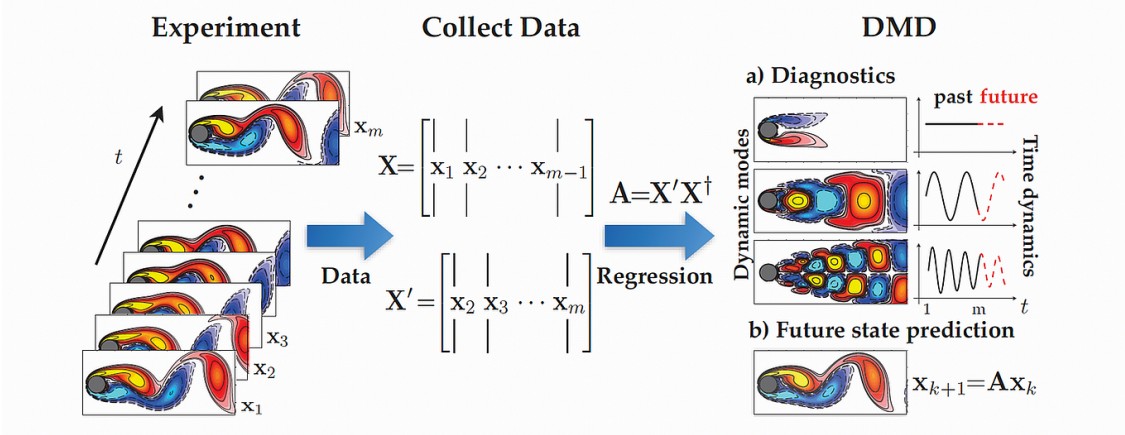


图 5.35: DMD 实施方法 [81]

5.9 PyTorch 与 libtorch

所有人讨论机器学习的时候，都在讨论 PyTorch。**PyTorch** 是一个提供 python 接口的张量计算库。然而很明确的在 PyTorch 官网主页，还会发现一个 libtorch 的东西，如图5.36。然而很多人却连 libtorch 听说都没听说过。这可能是行业差异导致的。在互联网领域，可以说基本人人都用 python。在高性能计算领域，20 年前基本人人都用 Fortran，近 20 年人人都用 C++。由于机器学习在早期并没有用于高性能计算而是图像识别，这些技术人员来自于互联网公司。比如 PyTorch 是 Meta 公司开发的。互联网公司用 python 比较顺手，因此早期的框架就基于 python 来写了 C++ 接口，叫 PyTorch。直到最近几年大家讲机器学习应用到了高性能计算。所以开发人员直接开发了 C++ 接口。另一方面，对

于传统的 CFD 开发人员，使用 C++ 的占大多数。乍眼一看，libtorch 对于 C++ 用户更为友好。因此有必要了解 libtorch 与 PyTorch 的关系。

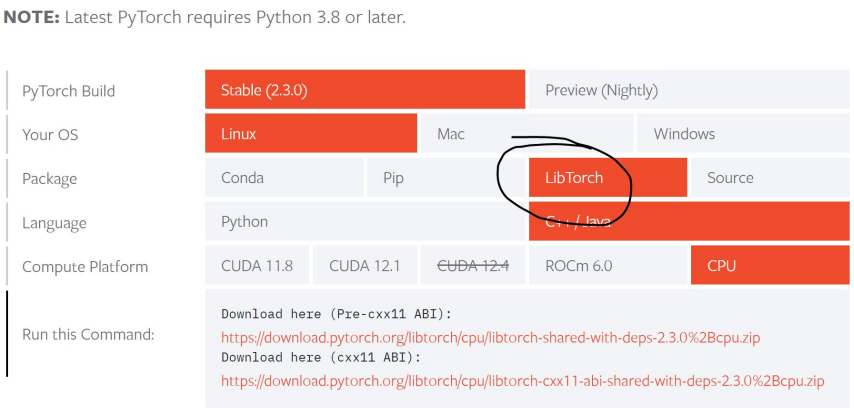


图 5.36: PyTorch 与 libtorch。

据 PyTorch 开发人员[19]表示：PyTorch 是基于 Torch 的一个融合了 python 语言的开源库。Torch 则是一个主要基于 C 语言的同时借助 Lua 语言做接口的开源库。PyTorch 官方还表示[20]：不能把 PyTorch 简单理解为核心为 C++ 并使用 python 作为了接口语言的计算库。但 PyTorch 的计算密集型任务实现为通过 C++ 来完成。另一方面，Meta 公司为 PyTorch 做 C++ 前端接口的 Goldsborough 表示[21]：

> *"Pytorch 的后端就是 C++，我过去六个月的工作就是为了这个后端的 C++ 程序同样的做出一个 C++ 的 API，也就是 libtorch"*

因此，**libtorch** 是可以实现与 PyTorch 同样功能的，提供 C++ 接口的计算库。既然与 PyTorch 具有同样的功能，那么就存在一个选择问题。官方对同一个软件包，缺提供了两种接口做出这样的解释[22]：

> *"我们提供一个 C++ 接口的 libtorch，理念是虽然 python 作为前端非常棒，应该尽可能的去使用。但是在一些环境下，对于性能的要求使得 python 接口不太合适。举例来说，python 对于高性能计算、要求极低延迟的一些算法表现比较差。我们发布的基于 C++ 接口的 libtorch 就是为了这些使用环境而适配的。"*

> *"郑重警告：Python is not necessarily slower than C++! PyTorch 的计算密集型任务都是通过 C++ 来编写。因为做计算大部分的时间都花费在了数值操作，而这部分恰恰是通过 C++ 来完成的。因此如果你们喜欢使用 Python，我们建议使*

[19] https://soumith.ch/posts/2021/02/growing-opensource/
[20] https://github.com/pytorch/pytorch
[21] 请在 youtube 搜索 "Machine Learning in C++ with PyTorch"。
[22] https://pytorch.org/cppdocs/frontend.html

用 *PyTorch*。然而如果你喜欢 *C++*，或者由于一些特定目的需要写一些 *C++* 程序，那么可以使用 *libtorch*。”

那么对于初学者,是使用 PyTorch 还是 libtorch?应该从几个方面来考虑。首先 PyTorch 的资料肯定要比 libtorch 的多。但反过来用户还需要学习 python 语言。另外，如果用户打算采用 OpenFOAM 来做相关交叉研究。因为 OpenFOAM 本身就是 C++ 环境，那么在 OpenFOAM 中直接挂 libtorch 看起来是水到渠成的。笔者没有严格的对比 libtorch 与 PyTorch 的速度，但是从 PyTorch 官方的描述来看，OpenFOAM 与 libtorch 混编对于高性能计算应该是完美的。并且，这种方式不需要附加 python 语言，会 C++ 即可。

5.10 数据驱动 CFD 侧重点

就像经典 CFD 与流体力学一样，虽然名词看起来很像，但是二者的研究重点完全不一样。甚至属于完全不同的研究领域。如果将机器学习与 CFD 相结合，同时也要注意研究方向的问题，不要跑偏。

专业的机器学习研究人员研究重点在于各种优化函数（比如上文中的梯度下降方法），很多的机器学习的相关研究在对比 Adam、L-BFGS、Adam+L-BFGS 各种优化函数的效率。在 libtorch 中也植入了一些优化函数。还有一些人专门来设计各种神经网络，比如从全连接神经网络演化到卷积神经网络，再到残差神经网络以及 Unet。这些属于传统的机器学习领域的人的研究范围，他们不需要任何的 CFD 以及偏微分方程的知识。更明显的，机器学习领域一个新的技术的首创，一般都是用来图像识别甚至玩游戏。感兴趣的读者可以在这里来看一下机器学习的研究内容：卷积神经网络 [89]、R-CNN[134]、Transformer[177]、深度 Q 网络 [122]，在这里仅列举一些，很明显正统机器学习领域的研究内容跟 CFD 毫无关系。反过来，这些相关技术被提出来之后，可以与 CFD 去深度结合。这就类似 CFD 领域中的多重网格求解器，多重网格求解器最开始并没有应用于 CFD 计算，但是目前是 CFD 领域众人皆知的算法。具体的如何将机器学习与 CFD 相结合，请参考前述章节。

有一个特殊的技术是 PINN。PINN 在提出的时候就是用来求解偏微分方程，在原始文章里面也用来求解了 CFD 相关问题 [131]。很明显 PINN 在提出的时候就枪打 CFD。那么如何看待 PINN 与 CFD 的关系？CFD 在 PINN 之前都是采用有限体积、有线差分等方法来进行。PINN 是一种完全的基于自动微分的方法来求解 CFD。因此可以把 PINN 看做一个全新的 CFD 求解方法。PINN 与有限体积法等没有任何关系。因此如果有限体积法研究人员打算使用 PINN 求解 NS 方程，这等于是完全换了个方向。一个会使用 PINN 求解 NS 方程的研究人员，很有可能不了解任何有限体积法的知识。一个有限体积法研究人员，非常有可能完全不熟悉 PINN。

一个 CFD 研究人员要与深度学习相结合，本质还是研究 CFD，在此基础之上，借助现有的机器学习模型来对 PDE 进行求解、或预测、或加速。数据驱动 CFD 是一个 CFD

的子方向。数据驱动 CFD 不会替代 CFD 也不会替代有限体积法，在以后的相当长的时间里会共存。另一方面，数据驱动 CFD 可以看做一个交叉学科，相关科研人员必然是机器学习与 CFD 都熟悉一些才更好的开展工作。比如在使用卷积神经网络进行全流场预测的时候，研究人员需要思考如何设计卷积层以及池化层，以及如何让机器训练的效率最好。在实操方面，科研人员不仅仅需要熟悉 CFD 代码（如 OpenFOAM），也需要熟悉机器学习的代码（如 libtorch 或 tensorflow）。这无外乎对科研人员的知识增加了更高的要求。

第六章　OpenFOAM 算法速查

6.1　OpenFOAM 中的非结构网格

在第4.11.2节的讨论中可以看出，结构网格由于存在编号易于实施。然而非结构网格由于存在不同的网格单元类型，且不存在规则的编号体系，在植入的时候要复杂很多。OpenFOAM 中的非结构网格定义成熟老练，现对其进行介绍总结。

OpenFOAM 中的非结构网格需要定义一系列的必要信息。主要有：

- `points`：存储网格顶点的位置矢量；
- `faces`：存储哪些顶点构成一个网格面；
- `owner`：存储每个面对应的主单元；
- `neighbour`：存储每个面对应的临单元；
- `boundary`：存储网格的边界信息构成；

这 5 个文件需要用户提供给 OpenFOAM，其构成了 CFD 的非结构网格基本元素。OpenFOAM 在这 5 个网格文件基础上，将其“理解”为非结构网格。因此，若使用第三方软件生成网格数据，必须存在的就是这五个文件。

在代码底层，OpenFOAM 中的网格主要存储在下面三个类型中：

- `fvMesh`：跟有限体积法相关的网格相关量，比如离散需要用到的网格体积、网格面矢量、网格主单元临单元等；
- `polyMesh`：网格的几何属性相关量，比如拓扑变形、网格分区、一些功能函数（如判断点在那个网格内）；
- `primitiveMesh`：网格基本操作。比如网格质量检查、网格单元编号、网格面心、质心、某网格单元相邻单元编号等；

三者为继承关系。`primitiveMesh` 最为底层，`polyMesh` 次之，`fvMesh` 位于顶层。在求解器层面，通过 `fvMesh` 来构建用户角度理解的网格。也就是说，我们通常理解的网格，具体的网格类型就是 `fvMesh`。例如在求解器层面会有下面的代码：

```
fvMesh mesh
(
    IOobject
    (
        "mesh",
        runTime.timeName(),
        runTime,
        IOobject::MUST_READ
    )
);
```

上述代码创建 mesh 类型，并在求解器层面可以调用。一些求解器层面可调用的函数可以在 fvMesh 类型里面找到。比如：

- 网格的体心 mesh.C()、面心 mesh.Cf()、网格单元体积 mesh.V()、面矢量 mesh.Sf() 等数据；
- 网格与离散矩阵的地址对应关系，网格的 owner()，网格的 neighbour() 等数据；

在 mesh 创建的时候，会同时创建 polyMesh 类型，也会创建更底层的 primitiveMesh 类型。上文提到的 points，faces，owner，neighbour 以及 boundary 均存储在 polyMesh 类型。例如在用户执行求解器的时候，代码会读取算例文件夹/constant/polyMesh 下面的 points，faces，owner，neighbour 以及 boundary，并将相关信息存储在 polyMesh 类型中。

6.1.1 polyMesh 中的 points

OpenFOAM 算例中的 constant/polyMesh/points 文件包含了非结构网格的所有的顶点的数据。如图6.1所示，points[1]文件用来存储网格顶点的位置矢量。点数据由一系列的矢量数据构成。因此对于一套非结构网格，首先需要存储网格的所有的网格单元顶点的信息。OpenFOAM 中的 points 就是一个 vector（矢量）[2]。pointIOField 就是一个 vectorIOField（矢量场）[3]。下面是一个真实的 points 文件示例：

```
/*---------------------------------*- C++ -*------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
```

[1]英文常称之为 vertices
[2]point.H
[3]pointIOField.H

```
7  \*-------------------------------------------------------------*/
8  FoamFile
9  {
10     format      ascii;
11     class       vectorField;
12     location    "constant/polyMesh";
13     object      points;
14 }
15 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   * * * //
16
17 18
18 (
19 (0 0 0)//第0个顶点
20 (0.05 0 0)//第1个顶点
21 (0.1 0 0)//第2个顶点
22 (0 0.05 0)//以此类推
23 (0.05 0.05 0)
24 (0.1 0.05 0)
25 (0 0.1 0)
26 (0.05 0.1 0)
27 (0.1 0.1 0)
28 (0 0 0.01)
29 (0.05 0 0.01)
30 (0.1 0 0.01)
31 (0 0.05 0.01)
32 (0.05 0.05 0.01)
33 (0.1 0.05 0.01)
34 (0 0.1 0.01)
35 (0.05 0.1 0.01)
36 (0.1 0.1 0.01)
37 )
```

如图6.1所示，其中 18 表示网格存在 18 个点。点从上至下进行编号。每一行对应的点的空间坐标。网格顶点的标号可以更改，比如上面代码中的低 28 行可以和任意一行进行更换，但如果进行更换，相应的其他的下文讨论的 faces 等都需要对应的进行替换。

6.1.2 polyMesh 中的 faces

除了点数据之外，OpenFOAM 需要读取面数据。如图6.2所示，这些面可以通过将点文件进行连接而形成。OpenFOAM 中的 constant/polyMesh/faces 文件包含了所有的面数

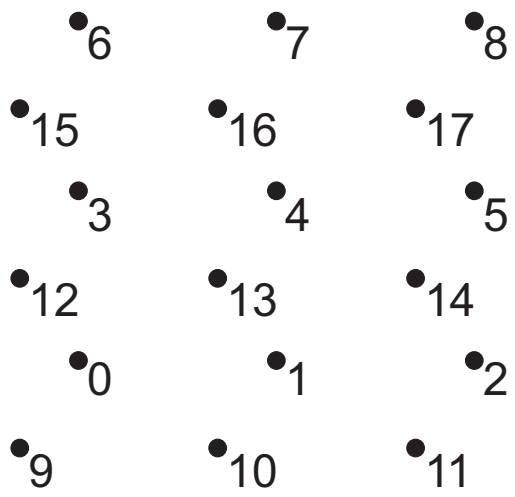


图 6.1: polyMesh 的 points 范例。

据。下面是一个 faces 文件实例：

```
/*--------------------------------*- C++ -*------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       faceList;
    location    "constant/polyMesh";
    object      faces;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

20
(
4(1 4 13 10)//首先从内部面开始，这一行表示第0个面，该面存在4个边，由4个顶点构成
4(3 12 13 4)//这一行表示第1个面（内部面）
4(4 13 14 5)//这一行表示第2个面（内部面）
4(4 7 16 13)//这一行表示第3个面（内部面）
4(6 15 16 7)//这一行表示第4个面（边界面）
4(7 16 17 8)//这一行表示第5个面（边界面）
```

```
4(0 9 12 3)
4(3 12 15 6)
4(2 5 14 11)
4(5 8 17 14)
4(0 1 10 9)
4(1 2 11 10)
4(0 3 4 1)
4(3 6 7 4)
4(1 4 5 2)
4(4 7 8 5)
4(9 10 13 12)
4(12 13 16 15)
4(10 11 14 13)
4(13 14 17 16)
)
```

如图6.2所示，其中 20 表示网格存在 20 个面。首先排布内部面，再次排布边界面。每一行前面的数字 4 表示这个面包含 4 个点，后面括号内的数字表示这四个点连接成的一个网格面。面从上至下进行编号（第 0 行表示第 0 个面，第 2 行表示第 1 个面，以此类推）。

上面的代码中，每个网格面上具有 4 个点，点的排布顺序决定了面的法向，例如图6.2中的 10 1 4 13 连接的面的指向遵循右手定则（也即为第 0 个内部面）。面矢量的大小通过方程(6.2)来计算。因此，某一个面上，点的排序的改动并不是任意的。如果将 10 1 4 13 排序改成 1 4 13 10，则没有问题。但若改为 13 4 1 10，则面矢量的方向发生改变（后续的主网格以及临网格也要对应着改变，否则报错）。

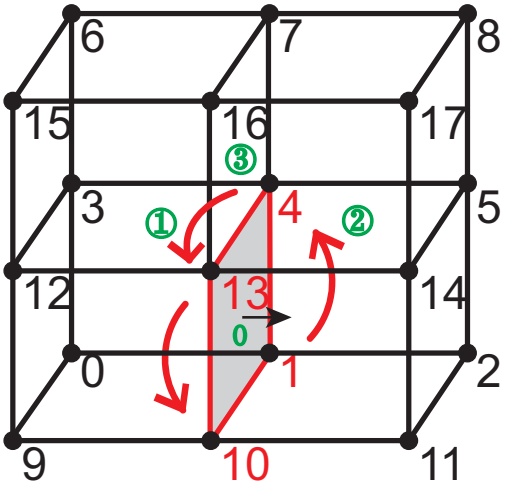


图 6.2: polyMesh 的 faces 范例。其中 faces 文件第一行的 4(1 4 13 10) 定义了图中的内部面，其中的面矢量方向也被定义（右手定则）。绿色的编号表示 4 个内部面定义，边界面未明示。

网格面的质心计算很重要[4]。考虑图6.3中一个最简单的三角形，面的质心的计算非常简单，其计算公式为[5]：

$$\mathbf{C}_f = \frac{1}{3}(\mathbf{p}_0 + \mathbf{p}_1 + \mathbf{p}_2) \tag{6.1}$$

其中 $\mathbf{C}_f$ 表示三角形面的质心，其为一个矢量，$\mathbf{p}_i, i = 0, 1, 2$ 表示三角形的三个顶点。三角形的面矢量可以通过下式计算：

$$\mathbf{S}_f = 0.5(\mathbf{p}_1 - \mathbf{p}_0) \times (\mathbf{p}_2 - \mathbf{p}_0) \tag{6.2}$$

其中 $\mathbf{S}_f$ 表示三角形的面矢量。这个公式也表示面矢量的方向遵循右手法则。三角形的面积则为面矢量 $\mathbf{S}_f$ 的模。方程(6.2)是遵循右手定则计算的方法（虽然可以用左手定则来计算，但工程物理一般都使用右手定则）。

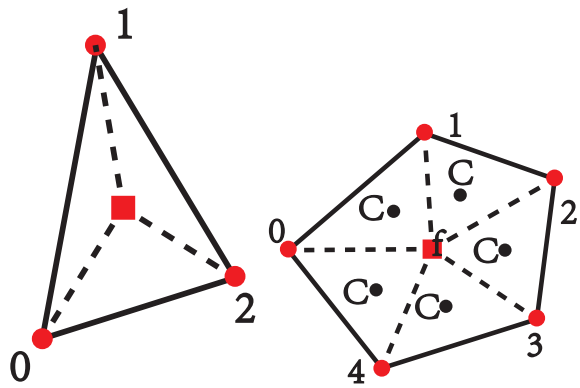


图 6.3: （左）红色方框就是图中三角形面的质心。（右）多边形分解。其中红色方框点的位置 **f** 通过公式(6.3)计算而来。

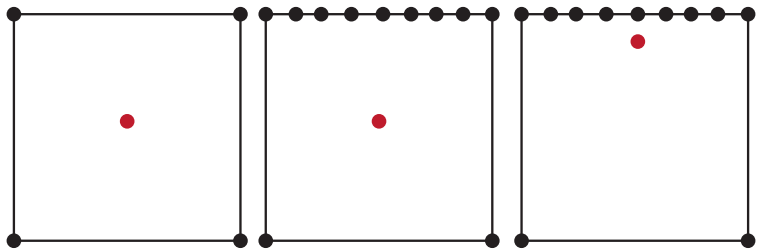


图 6.4: （左）标准四边形面的质心。（中）奇葩多边形通过公式(6.3)计算的面的质心。（右）奇葩多边形通过公式(6.4)计算的面的质心。

对于图6.3中所示任意的多边形，要获得其面的质心，首先需要对图6.3中的多边形进行分解成若干三角形[6]。三角形共有的顶点通过下式进行计算：

$$\mathbf{f} = \frac{\mathbf{p}_0 + \mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3 + \mathbf{p}_4}{5} \tag{6.3}$$

[4] Centroid 在这里翻译为质心，更多的讨论请参考链接

[5] primitiveMeshFaceCentresAndAreas.C

[6] 分解成其他形状也可以，但是三角形的质心最容易找。

分母中的 5 是表示多边形顶点的数量。这样，这个多边形就可以分解为 5 个三角形。对每个三角形求其质心可获得 $\mathbf{C}_0, ..., \mathbf{C}_4$，同时可以获得每个三角形的面矢量 $\mathbf{S}_0, ..., \mathbf{S}_4$。那么此多边形的面矢量，即为 $\mathbf{S}_0 + ... + \mathbf{S}_4$。此多边形面的质心即为

$$\mathbf{C}_f = \frac{\sum_{i=1} |\mathbf{S}_i| \mathbf{C}_i}{\sum_{i=1} |\mathbf{S}_i} \tag{6.4}$$

同时需要注意的是，对于多边形面的质心计算，公式(6.3)也是质心的一种计算方法。这种算法被通常被称之为几何平均法（arithmetic average）。但在 CFD 计算中，几何平均法计算的中心会有很大的误差。例如，对于图6.4中的奇葩多边形，使用公式(6.4)来计算质心可以保证其不受三角形分解的影响，然而使用公式(6.3)来计算则会导致偏移。最后我们总结面心的计算流程：

1. 通过几何平均法方程(6.3)估算一面心点；
2. 通过估算的面心点对面进行分解为三角形；
3. 求解三角形的质心；
4. 通过方程(6.4)计算面心；

6.1.3 polyMesh 中的 owner、neighbour

在网格的点、面被定义之后，`owner` 进一步定义所有网格面的主单元编号。这里的所有网格面指的是内部面以及边界面。每个面均存在主单元信息。`neighbour` 定义所有内部面的次单元编号（也被称之为临单元）。边界面不存在次单元。因此，存在多少个网格单元，`owner` 和 `neighbour` 内部定义的编号就不会大于这个值。假如存在 400 个网格单元，那么 `owner` 和 `neighbour` 内部的信息绝对不会出现大于 399 的编号[7]。

每一个面都对应一个 `owner` 信息，每一个内部面都对应一个 `neighbour` 编号。下面是一个 `owner` 文件实例：

```
/*--------------------------------*- C++ -*----------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------*/
FoamFile
{
```

[7]网格从 0 开始编号。400 个网格即从 0 到 399。如果出现大于 399 的编号，比如某个编号为 1023，那么求解器会认为一共存在 1024 个网格单元。

```
    format      ascii;
    class       faceList;
    location    "constant/polyMesh";
    object      owner;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

20
(
0
0
1
2
2
3
0
2
1
3
0
1
0
2
1
3
0
2
1
3
)
```

上面的 owner 隐含的信息为网格数量为 4（因为最大的网格单元编号为 3）。事实上，网格数量也是从 owner 以及 neighbour 中读取而来的[8]：

```
    label nCells = -1;

    forAll(owner_, facei)
    {
        nCells = max(nCells, owner_[facei]);
    }
```

[8] polyMeshInitMesh.C

```

    // The neighbour array may or may not be the same length as the owner
    forAll(neighbour_, facei)
    {
        nCells = max(nCells, neighbour_[facei]);
    }

    nCells++;
```

owner 编号有几个特点：

- 对于某个面，owner 编号一定小于 neighbour 编号[9]；
- owner 编号的第一行一定是从 0 开始，否则网格单元数量与标志不一致；
- 对于内部面，owner 编号一定是递增的；

下面是一个 neighbour 文件实例：

```
/*--------------------------------*- C++ -*------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  9
     \\/     M anipulation  |
\*---------------------------------------------------------------*/
FoamFile
{
    format      ascii;
    class       faceList;
    location    "constant/polyMesh";
    object      neighbour;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

4(1 2 3 3)
```

因为仅仅存在 4 个内部面，因此 neighbour 只需要定义 4 个值即可。因为只有 4 个值，因此仅仅构成了一行。

[9] primitiveMeshCheck.C: checkUpperTriangular() 函数

6.1.4 虚拟的 cells，体心，体积

OpenFOAM 里面的网格单元并没有明确的对其定义，也可以理解为 OpenFOAM 中的非结构网格并不需要读取 `cells` 数据（例如并不需要定义多少个网格，因为本身网格的数量已经通过 `ownner` 来定义）。换一种理解方法，对于结构网格，常规做法是定义若干的 `cells`，然后在做导数的时候，需要调取第 i 个网格以及第 i+1 个网格的数据。但是在 OpenFOAM 中，为了兼容非结构网格，不需要这样操作。而是通过上面讨论的面网格的连接性来进行，例如在做导数的时候，不需要调取第 i 个网格以及第 i+1 个网格的数据，在非结构网格中也不存在这种 i、i+1 的一一对应关系。因此在做导数的时候，需要调取面的 `own` 网格以及 `neighbour` 网格的数据。不仅仅是导数，如果需要做高斯积分，也只是需要通过网格连接性来进行。OpenFOAM 将 cells 定义为一系列的面包围的区域，OpenFOAM 中的 `cell` 就可以理解为一系列的面[10]，并可以实现一系列的功能，如通量加和、高斯积分等。对于一些比较重要的网格参数，如跟 `cells` 直接相关的体积以及体心，也不是从所谓的 `cells` 计算而来，而是从 `faces` 的定义中求出。

在这里可以思考一下，之所以需要 `cells` 的编号，是因为求解器需要调用一些定义在 `cells` 的数据，比如第 100 个网格点的速度，第 100 个网格单元的体积。对于速度与体积这两个变量，均为一个数组，因此可以将数组的第 100 个元素，理解为第 100 个网格单元的编号即可。例如，将网格单元体积理解为一个数组，其第 0 个元素，就是第 0 个网格单元的体积，其第 1 个元素，就是第 1 个网格单元的体积。因此，OpenFOAM 中 `cells` 的编号，其实就是各个场数组的编号。

不过再次重申，各个场数组的编号和网格编号是一一对应的关系。因此可以认为二者是等价的。但是更重要的，将 OpenFOAM 的 `cells`，考虑成一系列面包围的区域更好。在 OpenFOAM 算例文件中，基本的网格构成主要有 5 个文件（参考第6.1节），其中也没有 `cells` 的定义。

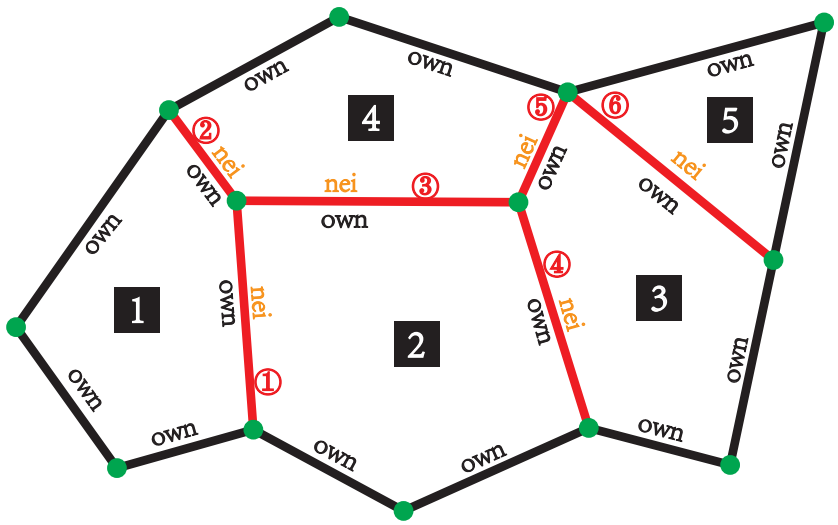
下面看怎么来计算网格单元体积。首先考虑几何平均法，虽然其不精准，但计算方法简单。在几何平均法中，网格单元的体心被定义为网格单元的面心加和除以网格单元的面数量[11]：

$$\mathbf{C}_c = \frac{\sum \mathbf{C}_{f,i}}{N_i} \tag{6.5}$$

其中 $\mathbf{C}_c$ 表示网格单元体心，$\mathbf{C}_{f,i}$ 表示网格单元的面心，N_i 表示网格单元的面的数量。在这里也可以看出，网格单元体心的计算，仅仅需要面数据。只要读取面数据，就可以计算网格单元的体心。实施方程(6.5)的时候，一种方法是对网格单元遍历，寻找每个网格单元对应的面，对面心进行加和并除以网格面数。OpenFOAM 并没有采用这一种方式。OpenFOAM 采用的是遍历面的方式。具体的算法如图6.5所示。在此进行详述。

[10]这在 primitiveMesh.H 中有明晰的体现。网格的基本信息如 point，face，owner，neighbour 都被定义为 Minimum mesh data，然而 `cells` 则被定义为 cell-faces 的连接性。

[11]primitiveMeshCellCentresAndVols.C


own
own
own

4. 其他面的 nei 的编号都不是 2，因此均无贡献；

5. 最终，网格单元体心第 2 个元素的值为 $(\mathbf{C}_{f,3}+\mathbf{C}_{f,10}+\mathbf{C}_{f,11}+\mathbf{C}_{f,0}++\mathbf{C}_{f,2})/5$；

在将 own、nei 遍历之后，网格单元体心数组全部被填充，即获得网格单元的体心场。网格单元的体心计算相应的代码如下：

```
// Clear the fields for accumulation
cellCtrs = Zero;
cellVols = 0.0;

const labelList& own = faceOwner();//内部面与边界面
const labelList& nei = faceNeighbour();//边界面

// first estimate the approximate cell centre as the average of
// face centres

vectorField cEst(nCells(), Zero);
labelField nCellFaces(nCells(), 0);

forAll(own, facei)//对网格所有的内部面以及边界面进行遍历
{
    cEst[own[facei]] += fCtrs[facei];//对某个面对应的own网格单元的贡献
    nCellFaces[own[facei]] += 1;//某网格单元标识为own面的数量
}

forAll(nei, facei)//对网格所有的内部面进行遍历
{
    cEst[nei[facei]] += fCtrs[facei];//对某个面对应的nei网格单元的贡献
    nCellFaces[nei[facei]] += 1;//某网格单元标识为nei面的数量
}

forAll(cEst, celli)
{
    cEst[celli] /= nCellFaces[celli];
}
```

几何平均法计算的体心并不精准。因此还需要参考类似面的质心的方法来进行处理。既然有了预估的体心，那么对于这个网格单元，就可以将其分解为多个多面体。每个多面体的基为面，顶点为预估的体心。如图6.6所示，多面体的体心定义为多面体的顶点 B 与基底的面的质心 A 的连线的下方 1/4 处（D 点）。可以看出 D 点的坐标为 $\mathbf{A}_f+0.25\mathbf{C}$，其

中 C 表示 A 点与 B 点之前的连线 $\mathbf{A}-\mathbf{B}$，因此 D 点的坐标为

$$\mathbf{D} = 0.75\mathbf{A}_f + 0.25\mathbf{B} \tag{6.6}$$

同时，多面体的体积被定义为

$$V = \frac{1}{3}\mathbf{S}_A \cdot (\mathbf{A}_f - \mathbf{B}) \tag{6.7}$$

最后，有多面体体心以及质心的计算流程：

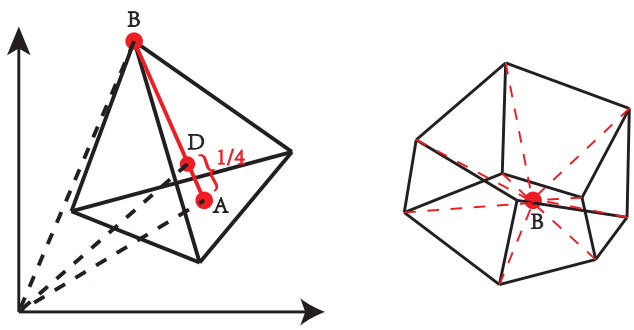


图 6.6: （左）多边形体心计算。（右）多面体分解。

1. 通过几何平均法方程(6.5)估算多面体体心点；
2. 通过估算的体心点对多面体进行分解为多个多面体；
3. 通过方程(6.6)计算分解后多面体的体心；
4. 通过方程(6.7)计算分解后多面体的体积；
5. 分解前的多面体体积为分解后多面体体心的体积加和：

$$V_c = \sum V_i \tag{6.8}$$

6. 分解前的多面体体心为分解后多面体体心的体积平均：

$$\mathbf{C}_c = \frac{\sum V_i \mathbf{C}_{c,i}}{\sum V_i} \tag{6.9}$$

如图6.7所示，现在我们考虑 `cells` 与 `faces` 连接性。再次重申，OpenFOAM 中的 `cells` 并不是我们想象的简单的网格单元标识。其最重要的功能是实现 `cells` 与 `faces` 的连接性[12]。因此，OpenFOAM 中的 `cells` 标识可以看为一个二维数组 [][]。数组第一个元素可以认为是 `cells` 的标识，例如如果有 30 个网格，那么 `cell[0]`，`cell[1]` 就标识第 0 个、第 1 个网格单元。数组的第二个元素（也是一个数组）标识这个 `cell` 包围的 `face`，比如 `cell[0][0]`，`cell[0][1]` 标识第 0 个网格里面的第 0 个面以及第 1 个面。因此，`cells()` 函数返回的是 `cellList`（可以理解为二维数组），`cells()[92]` 返回的是第 92 个元素（网格）相连接的面编号。相关的代码如下：

[12] primitiveMeshCells.C

```
forAll (cellFaceAddr, cellI)//cellFaceAddr可以认为是一个近似的二维数组，首先对数组的第一个元素进行遍历
{
    cellFaceAddr[cellI].setSize(ncf[cellI]);//对数组的第二个元素（数组）设置大小（并不设置值）。比如设置第0个网格单元具有3个值（数组大小为3），第1个网格单元具有4个值（数组大小为4）
}
ncf = 0;

forAll (own, faceI)//对网格所有内部面以及边界面进行遍历，此处own为faceOwner()
{
    label cellI = own[faceI];//对某个面，其owner对应一个网格编号，因此这个网格必然与这个face相连

    cellFaceAddr[cellI][ncf[cellI]++] = faceI;//将这个face的编号赋值给当前网格，也就是二维数组的第二个元素，成为第二个元素（数组）的一个元素
}
//通过这一次遍历，所有cell的对应的own面都被定义为相连接的face

forAll (nei, faceI)//对网格所有内部面进行遍历，此处own为faceNeighbour()
{
    label cellI = nei[faceI];//对某个面，其neighbour对应一个网格编号，因此这个网格必然与这个face相连

    if (cellI >= 0)
    {
        cellFaceAddr[cellI][ncf[cellI]++] = faceI;//将这个face的编号赋值给当前网格，也就是二维数组的第二个元素，成为第二个元素（数组）的一个元素
    }
}
//通过这一次遍历，所有cell的对应的nei面都被定义为相连接的face。因此，cell对应的nei面以及own面都已考虑
```

总结一下，OpenFOAM 的非结构网格必备数据为 point，face，owner，neighbour 以及 boundary。若通过第三方软件生成网格，在进行转换的时候最少提供这 5 个文件。有了这些文件之后，可以计算面心、面矢量等数据。同时不需要显性的定义 cell，就可以获得网格的体心、体积等数据。

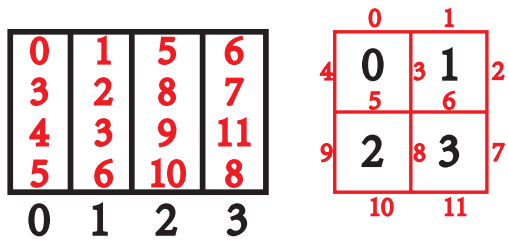


图 6.7: 2D 非结构网格 cell 与 face 的连接性。红色数字表示 face 编号，黑色数字表示 cell 编号。

6.1.5 line 和 edge

除了点之外，OpenFOAM 进一步定义 line，line 存在一个起始点、终止点、中心以及模长。line 的中心点的计算公式为：

$$\text{lineCenter} = 0.5(\text{lineStart} + \text{lineEnd}) \tag{6.10}$$

line 的模长即：$|\text{lineStart} + \text{lineEnd}|$。某点 **P** 距离线（起始点 **A**，终止点 **B**）上最近的点的计算公式为[13]：

$$\mathbf{P} = \mathbf{A} + \left(\frac{(\mathbf{B}-\mathbf{A})\cdot(\mathbf{P}-\mathbf{A})}{(\mathbf{B}-\mathbf{A})\cdot(\mathbf{B}-\mathbf{A})}\right)\cdot(\mathbf{B}-\mathbf{A}) \tag{6.11}$$

除 line 外，OpenFOAM 进一步定义 edge，二者的作用类似。需要注意的是，line 与 edge 数据不需要进行读取。

6.1.6 偏斜度

偏斜度存在各种不同的定义。如图6.8所示，公开的资料表示偏斜度可以通过一个几何内部的最大角以及最小角来计算。

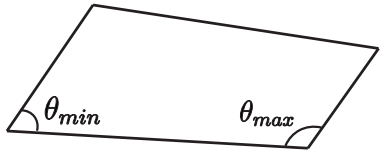


图 6.8: 一种计算偏斜度的方式。

OpenFOAM 并没有采用这种方式。OpenFOAM 中对网格的偏斜度定义在每个网格面上，针对图6.10所示的网格面，首先要求得 D 点的位置。OpenFOAM 中通过下式进行计算：

$$\mathbf{D} = \mathbf{A}\frac{d_{nei}}{d_{nei}+d_{own}} + \mathbf{B}\frac{d_{own}}{d_{nei}+d_{own}} \tag{6.12}$$

其中 d_{nei} 表示 B 点与 C 点的距离，d_{own} 表示 A 点与 C 点的距离。其在图6.10中的局部坐标系下可以更好的理解。D 点位于 A 与 B 点的连线上，其距离 A 点的位置为 $\frac{d_{own}}{d_{nei}+d_{own}}$。

[13] lineI.H

在获得 D 点的坐标后，最终偏斜度的定义为：

$$\text{skewness} = \frac{|\mathbf{C}-\mathbf{D}|}{|\mathbf{A}-\mathbf{B}|} \tag{6.13}$$

相应的代码如下[14]：

```
    scalar dOwn = mag(faceCtrs[faceI] - cellCtrs[own[faceI]]);
    scalar dNei = mag(faceCtrs[faceI] - cellCtrs[nei[faceI]]);

    point faceIntersection =
        cellCtrs[own[faceI]]*dNei/(dOwn+dNei)
      + cellCtrs[nei[faceI]]*dOwn/(dOwn+dNei);

    scalar skewness =
        mag(faceCtrs[faceI] - faceIntersection)
        /(mag(cellCtrs[nei[faceI]] - cellCtrs[own[faceI]]) + VSMALL);
```

图6.9所示的为一个真实的网格，其中的红色的网格单元面的偏斜度很差。偏斜度主要影响的是插值的精度问题。很明显，图6.9中两个点的插值，更倾向于表示红面之上的面的插值，而不是红面上的插值。

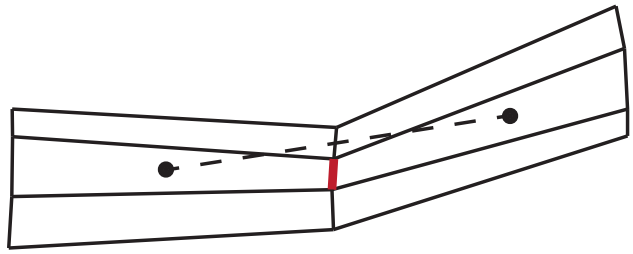


图 6.9: 一个比较差的 skewness 面（其中的红面）。

6.2 OpenFOAM 中的 TVD 格式

6.2.1 非结构网格 TVD 格式

参考图6.11，在结构网格中，TVD 格式通过定义变量 r 来计算通量限制器。在结构网格中，r 定义为

$$r = \frac{\psi_{\mathrm{C}} - \psi_{\mathrm{U}}}{\psi_{\mathrm{D}} - \psi_{\mathrm{C}}} \tag{6.14}$$

r 可以用来衡量网格面处的光滑度。如果在此处是光滑的，则 $r \approx 1$。如果此处存在间断，则 r 远远不等于 1。在非结构网格中植入 TVD 格式最大的问题在于：在非结构网格，针

[14] primitiveMeshCheck.C

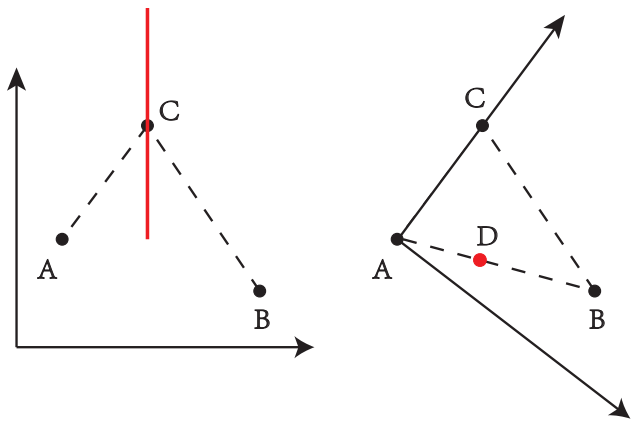


图 6.10: OpenFOAM 中计算网格偏斜度示意图。红线表示网格单元的某个面。A 点表示 own 网格单元体心，B 点表示 nei 网格单元体心，C 点表示面心。（左）全局坐标系示意图。（右）局部坐标系示意图，D 点表示通过公式(6.12)求得的矢量点。

对网格单元 C，并不存在 U 网格点的定义。在 90 年代，OpenFOAM 的前身 foam 就植入了非结构网格的 TVD 格式。相关工作也在 Jasak 的博士论文中被讨论 [73]。在这里简述这种非结构网格 TVD 格式，读者也可参考 2003 年 Darwish and Moukalled 的相关工作 [39]。

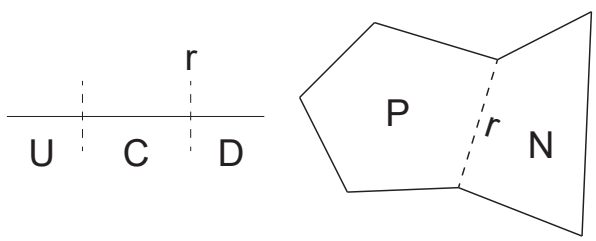


图 6.11: 结构网格（左侧）、非结构网格（右侧）示意图。虚线表示控制体面。其中 r 为 TVD 格式需要定义的变量

方程(6.14)中可以拓展为：

$$r = \frac{\psi_{\mathrm{C}} - \psi_{\mathrm{U}}}{\psi_{\mathrm{D}} - \psi_{\mathrm{C}}} = \frac{(\psi_{\mathrm{D}} - \psi_{\mathrm{U}}) - (\psi_{\mathrm{D}} - \psi_{\mathrm{C}})}{\psi_{\mathrm{D}} - \psi_{\mathrm{C}}} \tag{6.15}$$

在非结构网格中，$\psi_{\mathrm{D}} - \psi_{\mathrm{U}}$ 被认为是网格 C 点的梯度乘以网格 D 与 U 的方向矢量：

$$\psi_{\mathrm{D}} - \psi_{\mathrm{U}} = \nabla\psi_{\mathrm{C}} \cdot \mathbf{d}_{UD} \tag{6.16}$$

其中 $\mathbf{d}_{UD}$ 表示非结构网格上连接虚拟的 U 点与 D 点的矢量：

$$\mathbf{d}_{UD} = \mathbf{x}_D - \mathbf{x}_U \tag{6.17}$$

一旦 $\mathbf{d}_{UD}$ 可求，那么 $\psi_{\mathrm{D}} - \psi_{\mathrm{U}}$ 可求。目前，Darwish and Moukalled [39] 认为这个虚拟节点 U 的位置可以这样给定：

$$\mathbf{d}_{UD} = 2\mathbf{d}_{CD} \tag{6.18}$$

其本质上是假定 C 点在 U 与 D 点的中心，这是一种最简单的二阶假定。这样，有：

$$\psi_{\mathrm{D}} - \psi_{\mathrm{U}} = 2\nabla\psi_{\mathrm{C}}\mathbf{d}_{CD} \tag{6.19}$$

因此，非结构网格中 TVD 格式的 r 定义为：

$$r = \frac{2\nabla\psi_{\mathrm{C}}\mathbf{d}_{CD}}{\psi_{\mathrm{D}} - \psi_{\mathrm{C}}} - 1 \tag{6.20}$$

在图6.11右侧显示的非结构网格中，r 即为：

$$r = \frac{2\nabla\psi_{\mathrm{P}}\mathbf{d}_{PN}}{\psi_{\mathrm{N}} - \psi_{\mathrm{P}}} - 1 \tag{6.21}$$

在获得 r 值之后，有变量 W 在网格面上的值为：

$$\psi_f = \psi_{\mathrm{P}} + \frac{1}{2}\psi(r)(\psi_{\mathrm{N}} - \psi_{\mathrm{P}}) \tag{6.22}$$

其中 $\psi(r)$ 对应不同的高阶格式，如 vanLeer、Minmod 等。

6.2.2 附加限制器的数值格式

但凡需要调用限制器的格式，OpenFOAM 都将其归类为有界的格式（limitedSchemes）。例如对于 upwind 格式[15]，有：

$$\psi(r) = 0 \tag{6.23}$$

对于 vanLeer 格式，有[16]：

$$\psi(r) = \frac{r + |r|}{1 + |r|} \tag{6.24}$$

对于 limitedLinear 格式，有：

$$\psi(r) = \max\left(\min\left(\frac{2}{k}r, 1\right), 0\right) \tag{6.25}$$

其中 k 为用户需要指定的值，可见，当 $k \approx 0$ 的时候，limitedLinear 演变为：

$$\psi(r) = 1 \tag{6.26}$$

在这种情况下：

$$\psi_f = \frac{1}{2}(\psi_{\mathrm{N}} + \psi_{\mathrm{P}}) \tag{6.27}$$

其在均分网格下为中心格式。其他格式不一一类推，均为简单的代数方程。图6.12中是 OpenFOAM-12 中现存的有界的数值格式。

[15] src/finiteVolume/interpolation/surfaceInterpolation/limitedSchemes/upwind

[16] src/finiteVolume/interpolation/surfaceInterpolation/limitedSchemes/vanLeer

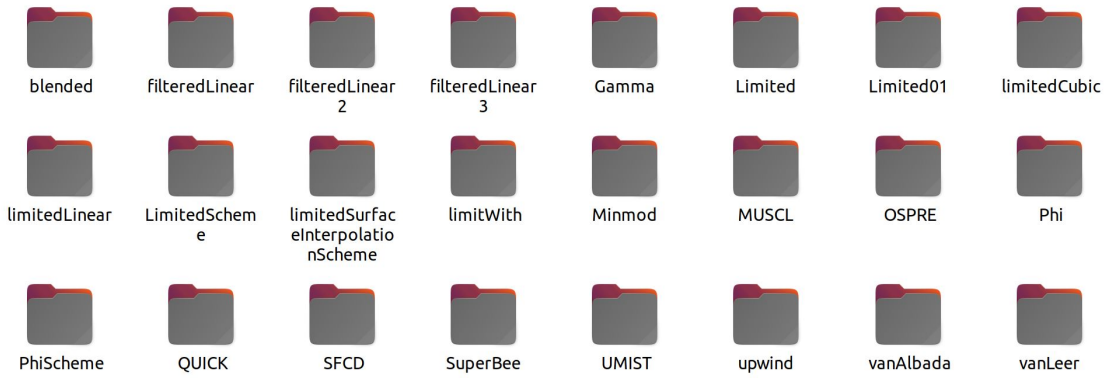


图 6.12: OpenFOAM-12 中现存的有界的数值格式。

6.2.3 显性修正的数值格式

OpenFOAM 中的一些数值格式没有做限制器。这些格式可能是无界的。例如中心格式，下风格式。但这些格式可能附带一些显性的数值修正。简单的情况，在计算的时候仅仅计算面网格相邻的网格单元的权重。例如中心格式有：

$$\psi_f = w_{\mathrm{N}}\psi_{\mathrm{N}} + w_{\mathrm{P}}\psi_{\mathrm{P}} \tag{6.28}$$

下风格式：

$$\psi_f = \psi_{\mathrm{N}} \quad \text{or} \quad \psi_f = \psi_{\mathrm{P}} \tag{6.29}$$

其他的格式大多需要显性修正。在显性修正的时候，需要计算梯度。比较常见的数值格式有线性迎风格式以及 LUST 格式。例如对于 LUST 格式，其计算权重的时候，简单的混合了 75% 的中心格式，以及 25% 的线性迎风格式。对于线性迎风格式，其为迎风格式的基础上，附加了显性的精度修正（使用当前的变量来计算梯度并修正）。因此，线性迎风格式与迎风格式的矩阵系数是相同的，但是矩阵的源项不同（显性修正）。假定上游是 N 网格，线性迎风格式为：

$$\psi_f = \psi_{\mathrm{N}} + \mathbf{d} \cdot \nabla\psi \tag{6.30}$$

其中的 $\mathbf{d}$ 表示网格面心距离上游网格体心的距离矢量。在对线性迎风做梯度限制的时候，其演变成一种有界的 TVD 格式。图6.13中是 OpenFOAM-12 中现存的显性修正的数值格式。

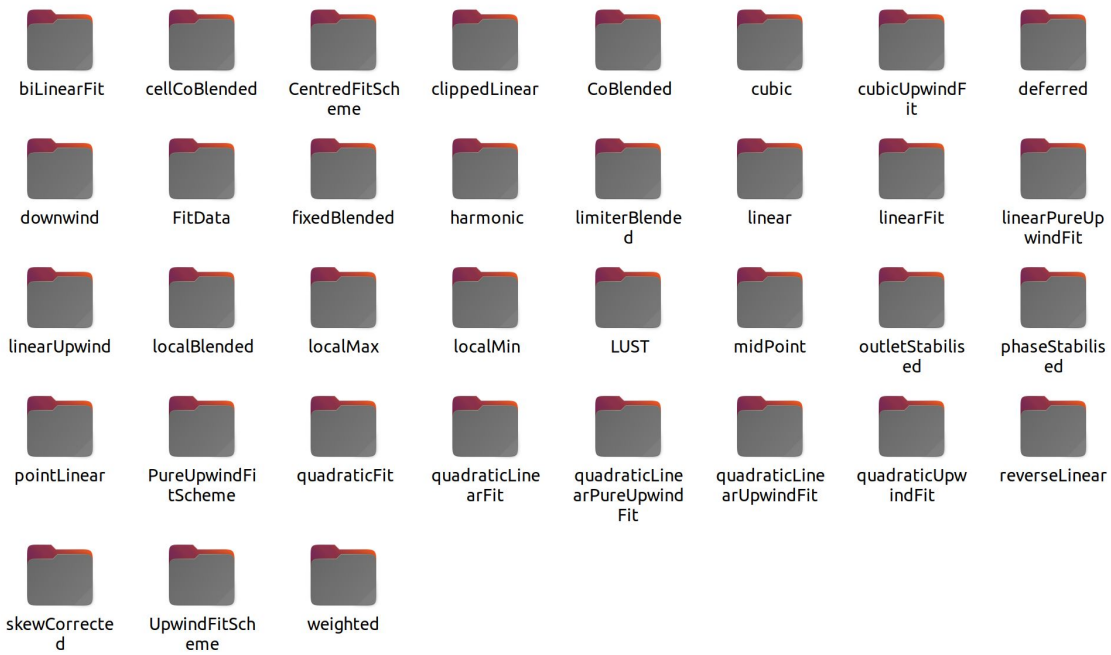


图 6.13: OpenFOAM-12 中现存的显性修正的数值格式。

6.3 OpenFOAM 中的 fvc 显性离散

6.3.1 fvc::ddt 显性时间项计算

OpenFOAM 中变量 T 的时间项可以表示为

$$\mathrm{fvc::ddt(T)} = \frac{\int_V \int_t \frac{\partial T}{\partial t} \mathrm{d}V\mathrm{d}t}{\Delta V \Delta t} \tag{6.31}$$

在这里要区分具体的时间格式，若采用 Euler 格式，则其可以进一步的写为：

$$\mathrm{fvc::ddt(T)} = \frac{\int_V \int_t \frac{\partial T}{\partial t} \mathrm{d}V\mathrm{d}t}{\Delta V \Delta t} = \frac{T^t - T^{t-\Delta t}}{\Delta t} \tag{6.32}$$

相应的代码如下[17]：

```
template<class Type>
tmp<GeometricField<Type, fvPatchField, volMesh> >
EulerDdtScheme<Type>::fvcDdt
(
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    ...
    else
```

[17] EulerDdtScheme.C

```
    {
        return tmp<GeometricField<Type, fvPatchField, volMesh> >
        (
            new GeometricField<Type, fvPatchField, volMesh>
            (
                ddtIOobject,
                rDeltaT*(vf - vf.oldTime())//函数定义
            )
        );
    }
}
```

6.3.2 fvc::div 显性对流项离散

OpenFOAM 中变量 T 的对流项，若显性离散，可以表示为[18]

$$\mathrm{fvc::div(phi, T)} = \frac{\int_V \int_t \nabla \cdot (T\mathbf{U})\mathrm{d}V\mathrm{d}t}{\Delta V \Delta t} = \mathrm{surfaceIntegrate}\left(\mathbf{U}_f \cdot \mathbf{S}_f T_f\right) \tag{6.33}$$

其中的变量均为已知量（因为为显性离散）。方程(6.33)中使用 surfaceIntegrate 来代替 $\sum$，这是因为 surfaceIntegrate 并不是简单的加和操作[19]。

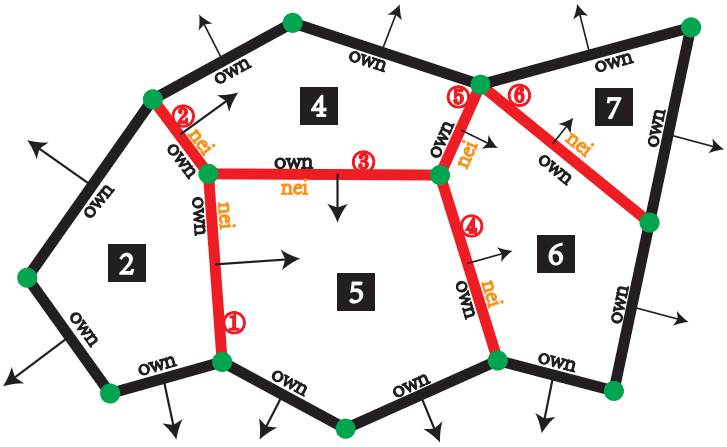


图 6.14: 2D 非结构网格示意图。黑线表示面矢量

下面对 surfaceIntegrate 进行解释，考虑整个网格系统的通量计算。一种方式是，对每个网格单元的面进行遍历操作，然后加和，获得当前网格单元的值。但由于每个面链接 2 个网格单元。这种操作实际上会遍历两次面单元。会导致效率损失。在开源 CFD 软件 OpenFOAM 中，仅仅需要遍历一次网格单元，即可求出面通量。具体的，考虑图6.14，针对

[18]注意下方公式除掉了网格单元体积以及时间步长 $\Delta V \Delta t$。

[19]OpenFOAM 中的 surfaceSum 函数与数学的 $\sum$ 函数作用相同。但最后注意除掉了网格体积。

图中的 2D 网格系统，黑色的箭头表示面矢量，以网格单元 5 为例，如果某个面速度矢量与面矢量同向，则表示流体流入网格 5，反之则表示流出网格 5。在实施的过程中，第一步，对所有的内部面进行遍历，以面 3 为例。假定此面的通量为 A（不管正负），那么 4 单元（own）的值要加上 3 面的通量。同时，5 网格单元将减去 A 贡献。这正是 surfaceIntegrate 函数的意义（依据 own 与 nei 的作用进行正负调节）。随后，对内部面遍历之后，则需要对边界面进行遍历，原理相同。计算通量的代码如下[20]：

```
const fvMesh& mesh = ssf.mesh();

const unallocLabelList& owner = mesh.owner();//内部面的own序号
const unallocLabelList& neighbour = mesh.neighbour();//内部面的nei序号

const Field<Type>& issf = ssf;

forAll(owner, facei)
{
    ivf[owner[facei]] += issf[facei];//针对某个面，own网格增加贡献
    ivf[neighbour[facei]] -= issf[facei];//针对某个面，nei网格减少贡献
}
//至此，所有内部面都已经考虑，内部面毗连的网格单元通量已经计算。但如果网格存在边界，其贡献还没考虑

forAll(mesh.boundary(), patchi)
{
    const unallocLabelList& pFaceCells =
        mesh.boundary()[patchi].faceCells();

    const fvPatchField<Type>& pssf = ssf.boundaryField()[patchi];

    forAll(mesh.boundary()[patchi], facei)
    {
        ivf[pFaceCells[facei]] += pssf[facei];
    }
}
//上述代码考虑边界face对网格的贡献

ivf /= mesh.V();//除掉网格体积
```

在上面的代码中，注意这里的 own 与之前的 own 的定义并不相同。这里的 own 定义为

[20] fvcSurfaceIntegrate.C

```
    const unallocLabelList& owner = mesh.owner();//内部面的own序号
```

在 fvMesh.H 中，其表示内部面的 own 网格单元。前述代码的 own 定义为：

```
    const labelList& own = faceOwner();//内部面与边界面own序号
```

在 primitiveMesh.H 中，其表示内部面与边界面的网格单元。nei 的定义以此类推。

6.3.3 fvc::laplacian 显性拉普拉斯项计算

OpenFOAM 中的显性拉普拉斯项可以定义为[21]

$$\text{fvc :: laplacian(T)} = \frac{\int_V \int_t \nabla \cdot (\nabla T) \mathrm{d}V \mathrm{d}t}{\Delta V \Delta t} = \text{surfaceIntegrage}\left((\nabla T)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} \right) |\mathbf{S}_f| \quad (6.34)$$

其中 $(\nabla T)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|}$ 当做面法向梯度进行离散即可（参考第6.3.4节）。相关代码如下[22]：

```
    tmp<GeometricField<Type, fvPatchField, volMesh> > tLaplacian
    (
        fvc::div(gamma*this->tsnGradScheme_().snGrad(vf)*mesh.magSf())
    );

    tLaplacian().rename("laplacian(" + gamma.name() + ',' + vf.name() +
    ')');

    return tLaplacian;
```

6.3.4 fvc::snGrad 显性面法向梯度项计算

OpenFOAM 中变量 T 的面法向梯度（面场）可以定义为：

$$\text{fvc :: snGrad(T)} = (\nabla T)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} \quad (6.35)$$

其中 $(\nabla T)_f$ 表示定义在面上的梯度。方程(6.35)在计算的过程中分为不同的情况，在最简单的情况下（正交网格），其可以表示为：

$$\text{fvc :: snGrad(T)} = (\nabla T)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} = \frac{T_{nei} - T_{own}}{|\mathbf{d}|} \quad (6.36)$$

其中的 $\mathbf{d}$ 表示从 own 单元指向 nei 单元的矢量，相对应的代码如下：

```
    const scalarField& deltaCoeffs = tdeltaCoeffs().internalField();//
    网格单元之间的距离的倒数

```

[21] 注意下方公式中除掉了网格单元体积与时间步长 $\Delta V \Delta t$。

[22] fvcLaplacian.C

```
    // owner/neighbour addressing
    const unallocLabelList& owner = mesh.owner();//内部面的own标识
    const unallocLabelList& neighbour = mesh.neighbour();//内部面的nei标识

    forAll(owner, faceI)//对内部面进行遍历
    {
        ssf[faceI] =
            deltaCoeffs[faceI]*(vf[neighbour[faceI]] - vf[owner[faceI]]);//某个面上的面梯度，在这里假定是正交的
    }

    forAll(vf.boundaryField(), patchI)
    {
        ssf.boundaryField()[patchI] = vf.boundaryField()[patchI].snGrad();//对于边界面，进一步调用边界面的计算函数（思想相同）
    }
```

然而在很多情况下，网格并不是正交的。在这里要引入非正交修正。在实现的过程中，首先按照正交网格通过方程(6.36)计算面法向梯度。然后加上非正交修正即可。OpenFOAM 中植入的为超松弛修正，即下面的方程形式：

$$\mathrm{fvc::snGrad(T)} = \frac{T_{nei} - T_{own}}{|\mathbf{d}|} + \mathrm{corr(T)} \tag{6.37}$$

其中 corr(T) 表示非正交修正，可以表示为

$$\mathrm{corr(T)} = \left(\frac{\mathbf{S}_f}{|\mathbf{S}_f|} - \frac{\mathbf{d}}{|\mathbf{d}|}\right) \cdot (\nabla T)_f \tag{6.38}$$

相应的代码如下：

```
    for (direction cmpt = 0; cmpt < pTraits<Type>::nComponents; cmpt++)
    {
        ssf.replace
        (
            cmpt,
            mesh.correctionVectors()
          & linear
            <
                typename
                outerProduct<vector, typename pTraits<Type>::cmptType>::type
            >(mesh).interpolate
            (
```

```
                gradScheme<typename pTraits<Type>::cmptType>::New
                (
                    mesh,
                    mesh.gradScheme(ssf.name())
                )()
                //gaussGrad<typename pTraits<Type>::cmptType>(mesh)
               .grad(vf.component(cmpt))
            )
        );
    }
    ...
    forAll(owner, faceI)
    {
        vector unitArea = areas[faceI]/magAreas[faceI];
        vector delta =
            cellCentres[neighbour[faceI]] - cellCentres[owner[faceI]];

        CorrVecs[faceI] = unitArea - delta*DeltaCoeffs[faceI];
    }
```

其中 $(\nabla T)_f$ 仅仅表示 ∇T 向网格面上的插值，将在梯度格式中介绍具体的求解方法。

6.3.5 fvc::grad 显性梯度项计算

OpenFOAM 中变量 T 的梯度项可以表示为

$$\mathrm{fvc::grad(T)} = \frac{\int_V \int_t \nabla T \mathrm{d}V \mathrm{d}t}{\Delta V \Delta t} = \frac{1}{\Delta V} \sum T_f \cdot \mathbf{S}_f \tag{6.39}$$

其中 T_f 为用户指定的插值格式。相应的代码如下[23]：

```
    forAll(owner, facei)
    {
        GradType Sfssf = Sf[facei]*issf[facei];

        igGrad[owner[facei]] += Sfssf;
        igGrad[neighbour[facei]] -= Sfssf;
    }
    ...
    igGrad /= mesh.V();
```

假如给定 4 个网格，假定 $\Delta x = 1$，其中第 1、2、3、4 个网格点的 ψ 值分别为 1、1、0、0，通过 `fvc::grad()` 进行计算，其第 1、2、3、4 个网格点梯度分别为 0、-0.5、-0.5、0。注意其并不是 0、-1、-1、0。这是因为 `fvc::grad()` 定义在网格点上，而不是网格面上。

[23] gaussGrad.C

OpenFOAM 中梯度项还可以使用最小二乘法计算，在这里不做详细介绍。

6.3.6 fvc 操作符的单位

fvc对场进行操作，其单位不变。例如，针对fvc::ddt(T)，其单位即为 $\frac{\partial T}{\partial t}$ 的真实形式，即 K/s。同理，fvc::div(phi,T)的单位即为 $\nabla\cdot(\mathbf{U}T)$ 的单位 K/s。其他以此类推。

6.4 OpenFOAM 中的 fvm 隐性离散

6.4.1 稀疏线性系统、lduMatrix，fvMatrix 等

CFD 求解器在离散之后，会形成一个稀疏线性系统，其可以用方程表示为：

$$\mathbf{A}\boldsymbol{\psi}=\mathbf{b} \tag{6.40}$$

其中 $\mathbf{A}$ 表示有限体积法离散后的矩阵 fvMatrix，$\boldsymbol{\psi}$ 表示待求变量 fvMatrix.psi，$\mathbf{b}$ 表示已知量（源项）fvMatrix.source。整个方程(6.40)都通过 fvMatrix 来进行存储以及相关的求解操作。fvMatrix 存在了一些成员：

- diagPtr_ 存储对角线系数的值；
- lowerPtr_ 存储下三角线系数的值；
- upperPtr_ 存储上三角线系数的值；
- lowerAddr_ 存储下三角线系数的位置；
- upperAddr_ 存储上三角线系数的位置；

现在以下面的代码举例说明矩阵构造过程：

```
...
#include createMesh.H
...
fvScalarMatrix TEqn(fvm::ddt(T));
...
```

1. 求解器首先通过 createMesh.H 来构建 fvMesh 类型，构建 fvMesh 类型之前，要构建 polyMesh 类型。在求解器运行时 fvMesh、polyMesh 会读取网格文件中的 owner、neighbour 等信息，然后将这些信息在 polyMesh 中进行存储。因此，在求解器运行时，一旦创建网格，owner、neighbour 等信息既可以随时调用；

2. fvm::ddt(T) 首先构建一个空的 fvMatrix[24]，并将这个 fvMatrix 的对角线系数、以及源项进行赋值；

3. 将上一步 fvm::ddt(T) 构建的 fvMatrix，赋值给 TEqn，因为上面的代码只有时间项，因此两个 fvMatrix 是相同的；

上述代码将初始化一个稀疏线性系统，其 **A** 为一个对角阵，**b** 存在值。注意因为其中不存在其他项，因此矩阵仅仅存在对角线系数。下面看一个稍微复杂一点的代码：

```
...
#include createMesh.H
...
fvScalarMatrix TEqn
(
    fvm::ddt(T)
 +  fvm::laplacian(nu, T)
);
...
```

这两个代码段的区别在于，后面的代码段构造的矩阵 **A** 存在非对角元素：

1. fvm::laplacian(nu, T) 首先构建一个空的 fvMatrix[25]，并将这个 fvMatrix 的 upper() 进行赋值。在赋值的过程中，会调用 lduMatrix 中的 upper() 函数，upper() 函数将值赋予 lduMatrix 中的 upperPtr_，即矩阵的上对角元素。需要强调的是，upperPtr_ 仅仅是一系列值，不具备位置信息；

2. 为了存储非对角元素的位置信息。必要需要调用某些函数。我们发现在对 upperPtr_ 赋值时，会自动调用 lduAddr().lowerAddr().size() 函数来获取上三角元素的场元素数量。同时在这一步，也确定了非对角线元素的位置信息。具体的：在调用这个函数的时候，lduAddr() 函数实际调用的是 lduMesh_.lduAddr()[26]。其中的 lduMesh_ 在创建 fvMatrix 以及 lduMatrix 的时候，就已经将 mesh 赋值给了 lduMesh 的具体类型 fvMesh。因此，lduMesh_.lduAddr() 调用的是 fvMesh 中的 lduAddr() 函数，这个函数进一步返回 lduAddressing 类型，其值通过 fvMeshLduAddressing 初始化构造函数来实现[27]。这个函数在进行初始化的时候，会对 lowerAddr_ 以及 upperAddr_ 进行赋值（读取网格的 owner 以及 neighbour 信息）[28]。至此，非对角阵的元素位置信息被确立。

[24] EulerDdtScheme.C
[25] gaussLaplacianScheme.C
[26] lduMatrix.H
[27] fvMeshLduAddressing.C
[28] fvMeshLduAddressing.H

3. 更新边界条件信息。

4. 将上一步 fvm::ddt(T) 构建的 fvMatrix，与拉普拉斯项构建的矩阵，进行加和，赋值给 TEqn。此时的矩阵存在对角线系数，非对角线系数，以及源项；

下述即为拉普拉斯项的计算过程。在第6.4.3节有更详细的信息。

```
{
    tmp<surfaceScalarField> tdeltaCoeffs =
        this->tsnGradScheme_().deltaCoeffs(vf);
    const surfaceScalarField& deltaCoeffs = tdeltaCoeffs();

    tmp<fvMatrix<Type> > tfvm
    (
        new fvMatrix<Type>
        (
            vf,
            deltaCoeffs.dimensions()*gammaMagSf.dimensions()*vf.dimensions()
        )
    );//一个空fvMatrix
    fvMatrix<Type>& fvm = tfvm();

    //更新对称的非对角线系数
    fvm.upper() = deltaCoeffs.internalField()*gammaMagSf.internalField();
    //更新对角线系数
    fvm.negSumDiag();

    //更新边界条件对对角线系数的影响
    forAll(fvm.psi().boundaryField(), patchI)
    {
        const fvPatchField<Type>& psf = fvm.psi().boundaryField()[patchI];
        const fvPatchScalarField& patchGamma =
            gammaMagSf.boundaryField()[patchI];

        fvm.internalCoeffs()[patchI] = patchGamma*psf.gradientInternalCoeffs();
        fvm.boundaryCoeffs()[patchI] = -patchGamma*psf.gradientBoundaryCoeffs();
    }

```

```
    return tfvm;
}
```

对流项的离散过程类似，在此不再赘述。需要强调的是，矩阵的非对角阵元素系数的位置，与具体的项有关（例如时间项离散就没有非对角阵元素，扩散项离散则有非对角阵元素）。具体的项的离散后的非对角阵元素位置，由网格直接确定。图6.15则解释了如何通过网格来确定矩阵非对角元素的位置信息。

同时，还可以调用下面的语句来获取相应的信息：

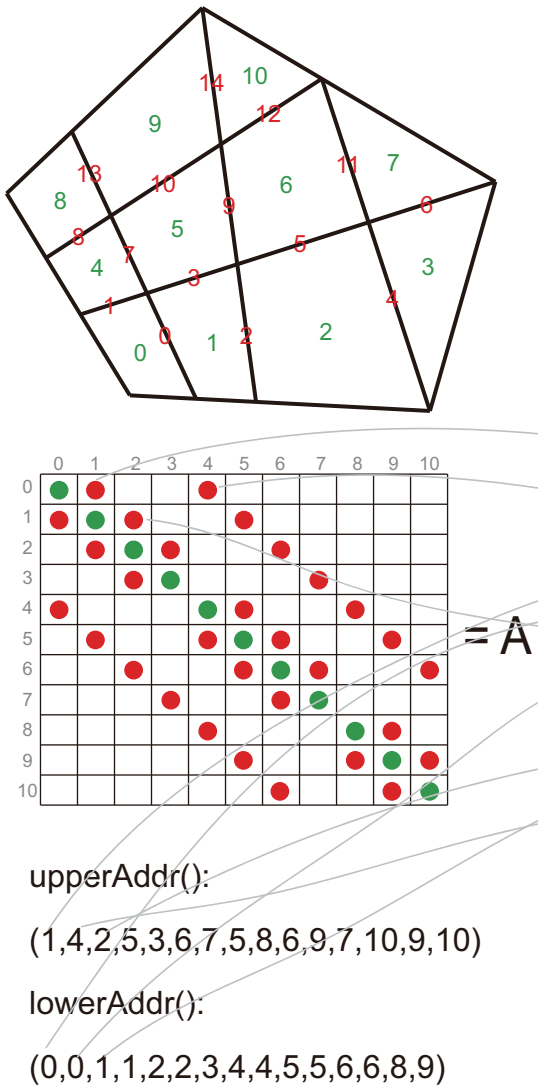
- `TEqn.diag()`：没有边界条件影响的对角线系数；
- `TEqn.D()`：考虑边界影响的矩阵的对角线系数。这是因为矩阵在离散的过程中，边界条件也会对对角线系数产生影响。因此，矩阵的对角线系数在组建的过程中，首先对内部场组建对角线系数 `TEqn.diag()`，然后将边界离散产生对对角线系数的影响通过 `internalCoeffs_` 进行赋值。`TEqn.D()` 的值为 `diag()` 和 `internalCoeffs_` 的值加和；
- `TEqn.internalCoeffs_`：边界条件对矩阵对角线系数的影响。参考第6.4.3节；
- `TEqn.boundaryCoeffs_`：边界条件对矩阵源项的影响。**b** 为 `source()` 与 `boundaryCoeffs_` 的和。参考第6.4.3节；
- `TEqn.A()`：`TEqn.D()` 除以网格单元体积，同时附加边界条件。也即其为一个几何场；
- `TEqn.H()`、`TEqn.H1()`：参考第6.5.3节；
- `TEqn.source()`：未考虑边界条件离散的矩阵源项；
- `TEqn.upper()`：矩阵的上对角线系数；
- `TEqn.lower()`：矩阵的下对角线系数；
- `TEqn.upperAddr()`：矩阵的上对角线系数存储位置；
- `TEqn.lowerAddr()`：矩阵的下对角线系数存储位置；

6.4.2 fvm::ddt 隐性时间项离散

现在考虑对时间项进行隐性离散。若采用欧拉方法，针对时间项，有：

$$\mathrm{fvm::ddt(T)} = \frac{\Delta V}{\Delta t}(T^{t+\Delta t} - T^t) \tag{6.41}$$

由于时间项不涉及到相邻的网格单元。因此时间项的离散不涉及到网格连接性。其中 $\frac{\Delta V}{\Delta t}$ 进入矩阵的对角阵，$\frac{\Delta V}{\Delta t}T^t$ 进入矩阵源项。



A的非对角元素的存储需要两部分信息，一部分是存储位置，一部分是存储的值。存储位置通过upperAddr()与lowerAddr()进行（下文简称u()与l()）。u()本质上即为每个内部面的nei网格标识。l()即为每个内部面的own网格标识。左侧2D网格的u()与l()分别为左下角所示。

以u()举例，其表示：
第0个内部面的nei网格编号为1
第1个内部面的nei网格编号为4
第2个内部面的nei网格编号为2
第3个内部面的nei网格编号为5
...

因此，在网格建立之后，即可以获得u()与l()。在有了u()与l()之后，可以构建A的非对角元素。OpenFOAM中按照网格连接性来构建对阵矩阵（非对阵矩阵存储元素值为0）。例如：

取l()的第一个元素0，u()的第一个元素1，构成A01
取l()的第二个元素0，u()的第二个元素4，构成A04
取l()的第三个元素1，u()的第三个元素2，构成A12
...

这些值即矩阵A的非零上三角矩阵，需要存储数据。同理，下三角矩阵为一种反向操作：

取u()的第一个元素1，u()的第一个元素0，构成A10
取u()的第二个元素4，u()的第二个元素0，构成A40
取u()的第三个元素2，u()的第三个元素1，构成A21
...

这样，稀疏非对称矩阵需要进行存储的元素位置即可被定义。

图 6.15: 网格与矩阵元素位置信息的直接关系。

在这里要注意 OpenFOAM 中上一个时间步存储的方法。假设有伪代码如下，其中的代码注释解释了时间步的推进方法。

```
// 求解器从Time = 0读取初始场，下面开始时间步推进
while (simple.loop(runTime))
{
    // 假定dt是1，因此当前时间步是1，Time = 1，变量在该时间步下"待"更新
    // 在solve之前，T可以理解为Time = 0的值，这是因为还没有solve，场并没有更新，因此T与T.old的值相同，都代表Time = 0时刻的T值
    // T.oldold实际代表Time = -1时间步，由于不存在，因此T.oldold与T.old相同

    solve(fvc::ddt(T) == S);
    // 求解器进行solve，更新T场，此刻T表示真正的Time = 1的值
    // T.old表示Time = 0时刻的值

    // 假设再进行一次solve，T会不会继续推进？
    // 并不会，因为在此T.old还是表示Time = 0时刻的值
    solve(fvc::ddt(T) == S);
    solve(fvc::ddt(T) == S);
    // 因此上述2行代码无效！即使多次solve，T并不会发生变化
}
```

如果存在多个变量，若采用欧拉方法，针对时间项，有：

$$\mathrm{fvm} :: \mathrm{ddt}(\alpha \mathrm{T}) = \frac{\Delta V}{\Delta t}\left(\alpha^{?} T^{?} - \alpha^{?} T^{?}\right) \tag{6.42}$$

问题是其中的时间步标志如何确定？首先我们看下面的方程：

$$\begin{aligned} \frac{\partial T}{\partial t} &= 1 \\ \frac{\partial \psi}{\partial t} &= \frac{2}{T} \end{aligned} \tag{6.43}$$

在 OpenFOAM 中，PDE 通常采用解耦的方法来进行求解。这也就意味着首先可以求解 T 方程，来获得 $T^{t+\Delta t}$ 的值，然后求解 ψ 方程。因此，方程(6.43)的离散形式为：

$$\frac{T^{t+\Delta t} - T^{t}}{\Delta t} = 1 \tag{6.44}$$

求解后获得 $T^{t+\Delta t}$ 后，有：

$$\frac{\psi^{t+\Delta t} - \psi^{t}}{\Delta t} = \frac{2}{T^{t+\Delta t}} \tag{6.45}$$

看下面的方程：

$$\begin{aligned} \frac{\partial T}{\partial t} &= 1 \\ \frac{\partial \psi T}{\partial t} &= \frac{2}{T} \end{aligned} \tag{6.46}$$

在 OpenFOAM 中，上面的方程的离散形式为

$$\frac{T^{t+\Delta t}-T^{t}}{\Delta t}=1 \tag{6.47}$$

求解后获得 $T^{t+\Delta t}$ 后，有：

$$\frac{T^{t+\Delta t}\psi^{t+\Delta t}-T^{t}\psi^{t}}{\Delta t}=2 \tag{6.48}$$

因此，针对 ddt 项，只要其中的 α 进行了更新，那么就是

$$\mathrm{fvm}::\mathrm{ddt}(\alpha\mathrm{T})=\frac{\Delta V}{\Delta t}\left(\alpha^{t+\Delta t}T^{t+\Delta t}-\alpha^{t}T^{t}\right) \tag{6.49}$$

如果 α 没有进行更新，那么就是

$$\mathrm{fvm}::\mathrm{ddt}(\alpha\mathrm{T})=\frac{\Delta V}{\Delta t}\left(\alpha^{t}T^{t+\Delta t}-\alpha^{t-\Delta t}T^{t}\right) \tag{6.50}$$

在这里进行拓展，在 OpenFOAM 的求解器中，会存在下面的代码：

```
fvm::ddt(rho, U) + ...
```

以及

```
fvm::ddt(psi, p) + ...
```

类似的相关代码。在这里要注意，上述代码的`rho`以及`psi`要进行更新，否则`rho`以及`psi`在时间步中会存在相同的值，导致该项失去作用。一般情况下，OpenFOAM 的可压缩代码都存在密度方程。因此不会出现问题。但是并不存在`psi`方程，所以`psi`一定要存储旧值[29]。

6.4.3 fvm::laplacian 隐性拉普拉斯项离散

首先定义一个单网格系统。考虑图6.16中的网格单元 5，如果其为内部网格：

- 网格面 3 的 own 网格单元为 1，nei 网格单元为 5；
- 网格面 7 的 own 网格单元为 4，nei 网格单元为 5；
- 网格面 9 的 own 网格单元为 5，nei 网格单元为 6；
- 网格面 10 的 own 网格单元为 5，nei 网格单元为 9；

如果其存在边界面，那么边界面与网格毗连的网格单元为边界面的 own 网格，且面矢量永远指向边界向外。

现在考虑稳态离散后的扩散方程[30]：

$$\mathrm{surfaceIntegrate}\left(\Gamma_f(\nabla T)_f\cdot\frac{\mathbf{S}_f}{|\mathbf{S}_f|}|\mathbf{S}_f|\right)=0 \tag{6.51}$$

[29] 可以参考 hePsiThermo.C 其中的 `correct()` 函数要更新 `oldTime()`，但是 heRhoThermo.C 中不需要。

[30] 真正的扩散方程需要添加负号。若不添加负号，离散后的对角线系数为负，非对角线系数为正。

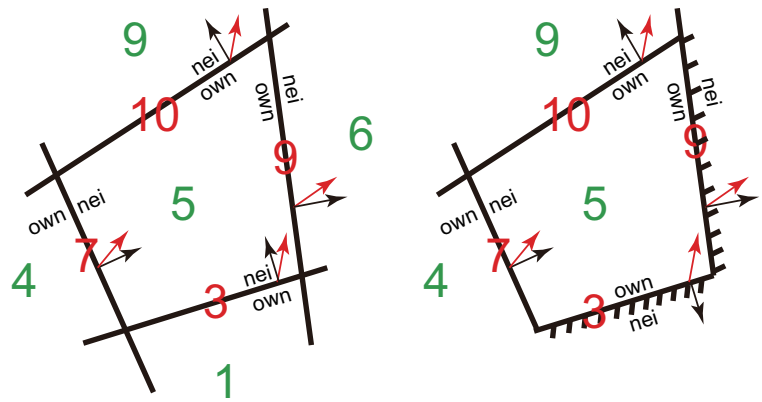


图 6.16: 左侧：内部网格单元 5 的面连接性。红箭头表示速度矢量，黑箭头表示面矢量。右侧：网格单元 5 存在两个边界面 3 与 9。

现假定网格为正交的，则方程(6.51)可以进一步写为

$$\text{surfaceIntegrate}\left(\Gamma_f(\nabla T)_f\cdot\frac{\mathbf{S}_f}{|\mathbf{S}_f|}|\mathbf{S}_f|\right)=\text{surfaceIntegrate}\left(\frac{T_{nei}-T_{own}}{|\mathbf{d}|}\Gamma_f|\mathbf{S}_f|\right)=0 \tag{6.52}$$

继续考虑图6.16中的网格单元 5，假定其所有网格面均为内部面，实施方程(6.52)有[31]：

$$-\frac{T_5-T_1}{|\mathbf{d}_{51}|}\Gamma_3|\mathbf{S}_3|-\frac{T_5-T_4}{|\mathbf{d}_{54}|}\Gamma_7|\mathbf{S}_7|+\frac{T_6-T_5}{|\mathbf{d}_{65}|}\Gamma_9|\mathbf{S}_9|+\frac{T_9-T_5}{|\mathbf{d}_{95}|}\Gamma_{10}|\mathbf{S}_{10}|=0 \tag{6.53}$$

整理后为：

$$\underbrace{\frac{\Gamma_3|\mathbf{S}_3|}{|\mathbf{d}_{51}|}}_{lower}T_1+\underbrace{\frac{\Gamma_7|\mathbf{S}_7|}{|\mathbf{d}_{54}|}}_{lower}T_4+\underbrace{\frac{\Gamma_9|\mathbf{S}_9|}{|\mathbf{d}_{65}|}}_{upper}T_6+\underbrace{\frac{\Gamma_{10}|\mathbf{S}_{10}|}{|\mathbf{d}_{95}|}}_{upper}T_9 \underbrace{-\left(\frac{\Gamma_3|\mathbf{S}_3|}{|\mathbf{d}_{51}|}+\frac{\Gamma_7|\mathbf{S}_7|}{|\mathbf{d}_{54}|}+\frac{\Gamma_9|\mathbf{S}_9|}{|\mathbf{d}_{65}|}+\frac{\Gamma_{10}|\mathbf{S}_{10}|}{|\mathbf{d}_{95}|}\right)}_{diag}T_5=0 \tag{6.54}$$

很明显，若针对所有网格点进行离散，类似的所有网格点的方程(6.54)会构成一个完全对称的矩阵。对角线元素，即为该行所有元素的加和的负值[32]。相应的代码如下[33]：

```
fvm.upper() = deltaCoeffs.internalField()*gammaMagSf.internalField();
//更新上三角矩阵，deltaCoeffs.internalField()即为面相邻的两个网格体心向量的模，gammaMagSf.internalField()为面扩散系数与网格面单元的模的乘积，在这里仅仅对upper进行初始化，如果不对lower进行初始化，默认lower=upper
```

[31]注意方程(6.53)中前两项添加的负号。这是因为第一项对应的是面 3，网格单元 5 是面 3 的 nei，则需要减去相应的贡献。第二项对应的是面 7，网格单元 5 是面 7 的 nei，同样需要减去相应的贡献。这也是 surfaceIntegrate 函数不同于 surfaceSum 函数的意义。

[32]由于为对称阵，也可以取该列所有元素，OpenFOAM 中通过 negSumDiag() 函数来实现。

[33]gaussLaplacianScheme.C

```
    fvm.negSumDiag();//按列进行非对角线元素的加和并取负值
```

现在考虑网格单元 5 存在边界面 3 与 9。在离散的过程中，网格面 7 与 10 按照上述流程正常处理。对于边界面 3，假定给定的为固定值边界条件 $T_f^{ref,3}$，在对边界面 3 离散的过程中，其离散形式为：

$$\frac{T_f^{ref,3}-T_5}{|\mathbf{d}|_{53}}\Gamma_3|\mathbf{S}_3| \tag{6.55}$$

对于网格边界面 9，给定零法向梯度边界条件，其离散形式为：

$$\frac{T_f^{9}-T_5}{|\mathbf{d}|_{59}}\Gamma_9|\mathbf{S}_9| = 0 \tag{6.56}$$

参考方程(6.53)的形式，如果网格单元 5 存在边界面，离散后的方程为：

$$\frac{T_f^{ref,3}-T_5}{|\mathbf{d}|_{53}}\Gamma_3|\mathbf{S}_3| + \frac{T_f^{9}-T_5}{|\mathbf{d}|_{59}}\Gamma_9|\mathbf{S}_9| - \frac{T_5-T_4}{|\mathbf{d}_{65}|}\Gamma_7|\mathbf{S}_7| - \frac{T_5-T_9}{|\mathbf{d}_{95}|}\Gamma_{10}|\mathbf{S}_{10}| = 0 \tag{6.57}$$

注意其中第二项为 0。可以看出，边界条件可以对边界面毗连的网格单元的矩阵产生影响。对于固定值边界条件，例如方程(6.55)，其影响主要有：

- 离散后的矩阵系数增量为 $\Gamma_d|\mathbf{S}_d|/|\mathbf{d}|_{53}$；
- 矩阵源项增量为 $\Gamma_d|\mathbf{S}_d|/|\mathbf{d}|_{53}T_f^{ref,3}$；

在代码中，边界条件带来对角线的系数增量用 `internalCoeffs_` 来表示，源项增量用 `boundaryCoeffs_` 来表示。因此，在对拉普拉斯项离散的时候，下列代码用来处理相应的增量：

```
//边界导致的对角线系数增量，一个负值
//由于未考虑负号的拉普拉斯项本身对角线系数为负
//因此考虑固定值边界条件之后，对角线系数的绝对值增加
//注意其中psf.gradientInternalCoeffs()为负值
fvm.internalCoeffs()[patchI]
  = patchGamma*psf.gradientInternalCoeffs();

//边界导致的源项增量，一个负值
//由于未考虑负号的拉普拉斯项本身无源项
//因此考虑固定值边界条件之后，源项变成一个负值
//注意其中psf.gradientBoundaryCoeffs()为正值
fvm.boundaryCoeffs()[patchI]
  = -patchGamma*psf.gradientBoundaryCoeffs();
```

6.4.4 拉普拉斯项非正交修正

在方程(6.52)中，假定网格是正交的。通常情况下，网格非正交的情况比较普遍。因此要考虑非正交修正。非正交修正的思想就是在对拉普拉斯项，假定正交隐性离散的基础上，添加一定量的非正交修正量。如图6.17所示，网格单元面法向矢量为 $\mathbf{V}$，其可以分解为正交矢量 $\mathbf{V}_{ortho}$ 以及非正交矢量 $\mathbf{V}_{unortho}$ 之和。用公式表示，即为：

$$(\nabla\phi)_f \cdot \mathbf{V} = (\nabla\phi)_f \cdot \mathbf{V}_{ortho} + (\nabla\phi)_f \cdot \mathbf{V}_{unortho} \tag{6.58}$$

其中的 $(\nabla\phi)_f \cdot \mathbf{V}_{ortho}$ 可隐性离散，$(\nabla\phi)_f \cdot \mathbf{V}_{unortho}$ 可显性离散。在这里存在若干的方法来

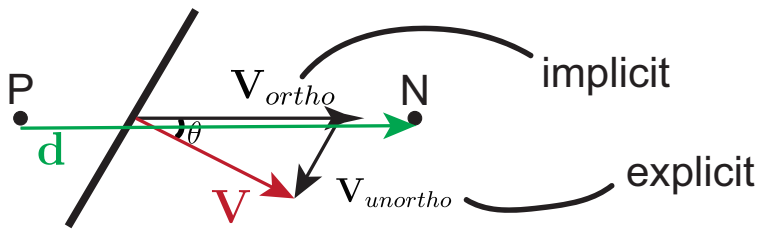


图 6.17: 非正交修正示意图。

处理 $\mathbf{V}_{ortho}$ 以及 $\mathbf{V}_{unortho}$ 的相对大小。需要注意的是，由于方程(6.58)中的 $(\nabla\phi)_f \cdot \mathbf{V}_{ortho}$ 的隐性处理，然而 $(\nabla\phi)_f \cdot \mathbf{V}_{unortho}$ 却为显性处理，其从一个已知量来进行计算，因此存在一个滞后。实操过程中，其需要进行多次迭代（非正交修正）来进行。

6.4.5 fvm::div 隐性对流项离散

现在考虑稳态离散后的对流方程：

$$\text{surfaceItegrate}\,(\phi_f T_f) = 0 \tag{6.59}$$

其中 $\phi_f = \mathbf{U}_f \cdot \mathbf{S}_f$ 为定义在面上的通量。在考虑隐性离散之前，假定所有网格单元面的通量 ϕ_f 均为 10（注意这里的通量是正号），T_f 均为 1，也即为已知的，首先我们尝试理解某个网格单元的通量守恒限定性条件的实施过程。在实施的过程中（参考6.33节的介绍），需要对面 3、7、9、10 上的通量进行处理：

- 网格单元 5 为网格单元 3 的 nei，因此要减去面 3 上 $\phi_f T_f$ 的值，即-10；
- 网格单元 5 为网格单元 7 的 nei，因此要减去面 7 上 $\phi_f T_f$ 的值，即-10；
- 网格单元 5 为网格单元 9 的 own，因此要加上面 9 上 $\phi_f T_f$ 的值，即 +10；
- 网格单元 5 为网格单元 10 的 own，因此要加上面 10 上 $\phi_f T_f$ 的值，即 +10；

因此，$(-10)+(-10)+(10)+(10)=0$，即表示流入网格单元 5 的 T 以及流出网格单元 5 的 T 是守恒的。

在隐性离散的情况下，面上的 T_f 未知，因此需要从网格单元体心插值而来。即：

$$\begin{aligned} T_{3,f} &= w_3T_1+(1-w_3)T_5 \\ T_{7,f} &= w_7T_4+(1-w_7)T_5 \\ T_{9,f} &= w_9T_5+(1-w_9)T_6 \\ T_{10,f} &= w_{10}T_5+(1-w_{10})T_9 \end{aligned} \tag{6.60}$$

其中 w 表示面上的权重。考虑最简单的情况，比如等距网格中心格式，那么 $w=1/2$。若采用不同的对流项格式，w 具有不同的值。将方程(6.60)代入到方程(6.59)中有：

$$\begin{aligned} &-(w_3T_1+(1-w_3)T_5)\phi_3-(w_7T_4+(1-w_7)T_5)\phi_7 \\ &\qquad +(w_9T_5+(1-w_9)T_6)\phi_9+(w_{10}T_5+(1-w_{10})T_9)\phi_{10}=0 \end{aligned} \tag{6.61}$$

整理有：

$$\begin{aligned} &-w_3\phi_3T_1-w_7\phi_7T_4+(1-w_9)\phi_9T_6+(1-w_{10})\phi_{10}T_9 \\ &\qquad +(-(1-w_3)\phi_3-(1-w_7)\phi_7+w_9\phi_9+w_{10}\phi_{10})T_5=0 \end{aligned} \tag{6.62}$$

参考图6.16，在针对网格单元 5 进行离散的情况下，其网格单元面分别为 3、7、9、10，因此要对图6.18中的网格单元 5 的横行处（浅灰色）红色点 3、7、9、10 进行填充（存在值）。具体的：

- 红点 3 填充的值为 $-w_3\phi_3$；
- 红点 7 填充的值为 $-w_7\phi_7$；
- 红点 9 填充的值为 $(1-w_9)\phi_9$；
- 红点 10 填充的值为 $(1-w_{10})\phi_{10}$；

以此类推，在对对流项进行隐形离散的情况下，下三角以及上三角的矩阵可以通过下面的方式进行组建[34]：

- 在下三角矩阵中，某点处的值均为 $-w_i\phi_i$（例如下三角矩阵中的红点 5 的值为 $-w_5\phi_5$）；
- 在上三角矩阵中，某点处的值均为 $(1-w_i)\phi_i$（例如上三角矩阵中的红点 5 的值为 $-(1-w_5)\phi_5$）；

相应的代码如下：

[34] gaussConvectionScheme.C

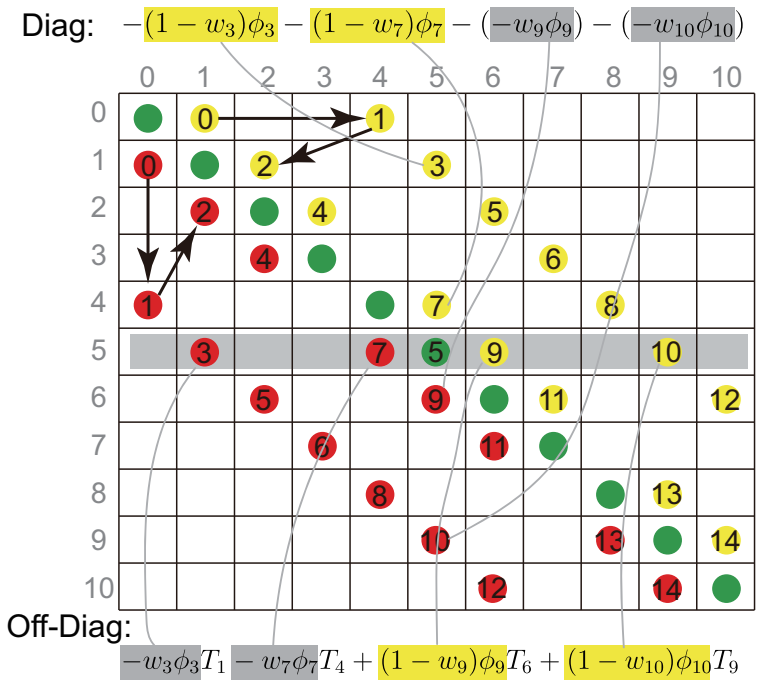


图 6.18: 网格单元 5 离散后方程与矩阵的对应性。红点上的数字为面编号。下三角矩阵从上至下排序。上三角矩阵从上至右排序。底部的方程为方程(6.62)中的非对角线项。顶部的方程为对角线项。

```
fvm.lower() = -weights.internalField()*faceFlux.internalField();
fvm.upper() = fvm.lower() + faceFlux.internalField();
```

至此，矩阵的非对角元素被组建。

现在看对角元素的组建，从方程(6.62)中可以看出对角线元素的值为：

$$-(1-w_3)\phi_3 - (1-w_7)\phi_7 - (-w_9\phi_9) - (-w_{10}\phi_{10}) \tag{6.63}$$

参考图6.18中可以看出，其为网格单元 5 所在列的元素的加和。OpenFOAM 正是通过这种方式来组建隐性对流项离散的对角线系数。相关的代码如下[35]：

```
    fvm.negSumDiag();
    ...
    for (register label face=0; face<l.size(); face++)
    {
        Diag[l[face]] -= Lower[face];
        Diag[u[face]] -= Upper[face];
    }
```

参考图6.18中顶部的公式，代码中之所以为负号是因为对角元素为相应的非对角元素的负值的加和。

[35] gaussConvectionScheme.C,lduMatrixOperations.C

6.4.6 fvm 操作符的单位

fvm对场进行操作，其会形成一个fvMatrix稀疏线性系统。因此其不能与fvc对场操作生成的场来进行一一比较。但fvm操作与fvc操作是可以进行操作符计算的，这将在第6.5.4节进行介绍。fvm对场操作后生成的fvMatrix稀疏线性系统的单位，在代码植入的过程中并没有统一的处理方式，但是其单位是fvc对场操作生成的场的单位乘以一个网格体积的单位。只有在这种情况下，fvm与fvc进行操作符操作后在数学上才是相符的。

6.5 OpenFOAM 中的 fvMatrix 稀疏线性系统

6.5.1 fvMatrix 之 Gauss-Seidel 求解器

在构建整个稀疏显性系统 $\mathbf{A}\boldsymbol{\psi} = \mathbf{b}$ 之后，需要对其求解。OpenFOAM 中提供了各种不同的方法进行求解，本节讨论其中的 smoother 类之 Gauss-Seidel 迭代求解方法。假定矩阵 $\mathbf{A}$ 具有如下形式：

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,0} & a_{n,1} & \cdots & a_{n,n} \end{bmatrix} \tag{6.64}$$

那么 Gauss-Seidel 迭代方法的解 $\boldsymbol{\psi}$ 可以表示为：

$$\psi_i^{(k+1)} = \frac{1}{a_{ii}} b_i - \frac{1}{a_{ii}} \sum_{j=i+1}^{n} a_{ij}\psi_j^{(k)} - \frac{1}{a_{ii}} \sum_{j=0}^{i} a_{ij}\psi_j^{(k+1)}, \quad i = 1, 2, \ldots, n \tag{6.65}$$

也可以用文字直观地描述如下：

1. 首先给定初始值 $\psi_0^{(k)}, ..., \psi_n^{(k)}$；
2. 通过 $\psi_1^{(k)}, ..., \psi_n^{(k)}$，来计算 $\psi_0^{(k+1)}$；
3. 通过 $\psi_0^{(k+1)}$、$\psi_2^{(k)}, ..., \psi_n^{(k)}$，来计算 $\psi_1^{(k+1)}$；
4. 以此类推...；
5. 通过 $\psi_0^{(k+1)}, ..., \psi_{n-2}^{(k+1)}$、$\psi_n^{(k)}$，来计算 $\psi_{n-1}^{(k+1)}$；
6. 通过 $\psi_0^{(k+1)}, ..., \psi_{n-1}^{(k+1)}$，来计算 $\psi_n^{(k+1)}$；
7. 回到第一步，进行下一次迭代；

如果以网格单元 5 为范例，有：

$$\psi_5^{(k+1)} = \frac{1}{a_{55}} b_5 - \frac{1}{a_{55}} a_{56}\psi_6^{(k)} - \frac{1}{a_{55}} a_{59}\psi_9^{(k)} - \frac{1}{a_{55}} a_{51}\psi_1^{(k+1)} - \frac{1}{a_{55}} a_{54}\psi_4^{(k+1)} \tag{6.66}$$

仔细观察，会发现矩阵的上三角元素，都是乘以已知的值，例如 a_{56} 是上三角元素，其乘以的 $\psi_6^{(k)}$ 为当前迭代步 k 的已知值。a_{59} 是上三角元素，其乘以的 $\psi_9^{(k)}$ 为当前迭代步 k 的已知值。

更具体地，ψ_0^{k+1} 可以通过下式计算 [36]：

$$\psi_0^{k+1} = \frac{1}{a_{00}}\left(b_0 \underbrace{-a_{01}\psi_1^{(k)} - \ldots - a_{0n}\psi_n^{(k)}}_{code1} \underbrace{-0}_{code2}\right) \tag{6.67}$$

其中 $a_{01}, \ldots, a_{0n}$ 为上三角元素（对应`upperPtr`）。有了 ψ_0^{k+1} 后，可以获得 ψ_1^{k+1} [37]：

$$\psi_1^{k+1} = \frac{1}{a_{11}}\left(b_1 \underbrace{-a_{02}\psi_2^{(k)} - \ldots - a_{0n}\psi_n^{(k)}}_{code1} \underbrace{-a_{10}\psi_0^{(k+1)}}_{code2}\right) \tag{6.68}$$

其中 $a_{02}, \ldots, a_{0n}$ 为上三角元素（对应`upperPtr`），a_{10} 为下三角元素（对应`lowerPtr`）。有了 $\psi_0^{k+1}, \psi_1^{k+1}$ 后，可以获得 ψ_2^{k+1} [38]：

$$\psi_2^{k+1} = \frac{1}{a_{22}}\left(b_2 \underbrace{-a_{03}\psi_3^{(k)} - \ldots - a_{0n}\psi_n^{(k)}}_{code1} \underbrace{-a_{10}\psi_0^{(k+1)} - a_{11}\psi_1^{(k+1)}}_{code2}\right) \tag{6.69}$$

随后以此类推，获得 $k+1$ 次的迭代解。OpenFOAM 中的代码片段如下

```
    bPrime = source;
    ...
    // code2
    psii = bPrimePtr[celli];
    ...
    // code1
    for (label facei=fStart; facei<fEnd; facei++)
    {
        psii -= upperPtr[facei]*psiPtr[uPtr[facei]];
    }

    psii /= diagPtr[celli];
```

36

$$\psi_0^{k+1} = \frac{1}{a_{00}}b_0 - \frac{1}{a_{00}}\sum_{j=1}^{n} a_{0j}\psi_j^{(k)}, \quad i = 1, 2, \ldots, n$$

37

$$\psi_1^{k+1} = \frac{1}{a_{11}}b_1 - \frac{1}{a_{11}}\sum_{j=2}^{n} a_{0j}\psi_j^{(k)} - \frac{1}{a_{11}}\sum_{j=0}^{1} a_{j0}\psi_j^{(k+1)}, \quad i = 1, 2, \ldots, n$$

38

$$\psi_2^{k+1} = \frac{1}{a_{22}}b_2 - \frac{1}{a_{22}}\sum_{j=3}^{n} a_{0j}\psi_j^{(k)} - \frac{1}{a_{22}}\sum_{j=0}^{2} a_{j0}\psi_j^{(k+1)}, \quad i = 1, 2, \ldots, n$$

```

    // code2
    for (label facei=fStart; facei<fEnd; facei++)
    {
        bPrimePtr[uPtr[facei]] -= lowerPtr[facei]*psii;
    }

    psiPtr[celli] = psii;
```

6.5.2 fvMatrix 之 PCG、PBiCG 求解器

PCG 求解器为预条件共轭梯度求解器。共轭梯度类求解器（CG）的求解思想有比较直观的几何描述。这一类求解器通过寻找在等值面上的最快的下降方向，获得很快的收敛速度。在这里 PCG 的几何思想不会被介绍，仅使用一例来归纳 CG（不带预条件）以及 PCG 的数学方法。

考虑一个稀疏线性系统 $\mathbf{A}\cdot\mathbf{x}=\mathbf{b}$[39]。其中：

$$\mathbf{A}\cdot\mathbf{x}=\begin{bmatrix}4 & 1\\ 1 & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1\\ 2\end{bmatrix} \tag{6.70}$$

假设 $\mathbf{x}$ 的初始值为：

$$\mathbf{x}_0=\begin{bmatrix}2\\ 1\end{bmatrix} \tag{6.71}$$

首先，要计算初始残差 $\mathbf{r}_0=\mathbf{b}-\mathbf{A}\cdot\mathbf{x}_0$:

$$\mathbf{r}_0=\begin{bmatrix}1\\ 2\end{bmatrix}-\begin{bmatrix}4 & 1\\ 1 & 3\end{bmatrix}\begin{bmatrix}2\\ 1\end{bmatrix}=\begin{bmatrix}-8\\ -3\end{bmatrix}=\mathbf{p}_0 \tag{6.72}$$

其中的 $\mathbf{p}_0$ 被当做是下降的方向。上述的步骤是必要的且不需要进行迭代。

下一步计算 α_0:

$$\alpha_0=\frac{\mathbf{r}_0^\top\mathbf{r}_0}{\mathbf{p}_0^\top\mathbf{A}\mathbf{p}_0}=\frac{\begin{bmatrix}-8 & -3\end{bmatrix}\begin{bmatrix}-8\\ -3\end{bmatrix}}{\begin{bmatrix}-8 & -3\end{bmatrix}\begin{bmatrix}4 & 1\\ 1 & 3\end{bmatrix}\begin{bmatrix}-8\\ -3\end{bmatrix}}=\frac{73}{331}\approx 0.2205 \tag{6.73}$$

有了 α_0 之后，可以更新 $\mathbf{x}_1$：

$$\mathbf{x}_1=\mathbf{x}_0+\alpha_0\mathbf{p}_0=\begin{bmatrix}2\\ 1\end{bmatrix}+\frac{73}{331}\begin{bmatrix}-8\\ -3\end{bmatrix}\approx\begin{bmatrix}0.2356\\ 0.3384\end{bmatrix} \tag{6.74}$$

[39] 本算例参考 wiki 百科相关内容

同时可以通过更新 $\mathbf{r}_1$:

$$\mathbf{r}_1 = \mathbf{r}_0 - \alpha_0 \mathbf{A}\mathbf{p}_0 = \begin{bmatrix} -8 \\ -3 \end{bmatrix} - \frac{73}{331} \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} -8 \\ -3 \end{bmatrix} \approx \begin{bmatrix} -0.2810 \\ 0.7492 \end{bmatrix} \tag{6.75}$$

然后计算 β_0，其被用来计算下一次的下降方向 $\mathbf{p}_1$:

$$\beta_0 = \frac{\mathbf{r}_1^\top \mathbf{r}_1}{\mathbf{r}_0^\top \mathbf{r}_0} \approx \frac{\begin{bmatrix} -0.2810 & 0.7492 \end{bmatrix} \begin{bmatrix} -0.2810 \\ 0.7492 \end{bmatrix}}{\begin{bmatrix} -8 & -3 \end{bmatrix} \begin{bmatrix} -8 \\ -3 \end{bmatrix}} = 0.0088 \tag{6.76}$$

$$\mathbf{p}_1 = \mathbf{r}_1 + \beta_0 \mathbf{p}_0 \approx \begin{bmatrix} -0.2810 \\ 0.7492 \end{bmatrix} + 0.0088 \begin{bmatrix} -8 \\ -3 \end{bmatrix} = \begin{bmatrix} -0.3511 \\ 0.7229 \end{bmatrix} \tag{6.77}$$

类似的，通过方程(6.73)来计算 α_1：

$$\alpha_1 = \frac{\mathbf{r}_1^\top \mathbf{r}_1}{\mathbf{p}_1^\top \mathbf{A}\mathbf{p}_1} \approx \frac{\begin{bmatrix} -0.2810 & 0.7492 \end{bmatrix} \begin{bmatrix} -0.2810 \\ 0.7492 \end{bmatrix}}{\begin{bmatrix} -0.3511 & 0.7229 \end{bmatrix} \begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} -0.3511 \\ 0.7229 \end{bmatrix}} = 0.4122 \tag{6.78}$$

有了 α_1 之后，可以通过方程(6.74)来更新 $\mathbf{x}$:

$$\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1 \approx \begin{bmatrix} 0.2356 \\ 0.3384 \end{bmatrix} + 0.4122 \begin{bmatrix} -0.3511 \\ 0.7229 \end{bmatrix} = \begin{bmatrix} 0.0909 \\ 0.6364 \end{bmatrix} \tag{6.79}$$

总体来说，CG 方法可以归纳为下述迭代步：

1. 预估初始的 $\mathbf{x}$，计算初始残差 $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$，$\mathbf{p}_0 = \mathbf{r}_0$;
2. 计算 $\alpha_0 = \frac{\mathbf{r}_0^\top \mathbf{r}_0}{\mathbf{p}_0^\top \mathbf{A}\mathbf{p}_0}$;
3. 更新 $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0$；
4. 计算 $\mathbf{r}_1 = \mathbf{r}_0 - \alpha_0 \mathbf{A}\mathbf{p}_0$;
5. 计算 $\beta_0 = \frac{\mathbf{r}_1^\top \mathbf{r}_1}{\mathbf{r}_0^\top \mathbf{r}_0}$;
6. 计算 $\mathbf{p}_1 = \mathbf{r}_1 + \beta_0 \mathbf{p}_0$;
7. 更新 $\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1$; 回到第二步直到收敛为止；

CG 算法的收敛性很大部分取决于矩阵 $\mathbf{A}$ 的条件数。在一些情况下，$\mathbf{A}$ 的收敛性比较差。在这种情况下，可以将原始问题 $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ 转化为 $(\mathbf{P}^{-1}\mathbf{A}) \cdot \mathbf{x} = \mathbf{P}^{-1}\mathbf{b}$，其中 $(\mathbf{P}^{-1}\mathbf{A})$ 具有更好的收敛性。PCG 求解器在 CG 求解器的基础上进行预条件。计算方法可以归纳为：

1. 预估初始的 $\mathbf{x}$，计算初始残差 $\mathbf{r}_0 = \mathbf{b} - \mathbf{A} \cdot \mathbf{x}_0$，$\mathbf{z}_0 = \mathbf{M}^{-1}\mathbf{r}_0$，$\mathbf{p}_0 = \mathbf{z}_0$;
2. 计算 $\alpha_0 = \frac{\mathbf{r}_0^\top \mathbf{z}_0}{\mathbf{p}_0^\top \mathbf{A}\mathbf{p}_0}$;
3. 更新 $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0$;
4. 计算 $\mathbf{r}_1 = \mathbf{r}_0 - \alpha_0 \mathbf{A}\mathbf{p}_0$;
5. 计算 $\mathbf{z}_1 = \mathbf{M}^{-1}\mathbf{r}_1$
6. 计算 $\beta_0 = \frac{\mathbf{r}_1^\top \mathbf{z}_1}{\mathbf{r}_0^\top \mathbf{z}_0}$;
7. 计算 $\mathbf{p}_1 = \mathbf{z}_1 + \beta_0 \mathbf{p}_0$;
8. 更新 $\mathbf{x}_2 = \mathbf{x}_1 + \alpha_1 \mathbf{p}_1$; 回到第二步直到收敛为止；

那么接下来的问题就是，如何确定一个矩阵的预条件矩阵？在这里有多钟不同的方法。比如不完全 LU 分解（ILU），不完全 Cholesky 分解（IC）、对角不完全 LU 分解（DILU）以及对角不完全 Cholesky 分解（DIC）等。OpenFOAM 中比较常见的为 DILU 以及 DIC，后者可以理解为前者的对称版本，即可以用于对称矩阵。DILU 分解将矩阵系数 $\mathbf{A}$ 分解为对角矩阵 $\mathbf{D}$，上三角矩阵 $\mathbf{U}$ 以下三角矩阵 $\mathbf{L}$。DILU 构建的预条件矩阵为 [181]：

$$\mathbf{P} = (\mathbf{E} + \mathbf{L})\mathbf{E}^{-1}(\mathbf{E} + \mathbf{U}) \tag{6.80}$$

其中 $\mathbf{E}$ 的计算方法为：

$$\mathbf{P} \tag{6.81}$$

另外一个问题是，PCG 算法的矩阵 $\mathbf{A}$ 必须是对称且正定的（例如拉普拉斯方程离散出来的即为对称矩阵）。如果 $\mathbf{A}$ 是非对称矩阵（例如含有对流项的离散方程），则可以使用 BiCG 求解器，在附加预条件的情况下，衍生为 PBiCG 求解器。BiCG 类求解器的稳定性相对较差，为了更好的稳定性，可以使用 PBiCGStab 求解器。这些求解器的算法与 PCG 的迭代算法类似，但越来越复杂，本资料在此不做详细介绍。目前的 OpenFOAM 可用的求解器均为附加预处理的求解器，如 PBiCG 以及 PBiCGStab。仅仅在非常老的 OpenFOAM 中存在的 BICG 等求解器。

6.5.3 fvMatrix 的 H()、H1()

OpenFOAM 中 fvMatrix 的 H() 函数均涉及到一些矩阵的操作[40]。fvMatrix 的 H() 函数返回 lduMatrix 的 H() 函数的值，同时加上 fvMatrix 的源项。在这里看 lduMatrix 的

[40] fvMatrix.C

H() 函数。H() 函数与 OpenFOAM 中计算通量的代码非常相似，原理上是将临点的贡献进行加和并取负。例如，若有下述离散方程：

$$a_P\psi_P + \sum a_N\psi_N = S \tag{6.82}$$

lduMatrix 中 H() 操作符的数学表达形式即为：

$$\mathrm{H}() = -\sum a_N\psi_N \tag{6.83}$$

fvMatrix 中 H() 操作符的数学表达形式即为：

$$\mathrm{H}() = -\sum a_N\psi_N + S \tag{6.84}$$

lduMatrix 中 H1() 操作符用于将矩阵的非对角线元素加和并取负。其数学表达形式即为：

$$\mathrm{H1}() = -\sum a_N \tag{6.85}$$

下面即 lduMatrix 中 H() 操作符的代码：

```
    if (lowerPtr_ || upperPtr_)
    {
        Field<Type> & Hpsi = tHpsi();

        Type* __restrict__ HpsiPtr = Hpsi.begin();

        const Type* __restrict__ psiPtr = psi.begin();

        const label* __restrict__ uPtr = lduAddr().upperAddr().begin();
        const label* __restrict__ lPtr = lduAddr().lowerAddr().begin();

        const scalar* __restrict__ lowerPtr = lower().begin();
        const scalar* __restrict__ upperPtr = upper().begin();

        const label nFaces = upper().size();

        for (label face=0; face<nFaces; face++)
        {
            HpsiPtr[uPtr[face]] -= lowerPtr[face]*psiPtr[lPtr[face]];
            HpsiPtr[lPtr[face]] -= upperPtr[face]*psiPtr[uPtr[face]];
        }
    }
```

下面即 lduMatrix 中 H1() 操作符的代码，可见其与 H() 函数的代码非常类似。

```
if (lowerPtr_ || upperPtr_)
```

```
    {
        scalarField& H1_ = tH1();
        ...
        for (label face=0; face<nFaces; face++)
        {
            H1Ptr[uPtr[face]] -= lowerPtr[face];
            H1Ptr[lPtr[face]] -= upperPtr[face];
        }
    }
```

6.5.4 fvMatrix 的操作符重载

fvMatrix对很多操作符进行了重载，本节讨论“减法”操作符。以fvScalarMatrix举例，Tbyt为体标量场 T/t，单位为 T 的单位除以时间 t 的单位。求解器中可以定义下面的函数：

```
fvScalarMatrix TEqn(fvm::ddt(T) - Tbyt);
```

以三网格系统为例，若考虑全部为零法向梯度边界条件，上述代码首先通过fvm::ddt(T)构建矩阵：

$$\begin{bmatrix} \frac{\Delta V}{\Delta t} & 0 & 0 \\ 0 & \frac{\Delta V}{\Delta t} & 0 \\ 0 & 0 & \frac{\Delta V}{\Delta t} \end{bmatrix} \begin{bmatrix} T_0^{t+\Delta t} \\ T_1^{t+\Delta t} \\ T_2^{t+\Delta t} \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t} T_0^t \\ \frac{\Delta V}{\Delta t} T_1^t \\ \frac{\Delta V}{\Delta t} T_2^t \end{bmatrix} \tag{6.86}$$

随后，- Tbyt将其影响添加至方程 Eq. (6.86) 右侧的源项中：

$$\begin{bmatrix} \frac{\Delta V}{\Delta t} & 0 & 0 \\ 0 & \frac{\Delta V}{\Delta t} & 0 \\ 0 & 0 & \frac{\Delta V}{\Delta t} \end{bmatrix} \begin{bmatrix} T_0^{t+\Delta t} \\ T_1^{t+\Delta t} \\ T_2^{t+\Delta t} \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t} T_0^t + \Delta V T/t \\ \frac{\Delta V}{\Delta t} T_1^t + \Delta V T/t \\ \frac{\Delta V}{\Delta t} T_2^t + \Delta V T/t \end{bmatrix} \tag{6.87}$$

因此，其数学形式即为：

$$\frac{\partial T}{\partial t} - T/t = 0 \tag{6.88}$$

“减法”操作符对应的源代码如下[41]：

```
template<class Type>
Foam::tmp<Foam::fvMatrix<Type> > Foam::operator-
(
    const fvMatrix<Type>& A,
    const dimensioned<Type>& su
)
{
```

[41] fvMatrix.C

```
    checkMethod(A, su, "-");
    tmp<fvMatrix<Type> > tC(new fvMatrix<Type>(A));
    tC().source() += su.value()*tC().psi().mesh().V();
    return tC;
}
```

注意，在方程(6.87)中，- Tbyt的贡献要乘上网格体积，其在上方的代码中也有体现。类似的，如果代码为

```
fvScalarMatrix TEqn(fvm::ddt(T) - fvc::ddt(T));
```

其中 fvc::ddt(T)离散后的数学形式为 $\Delta T/\Delta t$，同样要乘上网格体积才能进行操作符减法操作。

6.5.5 fvMatrix 的 correction()

OpenFOAM 在处理瞬态可压缩流动的情况下，经常会调用下面的函数来组建矩阵：

```
fvScalarMatrix pDDtEqn
(
    fvc::ddt(rho) + psi*correction(fvm::ddt(p))
  + fvc::div(phiHbyA) + fvm::div(phid, p)
);
```

本节主要讨论其中的 correction() 函数。依据 correction() 函数的定义：

```
tmp<Foam::fvMatrix<Type>> tAcorr = tA - (tA() & tA().psi());

return tAcorr;
```

可以看出计算流程为：1）通过 fvm:ddt(p) 来构建一个压力离散的对角矩阵，2）correction() 函数将对这个离散矩阵的系数进行修正，返回一个离散对角矩阵。

以三网格系统为例，fvm:ddt(p) 离散后的矩阵为：

$$\begin{bmatrix} \frac{\Delta V}{\Delta t} & 0 & 0 \\ 0 & \frac{\Delta V}{\Delta t} & 0 \\ 0 & 0 & \frac{\Delta V}{\Delta t} \end{bmatrix} \begin{bmatrix} p_0^* \\ p_1^* \\ p_2^* \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t} p_0^t \\ \frac{\Delta V}{\Delta t} p_1^t \\ \frac{\Delta V}{\Delta t} p_2^t \end{bmatrix} \tag{6.89}$$

其中 p^* 表示待求变量，fvm:ddt(p) 的 source 为方程(6.89)右侧的变量。方程(6.89)即为 fvm:ddt(p) 离散后形成的矩阵，也为传入 correction() 函数代码中的 tA。

下面细分 tA() & tA().psi() 操作。首先，tA().psi() 表示下面的源项：

$$\begin{bmatrix} p_0^t \\ p_1^t \\ p_2^t \end{bmatrix} \tag{6.90}$$

tA().psi() 表示当前的压力。tA() & tA().psi() 是将离散矩阵与 tA().psi() 进行一个乘积操作，其通过在 fvMatrix.C 文件中定义一个 & 操作符来进行。tA() & tA().psi() 分为两步，第一步的简要代码如下：

```
Mphi.primitiveFieldRef().replace(cmpt, -boundaryDiagCmpt*psiCmpt);
```

其将创造一个体标量场，值为：

$$\begin{bmatrix} -\frac{\Delta V}{\Delta t}p_0^t \\ -\frac{\Delta V}{\Delta t}p_1^t \\ -\frac{\Delta V}{\Delta t}p_2^t \end{bmatrix} \tag{6.91}$$

第二步的简要代码如下：

```
Mphi.primitiveFieldRef() += M.lduMatrix::H(psi.field()) + M.source();
```

其在第一步的基础上将，将第一步创建的体标量场更改为[42]：

$$\begin{bmatrix} -\frac{\Delta V}{\Delta t}p_0^t + \frac{\Delta V}{\Delta t}p_0^t \\ -\frac{\Delta V}{\Delta t}p_1^t + \frac{\Delta V}{\Delta t}p_1^t \\ -\frac{\Delta V}{\Delta t}p_2^t + \frac{\Delta V}{\Delta t}p_2^t \end{bmatrix} \tag{6.92}$$

第三部的的简要代码如下：

```
Mphi.primitiveFieldRef() /= -psi.mesh().V();
```

其将构造的体标量场更改为：

$$\begin{bmatrix} \frac{1}{\Delta t}p_0^t - \frac{1}{\Delta t}p_0^t \\ \frac{1}{\Delta t}p_1^t - \frac{1}{\Delta t}p_1^t \\ \frac{1}{\Delta t}p_2^t - \frac{1}{\Delta t}p_2^t \end{bmatrix} \tag{6.93}$$

然后，继续看 tA - (tA() & tA().psi()) 操作。其为一个 fvMatrix 与一个体标量场相减的操作，其中减法操作符可以参考第6.5.4节。其会返回如下矩阵：

$$\begin{bmatrix} \frac{\Delta V}{\Delta t} & 0 & 0 \\ 0 & \frac{\Delta V}{\Delta t} & 0 \\ 0 & 0 & \frac{\Delta V}{\Delta t} \end{bmatrix} \begin{bmatrix} p_0^* \\ p_1^* \\ p_2^* \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t}p_0^t + \left(\frac{\Delta V}{\Delta t}p_0^t - \frac{\Delta V}{\Delta t}p_0^t\right) \\ \frac{\Delta V}{\Delta t}p_1^t + \left(\frac{\Delta V}{\Delta t}p_1^t - \frac{\Delta V}{\Delta t}p_1^t\right) \\ \frac{\Delta V}{\Delta t}p_2^t + \left(\frac{\Delta V}{\Delta t}p_2^t - \frac{\Delta V}{\Delta t}p_2^t\right) \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t}p_0^t \\ \frac{\Delta V}{\Delta t}p_1^t \\ \frac{\Delta V}{\Delta t}p_2^t \end{bmatrix} \tag{6.94}$$

可见，在当前时间步下，第一次调用 correction(fvm::ddt(p)) 与 fvm::ddt(p) 是相同的。

然而，在 PISO 算法中，需要对压力方程进行多次求解。从第二步开始，correction(fvm::ddt(p)) 与 fvm::ddt(p) 将是有区别的。如果进行了压力求解，

[42] H() 操作符的元素为空，因为时间项不构造非对角线系数。

然后继续进行 `correction(fvm::ddt(p))` 的植入，那么在方程(6.91)中创立的则是

$$\begin{bmatrix} -\frac{\Delta V}{\Delta t}p_0^* \\ -\frac{\Delta V}{\Delta t}p_1^* \\ -\frac{\Delta V}{\Delta t}p_2^* \end{bmatrix} \tag{6.95}$$

方程(6.92)中创立的则变为：

$$\begin{bmatrix} \frac{1}{\Delta t}p_0^* + \frac{1}{\Delta t}p_0^t \\ \frac{1}{\Delta t}p_1^* + \frac{1}{\Delta t}p_1^t \\ \frac{1}{\Delta t}p_2^* + \frac{1}{\Delta t}p_2^t \end{bmatrix} \tag{6.96}$$

最终，`correction(fvm::ddt(p))` 将会返回如下矩阵：

$$\begin{bmatrix} \frac{\Delta V}{\Delta t} & 0 & 0 \\ 0 & \frac{\Delta V}{\Delta t} & 0 \\ 0 & 0 & \frac{\Delta V}{\Delta t} \end{bmatrix} \begin{bmatrix} p_0^{**} \\ p_1^{**} \\ p_2^{**} \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t}p_0^t + \left(\frac{1}{\Delta t}p_0^* - \frac{1}{\Delta t}p_0^t\right) \\ \frac{\Delta V}{\Delta t}p_1^t + \left(\frac{1}{\Delta t}p_1^* - \frac{1}{\Delta t}p_1^t\right) \\ \frac{\Delta V}{\Delta t}p_2^t + \left(\frac{1}{\Delta t}p_2^* - \frac{1}{\Delta t}p_2^t\right) \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t}p_0^* \\ \frac{\Delta V}{\Delta t}p_1^* \\ \frac{\Delta V}{\Delta t}p_2^* \end{bmatrix} \tag{6.97}$$

以此类推，在同一个时间步下，如果再次进行了压力方程求解，则 `correction(fvm::ddt(p))` 变为

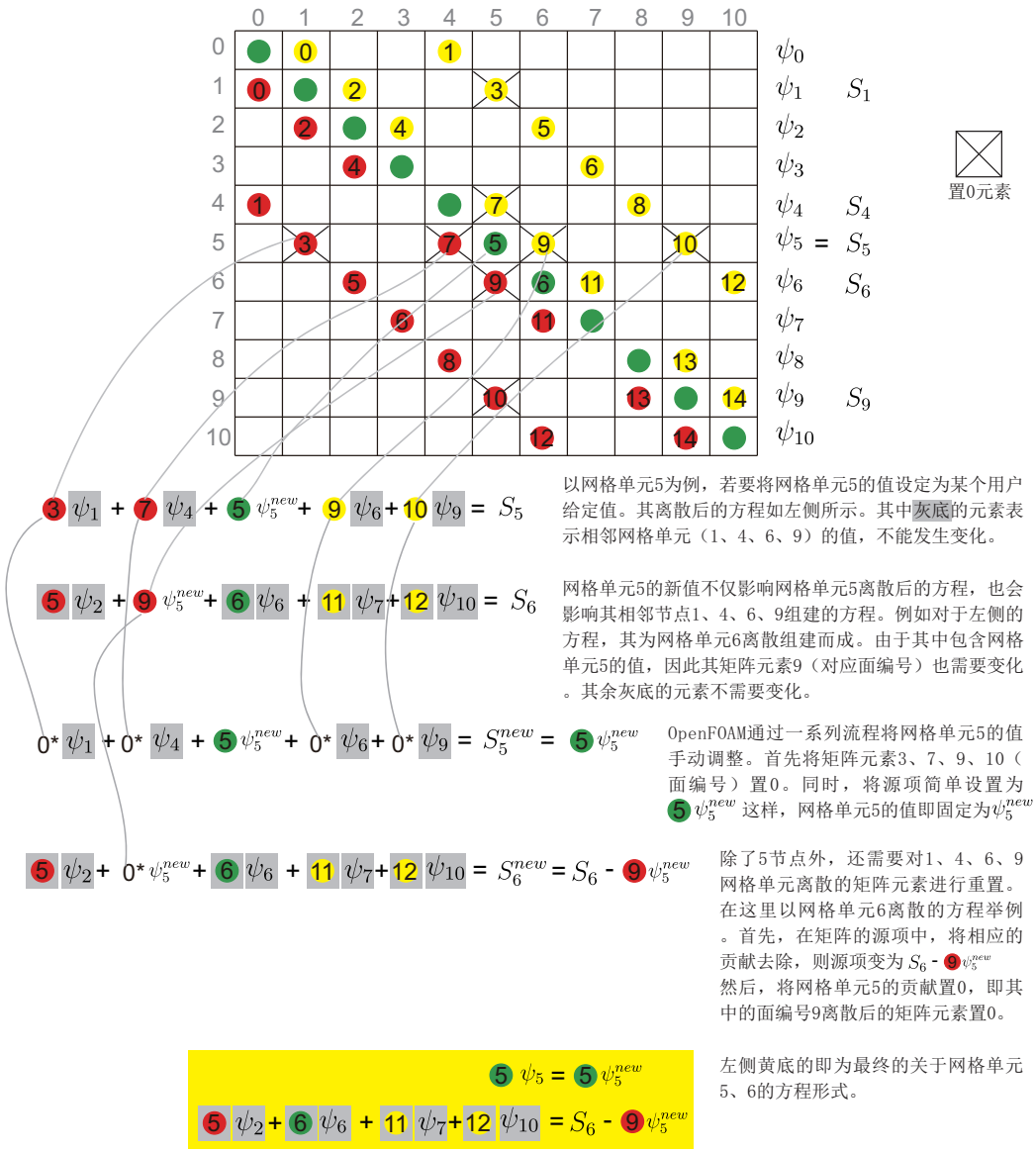
$$\begin{bmatrix} \frac{\Delta V}{\Delta t} & 0 & 0 \\ 0 & \frac{\Delta V}{\Delta t} & 0 \\ 0 & 0 & \frac{\Delta V}{\Delta t} \end{bmatrix} \begin{bmatrix} p_0^{***} \\ p_1^{***} \\ p_2^{***} \end{bmatrix} = \begin{bmatrix} \frac{\Delta V}{\Delta t}p_0^{**} \\ \frac{\Delta V}{\Delta t}p_1^{**} \\ \frac{\Delta V}{\Delta t}p_2^{**} \end{bmatrix} \tag{6.98}$$

可见，`correction()` 函数类似于 `fvm::ddt()` 函数，不同的是 `fvm::ddt()` 在同一个时间步之内，一直表示的为 $(p^* - p^t)/\Delta t$，其中 p^* 表示待求变量。若进行一次压力求解，`fvm::ddt()` 变为 $(p^{**} - p^t)/\Delta t$。继续进行压力求解，则变为 $(p^{***} - p^t)/\Delta t$。`correction(fvm::ddt())` 则在最初表示的为 $(p^* - p^t)/\Delta t$。进行一次压力求解，变成 $(p^{**} - p^*)/\Delta t$。再进行一次压力求解，变成 $(p^{***} - p^{**})/\Delta t$。同时也可以看出，在最终一个时间步收敛的情况下，$p^{***}$ 趋向于 p^{**}，因此 `correction(fvm::ddt())` 作用趋向于 0。

6.5.6 fvMatrix 中的 setValue()

通过对 OpenFOAM 中的矩阵操作，可以将某个网格单元的值设置为固定值。即 OpenFOAM 中矩阵的 setValues() 函数要实现的功能。结合图6.16与6.18，现假定要将网格单元 5 的值设置为一个给定的参数。如果形成的矩阵为一个对角阵，那么不需要考虑图6.18中的对角阵，只需要将矩阵的源项设定为矩阵的对角线系数乘以需要设定的值即可。比如网格单元 5 要给定的值为 10，其离散后的矩阵元素值为 3，那么将源项设定为 30 即可。对于非对角矩阵，OpenFOAM 通过一些列的操作可以实现将某个网格单元的值设置为固定值的功能。具体的实现原理可参考图6.19[43]。

[43] fvMatrix.C



以网格单元5为例，若要将网格单元5的值设定为某个用户给定值。其离散后的方程如左侧所示。其中灰底的元素表示相邻网格单元（1、4、6、9）的值，不能发生变化。

网格单元5的新值不仅影响网格单元5离散后的方程，也会影响其相邻节点1、4、6、9组建的方程。例如对于左侧的方程，其为网格单元6离散组建而成。由于其中包含网格单元5的值，因此其矩阵元素9（对应面编号）也需要变化。其余灰底的元素不需要变化。

OpenFOAM通过一系列流程将网格单元5的值手动调整。首先将矩阵元素3、7、9、10（面编号）置0。同时，将源项简单设置为 $5\ \psi_5^{new}$ 这样，网格单元5的值即固定为ψ_5^{new}

除了5节点外，还需要对1、4、6、9网格单元离散的矩阵元素进行重置。在这里以网格单元6离散的方程举例。首先，在矩阵的源项中，将相应的贡献去除，则源项变为 $S_6 - 9\ \psi_5^{new}$ 然后，将网格单元5的贡献置0，即其中的面编号9离散后的矩阵元素置0。

左侧黄底的即为最终的关于网格单元5、6的方程形式。

图 6.19: 将网格单元 5 的值设置为固定值的流程。

6.5.7 fvMatrix 的单位

OpenFOAM 中的 fvMatrix 的单位为待求方程项的单位乘以体积单位。现以下面的代码举例：

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)
);
```

上述代码通过 fvm::ddt(U) 来构建 UEqn 的单位[44]。其中未知的方程项为 $\frac{\partial \mathbf{U}}{\partial t}$。在 fvm::ddt() 函数中，会调用不同的离散格式，若采用欧拉方法，其单位被定义为速度的单位乘以体积单位再除以时间单位[45]：

```
tmp<fvMatrix<Type> > tfvm
(
    new fvMatrix<Type>
    (
        vf,
        vf.dimensions()*dimVol/dimTime
        //速度的单位乘以体积单位再除以时间单位
    )
);
```

若考虑下面的代码：

```
fvVectorMatrix UEqn
(
    fvm::ddt(U) + fvm::div(phi,U)
);
```

其单位同样是速度的单位乘以体积单位再除以时间单位，因为对流项与时间项的单位必然是一致的。另一方面，在对流项离散的代码中：

```
tmp<fvMatrix<Type> > tfvm
(
    new fvMatrix<Type>
    (
        vf,
        faceFlux.dimensions()*vf.dimensions()
        //速度的单位乘以面积的单位是faceFlux.dimensions()
        //再乘以速度的单位
    )
```

[44] fvmDdt.H

[45] EulerDdtScheme.C

```
    );
```

速度的单位乘以面积的单位再乘以速度的单位，与时间项离散的单位是一致的，均为 $m^4 s^{-2}$。

fvMatrix 可以进行一些数学操作（例如加减乘除），当 fvMatrix 与 fvMatrix 进行数学操作的时候，fvMatrix 的单位不变。fvMatrix 还可以与其他类型的变量进行数学才做，比如 fvMatrix 可以与 dimensionedField 进行数学操作，例如以等号举例，有源代码[46]：

```
template<class Type>
Foam::tmp<Foam::fvMatrix<Type>> Foam::operator+
(
    const fvMatrix<Type>& A,
    const DimensionedField<Type, volMesh>& su
)
{
    checkMethod(A, su, "+");
    tmp<fvMatrix<Type>> tC(new fvMatrix<Type>(A));
    tC.ref().source() -= su.mesh().V()*su.field();
    return tC;
}
```

可见，在进行赋值操作的时候，需要将 dimensionedField 每个网格上的值乘上网格体积后，再在 fvMatrix 的源项上面将其减掉。在这里同样要注意加减号的问题。

6.5.8 Bandwidth 带宽

给定一个稀疏线性系统，其具有一个特征量叫做带宽。例如对于下面的两个矩阵，他们的带宽可以理解为元素延拓的幅度。很明显，左边的矩阵带宽较大，右边的矩阵带宽较小。

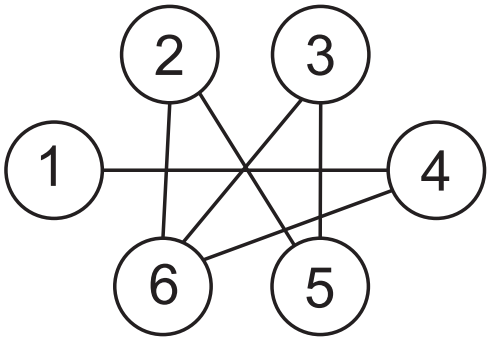
$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

对于大型稀疏线性系统，计算人员通常诉求带宽尽可能的小，来进一步的形成条带状稀疏线性系统。同时有资料表示 [6]，带宽减小算法并不能够改变 Krylov 子空间迭代求解器的效率，但可以增加 LU 分解类求解器的效率，但后者在 CFD 中用的较少。对于 Krylov 子

[46] fvMatrix.C

空间迭代求解器，减少带宽最重要的作用是做 CPU 的缓存优化。即减少带宽可以增加缓存命中率。这种效率的提升更倾向于是硬件效率的改变，而非算法效率类提升。

OpenFOAM 中可以调用不同的算法来减小稀疏线性系统的带宽。比如 Cuthill and McKee 方法 [38]，Sloan 方法 [148] 等。在这里简述 Reverse Cuthill-McKee 方法的实施思想 [11]。对于图6.20中的网格单元，可以依据网格连接性自动填充矩阵元素。例如，对于网格单元 1，其与网格单元 4 相连，因此对于离散出来的 6×6 矩阵，其第一行的第 1、4 个元素存在值。对于网格单元 2，其与网格单元 5、6 相连，因此对于离散出来的 6×6 矩阵，其第二行的第 2、5、6 个元素存在值。以此类推。其最终离散后形成的稀疏线性系统如图6.20所示。



$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

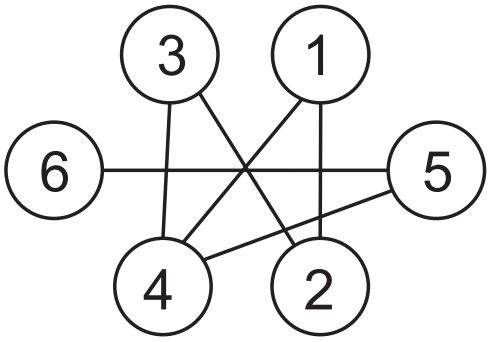
图 6.20: 一个 6 网格单元范例，以及离散后的稀疏线性系统实例。

Reverse Cuthill-McKee 方法首先选取其中网格连接性较少的网格单元，如网格单元 1（因为其仅仅与 4 相连接）。选取了网格单元 1 之后，构造一个数列，其中含有 6 个元素，并进行下述操作：

1. 首先，填充数列第一个元素为 1，数列形成为 (1, −, −, −, −, −)；
2. 网格单元 1 与网格单元 4 相连，且 4 未填充到数列中，那么数列形成为 (1, 4, −, −, −, −)；
3. 网格单元 4 与网格单元 1、6 相连，且 6 未填充到数列中，那么数列形成为 (1, 4, 6, −, −, −)；
4. 网格单元 6 与网格单元 2、3、4 相连，且 2、3 未填充到数列中，取其中的小数 2，那么数列形成为 (1, 4, 6, 2, −, −)；
5. 网格单元 2 与网格单元 5、6 相连，且 5 未填充到数列中，那么数列形成为 (1, 4, 6, 2, 5, −)；
6. 网格单元 5 与网格单元 2、3 相连，且 3 未填充到数列中，那么数列形成为 (1, 4, 6, 2, 5, 3)；

7. 将数列 (1, 4, 6, 2, 5, 3) 转置为 (3, 5, 2, 6, 4, 1)；

最后，Reverse Cuthill-McKee 方法重置的标号即为 (3, 5, 2, 6, 4, 1)，参考原数列 (1, 2, 3, 4, 5, 6)，即原始元素 3 的位置填充为 1，原始元素 5 的位置填充为 2，以此类推。如图6.21所示，很明显，其带宽要少，且呈现一种带状结构。



$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

图 6.21: 采用 Reverse Cuthill-McKee 方法调整网格编号后的稀疏线性系统实例。

6.6 OpenFOAM 中的 MULES

6.6.1 寻找最大值最小值

MULES 的全称为 Multi-dimensional Universal Limiter for Explicit Solution。MULES 为 OpenFOAM 基金会基于 FCT 算法 [21] 在 OpenFOAM 中的实现。FCT 算法通过反扩散的方法求解纯对流传输方程，既具有高阶精度，同时可以使的解严格的有界。在 FCT 算法的第一篇文章中 [21]，Boris and Book 将 FCT 算法用于一套特定的求解代码（SHASTA）中并用于求解激波，他们当时的计算显示了 FCT 算法的精度要高于一阶格式，并且解是单调无震荡的。在这篇文章中，也第一次的出现了通量限制（Flux limiting）这一概念。随后，FCT 算法被 Zalesak 拓展为了多维算法 [199]。这些文章的发表表明了 FCT 算法可以用于非常复杂的几何。OpenFOAM 中的 MULES 与 Zalesak 拓展的 FCT 算法基本一致，在更新限制器的时候略有不同。在这里通过一系列具体的例子，来理解 OpenFOAM 中 MULES 的主要流程。

MULES 的第一个关键步骤即通过遍历面以及 `own`、`nei` 的序号，来寻找相邻网格单元最大值。考虑一个计算域，在没有源项生成且速度散度为 0 的情况下，标量 ψ 的传输必然存在上下界。某个网格单元在下一个时间步可能的上界，就是与这个网格单元、毗连网格单元，这些所有网格单元的最大值。某个网格单元在下一个时间步可能的下界，就是与这个网格单元、毗连网格单元，这些所有网格单元的最小值。因此 MULES 的第一步就是通过代码将这些极值进行获取。

如图6.22所示，MULES 代码通过遍历面的方式，可以求得与网格单元 2 相邻的 0、1、

3、4、5 网格单元，共 6 个网格单元的最大值，并将这个值存储在网格单元 2 中。最小值同理。相应的代码如下[47]：

```
    scalarField psiMaxn(psiIf.size(), psiMin);
    scalarField psiMinn(psiIf.size(), psiMax);
    ....
    forAll(phiCorrIf, facei)
    {
        label own = owner[facei];
        label nei = neighb[facei];

        psiMaxn[own] = max(psiMaxn[own], psiIf[nei]);
        psiMaxn[nei] = max(psiMaxn[nei], psiIf[own]);
    }
```

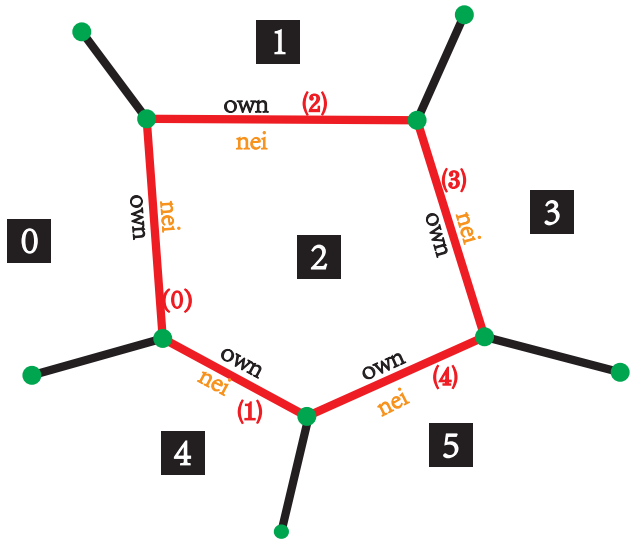


图 6.22: own、nei 应用实例示意图，用于计算全场内部网格与相邻网格单元的最大值。

1. 算法首先对 psiMaxn 赋予一个可能出现的最小值 ψ_M；

2. 算法对面进行遍历，首先从第 0 个面开始，第 0 个面的 own 网格单元序号为 0，nei 网格单元序号为 2，因此 psiMaxn[own] 判断 ψ_M 与 ψ_2 的大小，如果 ψ_2 大，则当前步 own 网格单元 0 的 $\text{psiMaxn}_0 = \psi_2$；同样的，psiMaxn[nei] 判断 ψ_M 与 ψ_0 的大小，如果 ψ_0 大，则当前步 nei 网格单元 2 的 $\text{psiMaxn}_2 = \psi_0$；

3. 进入第 1 个面，第 1 个面的 own 网格单元序号为 2，nei 网格单元序号为 4，因此 psiMaxn[own] 判断 ψ_2 与 ψ_4 的大小，如果 ψ_4 大，则当前步 own 网格单元 2 的

[47] MULEStemplate.C

psiMaxn$_2 = \psi_4$；同样的，psiMaxn[nei] 判断 ψ_M 与 ψ_2 的大小，如果 ψ_2 大，则当前步 nei 网格单元 4 的 psiMaxn$_4 = \psi_2$；

4. 进入第 2 个面，第 2 个面的 own 网格单元序号为 1，nei 网格单元序号为 2，因此 psiMaxn[own] 判断 ψ_M 与 ψ_2 的大小，如果 ψ_2 大，则当前步 own 网格单元 1 的 psiMaxn$_1 = \psi_2$；同样的，psiMaxn[nei] 判断 ψ_2 与 ψ_1 的大小，如果 ψ_1 大，则当前步 nei 网格单元 2 的 psiMaxn$_2 = \psi_1$；

其他的面以此类推。这几行代码就可以实现获得相邻网格单元的最大值。在本文中，网格单元 $_rP$ 的最大值最小值定义为 $\psi_{\mathrm{P}}^{\max}$ 和 $\psi_{\mathrm{P}}^{\min}$。

6.6.2 反扩散通量

FCT 算法中最重要的是反扩散（antidiffusion）的概念。之所以在 1973 年 Boris and Book 的原始算法中屡次提及反扩散的概念 [21]，是因为 Eq. (6.100) 右侧的最后一项可以看做是将一阶格式较大的扩散作用进行抑制，也即一种“反”扩散行为。同样的，这一项也可以看做是一种对低阶通量的修正，也即 FCT 算法中的“通量修正”的意义。具体这个低阶通量可以修正多少，通过 λ_f 来体现。

考虑一个被动传输的标量方程：

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\mathbf{U}\psi) = 0 \tag{6.99}$$

其中 ψ 是被传输的标量。考虑一阶时间离散，FCT 算法将其离散为：

$$\psi_{\mathrm{P}}^{t+\Delta t} = \psi_{\mathrm{P}}^{t} - \frac{\Delta t}{\Delta V_{\mathrm{P}}} \left(\sum_{i=0}^{f} \phi_f^{\mathrm{L}} \psi_f^{\mathrm{L}} + \sum_{i=0}^{f} \lambda_f \left(\phi_f^{\mathrm{H}} \psi_f^{\mathrm{H}} - \phi_f^{\mathrm{L}} \psi_f^{\mathrm{L}} \right) \right) \tag{6.100}$$

其中 ψ_{P} 表示当前网格点的变量值，ΔV_{P} 表示网格单元体积，Δt 表示时间步长，ψ_f^{H} 表示通过高阶格式离散后获得的通量，ψ_f^{L} 表示通过低阶格式离散后获得的通量，λ_f 表示定义在面上的限制器。很明显，$\lambda_f = 1$，FCT 算法为高阶格式，$\lambda_f = 0$，FCT 算法为低阶格式。FCT 算法的精髓，即获得各个面上的 λ_f 的值。在不越界的情况下，尽可能的坚持高阶格式。因此，$\phi_f^{\mathrm{H}} \psi_f^{\mathrm{H}} - \phi_f^{\mathrm{L}} \psi_f^{\mathrm{L}}$ 也即网格面上的反扩散通量（对应下述代码中的 phiCorr），相应的代码如下[48]：

```
surfaceScalarField& phiCorr = phiPsi;
phiCorr -= phiBD;
```

在这里令网格面上的反扩散通量为 A_f[49]，有 $A_f = \phi_f^{\mathrm{H}} \psi_f^{\mathrm{H}} - \phi_f^{\mathrm{L}} \psi_f^{\mathrm{L}}$，同时方程(6.100)可以写

[48] MULEStemplate.C

[49] 在 Zalesak 的原始文章中 [199]，反扩散通量也通过 A 表示。

为

$$\psi_{\mathrm{P}}^{t+\Delta t} = \psi_{\mathrm{P}}^{t} - \frac{\Delta t}{\Delta V_{\mathrm{P}}}\left(\sum_{i=0}^{f}\phi_f^{\mathrm{L}}\psi_f^{\mathrm{L}} + \sum_{i=0}^{f}\lambda_f A_f\right) \tag{6.101}$$

进一步的，求解器定义网格上的流入反扩散通量 P_{P}^{+} 以及流出反扩散通量 P_{P}^{-}[50]，并对其进行计算。考虑图6.23，其中第 0 个网格面的反扩散通量从第 0 个网格单元流向第 1 个网格单元。同时，反扩散通量的流向与面矢量方向通向，因此第 0 个网格面的反扩散通量的值为正值。另一方面，第 0 个网格面的反扩散通量，会导致网格单元 2 的值的增加，网格单元 0 的减小，因此，0 网格面的反扩散通量会对网格单元 2 的 P^{+} 产生贡献（青色箭头），对网格单元 0 的 P^{-} 产生贡献（青色箭头）。考虑第 2 个网格面，反扩散通量与面矢量的方向反向，因此反扩散通量为负值。同样的，2 网格面的反扩散通量，会导致网格单元 1 的值的增加，网格单元 2 的减小。因此，2 网格面的反扩散通量会对网格单元 1 的 P^{+} 产生贡献（青色箭头），对网格单元 2 的 P^{-} 产生贡献（青色箭头）。

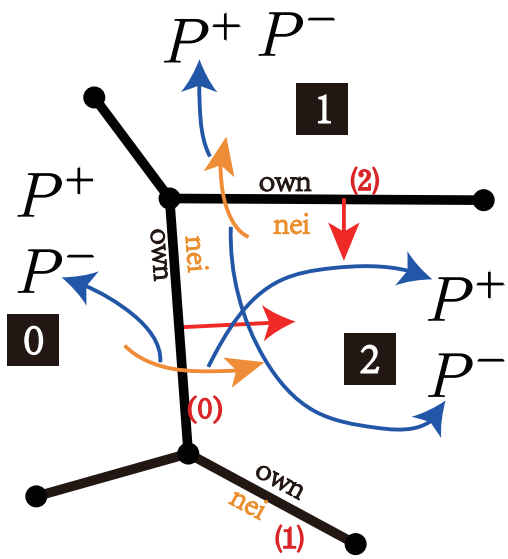


图 6.23: 方框的数字表示网格编号。括号数字表示网格面编号。红色的箭头表示面矢量。黄色的箭头表示反扩散通量的流向。青色的箭头表示反扩散通量的贡献指向。

下面结合代码来理解定义在网格单元上的反扩散通量的计算过程，相应的计算代码如下[51]：

```
forAll(phiCorrIf, facei)
{
    scalar phiCorrf = phiCorrIf[facei];
    if (phiCorrf > 0.0)//如果面上的反扩散通量大于0，则通量从own流向nei，则这个反扩散通量的贡献会添加到nei网格的P+上，以及own网格的P-上
    {
```

[50]再一次的，P_{P}^{+} 与 P_{P}^{-} 的符号采用与 Zalesak 的原始文章一致 [199]。

[51]MULEStemplate.C

```
        sumPhip[own] += phiCorrf;//P-
        mSumPhim[nei] += phiCorrf;//P+
    }
    else//如果面上的反扩散通量小于0，则通量从nei流向own，则这个反扩散通量的贡献会添加到own网格的P+上，以及nei网格的P-上
    {
        mSumPhim[own] -= phiCorrf;//P+，注意，此处的phiCorrf为负值，减去一个负值为正值
        sumPhip[nei] -= phiCorrf;//P-，注意，此处的phiCorrf为负值，减去一个负值为正值
    }
}
```

同时也可以看出，P^+ 与 P^- 为一个大于等于 0 的变量。现在假定某个网格具有 4 个面，方程(6.101)可以展开为

$$\psi_{\mathrm{P}}^{t+\Delta t}=\psi_{\mathrm{P}}^{t}-\frac{\Delta t}{\Delta V_{\mathrm{P}}}\left(\sum_{i=0}^{f}\phi_f^{\mathrm{L}}\psi_f^{\mathrm{L}}+\lambda_{f,1}A_{f,1}+\lambda_{f,2}A_{f,2}+\lambda_{f,3}A_{f,3}+\lambda_{f,4}A_{f,4}\right) \tag{6.102}$$

其中 $\lambda_{f,1}A_{f,1},\lambda_{f,2}A_{f,2},\lambda_{f,3}A_{f,3},\lambda_{f,4}A_{f,4}$ 表示附加限制器的反扩散通量。这些反扩散通量有一些流入 P 网格，一些流出 P 网格。要注意的是，λ 的符号肯定是大于 0 的。反扩散通量 A_f 如果大于 0，则必然流入网格单元（注意在这里 A_f 已经考虑了网格面矢量的方向），如果小于 0，必然流出网格单元。因此，P^+ 与 P^- 可以表示为：

$$\begin{aligned}P^+&=\max(A_{f,1},0)+\max(A_{f,2},0)+\max(A_{f,3},0)+\max(A_{f,4},0)\\P^-&=-\left(\min(A_{f,1},0)+\min(A_{f,2},0)+\min(A_{f,3},0)+\min(A_{f,4},0)\right)\end{aligned} \tag{6.103}$$

二者均为一个正值。如果不考虑反扩散通量限制器（均为 1），方程(6.102)也可以写为

$$\psi_{\mathrm{P}}^{t+\Delta t}=\psi_{\mathrm{P}}^{t}-\frac{\Delta t}{\Delta V_{\mathrm{P}}}\left(\sum_{i=0}^{f}\phi_f^{\mathrm{L}}\psi_f^{\mathrm{L}}+P^+-P^-\right) \tag{6.104}$$

其为高阶格式预测的结果。

6.6.3 可流入/流出的最大反扩散通量、反扩散通量限制器

在讨论可流入/流出的最大反扩散通量之前，首先讨论一下低阶格式以及高阶格式对计算结果可能带来的影响。如图6.24所示，考虑仅仅存在两个单元。高阶格式与低阶格式的区别在于传输的通量。对于高阶格式，假定左侧网格在 $t=0$ 时间步向 $t=1$ 时间步下步进的时候，向右侧网格移动了 10 个单位的通量。对于低阶格式，我们认为向右侧网格移动了 8 个单位的通量。因此，通量的差异，导致最终 $t=1$ 时间步下结果的差异。因此，高阶通量与低阶通量的数值并不相同。考虑图6.24中最右侧的情况，如果左侧网格流出 12

个网格单元，将导致左侧网格在 $t = 1$ 时间步下的值为-2，右侧网格出现的值为 12，分别越下界与上界。

通常情况下，低阶格式是一定有界的，高阶格式（如一些 TVD、NVD 格式）在一维格式下是有界的，在三维的情况下会有越界的情况出现。MULES 算法为了解决三维计算域下有界的问题，需要计算一套低阶有界的通量（格式），同时计算一套高阶的通量（格式），并对这两套通量进行存储，供后续流程进一步处理。

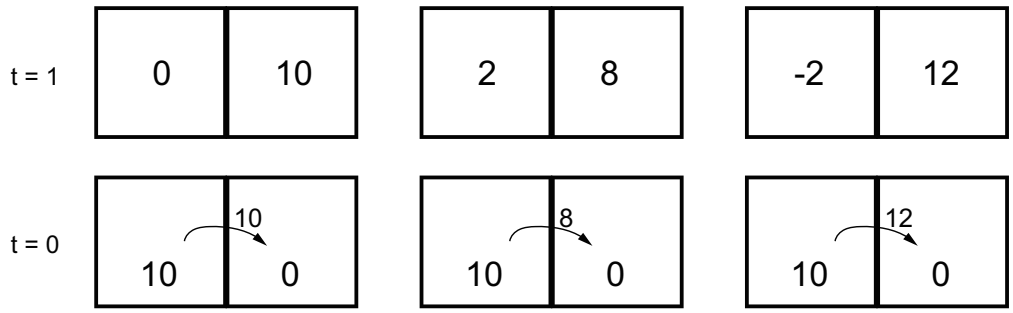


图 6.24: 2 个网格单元在不同格式下的通量变化。左：高阶格式。中：低阶格式。右：某格式导致的越界。

进一步考虑图6.24中 2 个网格单元在不同格式下的通量变化，在一阶格式的情况下，假定通量为 8，高阶格式的情况下，假定通量为 10，目前可以认为这两个通量都不会引起越界。另外可以获知的信息是这两个通量的差为 2，这说明在一阶格式的基础上，可以引入最多数值为 2 的反扩散通量，既可以将精度提高，同时也不会产生越界行为。这个数值 2，也就是最大可流入反扩散通量，其用 Q_{P}^{+} 来表示。同理，对于两个网格中的左侧的网格，存在最大可流出反扩散通量 Q_{P}^{-}，其值也为 2。

反扩散可以从另外一个角度来进行理解。反扩散的存在，一定会引起网格单元 ψ 的值得变化。要使得结果有界，这个变化后的 ψ 的值就不能超过第6.6.1节寻找的最大值与最小值。这个最大值与最小值，与第6.6.3节定义的最大最小可流入流出通量有关。反扩散通量的值不能够任意的给定，这可能会导致越界。其根本在于：

- 流入某网格单元的反扩散通量 P_{P}^{+}，要小于最大可流入反扩散通量 Q_{P}^{+}；
- 流出某网格单元的反扩散通量 P_{P}^{-}，要小于最大可流出反扩散通量 Q_{P}^{-}；

6.6.4 标准 FCT 限制器

OpenFOAM 中的 MULES 与标准 FCT 的区别在于限制器的不同。在讨论 OpenFOAM 中植入的限制器之前，首先讨论标准 FCT 算法的限制器是如何更新的。

如何确定最大可流入/流出反扩散通量？从方程(6.100)的形式上来看，Q_{P}^{+} 和 Q_{P}^{-} 的形式类似于 $\frac{\Delta V_{\mathrm{P}}}{\Delta t} f(\psi)$，此形式量纲正确，其为一个关于 ψ 的函数。结合上一节获得的网格单

元可能得最大值以及最小值 $\psi_{\mathrm{P}}^{\max}$ 和 $\psi_{\mathrm{P}}^{\min}$。标准 FCT 算法将 Q_{P}^{+} 和 Q_{P}^{-} 可以定义为：

$$
\begin{aligned}
Q_{\mathrm{P}}^{+} &= \frac{\Delta V_{\mathrm{P}}}{\Delta t}\left(\psi_{\mathrm{P}}^{\max}-\psi_{\mathrm{P}}^{\mathrm{BD}}\right)=\frac{\Delta V_{\mathrm{P}}}{\Delta t}\left(\psi_{\mathrm{P}}^{\max}-\psi_{\mathrm{P}}^{t}\right)+\sum_{i=0}^{f}\phi_{f}^{\mathrm{L}}\psi_{f}^{\mathrm{L}},\\
Q_{\mathrm{P}}^{-} &= \frac{\Delta V_{\mathrm{P}}}{\Delta t}\left(\psi_{\mathrm{P}}^{\mathrm{BD}}-\psi_{\mathrm{P}}^{\min}\right)=\frac{\Delta V_{\mathrm{P}}}{\Delta t}\left(\psi_{\mathrm{P}}^{t}-\psi_{\mathrm{P}}^{\min}\right)-\sum_{i=0}^{f}\phi_{f}^{\mathrm{L}}\psi_{f}^{\mathrm{L}}.
\end{aligned}
\tag{6.105}
$$

方程(6.105)中的极值在上一节已经可以计算，低阶通量 $\sum_{i=0}^{f}\phi_{f}^{\mathrm{L}}\psi_{f}^{\mathrm{L}}$ 需要处理，相关代码如下[52]：

```
forAll(phiCorrIf, facei)
{
   ...
    sumPhiBD[own] += phiBDIf[facei];
    sumPhiBD[nei] -= phiBDIf[facei];
    ...
}
```

需要注意的是，上述代码与第6.33节中的 surfaceIntegrate 函数的代码一致。因此在此不再详述。上述代码 `sumPhiBD` 的数学形式即 $\sum_{i=0}^{f}\phi_{f}^{\mathrm{L}}\psi_{f}^{\mathrm{L}}$。进一步的，计算 Q_{P}^{+}、Q_{P}^{-} 的相关代码如下：

```
    psiMaxn =
        V
       *(
           (rho.field()*rDeltaT - Sp.field())*psiMaxn
         - Su.field()
         - (rho.oldTime().field()*rDeltaT)*psi0
        )
      + sumPhiBD;
    psiMinn =
        V
       *(
           Su.field()
         - (rho.field()*rDeltaT - Sp.field())*psiMinn
         + (rho.oldTime().field()*rDeltaT)*psi0
        )
      - sumPhiBD;
```

[52] MULEStemplate.C

标准 FCT 算法的限制器可以定义为 λ，其计算公式为：

$$\begin{aligned}\lambda_{\mathrm{P}}^{+} &= \max\left(\min\left(1, \frac{Q_{\mathrm{P}}^{+}}{P_{\mathrm{P}}^{+}}\right), 0\right) \\ \lambda_{\mathrm{P}}^{-} &= \max\left(\min\left(1, \frac{Q_{\mathrm{P}}^{-}}{P_{\mathrm{P}}^{-}}\right), 0\right)\end{aligned} \tag{6.106}$$

其中 λ^{+} 与 λ^{-} 分别表示放置越上界的限制器与放置越下界的限制器。方程(6.106)的物理意义很好理解，如果 P_{P}^{+} 的值小于 Q_{P}^{+}，也就是说反扩散通量不会超过最大可允许流入通量，因此 $\lambda^{+}=1$，即高阶格式不会引起越上界。$\lambda^{+}=1$ 亦如此。

由于这些限制器被定义在网格上（λ），算法需要将其插值到网格面处（λ_f）。再次考虑图6.23中的网格，对于网格面 0，反扩散通量 A_f 从网格单元 0 流向 2，因此可能会导致网格单元 0 的越下界，网格单元 2 越上界。要保证网格单元 0 不会越下界，面 0 上的 λ_f 要小于等于网格单元 0 的 λ^{-}，也即 $\lambda_{f,0} <= \lambda_0^{-}$。同理，要保证网格单元 2 不会越上界，面 0 上的 λ_f 要小于等于网格单元 2 的 λ^{+}，也即 $\lambda_{f,0} <= \lambda_2^{+}$。由于求解器要保证同时不会越上界以及越下界，因此最终的 λ_f 可以表示为：

$$\lambda_f = \min(\lambda_{own}^{-}, \lambda_{nei}^{+}), A_f > 0 \tag{6.107}$$

同理，如果 $A_f < 0$（例如对于图6.23中的网格面 2），有

$$\lambda_f = \min(\lambda_{own}^{+}, \lambda_{nei}^{-}), A_f < 0 \tag{6.108}$$

相应的算法在 OpenFOAM 中并未植入，但伪代码可以写成如下：

```
    forAll(lambdaIf, facei)
    {
        if (phiCorrIf[facei] > 0.0)
        {
            lambdaIf[facei] =
                min(lambdaNeg[owner[facei]], lambdaPos[neighb[facei]]);
        }
        else
        {
            lambdaIf[facei] =
                min(lambdaPos[owner[facei]], lambdaNeg[neighb[facei]]);
        }
    }
```

6.6.5 限制器迭代更新算法

现在回头看标准 FCT 算法的意义。方程(6.106)可以理解为需要满足下述限定条件：

$$\begin{aligned}\lambda^+P^+ < Q^+, 0 \leqslant \lambda^+ \leqslant 1\\ \lambda^-P^- < Q^-, 0 \leqslant \lambda^- \leqslant 1\end{aligned} \tag{6.109}$$

也就是流入的反扩散通量，乘以限制器之后，要小于最大可流入反扩散通量。对于流出的反扩散通量亦然。

在 OpenFOAM 的 MULES 算法中，与标准 FCT 略有不同，MULES 认为流入的反扩散通量，乘以限制器之后，减去附加限制的流出的反扩散通量，要小于最大可流入反扩散通量。用方程可以写为：

$$\lambda^+P^+ + \sum_{i=0}^{f} \min(\lambda_{f,i}A_{f,i}, 0) \leqslant Q^+ \tag{6.110}$$

同理，MULES 认为流出的反扩散通量，乘以限制器之后，加上附加限制的流入的反扩散通量，要小于最大可流出反扩散通量：

$$\lambda^-P^- - \sum_{i=0}^{f} \max(\lambda_{f,i}A_{f,i}, 0) \leqslant Q^- \tag{6.111}$$

对上述方程移项有：

$$\lambda^+ \leqslant \frac{-\sum_{i=0}^{f} \min(\lambda_{f,i}A_{f,i}, 0) + Q^+}{P^+} \tag{6.112}$$

$$\lambda^- \leqslant \frac{\sum_{i=0}^{f} \max(\lambda_{f,i}A_{f,i}, 0) + Q^-}{P^-} \tag{6.113}$$

很明显，方程(6.112)和(6.113)中的 λ 在方程左右均存在。因此并不能够获得结果。在这里需要引入迭代。类似线性系统的迭代算法（如高斯赛德尔），可以将方程(6.112)和(6.113)改写为

$$\lambda^{+,n+1} \leqslant \frac{-\sum_{i=0}^{f} \min(\lambda_{f,i}^{n}A_{f,i}, 0) + Q^+}{P^+} \tag{6.114}$$

$$\lambda^{-,n+1} \leqslant \frac{\sum_{i=0}^{f} \max(\lambda_{f,i}^{n}A_{f,i}, 0) + Q^-}{P^-} \tag{6.115}$$

参考方程(6.106)，进一步可以写为：

$$\begin{aligned}\lambda_{\mathrm{P}}^{+,n+1} &= \max\left(\min\left(1, \frac{-\sum_{i=0}^{f} \min(\lambda_{f,i}^{n}A_{f,i}, 0) + Q_{\mathrm{P}}^{+}}{P_{\mathrm{P}}^{+}}\right), 0\right)\\ \lambda_{\mathrm{P}}^{-,n+1} &= \max\left(\min\left(1, \frac{\sum_{i=0}^{f} \max(\lambda_{f,i}^{n}A_{f,i}, 0) + Q_{\mathrm{P}}^{-}}{P_{\mathrm{P}}^{-}}\right), 0\right)\end{aligned} \tag{6.116}$$

在引入迭代的过程中，最初可以假定面上的 $\lambda = 1$。在通过方程(6.116)获得网格单元上的 λ 之后，面上的 λ 可以通过方程(6.107)与(6.108)获得。方程(6.116)的代码如下[53]：

[53] MULESTemplate.C

```
for (int j=0; j<nLimiterIter; j++)
//其中的nLimiterIter在算例中的fvSolution中设置
{
    forAll(lambdaIf, facei)
    {
        const label own = owner[facei];
        const label nei = neighb[facei];

        scalar lambdaPhiCorrf = lambdaIf[facei]*phiCorrIf[facei];

        if (lambdaPhiCorrf > 0)
        {
            sumlPhip[own] += lambdaPhiCorrf;
            mSumlPhim[nei] += lambdaPhiCorrf;
        }
        else
        {
            mSumlPhim[own] -= lambdaPhiCorrf;
            sumlPhip[nei] -= lambdaPhiCorrf;
        }
    }
    ....
    forAll(sumlPhip, celli)
    {
        sumlPhip[celli] =
            max(min
            (
                (sumlPhip[celli] + psiMaxn[celli])
               /(mSumPhim[celli] + rootVSmall),
                1.0), 0.0
            );

        mSumlPhim[celli] =
            max(min
            (
                (mSumlPhim[celli] + psiMinn[celli])
               /(sumPhip[celli] + rootVSmall),
                1.0), 0.0
            );
    }
    ...
```

```
}
```

最后要注意的是，所有的算法针对边界面也需要进行处理，大同小异，在这里不做介绍。

6.7 OpenFOAM 中的热物理模型

OpenFOAM 一些求解器不需要求解能量方程，例如一些不可压求解器比如 simpleFoam、interFoam。但对于可压缩流动，通常需要求解能量方程。因为可压缩流动的密度通常跟流体的温度相关。OpenFOAM 的热物理模型库主要附加在可压缩求解器中，负责能量方程以及密度压力之间的耦合联系更新求解。

6.7.1 能量方程

质量守恒、动量守恒、能量守恒分别为三大守恒定律。由这些守恒定律可以分别推导出连续性方程、动量方程以及能量方程。能量守恒，定义为绝热系统的总能量是一个常数，总能量不能凭空的产生和消失。能量的单位为 $\mathrm{J} = \mathrm{N} \cdot \mathrm{m} = \frac{\mathrm{kg} \cdot \mathrm{m}^2}{\mathrm{s}^2}$。在 CFD 中，可以用下述变量表示能量相关量：

- 比内能 e：单位质量的内能；
- 比机械能 K：单位质量的动能，可以定义为 $K = \frac{1}{2}|\mathbf{U}|^2$；
- 比能 E：单位质量的总能量。$E = K + e$；
- 比焓 h：单位质量的焓；

上述变量单位均为 $\mathrm{m}^2/\mathrm{s}^2$。很明显，与湍流动能的单位一致。在这里显性的称呼各类变量为比变量。一些论文中可能会忽略比字。

能量守恒表示其中比能 E 为守恒变量。在无源项和沉降项的时候，比能其值不随时间变化。只会相互转换。在推导能量方程的过程中，还需要考虑做功的影响。热力学第一定律表明物体内能的增加，等于物体吸热的能量变化和对物体所作的功的总和。考虑一拉格朗日粒子，其每单位体积比能的时间变化率，即为每单位质量的比能的变化率再乘以密度：

$$\rho \frac{\mathrm{D}E}{\mathrm{D}t} = \rho \frac{\mathrm{D}(K+e)}{\mathrm{D}t} \left[\frac{\mathrm{kg}}{\mathrm{s}^3 \cdot \mathrm{m}} \right] \tag{6.117}$$

方程(6.117)够成能量方程，也即比能方程的左边。

现考虑热通量导致的能量变化。定义热通量为 $\mathbf{q}$，其单位为 $\mathbf{q}\left[\frac{\mathrm{w}}{\mathrm{m}^2} = \frac{\mathrm{kg}}{\mathrm{s}^3}\right]$。从物理意义上理解，热通量与连续性方程中的通量 $\rho\mathbf{U}$ 类似。$\rho\mathbf{U}$ 表示速度导致的每单位时间每单位面积的质量变化。$\mathbf{q}$ 则表示温度导致的每单位时间每单位面积的能量变化。参考连续性方程，在连续性方程右侧，速度导致的每单位时间、每单位体积的质量变化可以用 $-\nabla \cdot (\rho\mathbf{U})$ 来表

示。同样的，能量方程中热通量导致的每单位时间、每单位体积的能量变化可以用 $-\nabla\cdot(\mathbf{q})$ 来表示。在这里要注意：热通量为一个矢量。同时，虽然其被称之为热通量，但其表示每单位时间、每单位面积的能量。或许叫能量通量更好。

热通量 $\mathbf{q}$ 与比焓的关系为：

$$\mathbf{q} = -\alpha_{eff,h}\nabla h \tag{6.118}$$

其中 $\alpha_{eff,h}$ 可以进一步写为[54]：

$$\alpha_{eff,h} = \frac{\kappa + C_p\alpha_t}{C_p} \tag{6.119}$$

α_t 表示湍流热导率。热通量 $\mathbf{q}$ 与比能的关系为：

$$\mathbf{q} = -\alpha_{eff,e}\nabla e \tag{6.120}$$

其中 $\alpha_{eff,e}$ 可以进一步写为[55]：

$$\alpha_{eff,e} = \frac{\kappa + C_p\alpha_t}{C_v} \tag{6.121}$$

其中 κ 为导热系数。

现在考虑做功导致的能量变化。力对物体做功，必然引起能量的变化。现在回想动量方程的相关项。在动量方程中，单位面积上的应力张量 $\boldsymbol{\sigma}$ 导致的动量变化可以表示为 $\nabla\cdot\boldsymbol{\sigma}$。在能量方程中，单位面积上的应力张量相对应的功率定义为单位面积上的应力张量与速度的乘积 $\boldsymbol{\sigma}\cdot\mathbf{U}$。类似的，能量方程中单位面积上的应力张量导致的能量变化可以表示为 $\nabla\cdot(\boldsymbol{\sigma}\cdot\mathbf{U})$。现将 $\boldsymbol{\sigma}$ 进行分解为压力贡献以及粘性贡献：$\boldsymbol{\sigma} = \boldsymbol{\tau} - p\mathbf{I}$。有：

$$\nabla\cdot(\boldsymbol{\sigma}\cdot\mathbf{U}) = \nabla\cdot((\boldsymbol{\tau} - p\mathbf{I})\cdot\mathbf{U}) = \nabla\cdot(\boldsymbol{\tau}\cdot\mathbf{U}) - \nabla\cdot(p\mathbf{U}) \tag{6.122}$$

同时有：

$$\nabla\cdot(\boldsymbol{\tau}\cdot\mathbf{U}) = (\nabla\cdot\boldsymbol{\tau})\cdot\mathbf{U} + \boldsymbol{\tau}:\nabla\mathbf{U} \tag{6.123}$$

$\boldsymbol{\tau}:\nabla\mathbf{U}$ 为一个恒正的量。其可以理解为将机械能耗散为内能。

至此，能量方程可以表示为：

$$\rho\frac{\mathrm{D}E}{\mathrm{D}t} - \nabla\cdot(\alpha_{eff,e}\nabla e) = \nabla\cdot(\boldsymbol{\tau}\cdot\mathbf{U}) - \nabla\cdot(p\mathbf{U}) \tag{6.124}$$

其中第二项表示热通量（扩散），等号右边表示做功。其也可以写为：

$$\frac{\partial\rho E}{\partial t} + \nabla\cdot(\rho\mathbf{U}E) - \nabla\cdot(\alpha_{eff,e}\nabla e) + \nabla\cdot(p\mathbf{U}) = \nabla\cdot(\boldsymbol{\tau}\cdot\mathbf{U}) \tag{6.125}$$

其可以继续分解为：

$$\frac{\partial\rho e}{\partial t} + \nabla\cdot(\rho\mathbf{U}e) + \frac{\partial\rho K}{\partial t} + \nabla\cdot(\rho\mathbf{U}K) - \nabla\cdot(\alpha_{eff,e}\nabla e) + \nabla\cdot(p\mathbf{U}) = \nabla\cdot(\boldsymbol{\tau}\cdot\mathbf{U}) \tag{6.126}$$

[54] heThermo.C

[55] heThermo.C

同时我们应用焓等式：

$$e = h - p/\rho \tag{6.127}$$

将方程(6.127)代入到(6.126)中有：

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{U} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho \mathbf{U} K) - \nabla \cdot (\alpha_{eff,h} \nabla h) - \frac{\partial p}{\partial t} = \nabla \cdot (\boldsymbol{\tau} \cdot \mathbf{U}) \tag{6.128}$$

方程(6.126)和(6.128)是无热源无体积力的情况下，比焓以及比内能形式的能量方程。对于跨音速或者超音速流动，不建议忽略任何项。

然而不可压缩的情况下，可以对能量方程进行简化。考虑动量方程

$$\rho \frac{\mathrm{D}\mathbf{U}}{\mathrm{D}t} = \nabla \cdot \boldsymbol{\tau} - \nabla p \tag{6.129}$$

如果将动量方程中的每个速度分量方程乘以速度分量并加和有：

$$\rho \frac{\mathrm{D}\left[\frac{1}{2}\left(u_x^2 + u_y^2 + u_z^2\right)\right]}{\mathrm{D}t} = \rho \frac{\mathrm{D}K}{\mathrm{D}t} = (\nabla \cdot \boldsymbol{\tau}) \cdot \mathbf{U} - \mathbf{U} \cdot \nabla p \tag{6.130}$$

方程(6.130)与(6.126)相减有：

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho \mathbf{U} e) + p \nabla \cdot \mathbf{U} - \nabla \cdot (\alpha_{eff} \nabla e) = \boldsymbol{\tau} : \nabla \mathbf{U} \tag{6.131}$$

对于不可压缩流动，其中粘性耗散 $\boldsymbol{\tau} : \nabla \mathbf{U}$ 通常被忽略，且速度散度项 $p \nabla \cdot \mathbf{U}$ 可以被忽略，考虑到密度为常数，比内能方程可以写为：

$$\frac{\partial e}{\partial t} + \nabla \cdot (\mathbf{U} e) - \nabla \cdot \left(\frac{\alpha_{eff}}{\rho} \nabla e\right) = 0 \tag{6.132}$$

对于理想气体，有

$$e = C_v T \tag{6.133}$$

进一步有：

$$\frac{\partial T}{\partial t} + \nabla \cdot (\mathbf{U} T) - \nabla \cdot \left(\frac{\alpha_{eff}}{\rho} \nabla T\right) = 0 \tag{6.134}$$

即为最常见的温度方程。

具体的有关方程如何选取。如果不可压缩，可以求解比内能方程(6.132)甚至温度方程(6.134)。如果方程弱可压缩，一般求解比能方程(6.126)[56]。如果方程可压缩，一般求解比能方程(6.126)或者比焓方程(6.128)。

6.7.2 速度压力密度耦合、psiThermo 与 rhoThermo

OpenFOAM 中存在不同的密度更新方法。这些方法主要通过 psiThermo 与 rhoThermo 实现[57]。psiThermo 与 rhoThermo 的主要区别就在于，psiThermo 类型[58]中的

[56] https://github.com/OpenFOAM/OpenFOAM-dev/commit/f9971f80d72a5318cdc81120a925ab9817b3ec62

[57] 在 OpenFOAM-12 中，heRhoThermo 定义在 rhoFluidThermo.C 中，hePsiThermo 定义在 psiThermo.C 中。

[58] psiThermo.C

密度与压力满足 $\rho = p\psi$ 的状态方程。rhoThermo 类型中的密度与压力存在更负责的对应关系。因为 psiThermo 一直满足 $\rho = p\psi$，因此该类型不需要显性的包含密度，在模型内部，每一次调用密度（执行 thermo.rho() 函数）都通过 $\rho = p\psi$ 来进行计算。rhoThermo 在 psiThermo 的基础上，包含了密度的定义。执行 thermo.rho() 函数直接返回该类型本身的`rho`（通过状态方程计算而来）。rhoThermo 可以调用不满足 $\rho = p\psi$ 的状态方程。

在求解器层面，求解能量变量后，主要存在下面两行关键代码：

```
thermo.correct(); //更新温度、可压缩性、密度等
```

以及

```
thermo.correctRho(psi*p - psip0); //修正密度
```

第一行代码会对相关的变量进行更新。例如在 psiThermo 类型中要更新温度`T`，可压缩性`psi`。在 rhoThermo 类型中，不仅要更新温度和可压缩性，还需要更新密度`rho`。第二行代码存在的原因是因为在压力方程求解之后，还需要进一步修正密度[59]。该代码在执行的时候，有两种方法：

- rhoThermo：$\varrho^{**} = \varrho^{*} + \psi^{*} p'_{\mathrm{rgh}}$；
- psiThermo：第二行代码不执行。

在此进行总结，rhoThermo 算法：

1. 求解连续性方程，更新密度 ρ^{*}。利用 ρ^{*} 组建通量，求解速度方程，求解能量方程；
2. 通过 thermo.correct() 函数更新温度 T^{*}，通过状态方程更新可压缩性 ψ^{*}，通过状态方程更新 rhoThermo 的密度 ϱ^{*}；
3. 进入几次压力迭代，每次压力迭代后，更新 rhoThermo 的密度 $\varrho^{**} = \varrho^{*} + \psi^{*} p'_{\mathrm{rgh}}$。同时更新速度直至收敛；
4. 利用更新后的速度和密度组建通量，求解连续性方程，更新密度 ρ^{**}；
5. 假定压力迭代 3 次，对比 ϱ^{***} 和 ρ^{***} 的差异，得到连续性误差；

psiThermo 算法：

1. 求解连续性方程，更新密度 ρ^{*}，利用 ρ^{*} 组建通量，求解速度方程，求解能量方程；
2. 通过 thermo.correct() 函数（调用状态方程）更新温度 T^{*}，更新可压缩性 $\psi^{*} = \frac{1}{RT^{*}}$；
3. 进入几次压力迭代，其中如果需要调用密度，通过 $\varrho = \psi p_{\mathrm{rgh}}$ 进行计算。同时更新速度直至收敛；

[59] http://dyfluid.com/compiso.html

4. 利用更新后的速度和密度组建通量，求解连续性方程，更新密度 ρ^{**}；

5. 假定压力迭代 3 次，对比 $\psi^* p_{\text{rgh}}^{***}$ 和 ρ^{**} 的差异，得到连续性误差；

6.7.3 更新 ρ 的 EOS 状态方程

OpenFOAM 热物理模型库中的状态方程主要用于更新 ρ, ψ。在第6.7.2的讨论中可以看出，耦合算法求解第6.7.1节讨论的能量方程之后，需要更新可压缩性、密度等变量。这些算法均被植入到了状态方程中[60]。状态方程的植入非常简单，均为数学代数方程。例如，`rhoConst`模型的计算方法为：

$$\rho = \rho_{\text{const}}, \psi = 0 \tag{6.135}$$

`perfectGas`模型的计算方法为：

$$\rho = \frac{p}{RT}, \psi = \frac{1}{RT} \tag{6.136}$$

其他模型可以参考相关文件，在此不一一列举。

6.7.4 更新 T 的 sensibleInternalEnergy 与 sensibleEnthalpy

在求解比内能或者比焓后，可以进行温度的更新。首先，hePsiThermo 或者 heRhoThermo 中的 calculate() 函数通过 THE() 函数来计算。进一步结合算例层面设置的 sensibleInternalEnergy 与 sensibleEnthalpy，THE() 函数执行情况不同。sensibleInternalEnergy 的 THE() 函数从比内能来计算温度，sensibleEnthalpy 从比焓来计算温度。具体的温度函数在 thermoI.H 中定义。

现在以从比内能来计算温度进行举例。假定求解能量传输方程之后，获得更新后的比内能 e。现在需要反推温度 T。那么方程就可以写为：

$$e(T^{new}) - e = 0 \tag{6.137}$$

也就是从 T^{new} 算出来的 $e(T^{new})$，要与当前的 e 相等。可见其为一个关于 T^{new} 的方程，要获取 T^{new} 的值，可以理解为对方程(6.137)求根。依据 Newton - Raphson 方法，针对 $f(x) = 0$ 的函数，其根为：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{6.138}$$

因此对于方程(6.137)，有：

$$T_{n+1} = T_n - \frac{e(T^n) - e}{C_v^n} \tag{6.139}$$

[60] src/thermophysicalModels/specie/equationOfState

6.7.5 更新 C_p, C_v 的 thermo 热模型、JANAF

thermo 热模型基于下述基本定义：

$$C_p = \frac{\partial h}{\partial T}, C_v = \frac{\partial e}{\partial T} \tag{6.140}$$

其主要更新流体的 C_p, C_v, h 等参数。例如考虑比内能，有：

$$e = \int_{T_{ref}}^{T} C_v \mathrm{d}T + e_{ref} \tag{6.141}$$

其中的 e_{ref} 表示在参考温度 T_{ref} 的情况下的比内能。`eConst`模型假定模型的 C_v 为常数，在这种情况下有

$$e = C_v(T - T_{ref}) + e_{ref} \tag{6.142}$$

`ePower`模型假定 C_v 是温度的指数：

$$C_v = C_0 \left(\frac{T}{T_{ref}}\right)^{n_0} \tag{6.143}$$

其中 T_{ref}, n_0 需要用户给定。将其代入到方程(6.146)有：

$$e = \frac{C_0(T^{n_0+1} - T_{std}^{n_0+1})}{T_{ref}^{n_0}(n_0+1)} \tag{6.144}$$

其中 T_{std} 表示标准温度 298.15 开尔文。

精准的计算化学反应，需要精准的比热容。在这种情况下通常选用 JANAF 表格来计算。JANAF 表格的 C_p 计算公式为：

$$C_p = (((a_4T + a_3)T + a_2)T + a_1)T + a_0 \tag{6.145}$$

其中的 a 就是 JANAF 表格针对某种物质给出的系数。在 OpenFOAM 中，某种物质的定压比热容这样指定：

```
thermodynamics
{
    Tlow            2;
    Thigh           3500;
    Tcommon         1000;
    highCpCoeffs    (0.0292664  0.00131225 -5.7329e-07 1.22293e-10 -1.01815e-14 -922.797 4.89163);
    lowCpCoeffs     (0.0329868  0.001484    -6.124e-07   1.144e-10 -9.597e-15  -917.935 3.662 );
}
```

可见，其中的 a 需要指定 7 个数值而不是 5 个。这是因为除了 C_p 的计算，OpenFOAM 中的 JANAF 模型还需要计算一些其他的参数，如吉布斯自由能。在计算吉布斯自由能的时候，需要 7 个 a 系数。同时，JANAF 需要指定三个温度。这三个温度构成两个区间。一个高温区间，一个低温区间。在高温区间，使用`highCoCoeffs`系数，在低温区间，使用`lowCoCoeffs`系数。除此之外还有其他的类似的模型可以用来计算 C_p，在这里不一一列举。

对于基于比焓的模型，其认为

$$h = \int_{T_{ref}}^{T} C_p \mathrm{d}T + h_{ref} \tag{6.146}$$

其与比内能模型大体相同。在这里也不进行一一介绍。

6.7.6 正压模型

前述模型中的密度不仅与压力还与温度有关。在正压模型中，密度仅仅跟压力有关：

$$\rho = \rho_0 + \frac{p - p_0}{c^2} \tag{6.147}$$

其中 c 为局部音速。通过正压模型（barotropic），可以构建一个全新的弱可压缩流体控制方程（可压缩流体连续性方程、可压缩流体动量方程、以及正压模型）。

正压模型在 OpenFOAM 中主要被应用于 cavitatingFoam。对于非界面类多相流，除了多流体模型、滑移速度模型（也叫混合模型、漂移通量模型等），还可以通过均相平衡模型来模化。均相平衡模型在 OpenFOAM 中的英文名称为 Homogeneous Equilibrium Model。均相平衡模型认为不同相之间只需要一个方程变量既可以描述，也即用单纯的一套 NS 方程既可以描述多相模型（两相具有相同的传输速度）。OpenFOAM-11 之前存在空化求解器 cavitatingFoam，其求解均相平衡模型并调用正压模型来计算空化。在这个正压模型中，是在均相平衡模型中的可压缩性的计算方法。各种不同的正压模型（并未原生的密度只跟压力有关的正压模型）都认为均相平衡模型中的可压缩性与饱和蒸气压、不同相本身的原生可压缩性、以及相分数有关。都是一些比较简单的数学代数方程。在 OpenFOAM-11 之后，均相平衡模型的 cavitatingFoam 被彻底删去[61]。

6.8 OpenFOAM 中的多重参考系

CFD 计算中经常会涉及到旋转部件。如图6.25所示，如果存在单一的旋转部件，可以使用单一参考系（Single Reference Frame，SRF）方法。如果计算域内存在旋转结构以及静止结构，则需要使用多重参考系（Multiple Reference Frame，MRF）方法。在定义旋转

[61] https://openfoam.org/release/11/

域的时候，需要指定旋转点 $\mathbf{o}$ 与旋转轴 $\mathbf{a}$。二者均为位置矢量。同时需要给定旋转角速度 Ω。这样，旋转域的绕轴方向 $\mathbf{\Omega}$ 即为:

$$\mathbf{\Omega} = \Omega \mathbf{a} \tag{6.148}$$

给定任意的网格单元，其距离旋转中心点 $\mathbf{o}$ 的距离矢量表示为 $\mathbf{r}$，在这种情况下，绝对速度 $\mathbf{U}$ 为 [9]：

$$\mathbf{U} = \mathbf{U}_r + \mathbf{\Omega} \times \mathbf{r} \tag{6.149}$$

其中 $\mathbf{U}_r$ 为相对速度。

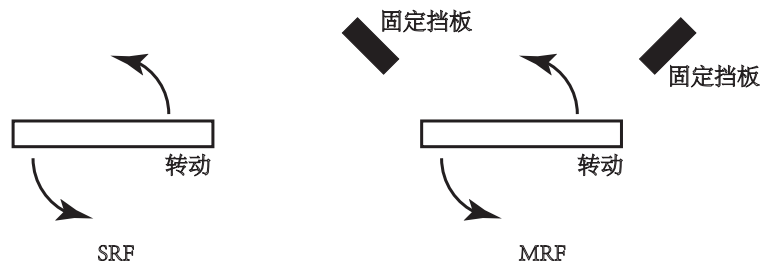


图 6.25: 通过开源 CFD 软件 OpenFOAM 模拟的粘弹性流体两相流。

在 CFD 计算中，为了调用旋转相关变量，绝对速度方程需要转变为相对速度方程，在 SRF 方法中，相对速度控制方程为：

$$\begin{gathered}\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}_r) = 0 \\ \frac{\partial \rho \mathbf{U}_r}{\partial t} + \nabla \cdot (\rho \mathbf{U}_r \mathbf{U}_r) + \mathbf{\Omega} \times \mathbf{\Omega} \times \mathbf{r} + 2\mathbf{\Omega} \times \mathbf{U}_r = \nabla p + \nabla \cdot (\mu \nabla \mathbf{U}_r)\end{gathered} \tag{6.150}$$

其中 $\mathbf{\Omega} \times \mathbf{\Omega} \times \mathbf{r}$ 表示离心力，$2\mathbf{\Omega} \times \mathbf{U}_r$ 表示科氏力。很明显，在对 SRF 控制方程进行求解的时候，可以将 $\mathbf{U}_r$ 当做求解变量，并在方程添加离心力以及科氏力源项即可。

在 MRF 方法中，进一步考虑不可压缩控制方程为：

$$\begin{gathered}\nabla \cdot \mathbf{U}_r = \nabla \cdot \mathbf{U} = 0 \\ \frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) - \nabla \cdot ((\mathbf{\Omega} \times \mathbf{r})\,\mathbf{U}) + \mathbf{\Omega} \times \mathbf{U}/\rho = \frac{\nabla p}{\rho} + \nabla \cdot (\nu \nabla \mathbf{U})\end{gathered} \tag{6.151}$$

其中连续性方程可用来组建压力泊松方程。动量方程可用于求解绝对速度 $\mathbf{U}$。

在 OpenFOAM 中，以 pimpleFoam 为例，速度方程代码如下：

```
tmp<fvVectorMatrix> tUEqn
(
    fvm::ddt(U) + fvm::div(phi, U)
  + MRF.DDt(U)
  + turbulence->divDevSigma(U)
 ==
```

```
    fvOptions(U)
);
```

其中的MRF.DDt(U)即为方程(6.151)中的 $\mathbf{\Omega}\times\mathbf{U}/\rho$。上述代码首先求解下述方程

$$\frac{\partial\mathbf{U}}{\partial t}+\nabla\cdot(\mathbf{UU})+\mathbf{\Omega}\times\mathbf{U}/\rho=\nabla\cdot(\nu\nabla\mathbf{U}) \tag{6.152}$$

获得预测速度 **HbyA**。然后通过预测速度 **HbyA** 组建通量 phiHbyA 以及压力方程[62]。同时，因为在方程(6.152)中并没有考虑 $-\nabla\cdot((\mathbf{\Omega}\times\mathbf{r})\,\mathbf{U})$，因此在组建 phiHbyA 后，需要减去其影响。这在 OpenFOAM 中通过下述代码实现：

```
MRF.makeRelative(phiHbyA);
```

6.9 OpenFOAM 中的多孔介质模型

OpenFOAM 中的多孔介质可以通过在动量方程中添加一个沉降项来实现。在这里需要注意的是，下文中的速度变量均为表观速度。在这里强调，为了与普适性 NS 方程保持一致，下文用 **U** 来表示表观速度而不是 $\overline{\mathbf{U}}$。考虑不可压缩流体，有添加多孔介质的动量方程为：

$$\frac{\partial\mathbf{U}}{\partial t}+\nabla\cdot(\mathbf{UU})=-\nabla\frac{p}{\rho}+\nabla\cdot(\nu\nabla\mathbf{U})+\mathbf{S}, \tag{6.153}$$

其中 **S** 表示多孔介质源项。其应为一个阻力源项，数值上为一种沉降项，可写为：

$$\mathbf{S}=-\left(\nu\mathbf{D}+\frac{1}{2}\mathbf{F}|\mathbf{U}|\right)\mathbf{U} \tag{6.154}$$

其中 **D**（Darcy 阻力系数，渗透率的倒数）[63]，**F**（Forchheimer 阻力系数）分别表示粘性损失以及惯性损失。各为一个二阶张量。可以看出，源项实际与方程(4.264)一致。

在流体速度比较低的情况下，通常只需要考虑粘性损失，即 $\mathbf{F}=0$。在这种情况下，如果忽略对流项以及扩散项，在稳态的情况下，有：

$$\nabla p=-\nu\mathbf{DU} \tag{6.155}$$

若进一步考虑各向同性的粘性损失（**D** 非对角线为 0，且对角线元素相同），则 **D** 可以写成一个标量 d。在这种情况下，方程(6.155)可以简化为 Darcy 定律：

$$\nabla p=-\nu d\mathbf{U} \tag{6.156}$$

在将多孔介质源项进行植入的时候需要进行离散，因此就涉及到显性隐性操作。若仅仅考虑时间项，方程(6.153)可以写为：

$$\frac{\partial\mathbf{U}}{\partial t}=-\nu\mathbf{DU} \tag{6.157}$$

[62] 更详细的内容请参考icoFoam 解析

[63] 其分量可能为负，但一定是正定且对称。

将其通过有限体积法离散，若采用纯隐性方法有[64]：

$$\frac{\mathbf{U}^{t+\Delta t}-\mathbf{U}^{t}}{\Delta t}\Delta V=-\nu\mathbf{D}\mathbf{U}^{t+\Delta t}\Delta V \tag{6.158}$$

若采用纯显性方法有：

$$\frac{\mathbf{U}^{t+\Delta t}-\mathbf{U}^{t}}{\Delta t}\Delta V=-\nu\mathbf{D}\mathbf{U}^{t}\Delta V \tag{6.159}$$

OpenFOAM 里采用一种显性隐性结合的方法。首先对方程(6.157)右侧进行一种数值操作：

$$\frac{\partial\mathbf{U}}{\partial t}=-\nu\left(\mathbf{D}+\mathrm{tr}(\mathbf{D})\mathbf{I}-\mathrm{tr}(\mathbf{D})\mathbf{I}\right)\mathbf{U} \tag{6.160}$$

分别对右侧进行部分显性离散，部分隐性离散有：

$$\begin{aligned}\frac{\mathbf{U}^{t+\Delta t}-\mathbf{U}^{t}}{\Delta t}\Delta V&=-\nu\left(\mathbf{D}-\mathrm{tr}(\mathbf{D})\mathbf{I}\right)\mathbf{U}^{t}\Delta V-\nu\left(\mathrm{tr}(\mathbf{D})\mathbf{I}\right)\mathbf{U}^{t+\Delta t}\Delta V\\ \frac{\mathbf{U}^{t+\Delta t}-\mathbf{U}^{t}}{\Delta t}\Delta V+\nu\left(\mathrm{tr}(\mathbf{D})\mathbf{I}\right)\mathbf{U}^{t+\Delta t}\Delta V&=-\nu\left(\mathbf{D}-\mathrm{tr}(\mathbf{D})\mathbf{I}\right)\mathbf{U}^{t}\Delta V\end{aligned} \tag{6.161}$$

方程(6.161)左侧的各向同性部分 $\nu\left(\mathbf{I}\mathrm{tr}(\mathbf{D})\right)\mathbf{U}^{t+\Delta t}\Delta V$ 进入到了离散后的矩阵对角线（类似正应力），右侧的各向异性部分 $-\nu\left(\mathbf{D}-\mathbf{I}\mathrm{tr}(\mathbf{D})\right)\mathbf{U}^{t}\Delta V$ 进入到了矩阵源项（类似偏应力）[65]。

在考虑各向异性的情况下，**D** 为一个二阶张量，既然为各向异性，必然各个方向的 Darcy 系数各不相同。因此，在使用的过程中，需要给定流通方向以及各个方向的 Darcy 系数。同时，在笛卡尔坐标系下，流动方向可能并不是完全平行于坐标轴（例如一块矩形蜂窝煤放置在一个倾斜的平板上）。因此会涉及到一个坐标转换过程。OpenFOAM 中的 **D** 通过一个矢量 d 与坐标系统来表示。首先考虑 d，如果矢量 d 某个分量为负（并无物理意义），则需要将其乘以最大的矢量分量的负值，来变成一个正值。例如，如果指定 d 为：

```
d    (5000 -1000 -1000);
```

其最大的分量为 5000，则其等于

```
d    (5000 -1000*5000*(-1) -1000*5000*(-1));
```

即

```
d    (5000 5000000 5000000);
```

在笛卡尔坐标系下，这表示该多孔介质在 x 方向有一定的渗透率。y、z 方向渗漏率很低。采用 (5000 -1000 -1000) 这种设置只是比较简便且清晰。

同时，在将 d 转换为二阶张量 **D** 的时候，需要考虑坐标系统。OpenFOAM 里存在不同的模型，在这里介绍非常基本的 axesRotation 方法。在调用 axesRotation 方法时，需要给定 e1 以及 e2 的方向。假设如果考虑流动方向为 x、y 平面，那么 e1 存在 x、y 分量，e2

[64]注意，这种方式 OpenFOAM 的分离式求解器并不支持，因为如果 **D** 为非对角阵，这需要使用耦合求解器。

[65]相关代码请参考 DarcyForchheimerTemplates.C。

可以简单设置为（0 0 1）。类似的，如果考虑流动方向为 x、z 平面，那么 e1 存在 x、z 分量，e2 可以简单设置为（0 1 0）。进一步的，其中 e1 的分量与角度有关。考虑图6.26中的多种渗透方向，其中与 x 方向的夹角为 θ，那么 e1 即为 $(\cos\theta, \sin\theta, 0)$[66]。在这里附上一种

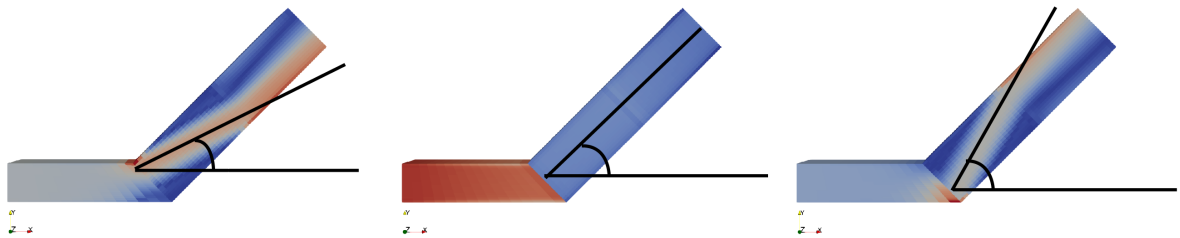


图 6.26: 设置多孔介质不同角度的模拟结果。左中右：30 度、45 度、60 度。

比较方便的设置代码：

```
    coordinateSystem
    {
        type    cartesian;
        origin  (0 0 0);
        coordinateRotation
        {
            angle   30.0;
            x       #calc "Foam::cos($angle*0.01745)";
            y       #calc "Foam::sin($angle*0.01745)";
            type    axesRotation;
            e1      ($x $y 0);
            e2      (0 0 1);
        }
    }
```

6.10 OpenFOAM 中的 IATE 模型

Interfacial Area Transport Equation（IATE）模型为一种可以预测气泡粒径分布的模型。IATE 模型从普适性群体平衡模型出发并对其简化，得到界面浓度传输方程并求解来获得粒径分布。在 IATE 模型中，数量密度函数 $n(V)$（单位 $1/m^6$）的控制方程可以写为[67]：

$$\frac{\partial n}{\partial t} + \nabla \cdot (n\mathbf{U}) = S_{bre} + S_{coa} \tag{6.162}$$

[66]其中 θ 取弧度值。

[67]注意此处的数量密度函数 $n(V)$ 的变量为体积 V。

其中 S_{bre}, S_{coa} 为聚并破碎源项。定义 $n(V)$ 的 k 阶矩为：

$$m_k = \int V^k n(V) \mathrm{d}V \tag{6.163}$$

有 m_0 表示每单位体积的粒子数量（单位 $1/m^3$），m_1 表示每单位体积的粒子体积（无单位），也即相分数 α。

对方程(6.162)求一阶矩，有关于 m_1 的体积传输方程：

$$\frac{\partial m_1}{\partial t} + \nabla \cdot (m_1 \mathbf{U}) = \int V S_{bre} \mathrm{d}V + \int V S_{coa} \mathrm{d}V \tag{6.164}$$

由于体积守恒（聚并破碎并不会改变总体积），因此方程右侧积分为 0，即

$$\int V S_{bre} \mathrm{d}V + \int V S_{coa} \mathrm{d}V = 0 \tag{6.165}$$

在下文中遵循传统写法，用 α 表示一阶矩 m_1，因此方程(6.164)可以简单的写为：

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{U}) = 0 \tag{6.166}$$

即无传质的相方程。定义 $A(V)$ 为体积 V 粒子的表面积，其和直径以及体积的关系为：

$$A(V) = \pi d^2, V = \frac{\pi d^3}{6} \tag{6.167}$$

进而有 $A(V)$ 和体积 V 的关系：

$$A(V) = 6^{2/3} \pi^{1/3} V^{2/3} \tag{6.168}$$

IATE 模型定义界面浓度 a 为

$$a = \int A(V) n(V) \mathrm{d}V \tag{6.169}$$

依据矩关系有：

$$A(V) = \frac{\int A(V) n(V) \mathrm{d}V}{\int n(V) \mathrm{d}V} = \frac{a}{m_0} \tag{6.170}$$

$$V = \frac{\int V n(V) \mathrm{d}V}{\int n(V) \mathrm{d}V} = \frac{\alpha}{m_0} \tag{6.171}$$

同时，对方程(6.162)左右两边乘以 $A(V)$ 并取矩有：

$$\frac{\partial a}{\partial t} + \nabla \cdot (a \mathbf{U}) = \int A(V) S_{bre} \mathrm{d}V + \int A(V) S_{coa} \mathrm{d}V \tag{6.172}$$

由于粒子界面浓度并不具有守恒性（同体积情况下粒子数量越多，表面积越大），因此

$$\int A(V) S_{bre} \mathrm{d}V + \int A(V) S_{coa} \mathrm{d}V \neq 0 \tag{6.173}$$

同时依据方程(6.170)和(6.171)的关系有：

$$A(V)=\frac{a}{m_0}=6^{2/3}\pi^{1/3}\left(\frac{\alpha}{m_0}\right)^{2/3}\rightarrow m_0=\psi\frac{a^3}{\alpha^2},A(V)=\frac{1}{\psi}\left(\frac{\alpha}{a}\right)^2 \tag{6.174}$$

其中

$$\psi=\frac{1}{36\pi} \tag{6.175}$$

依据积分关系：

$$\begin{aligned}\int A(V)S_{bre}\mathrm{d}V\approx\Delta A\int S_{bre}\mathrm{d}V,\\ \int A(V)S_{coa}\mathrm{d}V\approx\Delta A\int S_{coa}\mathrm{d}V\end{aligned} \tag{6.176}$$

Ishii and Kim 假定 [72]：

$$\Delta A=\frac{1}{3}A, \tag{6.177}$$

这是因为考虑聚并和破碎的情况下，粒子界面变化分别为：

$$\Delta A=-0.413A,\Delta A=0.26A \tag{6.178}$$

其中 $(|0.413|+|0.26|)/2\approx 1/3$。因此，有：

$$\begin{aligned}\Delta A\int S_{bre}\mathrm{d}V=\frac{1}{3}A\int S_{bre}\mathrm{d}V\\ \Delta A\int S_{coa}\mathrm{d}V=\frac{1}{3}A\int S_{coa}\mathrm{d}V\end{aligned} \tag{6.179}$$

在 IATE 算法中，通常将方程(6.179)中的积分表示为

$$\int S_{bre}\mathrm{d}V=R_{bre},\int S_{coa}\mathrm{d}V=R_{coa} \tag{6.180}$$

将方程(6.177)、(6.180)、(6.175)带入到(6.172)中有：

$$\frac{\partial a}{\partial t}+\nabla\cdot(a\mathbf{U})=\frac{1}{3}\frac{1}{\psi}\left(\frac{\alpha}{a}\right)^2(R_{bre}+R_{coa}) \tag{6.181}$$

其中 R_{bre},B_{coa} 需要进一步的模化。目前 OpenFOAM 中考虑三种机理：湍流引致破碎[68]，随机碰撞聚并[69]以及尾流夹带聚并[70]。聚并会导致界面浓度的降低，因此为沉降项（负号）。破碎会导致界面浓度的升高，因此为生成项（正号）。

[68]在 OpenFOAM 中为 turbulentBreakUp。
[69]在 OpenFOAM 中为 randomCoalescence。
[70]在 OpenFOAM 中为 wakeEntrainmentCoalescence。

6.11 OpenFOAM 中的化学反应

6.11.1 化学反应

OpenFOAM 中的化学反应参数设置一般通过读取 chemkin 的设置文件后通过 chemkinToFoam 来转换成 OpenFOAM 可以识别的格式。在这里不介绍 chemkin 的设置文件。仅仅介绍转换后 OpenFOAM 可以识别的格式。

例如对于经典的 Robertson 反应（更详细的可以参考第4.6.3节）：

$$\begin{aligned} A &\xrightarrow{k_1} B \\ 2B &\xrightarrow{k_2} B + C \\ B + C &\xrightarrow{k_3} A + C \end{aligned} \tag{6.182}$$

有反应速率：

$$\begin{aligned} \mathrm{RR}_1 &= k_1[A] \\ \mathrm{RR}_2 &= k_2[B]^2 \\ \mathrm{RR}_3 &= k_3[B][C] \end{aligned} \tag{6.183}$$

其中 $k_1 = 0.04, k_2 = 3 \times 10^7, k_3 = 10^4$。这里假定的反应速率常数为定值。普适性来讲，反应速率常数为温度的方程，如方程(4.188)。在这种情况下，需要给定三个系数。分别为上述代码中的`A,beta,Ta`。

```
reactions
{
    reaction1
    {
        type            irreversibleArrhenius;
        reaction        "A = B";
        A               0.04;
        beta            0;
        Ta              0;
    }

    reaction2
    {
        type            irreversibleArrhenius;
        reaction        "B + B = B + C";
        A               3e7;
        beta            0;
        Ta              0;
    }
```

```

    reaction3
    {
        type            irreversibleArrhenius;
        reaction        "C + B = A + C";
        A               1e4;
        beta            0;
        Ta              0;
    }
}
```

除了反应需要设置外，还需要对每个组分进行设置。下述代码是一个标准模版：

```
thermoType
{
    type            heRhoThermo;
    mixture         multiComponentMixture;
    transport       sutherland;
    //thermo          janaf;
    thermo          hConst;
    energy          sensibleEnthalpy;
    equationOfState perfectGas;
    specie          specie;
}

defaultSpecie N2;

species         4 (A B C N2);

A
{
    specie
    {
        molWeight       1.0;
    }
    thermodynamics
    {
        Cp              1;
        Hf              0;
    }
    transport
    {
        As              1.0e-05;
```

```
        Ts                300.0;
    }
}

B
{
    $A;
}

C
{
    $A;
}

N2
{
    $A;
}
```

如果反应共涉及到 3 个组分。那么需要给定 4 个组分（附加一个惰性物质）。需要对每个组分设置热物理模型库。

6.11.2 ODE 求解器

有关 ODE 求解理论部分请参考第4.6.4节。下面的代码给出了一个非常精简的 OpenFOAM 的 ODE 求解器范例。首先其中定义了方程数为 4。然后通过`derivatives`定义了 $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}x}$。通过`jacobian`定义了雅克比矩阵。$\mathbf{y}$ 的初始值均为 1。初始的时间步长`dxEst`定义为 0.5。然后通过`solve()`函数即可进行求解。相应的 ODE 系统如下：

$$
\begin{aligned}
\frac{\mathrm{d}y_0}{\mathrm{d}x} &= -y_1 \\
\frac{\mathrm{d}y_1}{\mathrm{d}x} &= y_0 - \frac{y_1}{x} \\
\frac{\mathrm{d}y_2}{\mathrm{d}x} &= y_1 - \frac{2y_2}{x} \\
\frac{\mathrm{d}y_3}{\mathrm{d}x} &= y_2 - \frac{3y_3}{x}
\end{aligned}
\tag{6.184}
$$

```
#include "argList.H"
#include "ODESystem.H"
#include "ODESolver.H"

using namespace Foam;
```

```

class testODE
:
    public ODESystem
{

public:

    testODE()
    {}

    label nEqns() const
    {
        return 4;
    }

    void derivatives
    (
        const scalar x,
        const scalarField& y,
        const label li,
        scalarField& dydx
    ) const
    {
        dydx[0] = -y[1];
        dydx[1] = y[0] - (1.0/x)*y[1];
        dydx[2] = y[1] - (2.0/x)*y[2];
        dydx[3] = y[2] - (3.0/x)*y[3];
    }

    void jacobian
    (
        const scalar x,
        const scalarField& y,
        const label li,
        scalarField& dfdx,
        scalarSquareMatrix& dfdy
    ) const
    {
        dfdx[0] = 0.0;
        dfdx[1] = (1.0/sqr(x))*y[1];
```

```
        dfdx[2] = (2.0/sqr(x))*y[2];
        dfdx[3] = (3.0/sqr(x))*y[3];

        dfdy(0, 0) = 0.0;
        dfdy(0, 1) = -1.0;
        dfdy(0, 2) = 0.0;
        dfdy(0, 3) = 0.0;

        dfdy(1, 0) = 1.0;
        dfdy(1, 1) = -1.0/x;
        dfdy(1, 2) = 0.0;
        dfdy(1, 3) = 0.0;

        dfdy(2, 0) = 0.0;
        dfdy(2, 1) = 1.0;
        dfdy(2, 2) = -2.0/x;
        dfdy(2, 3) = 0.0;

        dfdy(3, 0) = 0.0;
        dfdy(3, 1) = 0.0;
        dfdy(3, 2) = 1.0;
        dfdy(3, 3) = -3.0/x;
    }
};


int main(int argc, char *argv[])
{
    argList::validArgs.append("ODESolver");
    argList args(argc, argv);

    testODE ode;

    dictionary dict;
    dict.add("solver", args[1]);

    autoPtr<ODESolver> odeSolver = ODESolver::New(ode, dict);

    scalarField y(ode.nEqns());
    y[0] = 1.0;
    y[1] = 1.0;
```

```
    y[2] = 1.0;
    y[3] = 1.0;

    scalar dxEst = 0.5;
    odeSolver->relTol() = 1e-4;
    odeSolver->solve(1.0, 2.0, y, 0, dxEst);

    Info<< "Numerical:  y(2.0) = " << y << ", dxEst = " << dxEst <<
   endl;

    Info<< "\nEnd\n" << endl;

    return 0;
}
```

6.12 OpenFOAM 中的边界条件

6.12.1 边界条件计算方法

有限体积法中边界条件的作用通过改变稀疏矩阵的系数与源项来进行。考虑图6.27所示的一维网格单元，结合典型的无量纲化一维传输方程（各项系数均为一）：

$$\frac{\partial \phi}{\partial t}+\frac{\partial \phi}{\partial x}-\frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right)=S \tag{6.185}$$

对 A 网格点进行离散有[71]：

$$\frac{\phi_A^{t+\Delta t}-\phi_A^t}{\Delta t}+\frac{\phi_A^{t+\Delta t}-\phi_0}{0.5\Delta x}-\frac{\frac{\phi_B^{t+\Delta t}-\phi_A^{t+\Delta t}}{\Delta x}-\frac{\partial \phi_0}{\partial x}}{\Delta x}=S \tag{6.186}$$

很明显，方程(6.186)中存在未知的 ϕ_0 以及 $\frac{\partial \phi_0}{\partial x}$，这两项的存在，需要用户指定边界条件。有 2 点需要注意：

- 边界条件涉及到的变量可以显性离散，也可以隐性离散；
- 时间项与源项 S 的离散，和边界条件无关；
- 对于常见的 CFD 中的偏微分方程，边界条件主要影响的即为对流项和扩散项；

因此，分析对流项与扩散项的边界条件，是理解边界条件植入的精髓。

首先要说明的是在 OpenFOAM 中，对流项边界条件对矩阵对角线系数的影响通过`valueInternalCoeffs`进行更新[72]，对矩阵源项的影响通过`valueBoundaryCoeffs`进行更

[71] 假定正向迎风格式：$\phi_1=\phi_A$

[72] 注意区分控制方程的离散形式和连续形式，方程6.185为连续形式，对其进行有限体积法离散之后，每个网格点均具备方程6.186的形式，由于存在大量网格点，若干个方程6.186组成稀疏矩阵系统，通常写为 $a_P\phi_P+\sum a_N\phi_N=S$，这里的 S 即为矩阵源项，其不同于方程6.185中的 S。a_P 为矩阵对角线系数。

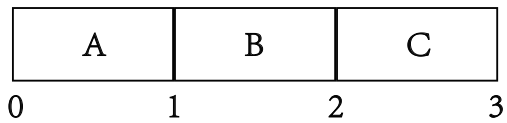


图 6.27: 一维有限体积网格。

新。扩散项边界条件对矩阵对角线系数的影响通过gradientInternalCoeffs进行更新，对矩阵源项的影响通过gradientBoundaryCoeffs进行更新。

对于对流项（隐性离散），固定值边界条件值 ϕ_0 为固定值，不管 ϕ_0 的值为多少，其并不改变对流项中 ϕ_A 的对角线系数 $\frac{1}{0.5\Delta x}$（valueInternalCoeffs）。同时，$\frac{\phi_0}{\Delta x}$ 将进入离散矩阵的源项部分并将其进行改变（valueBoundaryCoeffs）。若其为零法向梯度边界条件，则 $\phi_0 = \phi_A$，在离散方程中，对流项变为 0，因此，其改变 ϕ_A 的对角线系数，但并没有项进入矩阵源项部分，因此矩阵源项不变。类似的，对于扩散项（隐性离散），若其为固定值边界条件，有：

$$-\frac{\frac{\partial \phi_0}{\partial x}}{\Delta x} = -\frac{\frac{\phi_A^{t+\Delta t} - \phi_0}{0.5\Delta x}}{\Delta x} \tag{6.187}$$

因此，固定值边界条件将改变矩阵对角线系数（gradientInternalCoeffs），同时改变矩阵源项（gradientBoundaryCoeffs）。若其为零法向梯度边界条件，则 $\frac{\partial \phi_0}{\partial x} = 0$，则其不改变对角线系数也不改变源项。

6.12.2 OpenFOAM 边界条件速查

mixed：混合固定参考值/面法向梯度边界条件。也被称之为 Robin 边界条件。

$$\psi_f = w\psi_{\text{refValue}} + (1-w)\left(\psi_P + |\mathbf{d}|\nabla_f \psi\right) \tag{6.188}$$

其中 $\nabla_f \psi$ 表示面法向梯度，为一个标量，$|\mathbf{d}|$ 表示网格体心距离壁面的距离，ψ_{refValue} 为固定参考值。下面是一个使用范例：

```
    <patchName>
    {
        type            mixed;
        refValue        uniform (0 0 0);
        refGradient     uniform (0 0 0);
        valueFraction   uniform 0.5;
    }
```

其可以实现一种半滑移边界条件。

另一方面，考虑 $w = 0$ 的情况下，有：

$$\psi_f = \psi_P + |\mathbf{d}|\nabla_f \psi \tag{6.189}$$

这与面法向梯度的定义是一致的：

$$\frac{\psi_f - \psi_P}{|\mathbf{d}|} = \nabla_f \psi \tag{6.190}$$

advective：基于mixed边界条件的一种瞬态衍生出口边界条件。advective边界条件认为 ψ 的传输，尤其在边界处遵循下面的方程：

$$\frac{\partial \psi}{\partial t} + U \nabla_f \psi = 0 \tag{6.191}$$

$$U = \frac{\phi_f}{|\mathbf{S}_f|} \tag{6.192}$$

其中 U 表示传输速度标量，ϕ_f 表示边界面的通量，$\nabla_f \psi$ 表示面法向梯度，为一个标量。无反射边界不同于零梯度边界条件，其认为边界处的值是通过传输方程来计算而来的。这里假定采用欧拉时间格式且空间项隐性离散，针对边界处的面，其离散形式为

$$\frac{\psi_f^{t+\Delta t} - \psi_f^t}{\Delta t} + \frac{\phi_f}{|\mathbf{S}_f|} \frac{\psi_f^{t+\Delta t} - \psi_{\mathrm{P}}^{t+\Delta t}}{|\mathbf{d}|} = 0 \tag{6.193}$$

注意在这里假定边界面 $_f$ 存在网格点 $_{\mathrm{P}}$ 的右侧，且认为波的传输是纯粹的向右的，因此对流项的离散采用了向左的模板。可以认为是一种纯粹的迎风的思想。进一步推导有：

$$\psi_f^{t+\Delta t} - \psi_f^t + \alpha(\psi_f^{t+\Delta t} - \psi_{\mathrm{P}}^{t+\Delta t}) = 0 \tag{6.194}$$

$$\psi_f^{t+\Delta t} = \frac{1}{1+\alpha}\psi_f^t + \frac{\alpha}{1+\alpha}\psi_{\mathrm{P}}^{t+\Delta t} \tag{6.195}$$

定义权重：

$$w = \frac{1}{1+\alpha} \tag{6.196}$$

α 表示边界面的近似库朗数：

$$\alpha = \frac{\phi_f}{|\mathbf{S}_f|} \frac{\Delta t}{|\mathbf{d}|} \tag{6.197}$$

定义参考值：

$$\psi_{\mathrm{refValue}} = \psi_f^t \tag{6.198}$$

边界条件可以写为：

$$\psi_f^{t+\Delta t} = w\psi_{\mathrm{refValue}}^t + (1-w)\psi_{\mathrm{P}}^{t+\Delta t} \tag{6.199}$$

很明显，advective边界条件认为 $\psi_f^{t+\Delta t}$ 的值是基于固定值 ψ_f^t 和零法向梯度（$|\mathbf{d}|\nabla_f \psi = 0$）的边界条件组合。因此，其可以作为mixed边界条件的一种特殊形式。方程(6.199)就是最终的无反射边界条件。

文献 [129] 中将方程(??)称之为完美无反射边界条件。完美无反射边界条件可能是病态的。尤其是对于亚音速流动，NS 方程存在向左传输的特征变量。若使用完美的无反射边

界条件，则会导致一个变量缺失边界条件进而整个方程系统是病态的。通常情况下，出口压力需要给定固定值边界条件，这样可以解决方程病态问题。但其会导致向左传输的小扰动的气动压力的向左传输，也即存在一种波的反射。在大部分情况下，这种向左的小扰动的传输是不可见的，因为其幅值非常小。但在一些情况下，若流场结果存在反射，则不能使用压力出口固定值边界条件。

为了解决这个问题，文献 [129] 将其进行拓展成为一种非完美的，也即“部分”无反射边界条件：

$$\frac{\partial \psi}{\partial t} + U\nabla_f \psi + (-U)\left(\frac{\psi_{ref} - \psi}{l_{lnf}}\right) = 0 \tag{6.200}$$

ψ_{ref} 表示足够远处的 ψ 的值，也即远场的值。方程(6.200)的第三项表示远场向左侧传输的贡献。注意其中的传输方向向左，因此迎风的方向与第二项相反。如果出口的 $\psi_f = \psi_{ref}$，第三项失去作用。如果如果出口的 $\psi_f < \psi_{ref}$，第三项的贡献小于 0，会导致 ψ_f 的增加直至 $\psi_f = \psi_{ref}$。如果如果出口的 $\psi_f > \psi_{ref}$，第三项的贡献大于 0，会导致 ψ_f 的减小直至 $\psi_f = \psi_{ref}$。

方程(6.200)离散形式为：

$$\frac{\psi_f^{t+\Delta t} - \psi_f^t}{\Delta t} + \frac{\phi_f}{|\mathbf{S}_f|}\frac{\psi_f^{t+\Delta t} - \psi_{\mathrm{P}}^{t+\Delta t}}{|\mathbf{d}|} + \left(-\frac{\phi_f}{|\mathbf{S}_f|}\right)\frac{\psi_{ref} - \psi_f^{t+\Delta t}}{l_{inf}} = 0 \tag{6.201}$$

$$\psi_f^{t+\Delta t} - \psi_f^t + \alpha(\psi_f^{t+\Delta t} - \psi_{\mathrm{P}}^{t+\Delta t}) = \frac{\phi_f \Delta t}{|\mathbf{S}_f| l_{inf}}(\psi_{ref} - \psi_f^{t+\Delta t}) \tag{6.202}$$

定义

$$k = \frac{\phi_f \Delta t}{|\mathbf{S}_f| l_{inf}} \tag{6.203}$$

可见 k 与 α 的定义很相似。同时有

$$\psi_f^{t+\Delta t} - \psi_f^t + \alpha(\psi_f^{t+\Delta t} - \psi_{\mathrm{P}}^{t+\Delta t}) = k(\psi_{ref} - \psi_f^{t+\Delta t}) \tag{6.204}$$

$$(1 + \alpha + k)\psi_f^{t+\Delta t} - \psi_f^t = k\psi_{ref} + \alpha\psi_{\mathrm{P}}^{t+\Delta t} \tag{6.205}$$

$$\psi_f^{t+\Delta t} = \frac{1}{1+\alpha+k}(k\psi_{ref} + \psi_f^t) + \frac{\alpha}{1+\alpha+k}\psi_{\mathrm{P}}^{t+\Delta t} \tag{6.206}$$

$$\psi_f^{t+\Delta t} = w\frac{\psi_f^t + k\psi_{ref}}{1+k} + (1-w)\phi_{\mathrm{P}}^{t+\Delta t} \tag{6.207}$$

$$w = \frac{1+k}{1+\alpha+k} \tag{6.208}$$

waveTransmissive：基于`advective`边界条件进一步衍生的出口边界条件。其主要用于超音速出口 [129]。Poinsot 和 Lelef 在 1992 年发表在 JCP 的文章，主要讨论的就是`waveTransmissive`边界条件底层的算法。该文章在 2024 年被引用 4600 次。

waveTransmissive边界条件与advective边界条件的区别仅仅在于传输速度 U 的不同。waveTransmissive边界条件中的 U 表示为：

$$U = \frac{\phi}{|\mathbf{S}_f|} + \sqrt{\frac{\gamma}{\Psi}} \tag{6.209}$$

其中 $\gamma = C_p/C_v$，Ψ 表示可压缩性。

下面是waveTransmissive边界条件的使用实例：

```
outlet
{
    type     waveTransmissive;
    lInf     0.05;
    fieldInf     1600;
    gamma    1.4;
}
```

inletOutlet：在面通量小于 0 的时候（进口），给定固定值边界条件，在面通量大于 0 的时候（出口），给定零法向梯度边界条件。

$$\psi_f = w\psi_{\text{fixedValue}} + (1-w)\psi_P \tag{6.210}$$

其中 w 为权重，其值取决于面通量。如果是进口，$w = 1$，如果是出口，$w = 0$。考虑图6.28，翼形外圈的边界被处理成 inletOutlet，在这这种情况下，可以自由的调整攻角，而不需要重新生成边界面。类似的，对于一些存在回流的区域，回流面积会随着时间变化，inletOutlet 自动的将流入流指定固定值边界，流出流指定零法向梯度边界。

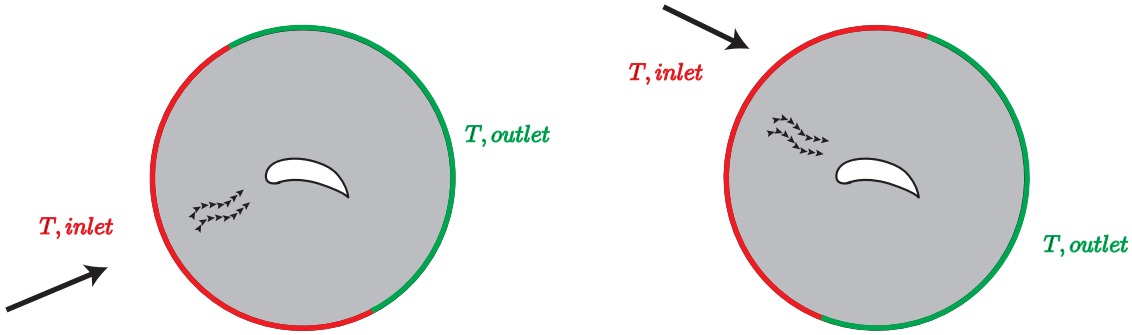


图 6.28: 翼形计算网格示意图。外围针对 T 变量使用 inletOutlet 边界。

freestreamPressure：对于压力的逐渐过渡的固定值边界条件与零法向梯度混合边界条件。其与mixed边界条件的区别仅仅在于权重的不同。

$$\psi_f = w\psi_{\text{fixedValue}} + (1-w)\left(\psi_P + |\mathbf{d}|\nabla_f\psi\right) \tag{6.211}$$

其中 w 为权重，在亚音速的情况下：

$$w = 0.5 + 0.5\frac{\mathbf{U}_f \cdot \mathbf{n}_f}{|\mathbf{U}_f|} \tag{6.212}$$

其中 $\mathbf{n}_f$ 表示面单位矢量，$\mathbf{U}_f$ 表示面上的速度。在超音速的情况下：

$$w = 0.5 - 0.5\frac{\mathbf{U}_f \cdot \mathbf{n}_f}{|\mathbf{U}_f|} \tag{6.213}$$

权重逐渐过渡，其值位于 0，1 之间。

freestreamVelocity：对于速度的逐渐过渡的固定值边界条件与零法向梯度混合边界条件。其与mixed边界条件的区别仅仅在于权重的不同。

$$\psi_f = w\psi_{\text{fixedValue}} + (1-w)\left(\psi_P + |\mathbf{d}|\nabla_f\psi\right) \tag{6.214}$$

其中 w 为权重：

$$w = 0.5 - 0.5\frac{\mathbf{U} \cdot \mathbf{n}_f}{|\mathbf{U}|} \tag{6.215}$$

$$\mathbf{U} = \frac{\mathbf{U}_f + \mathbf{U}_\mathrm{P}}{2} \tag{6.216}$$

权重逐渐过渡，其值位于 0，1 之间。

freestream：在面通量小于 0 的时候（进口），给定固定值边界条件，其值通过freestreamValue指定，在面通量大于 0 的时候（出口），给定零法向梯度边界条件。其等同于inletOutlet边界条件。在压力、速度基于freestreamPressure与freestreamValue指定时，其他变量（如湍流变量）可指定为freestream或inletOutlet。freestream的权重固定，要么是 0，要么是 1，不存在中间过渡状态。

上文所述的freestream一系列边界条件的使用，可以在一些特定的网格情况下自动适配。考虑图6.29中的翼形计算，其外部为一圆形边界，速度边界均指定为自由流速度，压力边界均指定为自由流压力。考虑计算域最左侧的面，由于 $\frac{\mathbf{U_f}\cdot\mathbf{n_f}}{|\mathbf{U_f}|} = -1$，因此freestreamVelocity变成固定值边界条件，freestreamPressure变成零梯度边界条件。考虑最右侧的面，freestreamVelocity变成零梯度边界条件，freestreamPressure变成固定值边界条件。在速度与计算域完美相切的情况下，等同于下文将要介绍的混合边界条件。

totalPressure：基于dynamicPressure边界条件的总压边界条件，为一种压力固定值边界条件，在比较新版本的 OpenFOAM 中，其具体的值取决于流入还是流出，以及流动的类型。较久版本的 OpenFOAM 中总压边界条件不具有流入流出的调节功能，其可以用于流入的进口。OpenFOAM 中的总压边界条件中的总压符号为 p_0。静压的值取决于边界处的通量是正值（流出）还是负值（流入）。如果是流出，p_0 的值就是静压的值。下面考虑流入的情况，进口速度的值依据压力的值自行调节，其静压也等于一个固定值边界条件，但不是给定的 p_0。

不可压缩流动低音速情况下静压可以这样计算：

$$p = p_0 - \frac{1}{2}|\mathbf{U}|^2 \tag{6.217}$$

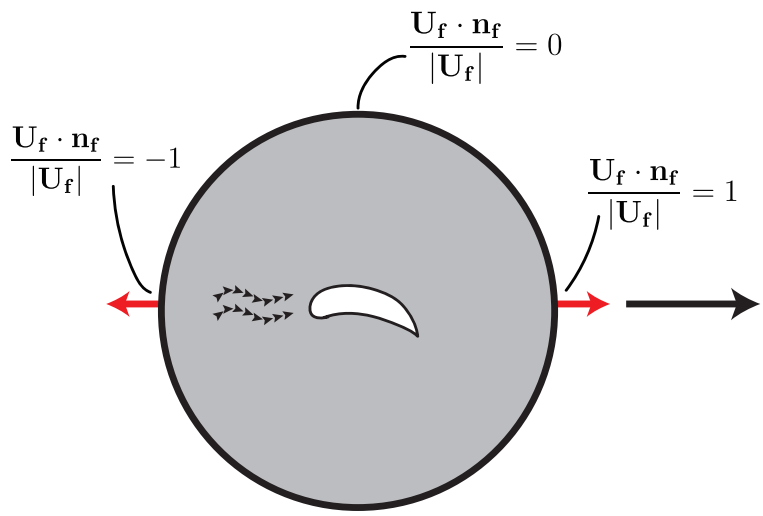


图 6.29: 翼形计算网格示意图。外围使用自由流边界条件。

在使用的时候需要给定p0的值。可压缩低音速情况下静压可以这样计算：

$$p = p_0 - \frac{1}{2}\rho|\mathbf{U}|^2 \tag{6.218}$$

在使用的时候需要给定p0、rho的值。可压缩近音速情况下静压可以这样计算：

$$p = \frac{p_0}{1+0.5\psi|\mathbf{U}|^2} \tag{6.219}$$

在使用的时候需要给定p0、psi的值。其中 ψ 表示可压缩性，其定义为 $\psi=\rho/p$。可压缩超音速情况下静压可以这样计算：

$$p = \frac{p_0}{(1+0.5\psi G|\mathbf{U}|^2)^{\frac{1}{G}}} \tag{6.220}$$

在使用的时候需要给定p0、psi、G的值。其中 G 为 $\frac{\gamma}{1-\gamma}$，其中 γ 为 C_p/C_v。

在一些开放边界中，推荐 totalPressure 总压边界条件因为其可以自行判断流动的流入以及流出。在流出的情况下，p0即为出口静压，在流入的情况下，p0即为进口总压。在超音速情况下，totalPressure 压力边界条件一般和 totalTemperature 总温边界条件一起使用。

下面是总压边界条件的使用实例：

```
inlet
{
    type            totalPressure;
    p0              uniform 100000;
    value           uniform 100000;
}
```

totalTemperature：总温边界条件。参考(4.399)，依据总温的定义，结合下列公式：

$$Ma = \frac{|\mathbf{U}|}{c}, c = \sqrt{\gamma RT}, \psi = \frac{1}{RT} \tag{6.221}$$

有

$$T = \frac{T_0}{1 + \frac{1}{2}\frac{\gamma-1}{\gamma}\frac{1}{RT}|\mathbf{U}|^2} \tag{6.222}$$

方程(6.222)即 OpenFOAM 植入的总温边界条件。

entrainmentPressure：总压边界条件变种。跟totalPressure非常类似，但是其只能用于不可压缩流。在流体流出的时候，为固定值压力边界条件。流体流入的时候，静压这样计算：

$$p = p_0 - \frac{1}{2}U_n|U_n|, \quad U_n = \frac{\mathbf{U}_f \cdot \mathbf{S}_f}{|\mathbf{S}_f|} \tag{6.223}$$

fanPressure：一种可以模拟风扇压力降的压力边界条件。风扇在出场设计的时候，通常会给定一个风扇曲线，来描述风扇产生的流量与风压的关系。风扇可以工作在不同的操作点。在固定转速下，风压大，风量小。想要同时增大风压，且提高风量，只能加大风扇转速。散热风扇风量是指风扇每分钟送出或吸入的空气总体积。风扇产品经常使用的风量单位是 CFM，即为每分钟立方英尺。风量越大的风扇其散热能力也越高。这是因为空气的热容量是一定的，更大的风量，也就是单位时间内更多的空气能带走更多的热量。散热风扇的风压又称为静压，指风扇在进气端与吹出端的气压差异。

在使用fanPressure边界条件的时候，要给定风扇曲线。下面是一个向外抽风的风扇的边界条件设置。

```
    outlet
    {
        type            fanPressure;
        rho             rho;
        psi             none;
        gamma           1;
        p0              uniform 0;
        value           uniform 0;
        fanCurve
        {
            type            table;
            values
            6
            (
                (0 50) //0立方米每分钟（阻抗非常大），50的风压
                (0.002 40)
                (0.004 30)
                (0.006 20)
                (0.008 10)
                (0.01  0) //0.01立方米每分钟，风压为0，基本无阻抗
            );
```

```
        }
        direction        out;
    }
```

该边界条件通过风扇曲线，依据边界的速度判断当前的风压是多少，然后通过下式对总压 p_0 进行更新：

$$p_0 = p_0 - d_p \tag{6.224}$$

其中 d_p 表示风压。在抽风的情况下，$d_p > 0$，反之 $d_p < 0$。在计算出总压后，该边界条件可以看做是总压边界条件。

在使用的过程中，对于抽吸类的模拟，流体出口可以给定`fanPressure`，流体入口可以给定总压边界条件。通过压差来驱使流动。

atmBoundaryLayer：中性大气边界条件。有关大气边界条件的更详细的理论请参考4.7一节。在 OpenFOAM 中，`atmBoundaryLayer`并不是一个可以实际使用的边界条件，而是一个抽象类型。在算例层面，需要给定具体的边界条件类型：

- `atmBoundaryLayerInletEpsilon`：进口湍流动能耗散率边界条件；
- `atmBoundaryLayerInletK`：进口湍流动能边界条件；
- `atmBoundaryLayerInletVelocity`：进口速度边界条件；
- `nutkAtmRoughWallFunction`：湍流粘度壁面函数边界条件；

目前 $k, \varepsilon, \mathbf{U}$ 植入的`atmBoundaryLayer`即为第4.7一节讨论的 RH93 边界条件。在 RH93 边界条件中并没有对湍流粘度进行调整。在 OpenFOAM 中，`nutkAtmRoughWallFunction`植入的壁面函数与方程(6.308)非常类似，但是其中的 E 要变为 $\tilde{E} = (y + z_0)/z_0$。

zeroGradient：零法向梯度边界条件。大量的用于出口以及壁面。正交网格条件下简单的可以理解为边界网格面的值与内部网格单元的值相等。对于任意的标量 ψ，从物理层面的理解，零法向梯度边界条件可以简写为：

$$(\nabla\psi\cdot)_f\mathbf{n}_f = 0 \tag{6.225}$$

另一方面，从数值方面可以获得相同的结果。例如压力的零法向梯度边界条件为 $(\nabla p)_f \cdot \mathbf{n}_f = 0$。考虑动量方程：

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla\cdot(\mathbf{U}\mathbf{U}) = -\nabla p + \nabla\cdot(\nu\nabla\mathbf{U}) \tag{6.226}$$

首先，壁面处均为不可穿透壁面，也即：

$$\mathbf{U}_f \cdot \mathbf{n}_f = 0 \tag{6.227}$$

对于网格面，同样的满足动量方程，因此左右两侧全部 $\cdot\mathbf{n}_f$ 有：

$$\left(\frac{\partial \mathbf{U}}{\partial t}+\nabla\cdot(\mathbf{U}\mathbf{U})\right)_f\cdot\mathbf{n}_f=\left(-\nabla p+\nabla\cdot(\nu\nabla\mathbf{U})\right)_f\cdot\mathbf{n}_f \tag{6.228}$$

其中对于时间项，有：

$$\frac{\partial \mathbf{U}_f}{\partial t}\cdot\mathbf{n}_f=\frac{\partial \mathbf{U}_f\cdot\mathbf{n}_f}{\partial t}=0 \tag{6.229}$$

对于对流项有：

$$\nabla\cdot(\mathbf{U}_f\mathbf{U}_f)\cdot\mathbf{n}_f=(\mathbf{U}_f\cdot\nabla\mathbf{U}_f+\mathbf{U}_f(\nabla\cdot\mathbf{U}_f))\cdot\mathbf{n}_f=(\mathbf{U}_f\cdot\nabla\mathbf{U}_f)\cdot\mathbf{n}_f+\mathbf{U}_f(\nabla\cdot\mathbf{U}_f)\cdot\mathbf{n}_f \tag{6.230}$$

方程(6.230)的最终两项中均需要乘 $\mathbf{U}_f$[73]，因此壁面处 $\nabla\cdot(\mathbf{U}_f\mathbf{U}_f)\cdot\mathbf{n}_f=0$。对于忽略粘性的扩散项有 $(\nabla^2\mathbf{U}_f)\cdot\mathbf{n}_f$，其看起来不是很好分析。如果将其展开，有 x,y 方向的分量分别为：

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2},\ \frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2} \tag{6.231}$$

这两项分别是速度对网格的二阶导数。对于流体的流动，如果是 x 方向的主流方向，那么 u 对于 x 的一阶导数应该近似为 0[74]，同时有 $\frac{\partial v}{\partial y}=0$（连续性方程）。在这种情况下，有 $(\nabla p)_f\cdot\mathbf{n}_f=0$。如果是 y 方向的主流方向，那么 v 对于 y 的一阶导数应该近似为 0，同时有 $\frac{\partial u}{\partial x}=0$（连续性方程）。在这种情况下，同样有 $(\nabla p)_f\cdot\mathbf{n}_f=0$。因此，压力在壁面处，存在零法向梯度边界条件。

fixedFluxPressure：一种压力近似零法向梯度边界条件。主要用于密度存在变化的流动，如多相流以及传热流动。在考虑重力的情况下，由于水压的存在，压力在法向上不再是零法向梯度。在考虑水压的情况下，OpenFOAM 求解器求解的为 p_{rgh} 方程，本质上 p_{rgh} 的梯度要与水压互相均衡。因此从物理层面来理解，考虑水压情况下的压力边界条件在三维情况下为：

$$(\nabla p_{rgh})_f\cdot\mathbf{n}_f=-\left(\mathbf{g}\cdot\mathbf{h}\nabla\rho\right)_f\cdot\mathbf{n}_f \tag{6.232}$$

因此壁面处 p_{rgh} 不再是零法向梯度边界条件，而是满足方程(6.232)。其也是老版本 OpenFOAM 中`buoyantPressure`边界条件的植入方法。

下面从数值方面来进行理解。考虑可压缩稳态流动，对于边界处有（考虑仅有重力作用的体积力）：

$$\mathbf{U}_f=\mathbf{HbyA}_f^*-\frac{1}{A_{\mathrm{P},f}^n}(\mathbf{g}\cdot\mathbf{h}\nabla\rho)_f-\frac{1}{A_{\mathrm{P},f}^n}(\nabla p_{rgh})_f \tag{6.233}$$

其中 $\mathbf{U}_f$ 为已知的量（壁面处一般是 0，或者与壁面毗连网格的值相同）。$\mathbf{HbyA}_f^*$ 为压力速度耦合迭代过程中的速度值。每一次的迭代会发生变化。方程移项有：

$$(\nabla p_{rgh})_f\cdot\mathbf{n}_f=\frac{\left(\mathbf{HbyA}_f^*-\frac{1}{A_{\mathrm{P},f}^n}(\mathbf{g}\cdot\mathbf{h}\nabla\rho)_f-\mathbf{U}_f\right)\cdot\mathbf{S}_f}{|\mathbf{S}_f|\frac{1}{A_{\mathrm{P},f}^n}} \tag{6.234}$$

[73]注意，$\mathbf{U}_f=0$，不代表 $\nabla\cdot\mathbf{U}=0$。但 $\mathbf{U}_f(\nabla\cdot\mathbf{U})$ 的模一定是 0。

[74]对于拐角处不为 0，但目前 CFD 软件均为考虑这个问题。应该是大多将其忽略罢了。

方程(6.234)右侧即为`fixedFluxPressure`针对压力需要指定的面法向梯度值[75]。其中的重力加速项，体现在 $\mathbf{HbyA}_f^* - \frac{1}{A_{\mathrm{P},f}^n}\mathbf{g}\cdot\mathbf{h}(\nabla\rho)_f$ 组建的通量中。在每一个时间步收敛的情况下，$\mathbf{HbyA}_f^* = \mathbf{U}_f$，方程(6.234)与方程(6.232)相同。

prghPressure：一种压力固定值边界条件。计算公式为

$$p_{rgh} = p - \rho\mathbf{g}\cdot\mathbf{h} \tag{6.235}$$

p 表示用户给定的压力固定值。其不同于 fixedValue 固定值边界条件（p_{rgh} 固定），尤其是出口可能存在多相时，prghPressure 可以依据密度进行调整。例如考虑一个平面出口，左边一半为液体，右边一半为空气，在这种情况下，p 固定为均一值，但 p_{rgh} 相差若干数量级（取决于密度）。存在 prghPressure 更适用于 interFoam 以及 multiPhaseEulerFoam 之类求解器（如图6.30所示范例）。

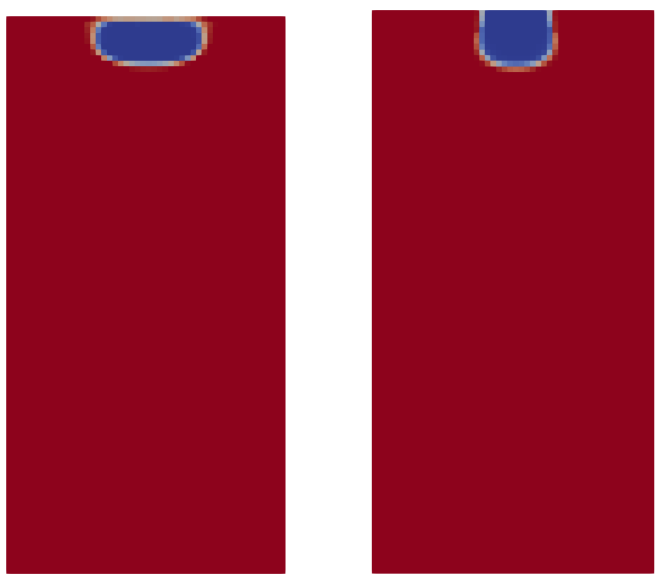


图 6.30: 对 p_rgh 场应用 fixedValue（左侧）以及 prghPressure（右侧）模拟的单气泡自由上浮。fixedValue 边界导致气泡困在流场中。

uniformFixedValue：一种简单的固定值边界条件，但是可以依据时间进行变化。下面是一个使用范例：

```
    inlet
    {
        type            uniformFixedValue;
        uniformValue    table
        (
            (0   (0 0 0))
```

[75]大部分求解器通过 constrainPressure 来计算这一项。也可以在求解器中直接调用 setSnGrad 函数来直接计算梯度。

```
            (5    (10 0 0))
        );
    }
```

上述代码实现的是随着时间变化的固定值边界条件。在 0-5 秒的时候，速度是 x 方向 0 m/s，在 5 秒之后，速度是 x 方向 10 m/s。

variableHeightFlowRate：进口值可变的固定值/零法向梯度混合边界条件。其需要输入一个最小值 $\alpha_{\min}$ 与最大值 $\alpha_{\max}$。α 在距离壁面的第一层网格如果在 $\alpha_{\min}$ 与 $\alpha_{\max}$ 之间且包含，则为一个零法向梯度边界条件。如果不在这个区间内，则为固定值边界条件，如果小于 $\alpha_{\min}$，则为固定值 $\alpha_{\min}$。反之亦然。这个边界条件可以用于可变高度水渠的相分数进出口（在进口更为重要）。在一些情况下，进口液面的高度可能是自由上升的。考虑一个竖直向上的单元，液面的上升导致上方某个网格单元的 α 值的增加，则这个网格单元边界处的 α 也设置为这个值。这个边界条件大体上等于零法向梯度边界条件，但是其更加稳定，因为其将进出口的 α 限定在了数值范围内。采用零法向梯度边界条件，如果某个网格的 α 越界，那么进一步的进口边界的 α 也会越界，进一步的导致求解稳定性问题。

activeBaffleVelocity：可穿透的速度挡板边界条件。是否穿透取决于压力。

```
    <patchName>
    {
        type            activeBaffleVelocity;
        p               p;
        cyclicPatch     cyclic1;
        orientation     1;
        openFraction    0.2;
        openingTime     5.0;
        maxOpenFractionDelta 0.1;
    }
```

$$x = x_{old} + sign(F_{net})\frac{\Delta t}{\Delta T} \tag{6.236}$$

其中 x 表示开放程度，x_{old} 表示上一个时间步的开放程度，F_{net} 表示单位面积的压力，其值为 $\sum p_f|\mathbf{S}_f|$，Δt 表示时间步，ΔT 表示用多少时间去决定是否开启或者关闭。

6.13 OpenFOAM 常用代码

6.13.1 div(U) = div(phi)?

`div(U)` 与 `div(phi)` 有一些细小的差别，但总体一致。在代码中，如果看到 `div(U)`，其数学形式为 $\sum \mathbf{U}_f \cdot \mathbf{S}_f$。如果看到 `div(phi)`，因为 `phi` 的数学形式为 $\mathbf{U}_f \cdot \mathbf{S}_f$，因此 `div(phi)` 的数学形式为 $\sum \mathbf{U}_f \cdot \mathbf{S}_f$。这也即二者的一致性。不一致的地方在于，在很多求

解器中，能够保证严格守恒的变量是 `phi`（数学形式 $\mathbf{U}_f \cdot \mathbf{S}_f$），而不是 `U`（数学形式 $\mathbf{U}$）。所以如果U是从算法计算出来的，那么很有可能出现 `div(U)` 与 `div(div(phi))` 不等的情况。如果 `phi` 是纯粹的从 `div(U)` 计算而来过来，则二者一致。

6.13.2 laplacian(ϕ) = div(grad(ϕ))?

在 OpenFOAM 中，扩散项 $\nabla \cdot (\nabla \phi)$ 的植入形式为

```
fvc::laplacian(psi)
```

从数学形式来讲，$\nabla \cdot (\nabla \phi)$ 似乎也可以写为

```
fvc::div(fvc::grad(psi))
```

然而，方程的连续形式和离散形式有时是不同的。OpenFOAM 创始人 Henry Weller 也表示：

> *"The consistency issue arises because laplacian uses a compact molecule and grad(div has to use an extended molecule. The only numerically consistent approach is to use the extended molecule for all three terms but then you loose the advantage of the compact molecule for laplacian."*

下面我们就证明二者的差异。

若采用拉普拉斯项形式对 $\nabla \cdot (\nabla \phi)$ 进行离散有[76]：

$$\int \nabla \cdot (\nabla \phi)\, \mathrm{d}V = \int \nabla \phi \mathrm{d}\mathbf{S} = \sum (\nabla \phi)_f \mathbf{S}_f = \sum \underbrace{\left((\nabla \phi)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} \right)}_{\text{snGrad}} \cdot |\mathbf{S}_f| \tag{6.237}$$

其中 snGrad 项可以调用紧致模板（Compact Stencil）进行离散。

若采用散体梯度的形式对 $\nabla \cdot (\nabla \phi)$ 进行离散有：

$$\int \nabla \cdot (\nabla \phi)\, \mathrm{d}V = \int \nabla \cdot \left(\frac{1}{\Delta V} \sum \phi_f \mathbf{S}_f \right) \mathrm{d}V = \sum \left(\frac{1}{\Delta V} \sum \phi_f \mathbf{S}_f \right)_f \cdot \mathbf{S}_f \tag{6.238}$$

其中 $\nabla \phi$ 调用 grad（梯度）离散格式，$\nabla \cdot$ 调用 div（散度）离散格式，整体上形成非紧致模板（Non-compact Stencil）。

俩种形式的区别在一维形式下更好理解，考虑 $\frac{\partial}{\partial x}\frac{\partial \phi}{\partial x}$，采用中心格式拉普拉斯形式的离散为

$$\int \frac{\partial}{\partial x}\frac{\partial \phi}{\partial x} \mathrm{d}V = \frac{\left(\frac{\partial \phi}{\partial x}\right)_w - \left(\frac{\partial \phi}{\partial x}\right)_e}{\Delta x} \Delta x \Delta y \Delta z = \frac{\phi_W + \phi_E - 2\phi_P}{\Delta x} \Delta y \Delta z \tag{6.239}$$

[76] /src/finiteVolume/finiteVolume/laplacianSchemes/gaussLaplacianScheme /gaussLaplacianScheme.C

若采用中心格式散体梯度的形式进行离散有：

$$\int \frac{\partial}{\partial x}\frac{\partial \phi}{\partial x}\mathrm{d}V = \int \frac{\partial}{\partial x}\left(\frac{\left(\frac{\phi_W+\phi_P}{2}-\frac{\phi_P+\phi_E}{2}\right)\Delta y\Delta z}{\Delta x\Delta y\Delta z}\right)\mathrm{d}V = \int \frac{\partial}{\partial x}\left(\frac{\phi_W-\phi_E}{2\Delta x}\right)\mathrm{d}V$$
$$= \frac{\phi_{WW}+\phi_{EE}-2\phi_P}{4\Delta x}\Delta y\Delta z \quad (6.240)$$

很明显，方程(6.239)调用的为相邻网格点，方程(6.240)调用了更远的网格点，有可能会引起震荡。

同时，正如 Henry Weller 所说，在 N-S 方程中，压力方程存在压力 p 的拉普拉斯项，只有其采用拉普拉斯形式进行离散的时候，才能保证调用紧致模板防止震荡。

6.13.3 fvm::ddt(alpha1) - fvc::ddt(alpha1)?

在 OpenFOAM 的一些求解器中存在这样的数值操作：

```
fvScalarMatrix alpha1Eqn
(
    fvm::ddt(alpha1) - fvc::ddt(alpha1) // WHY???
  - fvm::laplacian(alpha1alpha2f()*pPrimeByA_(), alpha1)
);

alpha1Eqn.relax();
alpha1Eqn.solve();
```

上述代码一般在这种情况下出现：先对alpha进行纯对流项求解更新场，然后添加扩散项。上述代码的主要问题是，为什么存在下面这一行代码：

```
fvm::ddt(alpha1) - fvc::ddt(alpha1)
```

其实这是一种纯粹的数值操作。现在对其进行解释。考虑下面的伪代码：

```
solve(fvm::ddt(alpha) + fvm::div(phi,alpha));

solve
(
    fvm::ddt(alpha1) - fvc::ddt(alpha1) // WHY???
  - fvm::laplacian(alpha1alpha2f()*pPrimeByA_(), alpha1)
);
```

其在进行第一个 solve 函数的时候，将 α 更新为 $\alpha^{t+\Delta t}$，同时其中的时间项表示为 $\frac{\alpha^{t+\Delta t}-\alpha^t}{\Delta t}$。在进行第二个 solve 函数的时候，`fvc::ddt(alpha1)` 可以表示为 $\frac{\alpha^{t+\Delta t}-\alpha^t}{\Delta t}$。因此考虑两个 solve 函数，第一个 solve 函数里面的 `fvm::ddt(alpha1)` 和第二个 solve 函数里面的 `fvc::ddt(alpha1)` 可以互相抵消。因此上面的代码等效于

```
solve
(
    fvm::ddt(alpha)
  + fvm::div(phi,alpha)
  - fvm::laplacian(alpha1alpha2f()*pPrimeByA_(), alpha1)
);
```

这种操作在数值上具有重要的意义。将其分为对流项以及扩散项单独求解，可以将对流项采用 MULES 算法保证对流项的有界。扩散项不会引起有界性的问题。

6.13.4 -fvc::flux(-phir,..) = fvc::flux(phir,..)?

OpenFOAM 中可以使用`flux()`函数计算通量。举例，对于 $\phi_f\alpha_f = \mathbf{U}_f \cdot \mathbf{S}_f\alpha_f$，可通过下述代码计算通量

```
phi = fvc::interpolate(U) & mesh.Sf();
phiAlpha = fvc::flux(phi,alpha);
```

在 interFoam 中，存在压面压缩项 $\nabla \cdot \left(\alpha\beta c|\mathbf{U}|\frac{\nabla\alpha}{|\nabla\alpha|}\right)$。注意其中 $\beta = 1-\alpha$。现在将其改写为 $\nabla \cdot (\mathbf{U}_r\alpha\beta)$，其中 $\mathbf{U}_r$ 表示压缩速度。界面压缩项的离散形式为

```
fvc::flux(-fvc::flux(-phir, beta, upwind), alpha, upwind)
```

在这里需要讨论的是这些负号，在调用合适的离散格式的时候，可以保证有界性。例如，考虑一维的压缩项，其可以简写为 $\partial u_r\alpha\beta/\partial x$。考虑网格单元面，按照上述代码，对这一项中的分子部分进行通量计算可以写为 $u_{r,f}\alpha_{\text{upwind}}\beta_{\text{downwind}}$，其原因下述，考虑下段代码：

```
-fvc::flux(-phir, beta, upwind)
```

`flux()`函数内的负号表示对 β 依据负的通量的方向进行迎风离散，即下风格式。由于`flux()`函数内添加的负号不仅改变了离散格式的选取方向，还改变了通量的正负，因此在`flux()`函数前应该添加负号将其进行还原。这段代码在数学上可以理解为 $-(-u_{r,f}\beta_{\text{downwind}})$。然后，$\alpha$ 依据 $-(-u_{r,f}\beta_{\text{downwind}})$ 进行离散。由于 $-(-u_{r,f}\beta_{\text{downwind}})$ 并没有改变方向，因此 α 迎风离散。

考虑下面三个网格单元，界面压缩项速度的方向源自于 α 的梯度。对于图示的 α 值，相应的速度方向倾向于将中间网格单元的值“挤压”到相邻网格，这也是界面压缩相存在的意义（使得界面更加尖锐）。然而，对于图示的 α 场，如果进行压缩，同时对 α 和 β 调用迎风格式：即 b 网格面的 α, β 均为 0.5，会导致 b 面存在相应的界面压缩通量，导致中心处网格单元的值小于 0.5（合理的），同时周围网格单元的 α 值大于 1。后者会导致越界，并不符合物理。a 面同理。

相反，如果对 α 调用迎风格式，β 调用下风格式，则 b 网格面的 α, β 分别为 0.5 和 0，进而界面压缩通量为 0，越界行为可以避免。

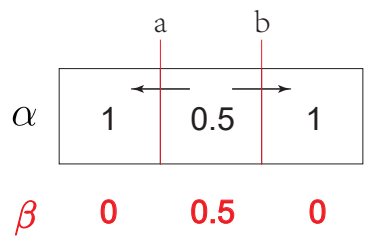


图 6.31: 界面压缩项网格示意图。a, b 表示网格单元面。

6.13.5 pEqn.flux() 与 fluxRequired?

通常来讲，对于给定变量在网格面上的通量，可以通过速度与变量以及网格面积的乘积来获得。例如考虑变量 T 的对流项 $\nabla\cdot(\mathbf{U}T)$ 对应的对流通量，其可以被定义为 $\mathbf{U}_f\cdot\mathbf{S}_f T_f$。考虑扩散通量 $\nabla\cdot(D\nabla T)$，其可以被定义为 $D_f(\nabla T)_f\cdot\mathbf{S}_f$，进一步可以定义为面法向梯度相关量 $D_f((\nabla T)_f\cdot\mathbf{S}_f/|\mathbf{S}_f|)|\mathbf{S}_f|$。以后者扩散通量举例，其可以通过下面的代码进行计算：

```
    fvc::snGrad(T)*mesh.magSf()*fvc::interpolate(D);
```

这是传统的通量计算方法。还有另外一种通量计算方法，其更加普适性且连贯。例如，如果对 T 方程组建稀疏线性系统，OpenFOAM 中的`flux()`可以直接调用稀疏线性系统的矩阵系数，反过来计算定义在网格面上的通量。这也即`flux()`函数的意义。例如对于下述代码

```
    fvScalarMatrix pEqn
    (
        fvm::laplacian(D, p) == fvc::div(phiHbyA)
    );
```

其为不可压缩单相流的压力泊松方程。对于其中的`pEqn`，其中的`fvc`部分进入源项，不参与`flux()`函数的计算。其左侧的拉普拉斯项，即表示压力的扩散项，参考前述 $\nabla\cdot(D\nabla T)$ 扩散通量的含义，有压力方程的扩散通量的数学意义：

$$\mathrm{fvm::laplacian(rAU, p)} = D_f\left((\nabla p)_f\cdot\frac{\mathbf{S}_f}{|\mathbf{S}_f|}\right)|\mathbf{S}_f| \tag{6.241}$$

方程(6.241)的右侧数学公式用代码可以表示为：

```
    fvc::interpolate(D)*fvc::snGrad(p)*mesh.magSf();
```

上述代码获得的通量场与`pEqn.flux()`是一致的。

需要注意的是，对于时间项以及源项，由于其并不参与网格间的流动，因此`flux()`函数并不考虑他们的贡献。同时，显性项离散不进入`flux()`函数的贡献，即使是对流项或者扩散项。

下面再进行举例，如果有如下代码：

```
    fvScalarMatrix pEqn
    {
        fvm::div(phiHbyA, p)
      - fvm::laplacian(D, p)
    }
```

pEqn.flux()即为：

$$\text{flux}() = p_f \mathbf{HbyA}_f \cdot \mathbf{S}_f - D_f \left((\nabla p)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|} \right) |\mathbf{S}_f| \tag{6.242}$$

最后需要注意的是，对于老版本的 OpenFOAM，要使用flux()函数，需要在算例层面开启fluxRequired关键词，如[77]：

```
    fluxRequired
    {
        p_rgh;
    }
```

否则会提示下列错误：

```
    flux requested but p_rgh not specified in the fluxRequired sub-
  dictionary of fvSchemes.
```

这是因为flux()函数需要存储矩阵的非对角系数，本身会增加存储以及计算消耗，因此默认并不会进行相关操作。只有显性指定的时候才会进行相应的存储。同时，如果开启fluxRequired，flux()函数最终的结果也需要包含非正交修正的贡献。这在下述代码中体现[78]：

```
    if (faceFluxCorrectionPtr_)
    {
        fieldFlux += *faceFluxCorrectionPtr_;
    }
```

这是因为对于方程(6.242)和(6.241)中的拉普拉斯项，其在非正交网格上需要进行修正。如果是正交网格，则若 if 判断为假，其为一个空指针[79]，代码并没有对其进行赋值：

```
    tmp<GeometricField<Type, fvsPatchField, surfaceMesh>>
  tfaceFluxCorrection
        = gammaSnGradCorr(SfGammaCorr, vf);

    if (this->tsnGradScheme_().corrected())
    {
```

[77] 请参考 fvMatrix.C 中的相关代码。
[78] fvMatrix.C
[79] gasuuLaplacianScheme.C

```
        tfaceFluxCorrection.ref() +=
            SfGammaSn*this->tsnGradScheme_().correction(vf);
    }

    fvm.source() -= mesh.V()*fvc::div(tfaceFluxCorrection())().
   primitiveField();

    if (mesh.schemes().fluxRequired(vf.name()))
    {
        fvm.faceFluxCorrectionPtr() = tfaceFluxCorrection.ptr();
    }
```

因此不会对 flux() 函数进行非正交修正。

在新版的 OpenFOAM 中，除了在算例层面指定外，也可以在求解器中增加下面一行来开启flux()函数：

```
mesh.schemes().setFluxRequired(p.name());
```

在这种情况下，不需要在算例层面显性的指定。

6.13.6 fvc::div(phiHbyA)

fvc::div(phiHbyA)在 OpenFOAM 中大量的出现，其主要用于计算通量的散度 $\nabla\cdot\mathbf{U}$。容易引起困惑的是，div为散度操作符，但是phiHbyA却是一个标量，对一个标量做散度操作这并不符合数学逻辑。接下来我们对这一问题进行解答[80]。

从数学形式来看，散度项 $\nabla\cdot\mathbf{U}$ 在有限体积法下可以离散为

$$\int \nabla\cdot\mathbf{U}\mathrm{d}V = \int \mathbf{U}_f\cdot\mathrm{d}\mathbf{S}_f = \sum \mathbf{U}_f\cdot\mathbf{S}_f \tag{6.243}$$

OpenFOAM 中将 $\sum \mathbf{U}_f\cdot\mathbf{S}_f$ 称之为phiHbyA（在这里认为 **U** 和 **HbyA** 都是表示速度）。其为一个surfaceScalarField面标量场。因此fvc::div(phiHbyA)在离散时，首先调用：

```
return GeometricField<Type, fvPatchField, volMesh>::New
(
    "div("+ssf.name()+')',
    fvc::surfaceIntegrate(ssf)
);
```

然后进一步调用surfaceIntegrate()函数，其为一个面积分函数。举例，如果代码中进

[80] 问题讨论参考本链接。

行surfaceIntegrate(phiHbyA)操作，其数学含义即为：

$$\text{fvc :: div(phiHbyA)} = \text{surfaceIntegrate(phiHbyA)} = \frac{1}{\Delta V}\sum \text{phiHbyA} = \frac{1}{\Delta V}\sum \mathbf{U}_f \cdot \mathbf{S}_f \quad (6.244)$$

因此，fvc::div(phiHbyA)中的div并不是散度操作，而是面积分的操作。同时需要注意，在组建phiHbyA的时候，可以通过fvc::flux(HbyA)函数调用不同的离散格式。

有关surfaceIntegrate()函数，更详细的可以参考第6.3.2节。

6.13.7 fvc::reconstruct()

fvc::reconstruct()函数主要用于处理动量方程中源项。例如表面张力项、压力项、浮力项、界面动量交换项等。此函数返回的场的变量要比输入的场变量高一阶。例如，若对标量进行 reconstruct，则得到矢量场，若对矢量做 reconstruct，则得到二阶张量场。现存一些讨论fvc::reconstruct()的文献很有意义，但并不完善 [1, 2]。本小节详述为什么需要fvc::reconstruct()以及其数学形式。

fvc::reconstruct()函数若给定计算域的面通量 ϕ_f，也即 $\mathbf{U}_f \cdot \mathbf{S}_f$，则可以重组出所有网格点的 $\mathbf{U}_\mathrm{P}$。首先看下面的方程：

$$\sum \phi_f = \sum (\mathbf{S}_f \cdot \mathbf{U}_f) \approx \sum (\mathbf{S}_f \cdot \mathbf{U}_\mathrm{P}) = \left(\sum \mathbf{S}_f\right) \cdot \mathbf{U}_\mathrm{P} \quad (6.245)$$

很明显方程中 $\mathbf{U}_f$ 与多个网格的速度相关，但是 $\mathbf{U}_\mathrm{P}$ 只和当前网格的速度相关，因此给定 $\sum \phi_f$，可以反推出 $\mathbf{U}_\mathrm{P}$。这就是fvc::reconstruct()的内部逻辑。但由于矢量不能求逆，也就是说下面的等式是不成立的，或者说是不能这么写：

$$\cancel{\left(\sum \mathbf{S}_f\right)^{-1} \sum \phi_f = \cdot \mathbf{U}_\mathrm{P}} \quad (6.246)$$

因此方程(6.245)虽然更易理解，但并不好计算。另一方面，方程(6.245)可以写为[81]：

$$\sum \mathbf{n}_f \phi_f \approx \sum (\mathbf{n}_f (\mathbf{S}_f \cdot \mathbf{U}_\mathrm{P})) = \sum ((\mathbf{n}_f \mathbf{S}_f) \cdot \mathbf{U}_\mathrm{P}) = \left(\sum \mathbf{n}_f \mathbf{S}_f\right) \cdot \mathbf{U}_\mathrm{P} \quad (6.247)$$

因此有[82]：

$$\mathbf{U}_\mathrm{P} \approx \left(\sum \mathbf{n}_f \mathbf{S}_f\right)^{-1} \sum \mathbf{n}_f \phi_f \quad (6.248)$$

代码中植入的即为方程(6.248).

之所以进行fvc::reconstruct()函数的原因在于可以防止数值震荡。举例说明：对于压力梯度项 ∇p，一种离散方法是将其写为fvc::grad(p)。其对应的代码为：

```
fvc::grad(p)
```

[81] 基本操作：$(\mathbf{ab}) \cdot \mathbf{c} = \mathbf{a}(\mathbf{b} \cdot \mathbf{c})$

[82] src/finiteVolume/finiteVolume/fvc/fvcReconstruct.C

其数学形式为[83]：

$$\frac{1}{\Delta V}\sum p_f \mathbf{S}_f = \frac{1}{\Delta V}\int \nabla p \mathrm{d}V = \nabla p_{\mathrm{P}} \tag{6.249}$$

其中调用了高斯定律将面加和转化为体积分。在第6.13.2节的讨论中可以看出，方程(6.249)的离散形式调用非紧致模板，跨越两个网格点的值，因此可能会产生数值震荡。

∇p 另外一种离散形式即通过`fvc::reconstruct()`的形式组建。对应的代码为

```
fvc::reconstruct
(
    fvc::snGrad(p)*mesh.magSf()
)
```

相应的数学形式为

$$\frac{\sum \mathbf{n}_f \underbrace{\left(\underbrace{(\nabla p)_f \cdot \frac{\mathbf{S}_f}{|\mathbf{S}_f|}}_{\text{snGrad}} \underset{\text{magSf}}{|\mathbf{S}_f|}\right)}_{input}}{\sum \mathbf{n}_f \mathbf{S}_f} = \nabla p_{\mathrm{P}} \tag{6.250}$$

参考第6.13.2节的讨论，可以看出方程(6.250)右侧的离散调用紧致模板，可以防止数值震荡。总体来说，`fvc::reconstruct()`函数结合面法向梯度，将普通的梯度离散导致的数值震荡抹平，即为`fvc::reconstruct()`函数存在的意义。

6.13.8 fvc::smooth()

`smooth()` 函数主要用于对某些场做平滑处理。这个函数在 OpenFOAM 中并不是非常常见，有两个相对重要的应用，就是对 LES 的亚格子尺度、以及局部时间步长的平滑处理上。`smooth()` 对某个场的平滑处理，并不遵循一定的物理，是一种纯数学操作。在 OpenFOAM 中，这种平滑处理的比例控制在 1.2（当然也可以用户自定义）。也就是说相邻网格的变量比要小于 1.2。现在以大涡模拟中的 `smoothDelta` 模型举例，在这个模型中，会将亚格子应力进行平滑处理。在下面的代码中[84]：

```
    forAll(geometricDelta, celli)
    {
        cellDeltaData[celli] = geometricDelta[celli];
    }

    // Set initial field on faces.
```

[83]在 OpenFOAM 中，虽然要对动量方程做体积分，但除了时间离散，诸如对流、散度、梯度等在离散时均除掉了网格体积。

[84]smoothDelta.C

```
    List<deltaData> faceDeltaData(mesh.nFaces());

    // Propagate information over whole domain.
    FaceCellWave<deltaData, scalar> deltaCalc
    (
        mesh,
        changedFaces,
        changedFacesInfo,
        faceDeltaData,
        cellDeltaData,
        mesh.globalData().nTotalCells()+1,  // max iterations
        maxDeltaRatio_
    );

    forAll(delta_, celli)
    {
        delta_[celli] = cellDeltaData[celli].delta();
    }
```

核心代码是 `FaceCellWave` 函数，其将 `cellDeltaData` 传输进来，进行平滑后，赋值给 `delta_`（亚格子应力)。

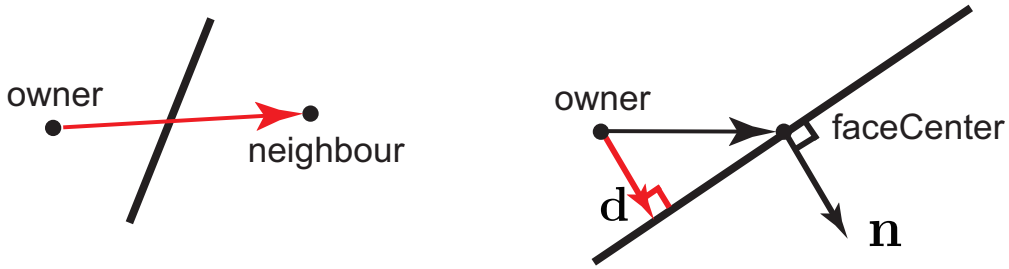


图 6.32: `deltaCoeffs` 函数中使用的 **d** 矢量（红色剪头）。

6.13.9 deltaCoeffs()

OpenFOAM 中的 `deltaCoeffs` 返回一个矢量的模的倒数。如图6.32所示，这个矢量对于内部面，即为这个内部面相对应的网格单元以及相邻网格单元的距离矢量 **d**。对于边界面，由于不存在相邻网格单元，因此这个距离矢量可以表示为边界面对应的网格单元距离边界的矢量。边界处的距离矢量 **d** 可以表示为：

$$\mathbf{d} = \mathbf{n}(\mathbf{n} \cdot (\mathbf{V}_{faceCenter} - \mathbf{V}_{owner})) \tag{6.251}$$

那么 `deltaCoeffs` 函数，就是返回的 $1/|\mathbf{d}|$。

6.13.10 R、sigma、prime2Mean、wallShearStress

要在结果中获取雷诺应力相关量，可以执行

```
simpleFoam -postProcess -func shearStress
```

在不可压缩的时候，其输出[85]：

$$\boldsymbol{\tau} = -\nu_{eff}\left(\nabla\mathbf{U} + \nabla\mathbf{U}^T - \frac{2}{3}\left(\nabla\cdot\mathbf{U}\right)\mathbf{I}\right) \tag{6.252}$$

如果单独需要壁面的壁面剪切应力，可以执行

```
simpleFoam -postProcess -func wallShearStress
```

wallShearStress 的计算公式为[86]

$$wallShearStress = \boldsymbol{\tau}\cdot\mathbf{n} = \begin{bmatrix} \tau_{11}n_1 + \tau_{12}n_2 + \tau_{13}n_3 \\ \tau_{21}n_1 + \tau_{22}n_2 + \tau_{23}n_3 \\ \tau_{31}n_1 + \tau_{32}n_2 + \tau_{33}n_3 \end{bmatrix} \tag{6.253}$$

可见，壁面剪切应力为一个矢量。考虑 y 法向 $\mathbf{n} = (0, 1, 0)$，有：

$$wallShearStress = \boldsymbol{\tau}\cdot\mathbf{n} = \begin{bmatrix} \tau_{12} \\ \tau_{22} \\ \tau_{32} \end{bmatrix} \tag{6.254}$$

旧版本 OpenFOAM 还存在`R()`与`sigma()`函数，与上文提到的 shearStress 类似，但略有区别。`R()`与`sigma()`表示[87]：

$$\mathbf{R} = -\nu_t\left(\nabla\mathbf{U} + \nabla\mathbf{U}^T - \frac{2}{3}\left(\nabla\cdot\mathbf{U}\right)\mathbf{I}\right) + \frac{2}{3}k\mathbf{I} \tag{6.255}$$

要注意 $\mathbf{R}$ 与 $\boldsymbol{\tau}$ 中粘度的度别，$\mathbf{R}$ 使用湍流粘度，$\boldsymbol{\tau}$ 使用有效粘度。非线性湍流模型中的`R()`需要添加非线性应力的贡献：

$$\mathbf{R} = -\nu_t\left(\nabla\mathbf{U} + \nabla\mathbf{U}^T - \frac{2}{3}\left(\nabla\cdot\mathbf{U}\right)\mathbf{I}\right) + \frac{2}{3}k\mathbf{I} + nonlinearContri \tag{6.256}$$

同时，依据 LES 模型以及 RANS 模型的不同，其中的变量 k 分别表示 LES 中模化的湍流动能以及 RANS 中的湍流动能。注意，在旧版本的 OpenFOAM 中，`R()`仅仅表示 $\frac{2}{3}k\mathbf{I}$。在这种情况下有：

$$\frac{1}{2}\mathrm{tr}(\mathbf{R}) = k \tag{6.257}$$

[85] 可压缩的时候要乘以密度。

[86] wallShearStress.C

[87] eddyViscosity.C

`prime2Mean`表示解析的雷诺应力，其表示 $\overline{\mathbf{U}'\mathbf{U}'}$。在 LES 模型中调用`prime2Mean`功能，可以获得解析的雷诺应力。其展开形式为：

$$\overline{\mathbf{U}'\mathbf{U}'} = \begin{bmatrix} \overline{u'u'} & \overline{u'v'} & \overline{u'w'} \\ \overline{v'u'} & \overline{v'v'} & \overline{v'w'} \\ \overline{w'w'} & \overline{w'v'} & \overline{w'w'} \end{bmatrix} \tag{6.258}$$

因为其为一个对称二阶张量，因此仅仅保留 6 个分量：

$$\overline{\mathbf{U}'\mathbf{U}'} = \begin{bmatrix} \overline{u'u'} & \overline{u'v'} & \overline{u'w'} \\ 0 & \overline{v'v'} & \overline{v'w'} \\ 0 & 0 & \overline{w'w'} \end{bmatrix} \tag{6.259}$$

在 OpenFOAM 中，其保存 6 个值，排序为 $\overline{u'u'}, \overline{u'v'}, \overline{u'w'}, \overline{v'v'}, \overline{v'w'}, \overline{w'w'}$。注意其中的第 1、4、6 个分量为平方项，因此只能为正。在 RANS 中不推荐使用`prime2Mean`，因为 RANS 求解的已经是时间平均后的流场。

6.13.11 fvm::ddt(psi, p) 与 psi*correction(fvm::ddt(p))?

可压缩瞬态算法的连续性方程需要计算密度时间项。其离散形式可以写为[88]：

$$\frac{\rho^{t+\Delta t} - \rho^t}{\Delta t} + \frac{1}{V_{\mathrm{P}}} \sum_f \rho_f^{t+\Delta t} \mathbf{U}_f^{t+\Delta t} \cdot \mathbf{S}_f = 0. \tag{6.260}$$

其中的第一项 $\frac{\rho^{t+\Delta t} - \rho^t}{\Delta t}$ 需要改写成为压力项后该方程可以用来组建压力方程。在第一种方法中，其可以处理为：

$$\frac{\rho^{t+\Delta t} - \rho^t}{\Delta t} = \psi^* \frac{p_{\mathrm{rgh}}^* - p_{\mathrm{rgh}}^t}{\Delta t} \tag{6.261}$$

这种处理方法在代码中的体现为压力方程存在下述一项：

```
fvm::ddt(psi, p)
```

其还可以处理为：

$$\frac{\rho^{t+\Delta t} - \rho^t}{\Delta t} = \psi^* \frac{p_{\mathrm{rgh}}^* - p_{\mathrm{rgh}}^t}{\Delta t} + \frac{\varrho^* - \rho^t}{\Delta t} \tag{6.262}$$

这种处理方法在代码中的体现为压力方程存在下述一项：

```
fvc::ddt(rho) + psi*correction(fvm::ddt(p))
```

在老版本，比如 OpenFOAM-2.3 版本中，大量的使用`ddt(psi, p)`代码书写方式。在 OpenFOAM-12 版本中，仅有一个被弃用的 PDRFoam 求解器在使用这种`ddt(psi, p)`代码书写方法。这是因为对于`correction`这种书写方法，如果不更新密度 ϱ^* 的话（采用 hePsiThermo 热物理模型库，该模型在瞬态内循环不更新密度的值，仅仅更新可压缩性），则

[88] 参考 CFD 的可压瞬态算法：http://dyfluid.com/compiso.html

其退化为方程(6.261)，如果采用 heRhoThermo 热物理模型库，方程(6.262)。很明显第二种代码书写方式具有更大的普适性。

6.13.12 bound()

bound 函数主要用于处理越界问题。其可以用于限定某个场的最小值（不能用于限定场的最大值）。在 OpenFOAM 的湍流模型求解时，经常会提示：

```
bounding k.water, min: -3.56016e-05 max: 0.09656 average: 0.000370
```

上述代码提示 water 相的湍流动能 k 的最小值为-3.56016e-05，最大值为 0.09656，平均值为 0.000370。依据湍流动能的定义，其为一个不为负值的函数。因此在出现负的湍流动能的情况下需要进行处理。bound 函数通过一种平均的操作对场进行操作使其有界。对于图6.33中的 3 个网格单元，其中间的网格单元出现负值，在进行 bound 操作的时候，首先从网格体心对网格面心的值进行线性插值操作求得面心的 k 值，然后通过下述公式三个面 S1，S2，S3 计算面平均值：

$$k_{vol} = \frac{S_1 k_{1,surf} + S_2 k_{2,surf} + S_3 k_{3,surf}}{S_1 + S_2 + S_3} \tag{6.263}$$

其中 $k_{i,surf}$ 是第 i 个网格面的 k 值，S_i 表示第 i 个网格面。

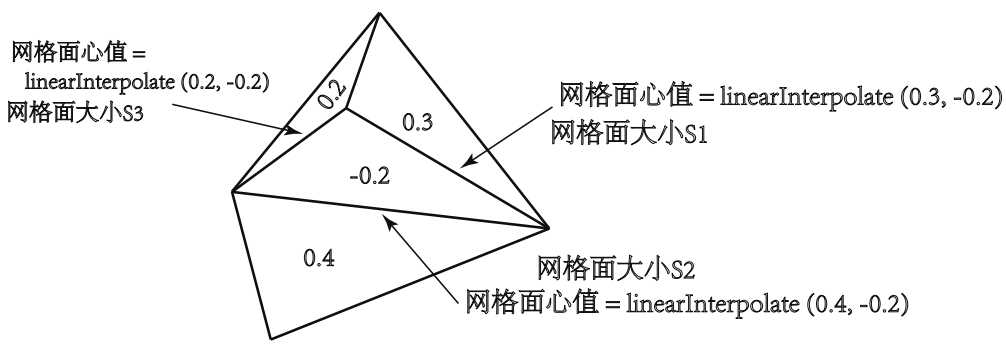


图 6.33: bound 函数示意图。

6.13.13 adjustPhi()

OpenFOAM 中的`adjustPhi`函数主要用于那些不存在压力固定值边界条件的算例。这是因为压力方程求解的实际就是连续性方程。连续性方程的根本是质量守恒。如果质量不守恒，则压力方程的求解必然会出现问题。因此在求解压力方程之前，要保证质量守恒。另一方面，对于计算域内所有的进出口，必然存在通量（除非速度为 0）。这些流入的通量必然要等于流出的通量，否则结果必然不符合物理。`adjustPhi`函数将边界条件分为三种类型：固定进口 ϕ_{in}、固定出口 ϕ_{out} 以及可调整的出口 ϕ_{adjust}。ϕ_{in} 即速度固定的流入的边界，ϕ_{out} 即速度固定的流出的边界，ϕ_{adjust} 即为速度可变的边界。`adjustPhi`函数对所有边界进

行遍历，求出所有的 ϕ_{in}，ϕ_{out}，以及 ϕ_{adjust}。adjustPhi函数输入 3 个参数，分别是通量、速度以及压力。其中压力用来判断是否需要指定压力参考值，仅此而已。速度用来判断是否是固定值速度边界条件，仅此而已。主要是传入的通量。

在执行的过程中，如果计算过程中 $\phi_{adjust} \neq 0$，表示存在可以调整的通量，则定义一个质量修正因子 Corr，其值为：

$$\mathrm{Corr} = \frac{|\phi_{in}| - |\phi_{out}|}{|\phi_{adjust}|} \tag{6.264}$$

同时对于可调整的边界，将其与质量修正因子相乘：

$$\phi_{adjust}^{new} = \mathrm{Corr} \cdot \phi_{adjust} \tag{6.265}$$

在这种情况下，有：

$$|\phi_{adjust}^{new}| = |\phi_{in}| - |\phi_{out}| \tag{6.266}$$

即可保证守恒。如果计算过程中 $\phi_{adjust} = 0$，则判断 $|\phi_{in}| - |\phi_{out}|$ 的大小。如果在机器精度以下，说明边界条件是守恒的。如果其值大于机器精度，则说明进出口通量不守恒，程序终止并抛出下列错误：

```
--> FOAM FATAL ERROR:
Continuity error cannot be removed by adjusting the outflow.
Please check the velocity boundary conditions and/or run potentialFoam
   to initialise the outflow.
```

在实际使用中，如果边界条件设置正确，通常不会抛出该错误。

6.13.14 pressureReference() 或 setReference()

pressureReference() 函数或老版本的 setReference() 函数在 OpenFOAM 中被大量的应用于求解压力方程中。这是因为在某些条件下（比如封闭的空腔中），压力方程不需要给定固定值边界条件，即所有边界条件都为固定梯度边界条件。这会导致构建的矩阵系统出现相容性问题。从数学上来讲，对于方程 $-\nabla \cdot (\nabla p) = 0$，其一定不能给定固定梯度边界条件。但若其存在其他项如时间项，如 $\frac{\partial p}{\partial t} - \nabla \cdot (\nabla p) = 0$（在这里我们假定量纲是一致的）则没有问题。

OpenFOAM 一旦存在类似的设置错误，会提示：

```
[4] --> FOAM FATAL IO ERROR:
[4] Unable to set reference cell for field p
    Please supply either pRefCell or pRefPoint
[4]
[4]
[4] file: IOstream.PIMPLE
```

现以一个横向二维网格为例：|0|1|2|，其网格编号从左至右为 0、1、2，同时左右分别给定零法向梯度边界条件，相对应的压力方程为

$$-\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right)=S \tag{6.267}$$

对其在 |0|1|2| 网格系统上进行离散之后，形成的方程为[89]：

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}\cdot\begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix}=\begin{pmatrix} S\Delta x^2 \\ S\Delta x^2 \\ S\Delta x^2 \end{pmatrix} \tag{6.268}$$

可以看出，方程(6.268)中的矩阵的秩为 2，这会导致 p 存在无穷多个解。也即方程奇异。但若对于附加时间项的压力方程：

$$\frac{\partial p}{\partial t}-\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial x}\right)=S \tag{6.269}$$

时间项的离散会改变离散矩阵，如：

$$\begin{pmatrix} 1+1/\Delta t & -1 & 0 \\ -1 & 2+1/\Delta t & -1 \\ 0 & -1 & 1+1/\Delta t \end{pmatrix}\cdot\begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix}=\begin{pmatrix} S\Delta x^2 \\ S\Delta x^2 \\ S\Delta x^2 \end{pmatrix} \tag{6.270}$$

其矩阵的阶数为 3，可以求解。

在 OpenFOAM 中，通过 setReference() 函数可以将某个网格单元施加一个参考量，并且使得方程增阶，进一步获得解。相应的代码如下[90]：

```
    source()[cell] += diag()[cell]*value;
    diag()[cell] += diag()[cell];
```

在实施的过程中，假定要对网格单元 2 给定某参考值 0，可以看出，方程(6.268)的源项不需要变化。但是方程(6.268)的矩阵最中间的元素变成了-4，即：

$$\begin{pmatrix} -1 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & -1 \end{pmatrix}\cdot\begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix}=\begin{pmatrix} S\Delta x^2 \\ S\Delta x^2 \\ S\Delta x^2 \end{pmatrix} \tag{6.271}$$

在这种情况下，方程(6.268)有解，且基于 0 上下浮动。本质上，针对离散的方程

$$a_P\psi_P+\sum a_N\psi_N=S \tag{6.272}$$

[89]注意其中矩阵对角线元素为负值，这是因为方程(6.267)左侧的拉普拉斯项为正值，其等于一种负的扩散项（扩散项本身为负）。

[90]fvMatrix.C

我们要假定某个网格单元的值在某个值上下浮动，在无任何因素影响的情况下，可以假定 $a_P\psi_P = a_P\psi_{ref}$，将其加到方程(6.273)中有：

$$a_P(\psi_P + \psi_P) + \sum a_N\psi_N = S + a_P\psi_{ref} \tag{6.273}$$

整理有：

$$(a_P + a_P)\psi_P + \sum a_N\psi_N = S + a_P\psi_{ref} \tag{6.274}$$

其与代码的操作是相符的。从方程(6.271)中还可以看出，其增加了矩阵的对角占优特性。

这一问题也可以这样来理解。如图6.34所示，对于一个变量，左右计算域边界均给定固定梯度边界条件的话，会存在无穷多个解。因此必须要固定一个参考值才可以。

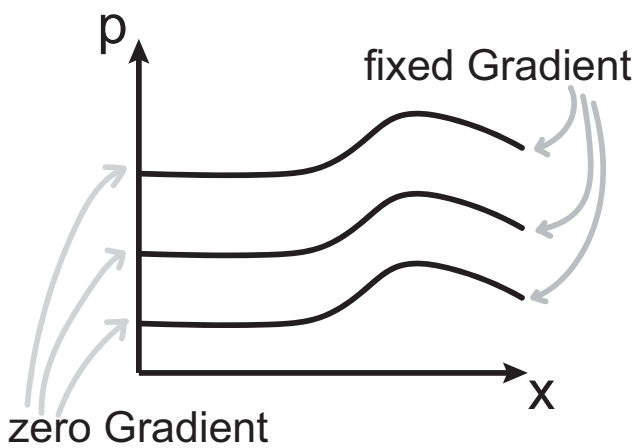


图 6.34: 变量相容性问题导致的无解示意图。

6.13.15 constrainHbyA()

constrainHbyA() 函数主要用来处理 **HbyA** 的边界条件。在速度压力耦合算法中，**HbyA** 理论上应该与速度的边界条件相同。因此 constrainHbyA() 在非特殊情况下，手动把用户给定的速度边界条件赋值给 **HbyA**。该代码数学意义非常简单，因此也不需要方程进行解释。相关代码如下[91]：

```
forAll(HbyAbf, patchi)
{
    if
    (
        !U.boundaryField()[patchi].assignable()
    && !isA<fixedFluxExtrapolatedPressureFvPatchScalarField>
        (
```

[91] constrainHbyA.C

```
            p.boundaryField()[patchi]
        )
    )
    {
        HbyAbf[patchi] = U.boundaryField()[patchi];
    }
}
```

6.13.16 continuityErrs 与 compressibleContinuityErrs

这两个分别代表不可压缩流的连续性误差以及可压缩流的连续性误差。不可压缩流的连续性误差计算非常简单，OpenFOAM 定义了sumLocalContErr和globalContErr，前者的计算公式为：

$$Err_1 = \Delta t \frac{\sum_i V_i |\sum_f \phi_f|}{\sum_i V_i} \tag{6.275}$$

sumLocalContErr在每一个时间步都会更新这个误差。globalContErr为一个累积的变量，计算公式为

$$Err_2 += \Delta t \frac{\sum_i (V_i (\sum_f \phi_f))}{\sum_i V_i} \tag{6.276}$$

可压缩流动与不可压缩不同。但其同样定了sumLocalContErr和globalContErr，前者的计算公式为：

$$Err_1 = \frac{\sum_i (\rho_i - \varrho_i) V_i}{\sum_i \rho_i V_i} \tag{6.277}$$

其中 ρ 为通过求解密度方程获得的密度值，ϱ 为通过热物理模型库求得的密度值。globalContErr的计算公式更加简单，即为sumLocalContErr的加和。

6.13.17 limitPressure 与 limitTemperature

limitPressure与limitTemperature可以用于强制限定压力预计温度为合理值。下述为一个典型的用法：

```
limitp
{
    type        limitPressure;

    minFactor   0.1;
    maxFactor   2;
}

limitT
```

```
{
    type        limitTemperature;

    min         101;
    max         1000;

    selectionMode all;
}
```

其数学含义非常简单。在 CFD 计算中某个网格的值会突然越界，但很有可能下一步会恢复正常，且仅仅是某个网格点的异常，因此这不应该直接导致计算停止。limitPressure与limitTemperature强制的将这些网格点的压力值以及温度值处理到正常的区间。比如上述代码将limitTemperature最小值设置为 101，最大值设置为 1000。对于压力，需要将边界条件的固定值或者计算出来的值作为参考当做一个参考压力，然后全场的压力最小值以及压力最大值进行可能得缩放。

6.13.18 fixedValueConstraint

fixedValueConstraint可以用于将变量的值，在某个网格区间限定为特定的值不变。例如下面的代码中，可以将多孔介质区域的湍流动能与湍流动能耗散率设定为固定值。

```
    porosityTurbulence
    {
        type            fixedValueConstraint;

        selectionMode   cellZone;
        cellZone        porosity;

        fieldValues
        {
            k           1;
            epsilon     150;
        }
    }
```

在算法层面，fixedValueConstraint主要采取第6.5.6节讨论的setValue()函数，在此不做详细介绍。

6.13.19 constrainPressure

constrainPressure主要用于更新压力的法向梯度值。constrainPressure在大部分求解器中都存在，但是其在一些求解器中并不是必要的。constrainPressure在使用中需要

结合fixedFluxPressure边界条件。如果算例使用了fixedFluxPressure边界条件，则必须进行constrainPressure，否则会导致程序计算报错。如果算例不需要fixedFluxPressure，则constrainPressure可以删掉。其核心代码如下[92]：

```
    forAll(pBf, patchi)
    {
        if (isA<fixedFluxPressureFvPatchScalarField>(pBf[patchi]))
        {
            refCast<fixedFluxPressureFvPatchScalarField>
            (
                pBf[patchi]
            ).updateCoeffs
            (
                (
                    phiHbyABf[patchi]
                  - rho.boundaryField()[patchi]
                   *MRF.relative(SfBf[patchi] & UBf[patchi], patchi)
                )
               /(magSfBf[patchi]*rhorAUBf[patchi])
            );
        }
    }
```

其计算公式即为方程(6.234)。

6.13.20 correctPhi

OpenFOAM 在网格发生变化后，例如在进行网格自适应之后，需要将原来的场进行插值到新网格。在这种情况下可能会发生通量的不一致。如图6.35所示，图中在插值后，虚线出的新的网格面的插值后的通量，导致 $\sum \mathbf{U}\cdot\mathbf{S}_f \neq 0$。correctPhi函数的意义，是通过数值计算，初步的将 $\sum \mathbf{U}\cdot\mathbf{S}_f = 0$。后续还需要通过速度压力耦合算法，进一步的更新速度以及压力。

回到速度压力耦合算法，在最终，连续性方程的离散形式可以整理为：

$$\sum \mathbf{U}_f^*\cdot\mathbf{S}_f = \sum \frac{1}{A_{\mathrm{P},f}^t}\left(\frac{1}{V_\mathrm{P}}\sum p_f^*\mathbf{S}_f\right)_f\cdot\mathbf{S}_f \tag{6.278}$$

该方程求解压力 p^* 一次后，就一定有满足连续性方程的通量：

$$\phi_f = \left(\mathbf{U}_f^* - \frac{1}{A_{\mathrm{P},f}^t}\left(\frac{1}{V_\mathrm{P}}\sum p_f^*\mathbf{S}_f\right)_f\right)\cdot\mathbf{S}_f \tag{6.279}$$

[92] constrainPressure.C

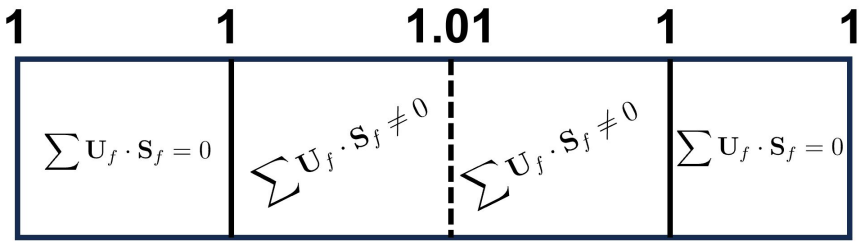


图 6.35: 网格自适应之后导致的通量不守恒。中间的虚线为新增的网格面。

所以可以看出，方程(6.278)中的 $\mathbf{U}^*$ 开始并不满足通量守恒，但经过方程(6.278)以及(6.279)两步处理之后，可以获得守恒的 ϕ_f。`correctPhi`函数的精髓就是求解方程(6.278)以及(6.279)。假定经过网格插值之后，$\mathbf{U}^*$ 在某一个网格面发生变化导致不守恒（如图6.35）。这导致周围的两个网格的 $\sum \mathbf{U} \cdot \mathbf{S}_f \neq 0$。代入到方程(6.278)中，其实等于是其中的源项。在这种情况下，直接感觉这两个网格求得的 $p \neq 0$[93]。其他网格中由于 $\sum \mathbf{U} \cdot \mathbf{S}_f = 0$。因此 p 在这些网格中也是 0。但实际并非如此，参考第A.1节的讨论，p 在所有的网格上都存在解。在获得 p 的解之后，方程(6.279)的修正可以获得守恒的 ϕ_f 场。

6.14 OpenFOAM 中的湍流模型

6.14.1 RANS 湍流模型

线性-可压缩-多相 kEpsilon 目前应用最广泛的模型就是 kEpsilon 湍流模型。最早的工作可以追溯到 1945 年抗日战争时期诞生于国立清华大学 [35]。依据 k 的数学定义，可以得到湍流动能的准确的传输方程，该方程右侧存在湍流动能耗散项或者湍流频率项。在 kEpsilon 模型中，k 方程右侧调用湍流动能耗散。在湍流动能耗散已知的情况下，k 方程可解。依据 ε 的定义，可以得到 epsilon 的准确的传输方程，若进一步调用湍流频率 ω 与 ε 的关系，还可以推导出 kOmega 模型。但 epsilon 的准确的传输方程并不封闭。因此 epsilon 方程需要进行大量的模化，并由此产生大量的 kEpsilon 湍流模型变种。OpenFOAM 中的 kEpsilon 模型为 Launder and Spalding 提出的 kEpsilon 湍流模型 [87]。其也被称之为标准 kEpsilon 模型。kEpsilon 模型在求解过程中非常稳健因此被广泛应用于各种普适性流动中。kEpsilon 湍流模型最大的缺点是对于存在逆压梯度的流动预测不够准确。这主要是由于 kEpsilon 大部分情况下会低估湍流动能耗散率，也就是预测的湍流尺度过高。kEpsilon 模型为一种高雷诺数模型，仅仅适用于高雷诺数流动。在应用于低雷诺数流动的情况下，需要在壁面附加低雷诺数修正模型。kEpsilon 模型中的 C_μ 为常数，很多工作认为使用一个变量来模化

[93]在代码中，一般被称之为 pcorr。

C_μ 可以获得更好的结果。

$$\frac{\partial \alpha\rho\varepsilon}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}\varepsilon)-\nabla\cdot(\alpha\rho D_\varepsilon\nabla\varepsilon)=\alpha\rho C_1 G\frac{\varepsilon}{k}-\alpha\rho\left(\frac{2}{3}C_1-C_3\right)(\nabla\cdot\mathbf{U})\varepsilon-\alpha\rho C_2\frac{\varepsilon}{k}\varepsilon+S_\varepsilon$$

$$\frac{\partial \alpha\rho k}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}k)-\nabla\cdot(\alpha\rho D_k\nabla k)=\alpha\rho G-\alpha\rho\frac{2}{3}(\nabla\cdot\mathbf{U})k-\alpha\rho\frac{\varepsilon}{k}k+S_k$$

$$G=\nu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right):\nabla\mathbf{U},\nu_t=C_\mu\frac{k^2}{\varepsilon},D_k=\frac{\nu_t}{\sigma_k}+\nu,D_\varepsilon=\frac{\nu_t}{\sigma_\varepsilon}+\nu$$

$$\sigma_k=1,\sigma_\varepsilon=1.3,C_\mu=0.09,C_1=1.44,C_2=1.92,C_3=0$$

kEpsilon 模型的边界条件通常这样选择：

- k : 进口固定值，可以通过公式 $k=\frac{1}{2}(u'2+v'^2+w'^2)$ 来计算，其中脉动速度可以自行预估。在比较强的湍流场中取平均速度的 0.15-0.2 左右。在中性湍流场取 0.1-0.15 左右。在比较小的湍流场取 0.5 左右。出口的 k 取零法向梯度。壁面采取壁面函数。

- ε : 进口固定值，可以通过公式 $\varepsilon=\frac{C_{mu}^{0.75}k^{1.5}}{L}$ 来估算，其中 $C_{mu}=0.09$，L 为特征长度。L 的取值没有固定标准，一些比较简单的算例，比如管道、槽道之类的可以取管道的直径，但对于复杂几何来说 L 的取值是一门玄学。实际操作过程中很大情况就是随便给，多次尝试，重要的是保证收敛。出口的 ε 取零法向梯度。壁面采取壁面函数。因为 ε 会出现在湍流粘度计算的分母中，ε 在计算中以及边界条件的设置中永远不能为 0，但可以为一个比较小的数。

- ν_t : 除了壁面之外，均从湍流变量计算而来，因此为 calculated 边界条件。在壁面处，需要使用壁面函数。

线性-可压缩-多相 realizableKE realizableKE 湍流模型认为标准的 kEpsilon 湍流模型可能会产生负的正应力，这是不可能的（在 CFD 领域通常被称之为是不可实现的）。realizableKE 湍流模型针对标准湍流模型最重要的两个区别在于 1）C_μ 不再是一个常数而是一个变量，2）湍流粘度模型基于可实现性理论，可确保法向雷诺应力为正。同样，realizableKE 湍流模型的 k 方程并没有变化，主要变动的是 epsilon 方程。从 realizableKE 湍流模型提出的原始文章数据中可以发现 [145]，realizableKE 湍流模型对旋转流动的预测提高较为明显，对于自由剪切、边界层流动、以及后向台阶流和标准 kEpsilon 模型近似相同。

$$\frac{\partial \alpha\rho\varepsilon}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}\varepsilon)-\nabla\cdot(\alpha\rho D_\varepsilon\nabla\varepsilon)=\alpha\rho C_1\sqrt{S_2}\varepsilon-\frac{\alpha\rho C_2\varepsilon}{k+\sqrt{\nu\varepsilon}}\varepsilon+S_\varepsilon$$

$$\frac{\partial \alpha\rho k}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}k)-\nabla\cdot(\alpha\rho D_k\nabla k)=\alpha\rho G-\alpha\rho\frac{2}{3}(\nabla\cdot\mathbf{U})k-\alpha\rho\frac{\epsilon}{k}k+S_k$$

$$G=\nu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right):\nabla\mathbf{U},\nu_t=C_\mu\frac{k^2}{\varepsilon},D_k=\frac{\nu_t}{\sigma_k}+\nu,D_\varepsilon=\frac{\nu_t}{\sigma_\varepsilon}+\nu$$

$$S_2 = 2\left|\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2} - \frac{1}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right|^2, C_1 = \max\left(\frac{\eta}{5+\eta}, 0.43\right), \eta = \frac{\sqrt{S_2}k}{\varepsilon}, C_\mu = \frac{1}{A_0 + \frac{A_s U_s k}{\varepsilon}}$$

$$A_s = \sqrt{6}\cos(\phi_s), \phi_s = \frac{1}{3}\mathrm{acos}\left(\min\left(\max\left(\sqrt{6}W, -1\right), 1\right)\right), U_s = \left(\frac{S_2}{2} + \left|\frac{\nabla\mathbf{U}-\nabla\mathbf{U}^T}{2}\right|^2\right)^{\frac{1}{2}}$$

$$A_0 = 4, C_2 = 1.9, \sigma_k = 1, \sigma_\varepsilon = 1.2$$

RNGkEpsilon 模型与 kEpsilon 模型的边界条件设置原则相同。

线性-可压缩-多相 RNGkEpsilon RNGkEpsilon 湍流模型采用重整化 NS 方程进行模化，意图包含小尺度湍流的影响。OpenFOAM 中植入的 RNGkEpsilon 湍流模型来自于 Yakhot 的工作 [193]。在 RNGkEpsilon 被提出时，强调此湍流模型的数学连贯性，批判 kEpsilon 的验证算例不充分，认为 RNGkEpsilon 是一个更普适性的湍流模型。经验表明，RNGkEpsilon 在一些算例中确实对预测结果略有改善 [156]，但求解略微刚性。RNGkEpsilon 和 kEpsilon 模型的方程形式完全一样，只不过模型系数是经过理论分析而来。需要注意的是，目前一些商软中植入了提高版本的更适用于旋转流动的 RNGkEpsilon 湍流模型。在此需要跟 OpenFOAM 的原始版本进行区分（区别主要在于湍流粘度项）。

$$\frac{\partial\alpha\rho\varepsilon}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}\varepsilon) - \nabla\cdot(\alpha\rho D_\varepsilon\nabla\varepsilon) = \alpha\rho\left(C_1 - R\right)G\frac{\varepsilon}{k} - \alpha\rho\left(\frac{2}{3}C_1 - C_3\right)(\nabla\cdot\mathbf{U})\varepsilon - \alpha\rho C_2\frac{\varepsilon}{k}\varepsilon + S_\varepsilon$$

$$\frac{\partial\alpha\rho k}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}k) - \nabla\cdot(\alpha\rho D_k\nabla k) = \alpha\rho G - \frac{2}{3}\alpha\rho(\nabla\cdot\mathbf{U})k - \alpha\rho\frac{\varepsilon}{k}k + S_k$$

$$G = \nu_t S_2, S_2 = \left(\nabla\mathbf{U} + \nabla\mathbf{U}^T - \frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right) : \nabla\mathbf{U}, \nu_t = C_\mu\frac{k^2}{\epsilon}, D_k = \frac{\nu_t}{\sigma_k} + \nu, D_\varepsilon = \frac{\nu_t}{\sigma_\varepsilon} + \nu$$

$$R = \frac{\eta\left(-\eta/\eta_0 + 1\right)}{\beta\eta^3 + 1}, \eta = \frac{\sqrt{|S_2|}k}{\varepsilon}$$

$$C_\mu = 0.0845, C_1 = 1.42, C_2 = 1.68, \eta_0 = 4.38, C_3 = 0, \sigma_k = 0.71942, \sigma_\varepsilon = 0.71942, \beta = 0.012$$

RNGkEpsilon 模型与 kEpsilon 模型的边界条件设置原则相同。

线性-可压缩-多相-低雷诺数 LaunderSharmaKE 标准 kEpsilon、RNGkEpsilon、以及 realizableKEpsilon 模型最大的缺点就是他们都是高雷诺数湍流模型。在壁面网格 $y^+ > 30$ 的时候，高雷诺数湍流模型需要结合壁面函数来使用。然而如果网格在壁面足够细，例如在 $y^+ < 5$ 的情况下，就可以不结合壁面函数使用高雷诺数湍流模型了么？答案是未必。在这种情况下，高雷诺数湍流模型同样需要附加阻尼函数，即形成低雷诺数湍流模型。LaunderSharmaKE 湍流模型就属于低雷诺数湍流模型。低雷诺数模型跟高雷诺数模型的区别主要在于前者的 C_μ 需要乘以一个系数 f_μ 来体现壁面附近分子粘度的作用。在使用的过程中需要保证 $y^+ \approx 1$ [86]。低雷诺数湍流模型定义湍流雷诺数 Re 为 $k^2/\nu/\varepsilon$，在 kEpsilon 模型的基础上建立与湍流雷诺数的关系式，对湍流动能在壁面附近的产生添加阻尼，以及湍流动能耗散率在壁面附近的消耗进行修正，从而可以应用于低雷诺数领域。低雷诺数湍

流模型同样被称之为附加阻尼函数（damping function）的湍流模型。低雷诺数湍流模型的第一层网格点要布置在粘性支层，y^+ 建议在 1 以下，不建议超过 4。同时建议在 $y^+ < 11.5$ 的区域至少布置 5 个网格点。另外，在这里所提及的低雷诺数模型并不意味该模型只能用于低雷诺数的流动，而是表面在壁面区域，可以通过细化网格来模拟壁面附近的低雷诺数流动。相对于其他高雷诺数模型，即使壁面做网格细化，也并不会预测精准的结果。

$$\frac{\partial \alpha\rho\varepsilon}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}\varepsilon)-\nabla\cdot(\alpha\rho D_\varepsilon\nabla\varepsilon)=\alpha\rho C_1 G\frac{\varepsilon}{k}-\alpha\rho\left(\frac{2}{3}C_1-C_3\right)(\nabla\cdot\mathbf{U})\varepsilon-\alpha\rho C_2 f_2\frac{\varepsilon}{k}\varepsilon+\alpha\rho E+S_\varepsilon$$

$$\frac{\partial \alpha\rho k}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}k)-\nabla\cdot(\alpha\rho D_k\nabla k)=\alpha\rho G-\alpha\rho\frac{2}{3}(\nabla\cdot\mathbf{U})k-\alpha\rho\frac{\varepsilon+D}{k}k+S_k$$

$$G=\nu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T-\frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right):\nabla\mathbf{U},\nu_t=C_\mu f_\mu\frac{k^2}{\varepsilon},D_k=\frac{\nu_t}{\sigma_k}+\nu,D_\varepsilon=\frac{\nu_t}{\sigma_\varepsilon}+\nu$$

$$f_\mu=\exp\left(\frac{-3.4}{\left(1+\frac{k^2}{50\nu\varepsilon}\right)^2}\right),f_2=1-0.3\exp\left(-\min\left(\frac{k^4}{\nu^2\varepsilon^2},50\right)\right)$$

$$D=2\nu\left|\nabla(\sqrt{k})\right|^2,E=2\nu\nu_t\left|\nabla\nabla\mathbf{U}\right|^2$$

$$C_\mu=0.09,C_1=1.44,C_2=1.92,C_3=0,\sigma_k=1,\sigma_\varepsilon=1.3$$

LaunderSharmaKE 模型的边界条件与其他高雷诺数 kEpsilon 模型在进出口处的设置相同，但是在壁面处有较大区别，通常这样选择：

- k：壁面的 k 在网格非常精细的情况下也可以给定一个比较小的固定值如 1×10^{-15}，这是因为在壁面附近通常湍动较小。
- ε：壁面处的 ε 直接取一个非常小的值如 1×10^{-15}。因为 ε 会出现在湍流粘度计算的分母中，ε 在计算中以及边界条件的设置中永远不能为 0，但可以为一个比较小的数。
- ν_t：在壁面处，ν_t 需要指定为低雷诺数边界条件，或采用 calculated 边界条件；
- 低雷诺数模型在使用过程中需要保证壁面处的 $y^+ < 11$。

线性-不可压缩-单相-低雷诺数 LamBremhorstKE LamBremhorstKE 湍流模型也为一种湍流模型，其与 LaunderSharmaKE 湍流模型非常类似，实际上，大部分的低雷诺数湍流模型之间的区别仅仅在于 f_μ, f_2 f_1 函数以及模型参数。LamBremhorstKE 湍流模型的 f_μ 与 Re_t 以及 Re_y 均相关，LaunderSharmaKE 湍流模型的 f_μ 仅仅与 Re_t 相关。

$$\frac{\partial\varepsilon}{\partial t}+\nabla\cdot(\mathbf{U}\varepsilon)-\nabla\cdot(D_\varepsilon\nabla\varepsilon)=C_{\varepsilon_1}f_1G\frac{\varepsilon}{k}-C_{\varepsilon_2}f_2\frac{\varepsilon}{k}\varepsilon$$

$$\frac{\partial k}{\partial t}+\nabla\cdot(\mathbf{U}k)-\nabla\cdot(D_k\nabla k)=G-\frac{\epsilon}{k}k$$

$$G=\nu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T\right):\nabla\mathbf{U},\nu_t=C_\mu f_\mu\frac{k^2}{\varepsilon},D_k=\nu_t+\nu,D_\varepsilon=\frac{\nu_t}{\sigma_\varepsilon}+\nu$$

$$f_\mu=\left(1-\exp\left(-0.0165\frac{\sqrt{k}y}{\nu}\right)\right)^2\left(1+\frac{20.5}{R_t}\right),R_t=\frac{k^2}{\nu\varepsilon},f_1=1+\left(\frac{0.05}{f_\mu}\right)^3,f_2=1-\exp\left(-R_t^2\right)$$

$$C_\mu=0.09,C_{\varepsilon_1}=1.44,C_{\varepsilon_2}=1.92,\sigma_\varepsilon=1.3$$

LamBremhorstKE 模型的边界条件与 LaunderSharmaKE 模型相同。但是实际操作过程中发现，LamBremhorstKE 相对 LaunderSharmaKE 更难收敛，因此要注意湍流变量内部初始场的值，同时也要注意算法的松弛因子相关设置。

线性-不可压缩-单相-低雷诺数 LienLeschziner 前述低雷诺数湍流模型的湍流动能耗散率方程存在一切共同点与区别，LamBremhorstKE 的 ε 传输方程中存在 f_1,f_2 项，LaunderSharmaKE 中存在 f_2,E 项。LienLeschziner 湍流模型则同时考虑了 f_1,f_2,E 项（$f_1=1$）。其中 f_1,E 项主要影响壁面附近湍流变量的峰值大小。f_2 项的作用主要限制在粘性支层内，起到一种阻尼的作用。

$$\frac{\partial\varepsilon}{\partial t}+\nabla\cdot(\mathbf{U}\varepsilon)-\nabla\cdot(D_\varepsilon\nabla\varepsilon)=C_{\varepsilon,1}G\frac{\varepsilon}{k}-C_{\varepsilon,2}f_2\frac{\varepsilon}{k}\varepsilon+E$$

$$\frac{\partial k}{\partial t}+\nabla\cdot(\mathbf{U}k)-\nabla\cdot(D_k\nabla k)=G-\frac{\varepsilon}{k}k$$

$$G=\nu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T\right):\nabla\mathbf{U},\nu_t=C_\mu f_\mu\frac{k^2}{\varepsilon},f_\mu=\frac{1-e^{-A_\mu y^*}}{1-e^{-A_\varepsilon y^*}},f_2=1-0.3e^{-R_t^2},y^*=\frac{\sqrt{k}y}{\nu}$$

$$R_t=\frac{k^2}{\nu\varepsilon},E=C_{\varepsilon,2}C_\mu^{0.75}\left(\frac{f_2\sqrt{k}\varepsilon}{le}\right)e^{-A_E y^{*2}},le=\kappa y\left(1-e^{-A_\varepsilon y^*}\right),D_\varepsilon=\frac{\nu_t}{\sigma_\varepsilon}+\nu,D_k=\frac{\nu_t}{\sigma_k}+\nu$$

$$C_{\varepsilon,1}=1.44,C_{\varepsilon,2}=1.92,\sigma_k=1,\sigma_\varepsilon=1.3,C_\mu=0.09,\kappa=0.41$$

$$A_\mu=0.016,A_\varepsilon=0.263,A_E=0.00222$$

LienLeschziner 模型的边界条件与 LaunderSharmaKE 模型相同。

非线性-不可压缩-单相-高雷诺数 ShihQuadraticKE 前述 kEpsilon 以及变种均调用的各向同性 Boussinesq 假定来进行模化雷诺应力。在充分发展的条件下会导致正应力相同（例如在二维情况下 $\boldsymbol{\tau}_{11}=\boldsymbol{\tau}_{22}=0$），这与实验不符。ShihQuadraticKE 认为这种各向同性假定会在一些湍流引致二次流的情况下带来较大的偏差 [144]。其认为假定雷诺应力与速度梯度的线性关系是不合理的，需要附加非线性修正。因此在 Shih 的模型中，直接对非线性的雷诺应力 $\boldsymbol{\tau}_n$ 进行计算。在计算的过程中，需要获取 k 和 ε 的值，因此需要 k 和 ε 的传输方程。在 OpenFOAM 中，其也被称之为 ShihQuadraticKE。其中的 Quadratic 体现为雷诺应力为速度梯度的 2 次幂关系（存在 $\mathbf{S}\cdot\mathbf{S}$）。需要注意的是 ShihQuadraticKE 并没有对 $\boldsymbol{\tau}_n$ 进行求解，否则将演变为雷诺应力模型（也被称之为二阶矩模型）。ShihQuadraticKE 湍流模型在推导的过程中并没有考虑分子粘度，因此为一个高雷诺数湍流模型，同样可以调用壁面函数。非线性类模型与线性类模型的区别在于雷诺应力项散度的计算，线性类模型这一项可以通过速度计算而来：$\nabla\cdot\boldsymbol{\tau}=\nabla\cdot(\mu_t\nabla\mathbf{U})+\nabla\cdot(\mu_t\mathrm{dev2}(\nabla\mathbf{U}^T))$，非线性模型除

了要考虑线性贡献，还需要考虑非线性的贡献：$\nabla\cdot\boldsymbol{\tau}+\nabla\cdot\boldsymbol{\tau}_n$。非线性湍流模型也被称之为各向异性湍流模型。

$$\frac{\partial\varepsilon}{\partial t}+\nabla\cdot(\mathbf{U}\varepsilon)-\nabla\cdot(D_\varepsilon\nabla\varepsilon)=C_{\varepsilon,1}G\frac{\varepsilon}{k}-C_{\varepsilon,2}\frac{\varepsilon}{k}\varepsilon$$

$$\frac{\partial k}{\partial t}+\nabla\cdot(\mathbf{U}k)-\nabla\cdot(D_k\nabla k)=G-\frac{\varepsilon}{k}k$$

$$G=\left(\nu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T\right)-\boldsymbol{\tau}_n\right):\nabla\mathbf{U},\nu_t=C_\mu\frac{k^2}{\varepsilon},D_k=\frac{\nu_t}{\sigma_k}+\nu,D_\varepsilon=\frac{\nu_t}{\sigma_\varepsilon}+\nu$$

$$\mathbf{S}=\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2},\mathbf{W}=\frac{\nabla\mathbf{U}-\nabla\mathbf{U}^T}{2},\bar{s}=\frac{k}{\varepsilon}\sqrt{2}|\mathbf{S}|,\bar{w}=\frac{k}{\varepsilon}\sqrt{2}|\mathbf{W}|$$

$$\begin{aligned}\boldsymbol{\tau}_n=\frac{k^3}{\varepsilon^2(C_\beta+\bar{s}^3)}\Bigg(&C_{\beta_1}\left(\mathbf{S}\cdot\mathbf{S}-\frac{1}{3}\mathrm{tr}(\mathbf{S}\cdot\mathbf{S})\mathbf{I}\right)\\&+C_{\beta_2}\left(\mathbf{S}\cdot\mathbf{W}+(\mathbf{S}\cdot\mathbf{W})^T\right)+C_{\beta_3}\left(\frac{\mathbf{W}\cdot\mathbf{W}+(\mathbf{W}\cdot\mathbf{W})^T}{2}-\frac{1}{3}\mathrm{tr}(\mathbf{W}\cdot\mathbf{W})\mathbf{I}\right)\Bigg)\end{aligned}$$

$$C_\mu=\frac{2/3}{C_{\mu_1}+\bar{s}+C_{\mu_2}\bar{w}},C_{\varepsilon,1}=1.44,C_{\varepsilon,2}=1.92,\sigma_k=1$$

$$\sigma_\varepsilon=1.3,C_{\mu_1}=1.25,C_{\mu_2}=0.9,C_\beta=1000,C_{\beta_1}=3,C_{\beta_2}=15,C_{\beta_3}=-19$$

ShihQuadraticKE 模型的边界条件与 kEpsilon 模型相同。

非线性-不可压缩-单相-低雷诺数 LienCubicKE lienCubicKE 模型与前述 ShinQuadraticKE 模型大同小异，均为非线性湍流模型。但 lienCubicKE 模型附加了低雷诺数修正。其中的 Cubic 体现在雷诺应力与速度梯度存在 3 次幂关系（存在 $(\mathbf{S}\cdot\mathbf{S})\mathbf{W}$）。

$$\frac{\partial\varepsilon}{\partial t}+\nabla\cdot(\mathbf{U}\varepsilon)-\nabla\cdot(D_\varepsilon\nabla\varepsilon)=C_{\varepsilon,1}G\frac{\varepsilon}{k}-C_{\varepsilon,2}f_2\frac{\varepsilon}{k}+E$$

$$\frac{\partial k}{\partial t}+\nabla\cdot(\mathbf{U}k)-\nabla\cdot(D_k\nabla k)=G-\frac{\varepsilon}{k}k$$

$$G=\left(\nu_t\left(\nabla\mathbf{U}+\nabla\mathbf{U}^T\right)-\boldsymbol{\tau}_n\right):\nabla\mathbf{U},\nu_t=C_\mu f_\mu\frac{k^2}{\varepsilon},D_k=\frac{\nu_t}{\sigma_k}+\nu,D_\varepsilon=\frac{\nu_t}{\sigma_\varepsilon}+\nu$$

$$\mathbf{S}=\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2},\mathbf{W}=\frac{\nabla\mathbf{U}-\nabla\mathbf{U}^T}{2},\bar{s}=\frac{k}{\varepsilon}\sqrt{2}|\mathbf{S}|,\bar{w}=\frac{k}{\varepsilon}\sqrt{2}|\mathbf{W}|$$

$$\begin{aligned}\boldsymbol{\tau}_n=\frac{k^3}{\varepsilon^2(C_\beta+\bar{s}^3)}\Bigg(&C_{\beta_1}\left(\mathbf{S}\cdot\mathbf{S}-\frac{1}{3}\mathrm{tr}(\mathbf{S}\cdot\mathbf{S})\mathbf{I}\right)\\&+C_{\beta_2}\left(\mathbf{S}\cdot\mathbf{W}+(\mathbf{S}\cdot\mathbf{W})^T\right)+C_{\beta_3}\left(\frac{\mathbf{W}\cdot\mathbf{W}+(\mathbf{W}\cdot\mathbf{W})^T}{2}-\frac{1}{3}\mathrm{tr}(\mathbf{W}\cdot\mathbf{W})\mathbf{I}\right)\Bigg)\\&-f_\mu k\left(\frac{C_\mu k}{\varepsilon}\right)^3\left(C_{\gamma,1}|\mathbf{S}|^2-C_{\gamma,2}|\mathbf{W}|^2\right)\mathbf{S}\\&-f_\mu k\left(\frac{C_\mu k}{\varepsilon}\right)^3C_{\gamma,4}\left((\mathbf{S}\cdot\mathbf{S})\cdot\mathbf{W}+((\mathbf{S}\cdot\mathbf{S})\cdot\mathbf{W})^T\right)\end{aligned}$$

$$f_\mu = \left(1 - e^{-A_\mu y^*}\right)\left(1 + \frac{2\kappa}{C_\mu^{0.75} y^*}\right), f_2 = 1 - 0.3e^{-R_t^2}, R_t = \frac{k^2}{\nu\varepsilon}$$

$$E = C_{\varepsilon,2} C_\mu^{0.75}\left(\frac{f_2\sqrt{k}\varepsilon}{le}\right) e^{-A_E y^{*2}}, y^* = \frac{\sqrt{k}y}{\nu}, C_\mu = \frac{2/3}{C_{\mu_1} + \bar{s} + C_{\mu_2}\bar{w}}$$

$$le = \frac{\kappa y}{1 + \frac{2\kappa}{C_\mu^{0.75} y^*}}, C_{\varepsilon,1} = 1.44, C_{\varepsilon,2} = 1.92, \sigma_k = 1, \sigma_\varepsilon = 1.3$$

$$C_{\mu_1} = 1.25, C_{\mu_2} = 0.9, C_\beta = 1000, C_{\beta_1} = 3, C_{\beta_2} = 15, C_{\beta_3} = -19, C_{\gamma,1} = 16, C_{\gamma,2} = 16$$

$$C_{\gamma,4} = -80, C_\mu = 0.09, \kappa = 0.41, A_\nu = 0.0198, A_E = 0.00375$$

LienCubicKE 模型的边界条件与其他低雷诺数 kEpsilon 模型相同。

线性-可压缩-多相 BuoyantKE kEpsilon 湍流模型是基于常密度流动而开发。对于一些密度差异比较大的流动，如燃烧、暖通等领域，密度变化会对湍流结构产生影响。因此，传统 kEpsilon 模型需要考虑浮升力的作用。因此，BuoyantKE 湍流模型在 kEpsilon 模型上添加 k, ε 方程源项：

$$S_k = -G_c k, S_\varepsilon = -C_1 \tanh(|v|/u) G_c \varepsilon$$

$$v = \frac{\mathbf{g}}{|\mathbf{g}|} \cdot \mathbf{U}, u = \left|\mathbf{U} - \frac{\mathbf{g}}{|\mathbf{g}|} v\right|, G_c = C_g C_\mu \alpha k (\mathbf{g} \cdot \nabla\rho)/\varepsilon, C_g = 1$$

BuoyantKE 模型的边界条件与其他低雷诺数 kEpsilon 模型相同。

可压缩-多相-雷诺应力 LRR LRR 是一个雷诺应力传输模型 [85]，也即二阶矩模型。同时为一个高雷诺数湍流模型，壁面处可以附加壁面函数求解。由于雷诺应力具有 6 个分量，因此雷诺应力湍流模型至少需要求解六个传输方程，若需要包含传热，可能需要求解更多的方程。传统的 kEpsilon 类方法在应力场变化比较复杂，尤其是那些旋转较为明显，或者包含体积力以及二次流的情况下预测偏差较大。在使用雷诺应力类模型的时候，需要给定雷诺应力边界条件以及初始条件。一般情况下，进口的雷诺应力可以通过湍流强度以及特征长度（比如管径）来获得，如 [178]：$\boldsymbol{\tau}_{11} = k$, $\boldsymbol{\tau}_{22} = 0.5k$, $\boldsymbol{\tau}_{33} = 0.5k$, $\boldsymbol{\tau}_{12} = \boldsymbol{\tau}_{21} = \boldsymbol{\tau}_{13} = 0$。或 $\boldsymbol{\tau}_{11} = \boldsymbol{\tau}_{22} = \boldsymbol{\tau}_{33} = \frac{2}{3}k$。历史上，雷诺应力传输模型并没有广为使用。其主要原因是相对于两方程模型，雷诺应力传输模型 1）需要求解六个传输方程，更加不稳定以及消耗更多的计算资源，2）在一些工况下可以对预测结果具有较大的改善，但在一些情况下反而较差。3）雷诺应力模型中的源项很多都需要进行模化，尤其是其中的压力应力再分配项，同时，很多项的物理机制尚不明确。Peter Bradshaw 曾这样评价雷诺应力传输模型 [8]

> *"RSM is the most disappointing thing for the modelling community because we put so much hope into it, but tends out to be not as successful."*

$$\frac{\partial \alpha\rho\varepsilon}{\partial t} + \nabla \cdot (\alpha\rho\mathbf{U}\varepsilon) - \nabla \cdot (\alpha\rho\mathbf{D}_{\varepsilon,eff} \cdot \nabla\varepsilon) = C_{\varepsilon,1}\alpha\rho G\frac{\varepsilon}{k} - C_{\varepsilon,2}\alpha\rho\frac{\varepsilon}{k}\varepsilon$$

$$\frac{\partial \alpha\rho\mathbf{R}}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}\mathbf{R})-\nabla\cdot(\alpha\rho\mathbf{D}_{\mathbf{R},eff}\cdot\nabla\mathbf{R})+C_1\alpha\rho\frac{\varepsilon}{k}\mathbf{R}=$$
$$\alpha\rho\mathbf{P}-\left(\frac{2}{3}\left(1-C_1\right)\mathbf{I}\right)\alpha\rho\varepsilon-C_2\alpha\rho\left(\mathbf{P}-\frac{1}{3}\mathrm{tr}(\mathbf{P})\mathbf{I}\right)$$

$$\mathbf{P}=-\mathbf{R}\cdot(\nabla\mathbf{U}+\nabla\mathbf{U}^T),k=0.5\left(\mathbf{R}_{xx}+\mathbf{R}_{yy}+\mathbf{R}_{zz}\right),G=0.5|\mathbf{P}_{xx}+\mathbf{P}_{yy}+\mathbf{P}_{zz}|$$

$$\mathbf{D}_{\varepsilon,eff}=C_\varepsilon\frac{k}{\varepsilon}\mathbf{R}+\nu\mathbf{I},\mathbf{D}_{\mathbf{R},eff}=C_s\frac{k}{\varepsilon}\mathbf{R}+\nu\mathbf{I}$$

$$C_\mu=0.09,C_1=1.8,C_2=0.6,C_{\varepsilon,1}=1.44,C_{\varepsilon,2}=1.92,C_s=0.25,C_\varepsilon=0.15,\kappa=0.41$$

$$C_{r1}=0.5,C_{r2}=0.3$$

可压缩-多相-雷诺应力 **SSG**

$$\frac{\partial \alpha\rho\varepsilon}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}\varepsilon)-\nabla\cdot(\alpha\rho\mathbf{D}_{\varepsilon,eff}\cdot\nabla\varepsilon)=C_{\varepsilon,1}\alpha\rho G\frac{\varepsilon}{k}-C_{\varepsilon,2}\alpha\rho\frac{\varepsilon}{k}\varepsilon$$

$$\frac{\partial \alpha\rho\mathbf{R}}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}\mathbf{R})-\nabla\cdot(\alpha\rho\mathbf{D}_{\mathbf{R},eff}\cdot\nabla\mathbf{R})+\left(\frac{C_1\varepsilon+C_{1,s}G}{2}\frac{\alpha\rho}{k}\right)\mathbf{R}=$$
$$\alpha\rho\mathbf{P}-\left(\left((2-C_1)\varepsilon-C_{1,s}G\right)\alpha\rho\right)\frac{1}{3}\mathbf{I}+C_2\alpha\rho\varepsilon\left(\mathbf{b}\cdot\mathbf{b}-\frac{1}{3}\mathrm{tr}(\mathbf{b}\cdot\mathbf{b})\mathbf{I}\right)+$$
$$\alpha\rho k\left(C_3-C_{3,s}|\mathbf{b}|\right)\left(\mathbf{S}-\frac{1}{3}\mathrm{tr}(\mathbf{S})\mathbf{I}\right)+$$
$$C_4\alpha\rho k\left(\mathbf{b}\cdot\mathbf{S}+(\mathbf{b}\cdot\mathbf{S})^T-\frac{2}{3}\mathrm{tr}(\mathbf{b}\cdot\mathbf{S})\mathbf{I}\right)+C_5\alpha\rho k\left(\mathbf{b}\cdot\mathbf{\Omega}+(\mathbf{b}\cdot\mathbf{\Omega})^T\right)$$

$$\mathbf{P}=-\mathbf{R}\cdot(\nabla\mathbf{U}+\nabla\mathbf{U}^T),k=0.5\left(\mathbf{R}_{xx}+\mathbf{R}_{yy}+\mathbf{R}_{zz}\right),\mathbf{\Omega}=\frac{\nabla\mathbf{U}-\nabla\mathbf{U}^T}{2},\mathbf{S}=\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2}$$

$$\mathbf{b}=\frac{\mathbf{R}-\frac{1}{3}\mathrm{tr}(\mathbf{R})\mathbf{I}}{2k},G=0.5|\mathbf{P}_{xx}+\mathbf{P}_{yy}+\mathbf{P}_{zz}|,\mathbf{D}_{\varepsilon,eff}=C_\varepsilon\frac{k}{\varepsilon}\mathbf{R}+\nu\mathbf{I},\mathbf{D}_{\mathbf{R},eff}=C_s\frac{k}{\varepsilon}\mathbf{R}+\nu\mathbf{I},\nu_t=C_\mu\frac{k^2}{\varepsilon}$$

$$C_\mu=0.09,C_1=3.4,C_{1,s}=1.8,C_2=4.2,C_3=0.8,C_{3,s}=1.3$$

$$C_4=1.25,C_5=0.4,C_{\varepsilon,1}=1.44,C_{\varepsilon,2}=1.92,C_s=0.25,C_\varepsilon=0.15$$

线性-可压缩-多相-高雷诺数 **SpalartAllmaras** SpalartAllmaras 湍流模型是一个具备问题属性的湍流模型。其在提出的时候就是针对空气动力学领域，例如近音速流动以及机翼设计。本质上，两方程湍流模型也是为了通过两个变量来求解湍流粘度。SpalartAllmaras 湍流模型直接对湍流粘度构建传输方程，因此不需要其他的传输方程，构成一方程湍流模型。OpenFOAM-10 中植入的 SpalartAllmaras 湍流模型为一种高雷诺数湍流模型，其原型为原始文献中的方程 4[153]。为了适应低雷诺数，应调用原始文献中的方程 12[153]。

$$\frac{\partial \alpha\rho\tilde{\nu}}{\partial t}+\nabla\cdot(\alpha\rho\mathbf{U}\tilde{\nu})-\nabla\cdot(\alpha\rho D_{\tilde{\nu}}\nabla\tilde{\nu})=\alpha\rho\frac{C_{b2}}{\sigma_{\nu_t}}|\nabla\tilde{\nu}|^2+\alpha\rho C_{b1}\tilde{\nu}\tilde{S}-\alpha\rho C_{w1}f_w\frac{\tilde{\nu}^2}{y^2}$$

$$f_w = g\left(\frac{1+C_{w3}^6}{g^6+C_{w3}^6}\right)^{\frac{1}{6}}, g = r + C_{w2}\left(r^6 - r\right), \chi = \frac{\tilde{\nu}}{\nu}, \Omega = \sqrt{2}\left|\frac{\nabla\mathbf{U} - \nabla\mathbf{U}^T}{2}\right|, \nu_t = \tilde{\nu}_t f_{v1}$$

$$\tilde{S} = \max\left(\Omega + \frac{f_{v2}\tilde{\nu}_t}{\kappa^2 y^2}, C_s\Omega\right), f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}}, f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}$$

$$r = \min\left(\frac{\tilde{\nu}_t}{\tilde{S}\kappa^2 y^2}, 10\right), D_{\tilde{\nu}} = \frac{\tilde{\nu}+\nu}{\sigma_{\nu_t}}, \sigma_{\nu_t} = 0.666666, \kappa = 0.41, C_{b1} = 0.1355$$

$$C_{b2} = 0.622, C_{w1} = \frac{C_{b1}}{\kappa^2} + \frac{1+C_{b2}}{\sigma_{\nu_t}}, C_{w2} = 0.3, C_{w3} = 2, C_{v1} = 7.1, C_s = 0.3$$

线性-可压缩-多相-低雷诺数 v2f 传统的 kEpsilon 模型认为壁面附近的特征速度尺度跟湍流动能 k 正相关。但研究工作表示这可能存在缺陷，v2f 湍流模型认为壁面附近的特征速度尺度与壁面法向应力 $\overline{\mathbf{U}_y'^2}$（在下文的方程中用 v_2 表示）正相关。因此，相关的湍流模型若要更好的捕获近壁能力，需要建立 k, ε,$\overline{\mathbf{U}_y'^2}$ 三变量方程。因此 v2f 湍流模型的原始版本也被称之为 $k-\varepsilon-\overline{\mathbf{U}_y'^2}$ 湍流模型 [46]。同时，大部分其他类型的低雷诺数湍流模型都需要定义一个 f_μ 函数来修正湍流粘度，然而这种方法由于高度的非线性会导致数值刚性。v2f 湍流模型则不需要阻尼函数。针对 $\overline{\mathbf{U}_y'^2}$ 的求解可以实现类似的动理学意义上的阻断（kinematic blocking）[102]。该模型在 kEpsilon 模型的基础上，增加两个传输方程，一个是 $\overline{\mathbf{U}_y'^2}$ 传输方程，一个是 f 传输方程。壁面法向应力可以理解为流向垂直方向的速度波动。也可用于表征壁面附近的各向异性。其大小与壁面距离高度相关。因此与 $\overline{\mathbf{U}_y'^2}$ 相关的项可以理解考虑为壁面阻尼后的效果。f 的定义为 $\overline{\mathbf{U}_y'^2}$ 传输方程的生成项与湍流动能 k 的比值。因此，若在 v2f 湍流模型为一个低雷诺数湍流模型，不需要调用壁面阻尼函数。

$$\frac{\partial\alpha\rho\varepsilon}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}\varepsilon) - \nabla\cdot(\alpha\rho D_\varepsilon\nabla\varepsilon) = C_{\varepsilon,1}\alpha\rho\frac{G}{Ts} - \left(\left(\frac{2}{3}C_{\varepsilon,1} + C_{\varepsilon,3}\right)\alpha\rho(\nabla\cdot\mathbf{U})\varepsilon\right) - C_{\varepsilon,2}\alpha\frac{\rho}{Ts}\varepsilon + S_\varepsilon$$

$$\frac{\partial\alpha\rho k}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}k) - \nabla\cdot(\alpha\rho D_k\nabla k) = \alpha\rho G - \frac{2}{3}\alpha\rho(\nabla\cdot\mathbf{U})k - \alpha\rho\frac{\epsilon}{k}k + S_k$$

$$-\nabla\cdot(\nabla f) = -\frac{1}{L_2}f - \frac{1}{L_2 k}\left(v_\alpha - C_2 G\right)$$

$$\frac{\partial\alpha\rho v_2}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}v_2) - \nabla\cdot(\alpha\rho D_k\nabla v_2) = \alpha\rho\min\left(kf, C_2 G - v_\alpha\right) - N\alpha\rho\frac{\varepsilon}{k}v_2$$

$$C_{\varepsilon,1} = 1.4\left(1 + 0.05\min\left(\sqrt{\frac{k}{v_2}}, 100\right)\right), L_2 = {L_s}^2, Ts = \max\left(\frac{k}{\varepsilon}, 6\sqrt{\frac{\nu}{\varepsilon}}\right)$$

$$L_s = C_l\max\left(\frac{k^{1.5}}{\varepsilon}, C_\eta\left(\frac{\nu^3}{\varepsilon}\right)^{0.25}\right), S_2 = 2\left|\frac{\nabla\mathbf{U} + \nabla\mathbf{U}^T}{2} - \frac{1}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right|^2$$

$$v_\alpha = \frac{1}{Ts}\left((C_1 - N)v_2 - \frac{2}{3}k(C_1 - 1)\right), \nu_t = \min\left(\frac{C_{\mu k\varepsilon}k^2}{\varepsilon}, C_\mu v_2 Ts\right)$$

$$G = \nu_t S_2, D_k = \frac{\nu_t}{\sigma_k} + \nu, D_\varepsilon = \frac{\nu_t}{\sigma_\varepsilon} + \nu, N = 6, C_\mu = 0.22, C_{\mu k\varepsilon} = 0.09, C_1 = 1.4, C_2 = 0.3, C_l = 0.23$$

$$C_\eta = 70, C_{\varepsilon,2} = 1.9, C_{\varepsilon,3} = -0.33, \sigma_k = 1, \sigma_\varepsilon = 1.3$$

v2f 模型中的 k 以及 ε 的边界条件可以参考前述低雷诺数模型。进口处的 $\overline{\mathbf{U}_y'^2}$ 可以设置为近似等于 $2/3k$，进口处的 f 则设置为零法向梯度。出口处的 $\overline{\mathbf{U}_y'^2}$ 可以设置为零法向梯度，出口处的 f 则设置为一个较小的固定值。在壁面处，$\overline{\mathbf{U}_y'^2}$ 可以设置为 0，f 设置为一个非常小的值。

线性-不可压缩-单相-低雷诺数 qZeta[94] qZeta 模型分别定义 q 与 ζ 变量（参考下面的方程），通过对 k 方程以及 ε 方程进行重整就可以得到 q 方程以及 ζ 方程。但是 qZeta 模型针对模型参数进行了调整，以满足低雷诺数的模型适用性。qZeta 模型目前并没有被广泛使用。

$$\frac{\partial \zeta}{\partial t} + \nabla \cdot (\mathbf{U}\zeta) - \nabla \cdot (D_\zeta \nabla \zeta) = (2C_1 - 1)G\frac{\zeta}{q} - \left((2C_2 f_2 - 1)\frac{\zeta}{q}\right)\zeta + E$$

$$\frac{\partial q}{\partial t} + \nabla \cdot (\mathbf{U}q) - \nabla \cdot (D_k \nabla q) = G - \frac{\zeta}{q}q$$

$$G = \frac{\nu_t 2\left|(\nabla \mathbf{U} + \nabla \mathbf{U}^T)/2\right|^2}{2q}, E = \frac{\nu\nu_t}{q}\left|\nabla\nabla(\mathbf{U})\right|^2, q = \sqrt{k}, \zeta = \frac{\varepsilon}{2q}$$

$$\nu_t = C_\mu f_\mu \frac{k^2}{\varepsilon}, f_2 = 1 - 0.3\exp(-R_t^2), R_t = \frac{qk}{2\nu\zeta}, f_\mu = \exp\left(\frac{-6}{\left(1+\frac{R_t}{50}\right)^2}\right)\left(1 + 3\exp\left(\frac{-R_t}{10}\right)\right)$$

$$C_\mu = 0.09, C_1 = 1.44, C_2 = 1.92, \sigma_\zeta = 1.3, D_q = \nu_t + \nu, D_\zeta = \frac{\nu_t}{\sigma_\zeta} + \nu$$

qZeta 湍流模型的边界条件可以参考低雷诺数 kEpsilon 模型，该模型不需要显性的指定 q 以及 ζ 的边界条件以及初始场，其从 k 以及 ε 中计算而来。

线性-可压缩-多相 kOmega kOmega 湍流模型最早起源于 Kolmogorov 在 1942 年提出的两方程模型。其也被认为是第一个两方程模型。随后 Wolcox、Saffman 等人基于前人的工作对 kOmega 模型进行了大量的改进。Kolmogorov 将 ω 认为是单位体积、单位时间下的能量耗散。Wilcox 简单的认为 ω 与 ε 以及 k 的比值正相关。OpenFOAM-8 中的 kOmega 模型原型为 Wilcox-1988 版[95][185]。但其中的区别在于 Wilcox-1988 版原始模型的 $\beta = 0.075, \gamma = 0.55556$。在上世纪 70 年代左右，在 kEpsilon 成为最受欢迎的湍流模型的前提下，很多工作发现 kEpsilon 模型在处理逆压梯度、二次流以及一些曲率较高的计算域内的流动并不完美。这就导致研究学者将目光转移到 kOmega 模型。kOmega 模型相对于 kEpsilon 模型更适用于附加逆压梯度的壁面流动以及旋转机械。同时，kOmega 模型本质是一个普适性模型可以用来处理低雷诺数以及高雷诺数，在处理低雷诺数的时候，只要壁面网格足够细致，就不需要附加阻尼函数或低雷诺数壁面函数。但是 kOmega 模型在壁面

[94] 本模型方程中的 f_μ 假定各项异性。

[95] 也可参考 NASA 的内容。

附近处，即使是低雷诺数流动，通常预测的 ω 也会趋向于无穷大。同时，kOmega 模型对于自由来流的边界条件比较敏感。

$$\frac{\partial \alpha\rho k}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}k) - \nabla\cdot(\alpha\rho D_k\nabla k) = \alpha\rho G - \frac{2}{3}\alpha\rho(\nabla\cdot\mathbf{U})k - C_\mu\alpha\rho\omega k + S_k$$

$$\frac{\partial \alpha\rho\omega}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}\omega) - \nabla\cdot(\alpha\rho D_\omega\nabla\omega) = \alpha\rho G\frac{\gamma\omega}{k} - \frac{2}{3}\gamma\alpha\rho(\nabla\cdot\mathbf{U})\omega - \beta\alpha\rho\omega^2 + S_\omega$$

$$G = \nu_t\left(\nabla\mathbf{U} + \nabla\mathbf{U}^T - \frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right) : \nabla\mathbf{U}, \nu_t = \frac{k}{\omega}, C_\mu = 0.09, \beta = 0.072, \alpha_k = 0.5$$

$$\alpha_\omega = 0.5, \gamma = 0.52, D_k = \alpha_k\nu_t + \nu, D_\omega = \alpha_\omega\nu_t + \nu$$

线性-可压缩-多相 kOmega2006 自 OpenFOAM-11 开始，OpenFOAM 引入了 kOmega 模型 2006 年版本 [187]。文献表示 kOmega 模型之所以对来流比较敏感就是因为 kOmega 模型中交叉扩散项的缺失 [186]。kOmega2006 版本与之前的 kOmega 版本最大的区别在与引入交叉扩散项。目前来说，当代 CFD 讨论的 kOmega 模型，主要是 kOmega2006 版本或者 kOmegaSST 模型。

$$\frac{\partial \alpha\rho k}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}k) - \nabla\cdot(\alpha\rho D_k\nabla k) = \alpha\rho G - \frac{2}{3}\alpha\rho(\nabla\cdot\mathbf{U})k - C_\mu\alpha\rho\omega k + S_k$$

$$\frac{\partial \alpha\rho\omega}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}\omega) - \nabla\cdot(\alpha\rho D_\omega\nabla\omega) = \alpha\rho G\frac{\gamma\omega}{k} - \frac{2}{3}\gamma\alpha\rho(\nabla\cdot\mathbf{U})\omega - \beta\alpha\rho\omega^2 + \alpha\rho C_{DK\omega} + S_\omega$$

$$C_{DK\omega} = \sigma_{do}\frac{\nabla k\cdot\nabla\omega}{\omega}, \beta = \beta_0\frac{1+85\chi_\omega}{1+100\chi_\omega}, \chi_\omega = \frac{|(\mathbf{\Omega}\cdot\mathbf{\Omega}):\mathbf{sh}|}{(\beta^*\omega)^3}$$

$$\mathbf{\Omega} = \frac{\nabla\mathbf{U} - \nabla\mathbf{U}^T}{2}, \mathbf{sh} = \frac{\nabla\mathbf{U} + \nabla\mathbf{U}^T}{2} - 0.5(\nabla\cdot\mathbf{U})\mathbf{I}$$

$$G = \nu_t\left(\nabla\mathbf{U} + \nabla\mathbf{U}^T - \frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right) : \nabla\mathbf{U}, C_\mu = 0.09, \alpha_k = 0.6, \beta^* = 0.09.\beta_0 = 0.0708$$

$$\alpha_\omega = 0.5, \gamma = 0.52, D_k = \alpha_k\nu_t + \nu, D_\omega = \alpha_\omega\nu_t + \nu, C_{li} = 0.875, \sigma_{do} = 0.125$$

$$\nu_t = \frac{k}{\max\left(\omega, C_{li}\sqrt{2/\beta^*}|\frac{\nabla\mathbf{U}+\nabla\mathbf{U}^T}{2} - \frac{1}{3}(\nabla\cdot\mathbf{U})\mathbf{I}|\right)}$$

线性-可压缩-多相 kOmegaSST kOmegaSST 为一种混合 kEpsilon 与 kOmega 模型的混合模型。为了将 kEpsilon 模型与 kOmega 模型的优点相结合并去除相关的缺点（kEpsilon 对于逆压梯度区域的预测不足以及 kOmega 模型的进口条件过于敏感），kOmegaSST 通过混合函数可自动在 kOmega 模型以及 kEpsilon 模型间自动转换。在壁面附近区域，kOmegaSST 模型通过混合函数激活 kOmega 模型，在自由来流以及核心区，kOmegaSST 模型转变为 kEpsilon 模型。由于在壁面附近 kOmegaSST 模型转变为 kOmega 模型，因此 kOmegaSST 模型在低雷诺数流动时不需要像 kEpsilon 模型那样需要低雷诺数湍流模型或阻尼函数。在自由流处，kOmegaSST 模型也对进口条件不敏感。kOmegaSST 模型可

以较好的预测逆压梯度以及分离流。模型中的 $B(a,b,c)$ 函数通过 a（也即 F_1）的值在 b 与 c 模型系数中进行切换。F_1 为 tanh 函数位于 0, 1 之间。kOmegaSST 模型的 $D_k, D_\omega, \beta, \gamma$ 均通过 $B(a,b,c)$ 函数进行系数混合转变。F_3 函数默认关闭，若激活可以处理粗糙壁面问题 [68]。OpenFOAM-8 中目前植入的版本为 Menter-2003 版 [119]（因为其中的系数 c_1 为 10）[96]。注意下文公式中的 ω 方程右侧的湍流产生项经过推导后即为 $\alpha\rho\frac{\gamma}{\nu_t}P_k$。kOmegaSST 在提出的过程中，参考了二阶矩模型的思想，即将 Bradshaw 假定作为根基，认为边界层的切应力与湍流动能呈线性关系：$\boldsymbol{\tau}_{12} = \rho a_1 k$，但在传统两方程模型中，这个关系为 $\boldsymbol{\tau}_{12} = \rho a_1 k(k_{production}/k_{dissipation})$。在逆压梯度附近，$k_{production}/k_{dissipation}$ 远大于 1。因此会导致剪切应力的高估。为了满足 Bradshaw 假定，kOmegaSST 将湍流粘度的定义除掉了 $\sqrt{S_2}$，即 $\nu_t = a_1 k/\sqrt{S_2}$。这样可以保证剪切应力不会过快的变化。进一步的在 ν_t 的定义中的分母中施加 max 函数，可以保证这个限制器仅仅在逆压梯度附近起作用。这也即为 SST 的来源 [118]。

$$\frac{\partial \alpha\rho k}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}k) - \nabla\cdot(\alpha\rho D_k\nabla k) = \alpha\rho P_k - \frac{2}{3}\alpha\rho k\left(\nabla\cdot\mathbf{U}\right) - \alpha\rho\beta^*\omega k + S_k$$

$$\begin{aligned}\frac{\partial \alpha\rho\omega}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}\omega) - \nabla\cdot(\alpha\rho D_\omega\nabla\omega) = \alpha\rho\gamma\min\left(\frac{G}{\nu_t}, \frac{c_1}{a_1}\beta^*\omega\max\left(a_1\omega, b_1F_2\sqrt{S_2}\right)\right)\\ - \frac{2}{3}\alpha\rho\gamma\left(\nabla\cdot\mathbf{U}\right)\omega - \alpha\rho\beta\omega^2 - \alpha\rho\left(F_1-1\right)\frac{CD_{k\omega}}{\omega}\omega + S_\omega\end{aligned}$$

$$P_k = \min\left(G, c_1\beta^*k\omega\right), CD_{k\omega} = \frac{2\alpha_{\omega,2}(\nabla k\cdot\nabla\omega)}{\omega}, G = \nu_t\left(\nabla\mathbf{U} + \nabla\mathbf{U}^T - \frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right):\nabla\mathbf{U}$$

$$S_2 = 2\left|\frac{\nabla\mathbf{U} + \nabla\mathbf{U}^T}{2}\right|^2, F_1 = \tanh\left(\min\left(\min\left(\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{\omega y^2}\right), \frac{4\alpha_{\omega,2}k}{CD_{k\omega}y^2}\right), 10\right)\right)^4$$

$$F_2 = \tanh\left(\min\left(\max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{\omega y^2}\right), 100\right)\right)^2, F_3 = 1 - \tanh\left(\min\left(\frac{150\nu}{\omega y^2}, 10\right)\right)^4$$

$$D_k = B(F_1, \alpha_{k1}, \alpha_{k2})\nu_t + \nu, D_\omega = B(F_1, \alpha_{\omega1}, \alpha_{\omega2})\nu_t + \nu, \beta = B(F_1, \beta_1, \beta_2)$$

$$\gamma = B(F_1, \gamma_1, \gamma_2), B(a,b,c) = ab + (1-a)c$$

$$\alpha_{k,1} = 0.85, \alpha_{k,2} = 1, \alpha_{\omega,1} = 0.5, \alpha_{\omega,2} = 0.856, \gamma_1 = \frac{5}{9}, \gamma_2 = 0.44, \beta_1 = 0.075, \beta_2 = 0.0828,$$

$$\beta^* = 0.09, a_1 = 0.31, b_1 = 1, c_1 = 10, \nu_t = \frac{a_1 k}{\max\left(a_1\omega, b_1F_2\sqrt{S_2}\right)}, \varepsilon = \beta^*k\omega$$

[96] 也可参考 NASA 的内容。

6.14.2 LES 湍流模型

可压缩-多相 Smagorinsky[97] Smagorinsky 湍流模型是目前应用最广泛的 LES 模型。Smagorinsky 模型中的在大部分的文献中表示为 $\nu_t = (C_k\Delta)^2\sqrt{2\overline{\mathbf{D}}:\overline{\mathbf{D}}}$。在高剪切流域（$\overline{\mathbf{D}}$ 较大），Smagorinsky 模型会导致比较大的耗散。因此在这些区域，C_k 的值需要降低。尤其在壁面处，虽然近似与层流，由于速度梯度较大，Smagorinsky 模型偏向与高估湍流粘度。另一方面，Smagorinsky 假定湍流是均衡的，因此在近壁面处略有不妥。同时，Smagorinsky 模型在壁面处的 Δ 并不为零（由于梯度不为零），这将导致壁面附近湍流粘度并不为零，并不符合实际。这可以通过 van Driest 修正来实现：将 Δ 建立与壁面距离的关系式并实现一种阻尼效果（$\Delta \to 0, y \to 0$）。van Driest 将 Δ 建立在 y^+ 基础之上，因此 van Driest 模型需要在壁面布置较多的网格点来获取精准的 y^+。另一方面，由于 Smagorinsky 模型高估湍流粘度，其求解也相对稳定。同时不需要附加任何的传输方程，计算快速。因此其在应用中较为广泛。

$$\nu_t = C_k\Delta\sqrt{k_{sgs}}, \overline{\mathbf{D}} = \frac{\nabla\overline{\mathbf{U}} + \left(\nabla\overline{\mathbf{U}}\right)^T}{2}, \varepsilon_{sgs} = \frac{C_e k_{sgs}^{3/2}}{\Delta}, C_k = 0.094, C_e = 1.048$$

$$k_{sgs} = \left(\frac{-b+\sqrt{b^2+4ac}}{2a}\right)^2, a = \frac{C_e}{\Delta}, b = \frac{2}{3}\mathrm{tr}(\overline{\mathbf{D}}), c = 2C_k\Delta\left(\left(\overline{\mathbf{D}} - \frac{1}{3}\mathrm{tr}\left(\overline{\mathbf{D}}\right)\mathbf{I}\right):\overline{\mathbf{D}}\right)$$

可压缩-多相 WALE 虽然 Smagorinsky 可以结合 van Driest 函数来对壁面湍流粘度进行修正。但其需要在壁面布置较多的网格点形成边界层，因此对于复杂几何较难实施。WALE 模型将 ν_{sgs} 与旋转率联系起来，因此对于任意的表面，由于旋转率趋向于 0，因此 ν_{sgs} 也趋向于零。这使得 WALE 相对于 Smagorinsky 模型更加易于实施且更加具有稳定性。

$$\nu_t = C_k\Delta\sqrt{k_{sgs}}, \overline{\mathbf{S}_d} = \frac{\nabla\overline{\mathbf{U}}\cdot\nabla\overline{\mathbf{U}} + \left(\nabla\overline{\mathbf{U}}\cdot\nabla\overline{\mathbf{U}}\right)^T}{2} - \frac{1}{3}\mathrm{tr}\left(\frac{\nabla\overline{\mathbf{U}}\cdot\nabla\overline{\mathbf{U}} + \left(\nabla\overline{\mathbf{U}}\cdot\nabla\overline{\mathbf{U}}\right)^T}{2}\right)\mathbf{I}, C_e = 1.048$$

$$k_{sgs} = \left(C_w\frac{\Delta}{C_k}\right)^2\frac{|\overline{\mathbf{S}_d}|^5}{\left(|\nabla\overline{\mathbf{U}} + \nabla\overline{\mathbf{U}}^T|^5 + |\mathbf{S}_d|^{5/2}\right)^2}, \varepsilon_{sgs} = \frac{C_e k_{sgs}^{3/2}}{\Delta}, C_k = 0.094, C_w = 0.325$$

可压缩-多相 kEqn LES 中的 kEqn 模型类似于 RANS 中的 kEpsilon 模型。其中的 k_{sgs} 采用传输方程进行描述，方程左侧的各项表示时间项、对流项、扩散项，方程右侧各项表示产生项、可压缩项以及耗散项。kEqn 模型的模型系数是固定的。因此并不是一个动态的模型（相比较于下文的 dynamicKEqn 模型）。

$$\frac{\partial\alpha\rho k_{sgs}}{\partial t} + \nabla\cdot\left(\alpha\rho\overline{\mathbf{U}}k_{sgs}\right) - \nabla\cdot\left(\alpha\rho D_k\nabla k_{sgs}\right) = \alpha\rho\overline{G} - \frac{2}{3}\alpha\rho(\nabla\cdot\overline{\mathbf{U}})k_{sgs} - \alpha\rho\varepsilon_{sgs} + S_k$$

$$\overline{G} = \nu_t\left(\nabla\overline{\mathbf{U}} + \nabla\overline{\mathbf{U}}^T - \frac{2}{3}(\nabla\cdot\mathbf{U})\mathbf{I}\right):\nabla\overline{\mathbf{U}}, \nu_t = C_k\Delta\sqrt{k_{sgs}}, \varepsilon_{sgs} = C_e\frac{k_{sgs}^{3/2}}{\Delta}$$

[97] Δ 表示截止尺度模型

$$C_e = 1.048, C_k = 0.094$$

可压缩-多相 dynamicKEqn[98] 动态 dynamicKEqn 模型来自于 Kim 在 2004 年发表的文章 [77]。相比较与 Germano et al. 的动态 Smagorinsky 模型，动态 dynamicKEqn 模型附加了 k_{sgs} 传输方程。在非动态的 kEqn 模型中，C_k 以及 C_e 都是固定的。动态 dynamicKEqn 模型这两个模型系数都是局部可变的。C_k 的求解思想参考了 Germano 恒等式，但 dynamicKEqn 并没有直接调用，而是通过前人的实验数据将 Leonard 应力 $\mathbf{L}$ 与亚格子应力建立恒等关系。这样的话在方程右侧不需要对 C_s 系数做二次滤波，因此也不需要做简化假定。最终导致 $\mathbf{M}$ 的模化方法不同。C_e 的求解思想调用了 Smagorinsky 模型的局部均衡假定。另外，传统的动态模型最大的问题就是用于预测一个负的 C_k，一个负的 C_k 会产生一个负的湍流粘度进而对计算产生非常大的稳定性问题。在 Germano 等的文章中采用一种空间平均的技术来将部分的负的 C_k 抹平 [57]，OpenFOAM 则通过 simple 滤波以及 $|C_k'| + C_k'$ 的方式尽可能的减少负的 C_k 的产生。原始的动态 Smagorinsky 模型由于负系数导致的收敛性问题，OpenFOAM 已经将 dynSmagorinsky 模型彻底删掉[99]。

$$\frac{\partial \alpha\rho k_{sgs}}{\partial t} + \nabla\cdot\left(\alpha\rho\overline{\mathbf{U}}k_{sgs}\right) - \nabla\cdot\left(\alpha\rho D_k\nabla k_{sgs}\right) = \alpha\rho G - \frac{2}{3}\alpha\rho(\nabla\cdot\overline{\mathbf{U}})k_{sgs} - \alpha\rho\varepsilon_{sgs} + S_k$$

$$\varepsilon_{sgs} = \frac{C_e k_{sgs}^{3/2}}{\Delta}, \nu_t = C_k\Delta\sqrt{k_{sgs}}, \overline{\mathbf{D}} = \frac{\nabla\overline{\mathbf{U}} + \left(\nabla\overline{\mathbf{U}}\right)^T}{2} - \frac{1}{3}(\nabla\cdot\overline{\mathbf{U}})\mathbf{I}, G = 2\nu_t\nabla\overline{\mathbf{U}} : \overline{\mathbf{D}}$$

$$\mathbf{L} = \left\langle \widetilde{\overline{\mathbf{U}}\,\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\widetilde{\overline{\mathbf{U}}} - \frac{1}{3}\mathrm{tr}\left(\widetilde{\overline{\mathbf{U}}\,\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\widetilde{\overline{\mathbf{U}}}\right)\mathbf{I}\right\rangle, \mathbf{M} = \left\langle -2\Delta\sqrt{k_{test}}\widetilde{\overline{\mathbf{D}}}\right\rangle$$

$$C_k' = \frac{1}{2}\frac{\langle \mathbf{L} : \mathbf{M}\rangle}{\langle \mathbf{M} : \mathbf{M}\rangle}, C_e' = \frac{1}{2}\left\langle \frac{\Delta(\nu + \nu_t)\left(\widetilde{\overline{\mathbf{D}} : \overline{\mathbf{D}}} - \widetilde{\overline{\mathbf{D}}} : \widetilde{\overline{\mathbf{D}}}\right)}{k_{test}^{3/2}}\right\rangle$$

$$C_k = \frac{1}{2}(|C_k'| + C_k'), C_e = \frac{1}{2}(|C_e'| + C_e'), k_{test} = \frac{1}{2}\left(\widetilde{\overline{\mathbf{U}}\cdot\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\cdot\widetilde{\overline{\mathbf{U}}}\right), D_k = \nu_t + \nu$$

注意公式中的 $\langle\rangle$ 表示光顺后的场，主要用于消除导致负的模型系数的情况，在 OpenFOAM 中对用`simpleFilter`。

可压缩-多相 dynamicLagrangian Meneveau 等人认为动态类模型通过各种空间平均技术预测的正的系数或多或少都存在缺陷 [117]。考虑一种流体粒子的运动，其运动轨迹符合拉格朗日运动。Meneveau 等认为流体的流动遵循流线，因此这种平均技术应该在流线上进行平均操作。dynamicLagrangian 模型力图从拉格朗日流线方程出发，使得方程(4.29)的误差最小化。

$$\frac{\partial \alpha\rho\mathcal{J}_{lm}}{\partial t} + \nabla\cdot\left(\alpha\rho\overline{\mathbf{U}}\mathcal{J}_{lm}\right) = \frac{1}{T}(\mathbf{L} : \mathbf{M} - \mathcal{J}_{lm})$$

[98] $\langle\rangle$ 表示 simple 滤波。

[99] 动态 Smagorinsky 模型 [57] 被认为是最具有影响力的显性大涡模拟模型。在 1990 年，Germano 等的工作所有的结果在 3 周搞定。从想法的构思，到稿件的提交。一共少于 4 个月。这一篇只有 6 页的 LES 的论文，核心公式只有 6 个，在 2025 年被引用破 10000 次。从引用数来判断，该文章位列人类历史上最有影响力的湍流论文第二位。

$$\frac{\partial \alpha\rho \jmath_{mm}}{\partial t} + \nabla\cdot\left(\alpha\rho\overline{\mathbf{U}}\jmath_{mm}\right) = \frac{1}{T}(\mathbf{M}:\mathbf{M} - \jmath_{mm})$$

$$\overline{\mathbf{S}} = \frac{\nabla\overline{\mathbf{U}} + \left(\nabla\overline{\mathbf{U}}\right)^T}{2} - \frac{1}{3}(\nabla\cdot\overline{\mathbf{U}})\mathbf{I}, \widetilde{\overline{\mathbf{S}}} = \frac{\nabla\widetilde{\overline{\mathbf{U}}} + \left(\nabla\widetilde{\overline{\mathbf{U}}}\right)^T}{2} - \frac{1}{3}(\nabla\cdot\widetilde{\overline{\mathbf{U}}})\mathbf{I},$$

$$\mathbf{L} = \widetilde{\overline{\mathbf{U}}\,\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\widetilde{\overline{\mathbf{U}}} - \frac{1}{3}\mathrm{tr}\left(\widetilde{\overline{\mathbf{U}}\,\overline{\mathbf{U}}} - \widetilde{\overline{\mathbf{U}}}\widetilde{\overline{\mathbf{U}}}\right)\mathbf{I}, \mathbf{M} = 2\Delta^2\left(\widetilde{|\overline{\mathbf{S}}|\overline{\mathbf{S}}} - 4|\widetilde{\overline{\mathbf{S}}}|\widetilde{\overline{\mathbf{S}}}\right)$$

$$\frac{1}{T} = \frac{\alpha\rho}{\theta\Delta}(\jmath_{lm}\jmath_{mm})^{1/8}, k_{sgs} = \left(\frac{2\jmath_{lm}}{\jmath_{mm}}\right)^{2/3} C_e^{-2/3}\Delta^2|\overline{\mathbf{S}}|^2$$

$$C_e = 1.048, \theta = 1.5, \nu_t = \frac{\jmath_{lm}}{\jmath_{mm}}\Delta^2|\overline{\mathbf{S}}|$$

可压缩-多相 **DeardorffDiffStress**

$$\frac{\partial\alpha\rho\mathbf{R}}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}\mathbf{R}) - \nabla\cdot\left(\left(\nu\mathbf{I} + C_s\frac{k_{sgs}}{\varepsilon}\mathbf{R}\right)\nabla\mathbf{R}\right) + C_m\frac{\alpha\rho\sqrt{k_{sgs}}}{\Delta}\mathbf{R} = \alpha\rho\mathbf{P} + \frac{4}{5}\alpha\rho k\mathbf{D} - \frac{2}{3}\left(1 - \frac{C_m}{C_e}\right)\alpha\rho\varepsilon\mathbf{I}$$

$$\mathbf{D} = \frac{\nabla\overline{\mathbf{U}} + \left(\nabla\overline{\mathbf{U}}\right)^T}{2}, \mathbf{P} = \mathbf{R}\cdot\nabla\mathbf{U} + (\mathbf{R}\cdot\nabla\mathbf{U})^T, \varepsilon_{sgs} = \frac{C_e k_{sgs}^{3/2}}{\Delta}, k_{sgs} = \frac{1}{2}(\mathbf{R}_{xx} + \mathbf{R}_{yy} + \mathbf{R}_{zz})$$

$$\nu_t = C_k\Delta\sqrt{k_{sgs}}, C_k = 0.094, C_m = 4.13, C_e = 1.05, C_s = 0.25$$

6.14.3 RANS-LES 混合模型

可压缩-多相 **kOmegaSSTSAS** 在 **kOmegaSST** 模型上添加 ω 方程源项：

$$S_\omega = \alpha\rho\omega\min\left(\max\left(\zeta_2\kappa S_2\left(\frac{L}{L_{vk}}\right)^2 - \frac{2C}{\sigma_\phi}k\max\left(\frac{|\nabla\omega|^2}{\omega^2}, \frac{|\nabla k|^2}{k^2}\right), 0\right), \frac{\omega}{0.1\Delta t}\right)$$

$$L_{vk} = \max\left(\frac{\kappa\sqrt{S_2}}{|\nabla\cdot(\nabla\mathbf{U})|}, C_s\sqrt{\frac{\kappa\zeta_2}{\beta/\beta^* - \gamma}}\Delta\right), L = \frac{\sqrt{k}}{\beta^{*0.25}\omega}$$

$$C_s = 0.11, \kappa = 0.41, \zeta_2 = 3.51, \sigma_\phi = \frac{2}{3}, C = 2, \Delta = (\Delta x\Delta y\Delta z)^{1/3}$$

可压缩-多相 **SpalartAllmarasDES** Smagorinsky 模型认为湍流粘度正比于 $\Delta^2\sqrt{2\mathbf{D}:\mathbf{D}}$。Spalart 认为在 SpalartAllmaras 湍流模型中，可以将壁面举例 y 替换为 $\min(C_{DES}\Delta, y)$[154]。

$$\frac{\partial\alpha\rho\tilde{\nu}}{\partial t} + \nabla\cdot(\alpha\rho\mathbf{U}\tilde{\nu}) - \nabla\cdot(\alpha\rho D_{\tilde{\nu}}\nabla\tilde{\nu}) - \frac{C_{b2}}{\sigma_{\nu_t}}\alpha\rho|\nabla\tilde{\nu}|^2 = C_{b1}\alpha\rho\tilde{S}\tilde{\nu} - C_{w1}\alpha\rho f_w\frac{\tilde{\nu}^2}{\tilde{d}^2}$$

$$f_w = g\left(\frac{1 + C_{w3}^6}{g^6 + C_{w3}^6}\right)^{\frac{1}{6}}, g = r + C_{w2}\left(r^6 - r\right), \chi = \frac{\tilde{\nu}}{\nu}, \Omega = \sqrt{2}\left|\frac{\nabla\mathbf{U} - \nabla\mathbf{U}^T}{2}\right|, \nu_t = \tilde{\nu}f_{v1}$$

$$\tilde{S}=\max\left(\Omega+\frac{f_{v2}\tilde{\nu}}{\kappa^2\tilde{d}^2},C_s\Omega\right),f_{v2}=1-\frac{\chi}{1+\chi f_{v1}},f_{v1}=\frac{\chi^3}{\chi^3+C_{v1}^3}$$

$$r=\min\left(\frac{\tilde{\nu}}{\tilde{S}\kappa^2\tilde{d}^2},10\right),D_{\tilde{\nu}}=\frac{\tilde{\nu}+\nu}{\sigma_{\nu_t}},\sigma_{\nu_t}=0.66666,\kappa=0.41,C_{b1}=0.1355$$

$$C_{b2}=0.622,C_{w1}=\frac{C_{b1}}{\kappa^2}+\frac{1+C_{b2}}{\sigma_{\nu_t}},C_{w2}=0.3,C_{w3}=2,C_{v1}=7.1,C_s=0.3$$

$$\tilde{d}=\min\left(C_{DES}\Delta,y\right),C_{DES}=0.65,k_{sgs}=\left(\frac{\nu_t}{C_k\tilde{d}}\right)^2,C_k=0.07$$

可压缩-多相 SpalartAllmarasDDES：在 **SpalartAllmarasDES** 模型的基础上，更改

$$\tilde{d}=\max\left(y-\left(1-\tanh\left((8r)^3\right)\right)\max\left(y-C_{DES}\Delta,0\right),0\right),r=\min\left(\frac{\nu+\nu_t}{|\nabla\mathbf{U}|\kappa^2y^2},10\right)$$

可压缩-多相 kOmegaSSTDES：将 **kOmegaSST** 模型的 k 方程的源项 $-\alpha\rho\beta^*\omega k$ 更改为（其中 F_1,F_2 取自 kOmegaSST 模型）：

$$-\alpha\rho\beta^*\omega k\rightarrow-\alpha\rho\beta^*\omega kF_{DES},F_{DES}=\max\left(\frac{L_t(1-F_{SST})}{C_{DES}\Delta},1\right),L_t=\frac{\sqrt{k}}{\beta^*\omega},C_{DES}=0.61$$

$$F_{SST}=\begin{cases}\max\left(\frac{L_t}{C_{DES}},1\right), & 0\\ \max\left(\frac{L_t(1-F_1)}{C_{DES}},1\right), & 1\\ \max\left(\frac{L_t(1-F_2)}{C_{DES}},1\right), & 2\end{cases}$$

LES 中的 simple 滤波函数：在显性 LES 方法中需要调用二次滤波函数。目前在 OpenFOAM 中植入的 simple 滤波与有限体积法相辅相成。其他的滤波方法也具有同样的特性。各种滤波方法都可以看做是一种对场的一种空间光顺。在下式中，考虑三网格点，其中左中右三个网格点的值分别为 ϕ_{i-1}、ϕ_i、ϕ_{i+1}。经过 simple 滤波计算后，其值为 $\frac{(\phi_{i-1}+\phi_i)/2+(\phi_i+\phi_{i+1})/2}{2}$。这与传统带网格编号的结构网格计算结果是一致的 $0.25\phi_{i-1}+0.5\phi_i+0.25\phi_{i-1}$。并且二次滤波的尺度为网格尺度的两倍。在非结构网格的情况下，二次滤波的尺度并不会严格的满足 2 倍但一定比网格尺度大。

$$\overline{\phi}=\frac{\sum|\mathbf{S}_f|\phi_f}{\sum|\mathbf{S}_f|}$$

LES 中的 laplace 滤波函数：laplace 滤波函数可以依据 $W_{widthCoeff}$ 的值来进行调节。同样考虑简单的三网格点，laplace 滤波可以写离散形式 $\phi_i+\frac{\Delta V^{2/3}}{W_{widthCoeff}}\frac{\phi_{i+1}-2\phi_i+\phi_{i-1}}{\Delta x^2}$。若继续认为是均一网格，有 $\Delta V=\Delta x^3$，即 $\frac{\phi_{i+1}+(W_{widthCoeff}-2)\phi_i+\phi_{i-1}}{W_{widthCoeff}}$。当 $W_{widthCoeff}=4$，laplace 滤波与 simple 滤波一致。$W_{widthCoeff}$ 的值越大，laplace 滤波的作用越小。

$$\overline{\phi}=\phi+\nabla\cdot\left(\frac{\Delta V^{2/3}}{W_{widthCoeff}}\nabla\phi\right)$$

α	ρ	ν	ε	k	k_{sgs}	ω	$\mathbf{R}$	y	Δ
相分数	密度	运动粘度	耗散率	湍动能	亚格子湍动能	湍流频率	雷诺应力	壁面距离	截断尺度

表 6.1: 湍流模型符号表

6.14.4 壁面函数

在普朗特之前，一些科学家曾经对管道内的速度以及摩擦系数进行研究，尝试总结出相应的规律。但是当时的发现发现每一套的数据均不一致。直到普朗特将速度处理为 u^+，将壁面处理为 y^+ 之后，才发现所有的实验数据，在壁面处，均贴近于壁面法则。

随着工业 CFD 问题的尺度越来越大，虽然计算机的计算能力一直在提高，但对于湍流求解，壁面网格一直不够致密去直接解析。如果壁面的流动无法精确的预测，则预测的壁面剪切力误差会非常大，进一步的相关变量如阻力系数等均会产生非常大的误差，甚至进一步影响到全场。最重要的是，CFD 经常被应用于空气动力学的阻力分析。壁面剪切力的预测不精准，直接导致 CFD 的预测结果无法用于工业设计。为了合理的对壁面流动行为进行预测。一些湍流模型要求壁面处的网格位于粘性支层内，这导致计算量大大增加。例如，压缩机叶片的研究需要全场的流动结构。如果叶片壁面网格位于粘性支层，会导致计算网格大量增加。壁面函数针对现存问题，可以通过较少的壁面网格结构获得精准的结果。如果应用壁面函数，壁面的网格点可放置于 log 区，大大减少网格数量并且毫不损失精度。例如，对于非均一分布（延伸率 1.15）的未使用壁面函数的壁面网格，对于基于壁面边界层厚度雷诺数为 5000 的的壁面流动，大约需要 40 个边界层网格。若使用壁面函数（第一层网格），只需要约 15 个壁面网格。大大节省了计算量。

如图6.36所示，大量的实验表明壁面处附近的无量纲速度 u^+ 与无量纲距离 y^+ 满足一定的型线。在 19 世纪，当时的流体力学研究都是 2 维的，因此 y 方向表示壁面垂直的方向。图中的两个蓝线分别为 u^+ 与 y^+ 的线性分布与 log 分布，红线表示实验检测的普适性 $u^+ - y^+$ 分布。可以看出：

- 在粘性支层 $y^+ < 5$，$u^+ - y^+$ 符合线性关系：

$$u^+ = y^+ \tag{6.280}$$

- 在 log 区 $y^+ > 30$，$u^+ - y^+$ 符合 log 分布。对于光滑壁面：

$$u^+ = \frac{\ln(Ey^+)}{\kappa} \tag{6.281}$$

其中 E 通常取值 9.8，κ 通常取值 0.4。对于粗糙壁面：

$$u^+ = \frac{\ln(Ey^+)}{\kappa} - C \tag{6.282}$$

粗糙壁面会导致 y^+ 曲线整体的下移。

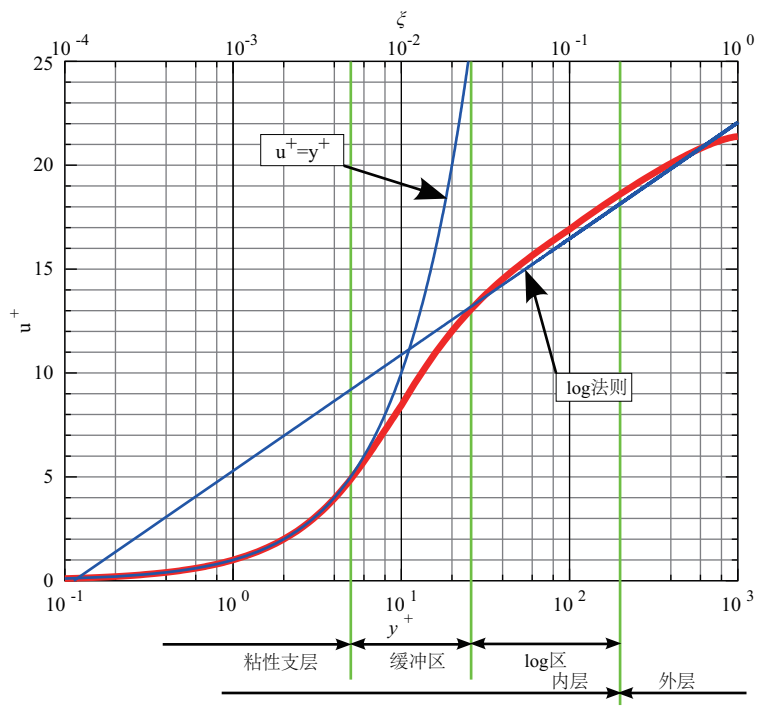


图 6.36: 壁面处 $u^+ - y^+$ 分布图。两个蓝线分别为 u^+ 与 y^+ 的线性分布与 log 分布，红线表示实验检测的普适性 $u^+ - y^+$ 分布。

- 在缓冲层 $5 < y^+ < 30$，在这一层湍流应力与粘性应力的贡献相当。由于二者同时存在，因此并不存在一个较好的速度型线定义。在 CFD 实施的过程中，最好避免将第一层网格布置在缓冲层。

很明显，参考图6.36，只有 $y^+ < 5$ 或者 $y^+ > 30$ 的情况下，存在线性的关系或者 log 关系。如果第一层网格不在这个区间，并没有一个合适的型线进行描述。然而在 CFD 代码中，并不能够在这个区间进行留空。因此 CFD 代码会存在不同的做法，一种是将其简单区分为 2 层，界限为 $y^+ = 11.5$，即为图6.36中两条直线的交叉处。$y^+ < 11.5$，满足线性分布，$y^+ > 11.5$，满足 log 分布。

另一种方法是重新定义 y^+, u^+ 的关系，如 [155]：

$$y^+ = u^+ + 0.1108\left[e^{0.4u^+} - 1 - 0.4u^+ - \frac{1}{2}\left(0.4u^+\right)^2 - \frac{1}{6}\left(0.4u^+\right)^3\right] \tag{6.283}$$

这种型线非常贴合图6.36中红线的数据。Spalding 认为该函数在 $y^+ < 300$ 的时候都有效。

在这里对壁面函数进行更深层次的讨论。二维情况下的壁面应力可以表示为：

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} \\ \tau_{yx} & \tau_{yy} \end{bmatrix} = (\mu_l + \mu_t)\begin{bmatrix} 2\frac{\partial u_1}{\partial x} & \frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y} \\ \frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y} & 2\frac{\partial u_2}{\partial y} \end{bmatrix} \tag{6.284}$$

在不可渗透壁面，$u_2 = 0$，因此仅仅 $\tau_{xy} \neq 0$：

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_{xx} & \tau_{xy} \\ \tau_{yx} & \tau_{yy} \end{bmatrix} = (\mu_l + \mu_t) \begin{bmatrix} 0 & \frac{\partial u_1}{\partial y} \\ \frac{\partial u_1}{\partial y} & 0 \end{bmatrix} \tag{6.285}$$

壁面剪切力一般用 τ_w 表示其中的 τ_{xy} 分量[100]。写成离散的形式：

$$\tau_w = (\mu_l + \mu_t) \frac{u_\mathrm{P}}{y_\mathrm{P}} \tag{6.289}$$

CFD 要保证的，即正确的预测壁面剪切力。为了引申出无量纲速度以及无量纲距离的概念，定义壁面摩擦速度 u_τ[101]：

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{6.290}$$

$$u_\tau = \sqrt{(\nu_l + \nu_t) \frac{u_\mathrm{P}}{y_\mathrm{P}}} \tag{6.291}$$

基于摩擦速度，可以引申出 u^+, y^+ 的定义：

$$\begin{aligned} u^+ &= \frac{u}{u_\tau} \\ y^+ &= \frac{u_\tau y}{\nu} \end{aligned} \tag{6.292}$$

将方程(6.292)带入到(6.291)中有：

$$u_\tau^2 = (\nu + \nu_t) \frac{u^+ u_\tau^2}{y^+ \nu} \tag{6.293}$$

即

$$\nu_t = \frac{y^+}{u^+} \nu - \nu \tag{6.294}$$

可见，在粘性支层，$\nu_t = 0$。在 log 区，方程(6.294)可以写为

$$\nu_t = \left(\frac{y^+ \kappa}{\ln E y^+} - 1 \right) \nu \tag{6.295}$$

另一方面，大量的实验以及 DNS 数据表明，在壁面处的 log 区内，湍流动能的产生与耗散相抵。基于这个假设，有另外一种摩擦速度的定义[102]：

$$u_\tau = C_\mu^{0.25} k^{0.5} \tag{6.299}$$

---

[100] 考虑壁面附近的动量方程，其可以简化为：

$$u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} = \frac{\partial}{\partial y} \left(\nu \frac{\partial u}{\partial y} - \overline{u'v'} \right) \tag{6.286}$$

因为 $u \frac{\partial u}{\partial x} \approx 0, v \frac{\partial u}{\partial y} \approx 0$，所以方程可以简化为

$$\frac{\partial}{\partial y} \left(\nu \frac{\partial u}{\partial y} - \overline{u'v'} \right) \approx 0 \tag{6.287}$$

其表明壁面附近的 $\nu \frac{\partial u}{\partial y} - \overline{u'v'}$ 近似为一个常数。其中的雷诺应力项 $\overline{u'v'}$ 可以写为 $-\nu_t \frac{\partial u}{\partial y}$。因此，上式可以继续写为

$$\frac{\partial}{\partial y} \left((\nu + \nu_t) \frac{\partial u}{\partial y} \right) \approx 0 \tag{6.288}$$

其表示 $(\nu + \nu_t) \frac{\partial u}{\partial y}$ 为一个常数，这个常数即为 τ_w。

[101] 下文中还会有另外一种摩擦速度的定义。

[102] 壁面附近的湍流动能的产生与耗散相抵，即表明 $\varepsilon = G$，其中 G 即为标准 $k - \varepsilon$ 模型中的产生项：

$$G = \nu_t \left(\nabla \mathbf{U} + \nabla \mathbf{U}^T - \frac{2}{3} (\nabla \cdot \mathbf{U}) \mathbf{I} \right) : \nabla \mathbf{U} \tag{6.296}$$

将方程(6.299)带入到(6.292)有：

$$y^+ = \frac{C_\mu{}^{0.25} y_{\mathrm{P}} k_{\mathrm{P}}^{0.5}}{\nu} \tag{6.300}$$

图 6.37: 壁面处速度型线分布图。右侧黑线的梯度并不等于左侧第一层网格蓝线的梯度，误差较大。

依据图6.37所示，红线为真实的速度分布，蓝色的实线表示 CFD 计算的速度梯度。可以看出在左侧网格比较密的情况下，分段的速度梯度可以比较好的满足真实的速度梯度。右侧当网格比较粗的情况下，速度梯度与真实的速度梯度差异较大（右侧黑线的梯度并不等于左侧第一层网格蓝线的梯度），因此并不能够合理的预测壁面剪切力。进一步的，预测的阻力系数升力系数必然不准确。理论上，在网格比较细密的情况下，壁面剪切力模型可以表示为：

$$\tau_w = \mu_l \frac{u_{\mathrm{P}}}{y_{\mathrm{P}}} \tag{6.301}$$

其中 μ_l 表示层流粘度以及湍流粘度，$u_{\mathrm{P}}, y_{\mathrm{P}}$ 表示第一层网格速度以及第一层的网格壁面距离。如果使用壁面函数，准确的壁面剪切力模型应该通过下面的公式进行计算：

$$\tau_w = \frac{u_\tau u_{\mathrm{P}}}{\ln(Ey^+)/\kappa} \tag{6.302}$$

通过(6.301)计算出来的 τ_w 是不准确的，也即：

$$\mu_l \frac{u_{\mathrm{P}}}{y_{\mathrm{P}}} \neq \frac{u_\tau u_{\mathrm{P}}}{\ln(Ey^+)/\kappa} \tag{6.303}$$

壁面函数的意义就是在壁面网格比较稀疏的情况下，通过对方程(6.303)中的壁面第一层网格的粘度进行修正，使得：

$$\tau_w = (\mu_l + \mu_t)\frac{u_{\mathrm{P}}}{y_{\mathrm{P}}} \approx \frac{u_\tau u_{\mathrm{P}}}{\ln(Ey^+)/\kappa} \tag{6.304}$$

在二维壁面处，G 可以简化为：

$$G = \nu_t \left(\nabla \mathbf{U} + \nabla \mathbf{U}^T\right) : \nabla \mathbf{U} = \nu_t \left(\begin{bmatrix} 0 & 0 \\ \frac{\partial u_1}{\partial y} & 0 \end{bmatrix} + \begin{bmatrix} 0 & \frac{\partial u_1}{\partial y} \\ 0 & 0 \end{bmatrix}\right) : \begin{bmatrix} 0 & 0 \\ \frac{\partial u_1}{\partial y} & 0 \end{bmatrix} = \nu_t \left(\frac{\partial u_1}{\partial y}\right)^2 \tag{6.297}$$

因此，壁面附近的湍流动能的产生与耗散相抵表明

$$\varepsilon = \nu_t \left(\frac{\partial u_1}{\partial y}\right)^2 \tag{6.298}$$

在这里我们认为壁面附近的 $y^+ - u^+$ 分布是以 11.5 为界限的两段线性模型（即最简单的模型）。首先考虑第一层网格的 $y^+ > 11.5$ 的情况，也即定义在 log 区域内。首先，壁面附近处的湍流动能 k_P 不需要进行改动，同时 k_P 简单的取零法向梯度边界条件即可。在 k_P 已知的情况下，可以通过下式计算壁面第一层网格的湍流动能耗散率：

$$\varepsilon_\mathrm{P} = \frac{C_\mu^{0.75} k_\mathrm{P}^{1.5}}{\kappa y_\mathrm{P}} \tag{6.305}$$

其中 C_μ 在通常取 0.09。同时，壁面附近第一层网格的湍流产生项 G_P 可以表示为

$$G_\mathrm{P} = (\nu_l + \nu_t)\frac{u_\mathrm{P}}{y_\mathrm{P}}\frac{C_\mu^{0.25} k_\mathrm{P}^{0.5}}{\kappa y_\mathrm{P}} \tag{6.306}$$

随后，将湍流产生项 G_P 代入到湍流动能方程以及湍流动能耗散率方程进行求解获得 k, ε。

另一方面，考虑第一层网格的 $y^+ < 11.5$ 的情况，在这里认为第一层网格的布置点不在 log 区域（网格很细，近似进行对壁面流动进行解析）。在这种情况下，壁面附近第一层的 G_P 近似为 0。ε_P 的值需要调用低雷诺数壁面函数来进行计算。

具体的，在附加壁面函数的 $k-\varepsilon$ 湍流模型的计算中，主要分为几步。在这里仅讨论第一层网格布置在 log 区，且比较简单的计算流程（基于摩擦速度的定义，以及不同的湍流粘度模型会有不同的计算公式）：

1. 依据流场结果，计算 y^+[103]：

$$y^+ = \frac{{C_\mu}^{0.25} y_\mathrm{P} k_\mathrm{P}^{0.5}}{\nu} \tag{6.307}$$

2. 依据 y^+ 的值，计算壁面第一层网格的 G_P、ε_P；
3. 构建 k 方程、ε 方程，参考第6.5.6节，固定第一层网格的 ε_P 值，对湍流方程进行求解；
4. 通过求解的 k, ε，更新湍流粘度；
5. 壁面一层的湍流粘度，在 $y^+ < 11.5$ 的情况下，$\mu_t = 0$，若大于 11.5[104]：

$$\nu_t = \frac{y^+ \nu \kappa}{\ln(Ey^+)} - \nu \tag{6.308}$$

需要注意的是，上面的流程计算出来的湍流粘度并不是连续的（间断点位于 $y^+ = 11.5$）[105]。在调用其他的 y^+ 计算公式的情况下，可以获得连续的湍流粘度[106]。

[103]方程 (6.307)是 OpenFOAM 中 `nutkWallFunction` 中的计算方法，其基于方程(6.299)。同理可以基于另外一个摩擦速度的定义，如方程(6.290)，获得不同的 y^+ 植入方法。比如对于 `nutUSpaldingWallFunction` 以及 `nutUWallFunction`，起始于 $\frac{1}{\kappa}\ln(Ey^+) = u^+ = \frac{u_\mathrm{P}}{u_\tau}$ 有：

$$y^+ \ln(Ey^+) - \frac{\kappa u_\mathrm{P} y_\mathrm{P}}{\nu} = 0$$

上述方程可以采用 Newton-Raphson 方法迭代求解 [107]：

$$y_{n+1}^+ = y_n^+ - \frac{f(y^+)}{f'(y+)} = y_n^+ - \frac{y_n^+ \ln(Ey_n^+) - \frac{\kappa y_\mathrm{P} U_\mathrm{P}}{\nu}}{1 + \ln(Ey_n^+)} = \frac{y_n^+ + \frac{\kappa y_\mathrm{P} U_\mathrm{P}}{\nu}}{1 + \ln(Ey_n^+)}$$

[104]同样的，这里的湍流粘度计算公式是基于方程(6.299)。在 `nutUSpaldingWallFunction` 壁面函数中，计算方法需要迭代进行。

[105]OpenFOAM 中的 `nutkWallFunction` 条件

[106]OpenFOAM 中的 `nutUSpaldingWallFunction` 条件

涉及到壁面流动的问题，主要是壁面第一层网格的高度如何获取。对于一些经典的流动情况（例如平板流动、翼形、甚至汽车外流场），可以尝试使用下面的经验方法来近似的计算第一层网格高度，来保证 y^+ 的值。

1. 首先通过七分之一定律来计算摩擦系数 $C_f = 0.0576Re^{-0.2}$（除此之外，也存在其他不同的方法来计算摩擦系数 C_f，大同小异）；
2. 通过 $\tau_w = \frac{1}{2}C_f\rho|\mathbf{U}_\infty|^2$ 计算壁面剪切力 τ_w；
3. 通过方程(6.290)计算摩擦速度；
4. 给定特定的 y^+ 值，通过方程(6.292)反推第一层网格距离 y。

附录 A　洞见偏微分方程

偏微分方程组需要能够反应真实的物理现象。物理世界中的各类守恒、扩散、对流等都要能够在偏微分方程组以及边界条件中找到答案。本章针对各种偏微分方程，洞见各种源项、扩散系数、对流系数、边界条件对结果的影响。

A.1　稳态扩散方程

如图A.1所示，本节对一个 10 网格点的一维稳态扩散方程进行离散，相应的控制方程如下[1]：

$$-\frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right)=K \tag{A.1}$$

在这个算例中,我们假定左侧为固定值边界条件 $\phi=\phi_w$,右侧为固定梯度边界条件 $\frac{\partial \phi}{\partial x}=G$。

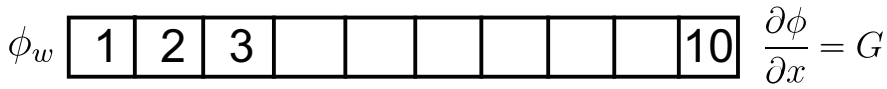


图 A.1: 一维 10 网格示意图。

且在网格中存在非均一分布的源项：

$$K_i \neq 0, i=5,6,7 \tag{A.2}$$

首先对内部网格点 $2-9$ 进行离散，有：

$$\begin{aligned}
-\phi_1+2\phi_2-\phi_3&=0,\\
-\phi_2+2\phi_3-\phi_4&=0,\\
-\phi_3+2\phi_4-\phi_5&=0,\\
-\phi_4+2\phi_5-\phi_6&=K_5\Delta x^2,\\
-\phi_5+2\phi_6-\phi_7&=K_6\Delta x^2,\\
-\phi_6+2\phi_7-\phi_8&=K_7\Delta x^2,\\
-\phi_7+2\phi_8-\phi_9&=0,\\
-\phi_8+2\phi_9-\phi_{10}&=0
\end{aligned} \tag{A.3}$$

[1]在这里并不能够严格的区分有限差分或者有限体积，因为离散后的形式是完全一样的。

对于左边的固定值边界：

$$3\phi_1 - \phi_2 = 2\phi_w \tag{A.4}$$

对于右边的固定梯度边界：

$$\phi_{10} - \phi_9 = G \tag{A.5}$$

对这些网格点组建矩阵系统有：

$$\begin{bmatrix} 3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10} \end{bmatrix} = \begin{bmatrix} 2\phi_w \\ 0 \\ 0 \\ 0 \\ K_5\Delta x^2 \\ K_6\Delta x^2 \\ K_7\Delta x^2 \\ 0 \\ 0 \\ G \end{bmatrix} \tag{A.6}$$

其解为：

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10} \end{bmatrix} = \begin{bmatrix} \phi_w + 0.5G + 0.5\Delta x^2(K_5 + K_6 + K_7) \\ \phi_w + 1.5G + 1.5\Delta x^2(K_5 + K_6 + K_7) \\ \phi_w + 2.5G + 2.5\Delta x^2(K_5 + K_6 + K_7) \\ \phi_w + 3.5G + 3.5\Delta x^2(K_5 + K_6 + K_7) \\ \phi_w + 4.5G + 4.5\Delta x^2(K_5 + K_6 + K_7) \\ \phi_w + 5.5G + 4.5\Delta x^2 K_5 + 5.5\Delta x^2(K_6 + K_7) \\ \phi_w + 6.5G + 4.5\Delta x^2 K_5 + 5.5\Delta x^2 K_6 + 6.5\Delta x^2 K_7 \\ \phi_w + 7.5G + 4.5\Delta x^2 K_5 + 5.5\Delta x^2 K_6 + 6.5\Delta x^2 K_7 \\ \phi_w + 8.5G + 4.5\Delta x^2 K_5 + 5.5\Delta x^2 K_6 + 6.5\Delta x^2 K_7 \\ \phi_w + 9.5G + 4.5\Delta x^2 K_5 + 5.5\Delta x^2 K_6 + 6.5\Delta x^2 K_7 \end{bmatrix} \tag{A.7}$$

现在对结果进行分析。如果方程(A.7)中没有源项（K 全部为 0），则结果简化为：

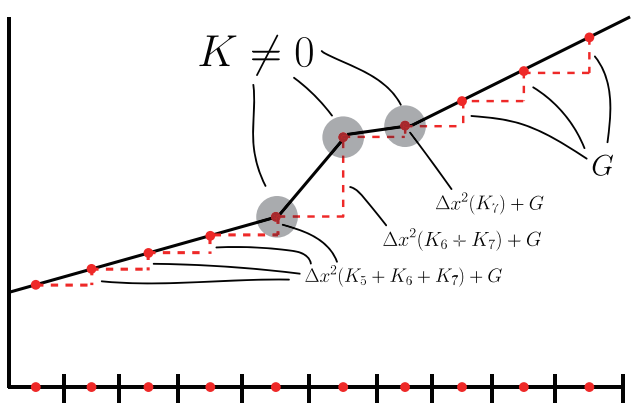


图 A.2: 一维 10 网格点计算的 ϕ。

$$\begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10} \end{bmatrix} = \begin{bmatrix} \phi_w + 0.5G \\ \phi_w + 1.5G \\ \phi_w + 2.5G \\ \phi_w + 3.5G \\ \phi_w + 4.5G \\ \phi_w + 5.5G \\ \phi_w + 6.5G \\ \phi_w + 7.5G \\ \phi_w + 8.5G \\ \phi_w + 9.5G \end{bmatrix} \tag{A.8}$$

ϕ 的结果如图A.2所示，可见：

- ϕ 在 x 方向的分布取决于右侧的固定梯度边界条件。固定梯度的值 G，决定了 ϕ 在 x 方向分布的斜率（梯度）。如果 $G = 0$（零法向梯度），则为均一分布。
- ϕ 左侧的固定值边界条件，决定了 ϕ 的参考水平。

若 K 部分不为 0，有：

- 从 $K \neq 0$ 最左侧的网格（含网格单元 5）向左起，对于网格单元 1、2、3、4、5，虽然 $K = 0$，但面上 ϕ 的梯度也收到 K 的值的影响进而发生变化，其值为 $\partial\phi/\partial x = \Delta x^2(K_5 + K_6 + K_7) + G$。
- 从 $K \neq 0$ 最右侧的网格（含网格单元 7）向右起，对于网格单元 7、8、9、10，其面上的梯度还是仅仅收到右侧固定梯度边界条件的影响：$\partial\phi/\partial x = G$。
- 对于其中的网格单元 5、6、7，其梯度发生变化，例如对于 5、6 网格单元，其面上的梯度 $\partial\phi/\partial x = \Delta x^2(K_6 + K_7) + G$。对于 6、7 网格单元，其面上的梯度 $\partial\phi/\partial x = \Delta x^2 K_7 + G$。

A.2 非均一源项

同样考虑下述控制方程：

$$-\frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) = K \tag{A.9}$$

左侧为固定值边界条件 $\phi = \phi_w$，右侧为固定梯度边界条件 $\frac{\partial\phi}{\partial x} = G$。但在网格中各点都存在非均一分布的源项：

$$K_i \neq 0, i = 1, 2, ..., 10 \tag{A.10}$$

离散后的矩阵为：

$$
\begin{aligned}
3\phi_1 - \phi_2 &= 2\phi_w + K_1\Delta x^2, \\
-\phi_1 + 2\phi_2 - \phi_3 &= K_2\Delta x^2, \\
-\phi_2 + 2\phi_3 - \phi_4 &= K_3\Delta x^2, \\
-\phi_3 + 2\phi_4 - \phi_5 &= K_4\Delta x^2, \\
-\phi_4 + 2\phi_5 - \phi_6 &= K_5\Delta x^2, \\
-\phi_5 + 2\phi_6 - \phi_7 &= K_6\Delta x^2, \\
-\phi_6 + 2\phi_7 - \phi_8 &= K_7\Delta x^2, \\
-\phi_7 + 2\phi_8 - \phi_9 &= K_8\Delta x^2, \\
-\phi_8 + 2\phi_9 - \phi_{10} &= K_9\Delta x^2, \\
\phi_{10} - \phi_9 &= G + K_{10}\Delta x^2
\end{aligned}
\tag{A.11}
$$

对这些网格点组建矩阵系统有：

$$
\begin{bmatrix}
3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
\phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10}
\end{bmatrix}
=
\begin{bmatrix}
2\phi_w + K_1\Delta x^2 \\ K_2\Delta x^2 \\ K_3\Delta x^2 \\ K_4\Delta x^2 \\ K_5\Delta x^2 \\ K_6\Delta x^2 \\ K_7\Delta x^2 \\ K_8\Delta x^2 \\ K_9\Delta x^2 \\ G + K_{10}\Delta x^2
\end{bmatrix}
\tag{A.12}
$$

矩阵存在逆矩阵，为：

$$
\begin{bmatrix}
0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\
0.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 \\
0.5 & 1.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 \\
0.5 & 1.5 & 2.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 \\
0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 4.5 & 4.5 & 4.5 & 4.5 & 4.5 \\
0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 5.5 & 5.5 & 5.5 & 5.5 \\
0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 6.5 & 6.5 & 6.5 \\
0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 7.5 & 7.5 \\
0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 8.5 \\
0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 9.5
\end{bmatrix}
\tag{A.13}
$$

其解为：

$$\begin{bmatrix}\phi_1\\ \phi_2\\ \phi_3\\ \phi_4\\ \phi_5\\ \phi_6\\ \phi_7\\ \phi_8\\ \phi_9\\ \phi_{10}\end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5\\ 0.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5 & 1.5\\ 0.5 & 1.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5 & 2.5\\ 0.5 & 1.5 & 2.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5 & 3.5\\ 0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 4.5 & 4.5 & 4.5 & 4.5 & 4.5\\ 0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 5.5 & 5.5 & 5.5 & 5.5\\ 0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 6.5 & 6.5 & 6.5\\ 0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 7.5 & 7.5\\ 0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 8.5\\ 0.5 & 1.5 & 2.5 & 3.5 & 4.5 & 5.5 & 6.5 & 7.5 & 8.5 & 9.5 \end{bmatrix} \cdot \begin{bmatrix} 2\phi_w + K_1\Delta x^2\\ K_2\Delta x^2\\ K_3\Delta x^2\\ K_4\Delta x^2\\ K_5\Delta x^2\\ K_6\Delta x^2\\ K_7\Delta x^2\\ K_8\Delta x^2\\ K_9\Delta x^2\\ G + K_{10}\Delta x^2 \end{bmatrix} \tag{A.14}$$

现在对结果进行分析。首先看左侧第一个网格，有：

$$\phi_1 = 0.5(\phi_w + \sum K_i\Delta x^2 + G) \tag{A.15}$$

如果方程(A.14)中右侧的 $G = 0$，即零梯度，有：

$$\phi_1 = 0.5(\phi_w + \sum K_i\Delta x^2) \tag{A.16}$$

$$\frac{\phi_1 - \phi_w}{\frac{1}{2}\Delta x} = \sum K_i\Delta x \tag{A.17}$$

可见，左侧边界处的梯度的值为 $(\sum K_i)\Delta x$。其表示的是：

- 对于扩散方程，左侧固定值，右侧零梯度，最左侧网格的梯度取决于所有网格点的源项。
- 方程(A.14)可以看出，对于扩散方程，如果每个网格存在源项，其会影响所有网格点的值。
- 右侧固定梯度边界条件的值 G，对所有网格点的值都存在影响。

A.3 扩散系数的影响

现在考虑另外一种方程形式：

$$-\frac{\partial}{\partial x}\left(\sigma\frac{\partial\phi}{\partial x}\right) = 0 \tag{A.18}$$

左侧为固定值边界条件 $\phi = \phi_w$，右侧为固定梯度边界条件 $\frac{\partial\phi}{\partial x} = G$。其中 σ_s 是一种非均一分布，其

$$\begin{gathered}\sigma_1 = \sigma_{ref}, \sigma_2 = \sigma_{ref}, \sigma_3 = \sigma_{ref}, \sigma_4 = \sigma_{ref}, \sigma_8 = \sigma_{ref}, \sigma_9 = \sigma_{ref}, \sigma_{10} = \sigma_{ref},\\ \sigma_5 = \sigma_6 = \sigma_7 = 0,\end{gathered} \tag{A.19}$$

后续会发现，由于扩散系数为 0，会导致方程奇异。在这里要注意的是，当扩散系数为 0 的时候，方程并无意义。但本算例仅仅为了显示数值算法的意义。

首先对内部网格点进行离散，有：

$$
\begin{aligned}
3\phi_1 - \phi_2 &= 2\phi_w, \\
-\phi_1 + 2\phi_2 - \phi_3 &= 0, \\
-\phi_2 + 2\phi_3 - \phi_4 &= 0, \\
-\phi_3 + \frac{3}{2}\phi_4 - \frac{1}{2}\phi_5 &= 0, \\
-\phi_4 + \phi_5 &= 0, \\
-\phi_5 + 2\phi_6 - \phi_7 &= 0, \\
\phi_7 - \phi_8 &= 0, \\
-\frac{1}{2}\phi_7 + \frac{3}{2}\phi_8 - \phi_9 &= 0, \\
-\phi_8 + 2\phi_9 - \phi_{10} &= 0, \\
\phi_{10} - \phi_9 &= G
\end{aligned}
\tag{A.20}
$$

对这些网格点组建矩阵系统有：

$$
\begin{bmatrix}
3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 3/2 & -1/2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1/2 & 3/2 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
\phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \\ \phi_5 \\ \phi_6 \\ \phi_7 \\ \phi_8 \\ \phi_9 \\ \phi_{10}
\end{bmatrix}
=
\begin{bmatrix}
2\phi_w \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ G
\end{bmatrix}
\tag{A.21}
$$

上述矩阵系统奇异，因此无解。在这里需要注意，对于固定扩散系数的扩散方程，左侧固定值，右侧零法向梯度边界条件，是不应该存在方程奇异的。但是针对本例，由于其中某些网格点的扩散系数为 0，会导致方程奇异，求解稳定性发生问题。这也是本算例要演示的意义。为了对其做数值求解，可以做这样一种尝试，对于 σ 近似于 0 的区域，可以将其

设置为一个非常小的值，如 1×10^{-6}。这样，离散后的矩阵为：

$$
\begin{aligned}
3\phi_1-\phi_2&=\phi_w,\\
-\phi_1+2\phi_2-\phi_3&=0,\\
-\phi_2+2\phi_3-\phi_4&=0,\\
-\phi_3+\frac{3}{2}\phi_4-\frac{1}{2}\phi_5&=0,\\
-\phi_4+1.000001\phi_5-0.000001\phi_6&=0,\\
-\phi_5+2\phi_6-\phi_7&=0,\\
-0.000001\phi_6+1.000001\phi_7-\phi_8&=0,\\
-\frac{1}{2}\phi_7+\frac{3}{2}\phi_8-\phi_9&=0,\\
-\phi_8+2\phi_9-\phi_{10}&=0,\\
\phi_{10}-\phi_9&=G
\end{aligned}
\tag{A.22}
$$

对这些网格点组建矩阵系统有：

$$
\begin{bmatrix}
3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
-1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & -1 & 3/2 & -1/2 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -1 & 1.000001 & -0.000001 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & -0.000001 & 1.000001 & -1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -1/2 & 3/2 & -1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{bmatrix}
\cdot
\begin{bmatrix}
\phi_1\\ \phi_2\\ \phi_3\\ \phi_4\\ \phi_5\\ \phi_6\\ \phi_7\\ \phi_8\\ \phi_9\\ \phi_{10}
\end{bmatrix}
=
\begin{bmatrix}
2\phi_w\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0\\ G
\end{bmatrix}
\tag{A.23}
$$

其解为：

$$
\begin{bmatrix}
\phi_1\\ \phi_2\\ \phi_3\\ \phi_4\\ \phi_5\\ \phi_6\\ \phi_7\\ \phi_8\\ \phi_9\\ \phi_{10}
\end{bmatrix}
=
\begin{bmatrix}
\phi_w+0.5G\\
0.5\phi_w+1.5G\\
0.5\phi_w+2.5G\\
0.5\phi_w+3.5G\\
0.5\phi_w+5.5G\\
0.5\phi_w+2000005.5G\\
0.5\phi_w+4000005.5G\\
0.5\phi_w+4000007.5G\\
0.5\phi_w+4000008.5G\\
0.5\phi_w+4000009.5G
\end{bmatrix}
\tag{A.24}
$$

在这里需要注意的是，一个原本奇异的矩阵系统，通过将 0 设定为一个非常小的数，暂且称之为稳定性因子，整个系统可以求解。但是，同时可以发现，矩阵的解，取决于用户自行设定的稳定性因子。本质上，这种方程是不现实的。实际操作中，最好通过多域求解来进行。

A.4 全局守恒与边界条件限定

本节考虑这样一个瞬态扩散方程：

$$\frac{\partial \phi}{\partial t} - \frac{\partial}{\partial x}\left(\sigma \frac{\partial \phi}{\partial x}\right) = S \tag{A.25}$$

其中 $S < 0$。方程(A.25)可用于描述固体导热，也可以用于描述对流传输。考虑简单的固体导热，针对一维网格，若左侧给定固定值边界条件（假设为左侧温度为固定值），中间某些网格存在一个负的源项，在计算一段时间后，如图A.4所示，方程(A.25)预测的解会逐步趋向于稳定不变。

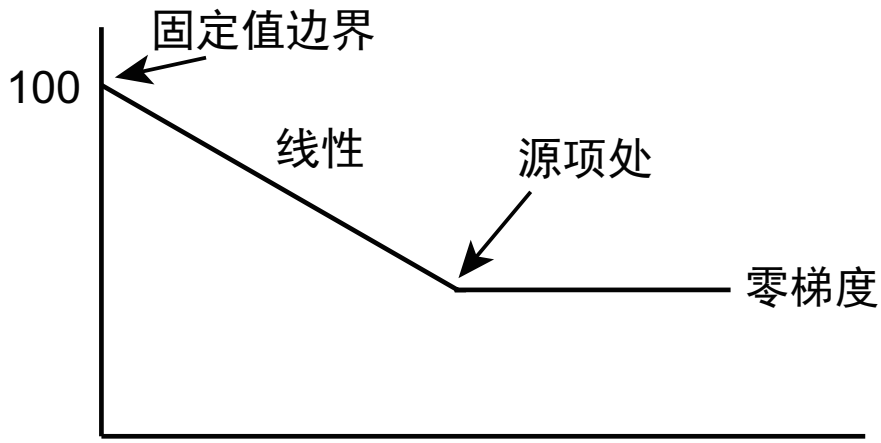


图 A.3: 方程(A.25)的解的示意图。

问题是图A.4中的解的线性部分下降的斜率是多少？很明显。如果 $S = 0$，那么这部分区域的斜率为零，整场呈现均一分布。因此这个斜率必然与 S 的大小有关。首先理解方程的解的左半部分为什么是线性的。这是因为在这部分并没有源项。在最终稳态的情况下，方程简化为

$$-\frac{\partial}{\partial x}\left(\sigma \frac{\partial \phi}{\partial x}\right) = 0 \tag{A.26}$$

其表示在网格单元面上任意一处的斜率相同。因此解为线性下降的。现考虑图A.4所示的考虑存在源项 S 的三网格点变量 ϕ 分布图。最终稳态的情况，$\frac{\partial \phi}{\partial t} = 0$。因此在稳态时，当前的网格的控制方程简化为

$$-\frac{\partial}{\partial x}\left(\sigma \frac{\partial \phi}{\partial x}\right) = S \tag{A.27}$$

其离散后为：

$$-\left(\left(\frac{\partial \phi}{\partial x}\right)_C - \left(\frac{\partial \phi}{\partial x}\right)_B\right) = \frac{S\Delta x}{\sigma} \tag{A.28}$$

由于在存在源项 S 的网格右侧，为零梯度边界条件，因此 C 面上的梯度 $\frac{\partial \phi}{\partial x}_C = 0$。方程可以简化为：

$$\left(\frac{\partial \phi}{\partial x}\right)_B = \frac{S\Delta x}{\sigma} \tag{A.29}$$

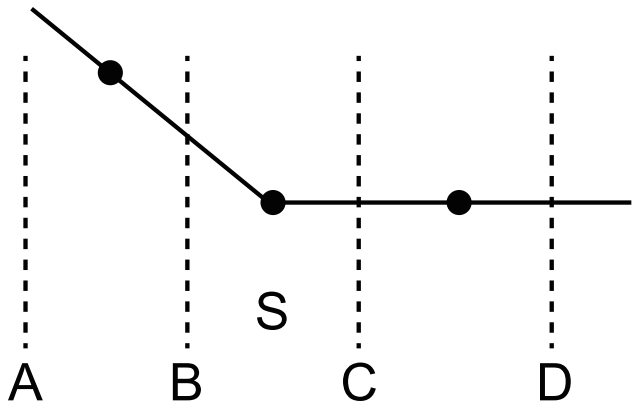


图 A.4: 存在源项 S 的三网格点变量 ϕ 分布。

很明显，$\left(\frac{\partial\phi}{\partial x}\right)_B$ 即为存在源项 S 的网格左侧的梯度，也即斜率。进一步的，源项处稳定的值也可以由此推出。

那么是否可以说，所有的瞬态方程，都可以把时间项去掉，来直接获得稳态解呢？未必。这是因为瞬态方程与稳态方程的方程特征不一样，瞬态方程由于存在时间项的原因，可以保证对角占优特性。把这个对求解有利的因素去掉，会导致求解发散。下面可以通过另外一套边界条件进行分析，在使用这一套边界条件的情况下，瞬态方程存在解，但是稳态方程在某些情况下会导致发散（这与矩阵迭代求解器以及预条件有关）。因此，瞬态方程的按照时间步进的解，不能简单通过将时间项去掉简化为稳态方程。

现在假定网格左侧为固定梯度边界条件 $\frac{\partial\phi}{\partial x}=G$，右侧同样为零梯度边界条件，中间存在一个负的源项。随后会发现，这一套边界条件，如果满足某些特殊的限定性条件之后，变量 ϕ 的结果可以趋向于稳态。若不满足，则会随着时间一直增加或减小。对于后者，可以认为是一种全局的不守恒。即持续的有变量的进入或流出导致变量在计算域内堆积（但斜率相同）。若希望 ϕ 的结果可以趋向于稳态，时间项最后的贡献为 0，因此求解域内的线性区域，满足方程(A.29)。在这里要注意，由于方程左侧给定的是固定梯度边界条件，因此这个固定梯度的值，也需要满足方程(A.29)。只有在这种情况下，可以全局守恒。

现在考虑三维的情况。在三维情况下，对于下面的瞬态扩散方程：

$$\frac{\partial\phi}{\partial t}-\nabla\cdot(D\nabla\phi)=S \tag{A.30}$$

若给定固定梯度边界条件，同时存在一个负的沉降项（$S<0$）。同样也可以存在一种全局守恒的情况。同理一维情况，三维情况的全局守恒的边界条件限定性限制为：

$$(\nabla\phi)_f\cdot\mathbf{n}_f|\mathbf{S}_f|=\frac{S\Delta V}{D} \tag{A.31}$$

进一步的，三维情况通常边界处存在多个面，同时源项的分布可以布置于一定的网格区域。

在这种情况下，全局守恒的边界条件限定性限制为：

$$\sum (\nabla\phi)_{f,facei} \cdot \mathbf{n}_{f,facei} |\mathbf{S}_{f,facei}| = \sum \frac{S\Delta V_{celli}}{D} \tag{A.32}$$

A.5 OpenFOAM 离散的矩阵系数

首先我们来看内部网格点的矩阵系数。在 CFD: 不可压 + 稳态算法中，有针对动量离散后的方程形式为

$$\frac{1}{V_\mathrm{P}}\left(\sum \frac{F_f^n}{2} + \sum \nu \frac{|\mathbf{S}_f|}{|\mathbf{d}|}\right)\mathbf{U}_\mathrm{P}^* = -\frac{1}{V_\mathrm{P}}\sum\left(\frac{F_f^n}{2} - \nu\frac{|\mathbf{S}_f|}{|\mathbf{d}|}\right)\mathbf{U}_\mathrm{N}^* - \frac{1}{V_\mathrm{P}}\sum p_f^n \mathbf{S}_f \tag{A.33}$$

略去压力项、对流项有：

$$\frac{1}{V_\mathrm{P}}\left(\sum \nu \frac{|\mathbf{S}_f|}{|\mathbf{d}|}\right)\mathbf{U}_\mathrm{P}^* = \frac{1}{V_\mathrm{P}}\sum\left(\nu\frac{|\mathbf{S}_f|}{|\mathbf{d}|}\mathbf{U}_\mathrm{N}^*\right) \tag{A.34}$$

假定面网格矢量大小为 1，距离为 1，粘度为 1。同时针对一维问题的内网格，针对第 2 个网格点存在两个网格面，则有：

$$\frac{1}{1}\left(1\frac{1}{1} + 1\frac{1}{1}\right)\mathbf{U}_2^* = \frac{1}{1}\left(1\frac{1}{1}\mathbf{U}_1^*\right) + \frac{1}{1}\left(1\frac{1}{1}\mathbf{U}_3^*\right) \tag{A.35}$$

也即：

$$2\mathbf{U}_2^* - \mathbf{U}_1^* - \mathbf{U}_3^* = 0 \tag{A.36}$$

同样针对第 3 个网格点存在同样的方程：

$$2\mathbf{U}_3^* - \mathbf{U}_2^* - \mathbf{U}_4^* = 0 \tag{A.37}$$

针对 4 网格系统，其内部点的离散矩阵系数为：

$$\begin{bmatrix} \times & \times & \times & \times \\ -1 & 2 & -1 & \times \\ \times & -1 & 2 & -1 \\ \times & \times & \times & \times \end{bmatrix} \tag{A.38}$$

为方便后面讨论，对其进行注释：

$$\begin{bmatrix} \mathrm{diag} & \mathrm{upper} & \times & \times \\ -1,\mathrm{lower} & 2.\mathrm{diag} & -1,\mathrm{upper} & \times \\ \times & -1,\mathrm{lower} & 2,\mathrm{diag} & -1,\mathrm{upper} \\ \times & \times & \mathrm{lower} & \mathrm{diag} \end{bmatrix} \tag{A.39}$$

其离散后输出的矩阵相关系数为：

```
diag(): 4(1 2 2 1)   //其中的2 2对应矩阵对角线系数，第一个1和最后一个1取决于边界网格，在这里先不做分析
upper(): 3{-1}  //一共3个-1，其中2个-1对应上对角线系数。如果更换离散格式，可能存在更多的数值
lower(): 3{-1}  //一共3个-1，其中2个-1对应下对角线系数。如果更换离散格式，可能存在更多的数值
source(): 4{(-0 -0 -0)}  //本算例没有源项
upperAddr(): 3(1 2 3)  //每个内部面对应的nei网格单元编号。如果更换离散格式，可能存在更多的数值
lowerAddr(): 3(0 1 2)  //每个内部面对应的own网格单元编号。如果更换离散格式，可能存在更多的数值
D(): 4(3 2 2 1)  //diag()的贡献同时附加边界影响
A():   //D()除以网格体积
    dimensions      [0 0 -1 0 0 0 0];
    internalField   nonuniform List<scalar> 4(3 2 2 1);
    boundaryField{...}
```

下面来看单独对流项的影响，针对方程(A.33)，略去压力项、扩散项有：

$$\frac{1}{V_{\mathrm{P}}}\left(\sum\frac{F_f^n}{2}\right)\mathbf{U}_{\mathrm{P}}^*=-\frac{1}{V_{\mathrm{P}}}\sum\left(\frac{F_f^n}{2}\mathbf{U}_{\mathrm{N}}^*\right) \tag{A.40}$$

假定内部网格速度都是 1，同时使用中心格式，针对网格单元 2 有：

$$\frac{1}{1}\left(\frac{1}{2}-\frac{1}{2}\right)\mathbf{U}_2^*=-\frac{1}{1}\left(\frac{1}{2}\mathbf{U}_3^*\right)-\frac{1}{1}\left(\frac{-1}{2}\mathbf{U}_1^*\right) \tag{A.41}$$

整理后有：

$$0\cdot\mathbf{U}_2^*=-0.5\mathbf{U}_3^*+0.5\mathbf{U}_1^* \tag{A.42}$$

同理，针对网格点 3 有：

$$0\cdot\mathbf{U}_3^*=-0.5\mathbf{U}_4^*+0.5\mathbf{U}_2^* \tag{A.43}$$

其内部点的离散矩阵系数为：

$$\begin{bmatrix}\times & \times & \times & \times\\ -0.5 & 0 & 0.5 & \times\\ \times & -0.5 & 0 & 0.5\\ \times & \times & \times & \times\end{bmatrix} \tag{A.44}$$

其离散后输出的矩阵相关系数为：

```
diag(): 4(0.5 0 0 -0.5)
upper(): 3{0.5}
lower(): 3{-0.5}
source(): 4{(0 0 0)}
```

附录 B　新版 OpenFOAM 之模块化求解器

B.1　版本适配

OpenFOAM 基金会版本的 OpenFOAM 代码，自从版本 11 开始，就进行了模块化的处理。所谓代码模块化，表示的是之前的求解器的核心算法都是植入到了主程序。代码模块化指的是，各类核心算法都被植入到了库文件中。主程序主要保留一个 foamRun，仅仅是为了调用各种算法模块。对于 CFD 应用者来说，OpenFOAM 的模块化并不会导致使用方较大的改变。对于 CFD 算法开发人员来说，需要适配新的 OpenFOAM 的代码模块化开发思路，也即把核心算法开发到库函数中而不是主程序中。但是即使是模块化的 OpenFOAM 代码，算法开发人员同样也可以把核心算法植入到主程序中。对于模块化，更像是“要遵守，就遵守，不遵守，可以自由发挥”。

不管是 CFD 应用，还是 CFD 开发，在初次接触新版 OpenFOAM[1]的时候，需要将模块化代码与 OpenFOAM 之前的求解器相对应。比如之前的 OpenFOAM 密度基求解器 rhoCentralFoam，对应新版的 shockFluid 模块。图B.1中将新版 OpenFOAM 模块与老版本的 OpenFOAM 求解器相对应。

B.2　代码实例

自从 2004 年 OpenFOAM 开源之后的 20 年，OpenFOAM 的求解器代码均不是模块化的。因此关于这部分的介绍非常多，本节重复介绍。仅介绍 OpenFOAM 新版的模块化代码流程。

OpenFOAM 模块化的求解器主要使用 foamRun 来进行计算。foamRun 可以理解为老版本 OpenFOAM 的主程序。其中的代码如下：

```
while (pimple.loop())
{
    //将网格动起来的函数，不需要动网格可忽略
    solver.moveMesh();

```

[1]这里的新版指的是基金会版本的 11 版本之后

图 B.1: OpenFOAM 模块与老版本的 OpenFOAM 求解器。

```
    //动网格相关函数，主要是实现correctPhi的功能，不需要动网格可忽略
    solver.motionCorrector();

    //一些源项，无源项可忽略
    solver.fvModels().correct();

    //修正粘度，例如一些非牛顿流体模型
    solver.prePredictor();

    //动量方程
    solver.momentumPredictor();

    //能量方程
    solver.thermophysicalPredictor();

    //压力方程
    solver.pressureCorrector();

    //求解湍流变量
    solver.postCorrector();
}
```

相比较之前的 OpenFOAM 主程序。相应的算法在上面的代码中都没有体现，而是在具体的某一个库中。因此所有的算例都可以通过 foamRun 并结合某个具体的库来运行。具体的流

程代码注释已经给出。下面看一个具体的 incompressibleFluid 实例。以上文中的 moveMesh 函数以及 motionCorrector 函数都定义在 moveMesh.C 中。moveMesh 函数调用下面的代码：

```
void Foam::solvers::incompressibleFluid::moveMesh()
{
    if (pimple.firstIter() || pimple.moveMeshOuterCorrectors())
    {
        // Move the mesh
        mesh_.move();
    }
}
```

类似的，motionCorrector 函数主要调用 correctPhi 函数。可以看出，其即为简单的 move 函数，实现网格运动的功能。同样的，老版本求解器的 pEqn.H，在模块化求解器中，其实是植入到了 pressureCorrector 函数中。

参考文献

[1] About fvc::reconstruct (1). https://www.jianguoyun.com/p/DeD7OAkQ9s3ZBhiAtPQC. Accessed: 2020-03-12.

[2] About fvc::reconstruct (2). https://www.jianguoyun.com/p/DVqidAUQ9s3ZBhiBtPQC. Accessed: 2020-03-12.

[3] Change of variables in multiple integrals (jacobians). https://math.libretexts.org/Courses/Monroe_Community_College/MTH_212_Calculus_III/Chapter_14%3A_Multiple_Integration/14.7%3A_Change_of_Variables_in_Multiple_Integrals_(Jacobians). Accessed: 2022-09-20.

[4] Coriolis-force. https://www.britannica.com/science/Coriolis-force. Accessed: 2023-02-21.

[5] The fourier coefficients. http://www.thefouriertransform.com/series/coefficients.php. Accessed: 2020-12-05.

[6] Is there a way to order nodes in a mesh to efficiently compute sparse matrices? https://www.cfd-online.com/Forums/main/196509-there-way-order-nodes-mesh-efficiently-compute-sparse-matrices.html. Accessed: 2022-01-03.

[7] Numerator layout. https://en.wikipedia.org/wiki/Matrix_calculus#Layout_conventions. Accessed: 2022-06-15.

[8] Reynolds stress models. https://www.cfd-online.com/Forums/main/1779-reynolds-stress-models.html. Accessed: 2021-08-03.

[9] Rotating frame of reference. https://www.physics-in-a-nutshell.com/article/29/rotating-frame-of-reference#bibliography. Accessed: 2021-04-13.

[10] Turbulent boundary layer (dns). https://www.youtube.com/watch?v=Wr984EOmNaY&t=12s. Accessed: 2020-11-05.

[11] Tutorial: Bandwidth reduction–the cuthill–mckee algorithm. https://ciprian-zavoianu.blogspot.com/2009/01/project-bandwidth-reduction.html. Accessed: 2022-01-03.

[12] E. Ajuria Illarramendi, A. Alguacil, M. Bauerheim, A. Misdariis, B. Cuenot, and E. Benazera. Towards an hybrid computational strategy based on deep learning for incompressible flows. In AIAA Aviation 2020 Forum, page 3058, 2020.

[13] A. Al Safwan, C. Song, and U.B. Waheed. Is it time to swish? Comparing activation functions in solving the Helmholtz equation using PINNS. In 82nd EAGE Annual Conference & Exhibition, volume 2021, pages 1–5. European Association of Geoscientists & Engineers, 2021.

[14] D.D. Apsley and I.P. Castro. A limited-length-scale k-ε model for the neutral and stably-stratified atmospheric boundary layer. Boundary-layer meteorology, 83:75–98, 1997.

[15] J. Bardina, J. Ferziger, and W.C. Reynolds. Improved subgrid-scale models for large-eddy simulation. In 13th fluid and plasmadynamics conference, page 1357, 1980.

[16] H. Baty and L. Baty. Solving differential equations using physics informed deep learning: a hand-on tutorial with benchmark tests. arXiv preprint arXiv:2302.12260, 2023.

[17] A. Beck, D. Flad, and C. Munz. Deep neural networks for data-driven LES closure models. Journal of Computational Physics, 398:108910, 2019.

[18] R. Beetstra, M.A. van der Hoef, and J.A.M. Kuipers. Drag force of intermediate reynolds number flow past mono-and bidisperse arrays of spheres. AIChE journal, 53:489–501, 2007.

[19] Afshar Y. Pan S. Bhatnagar, S. Prediction of aerodynamic flow fields using convolutional neural networks. Comput Mech, 64, 2019.

[20] G.A. Bird. Molecular gas dynamics. NASA STI/Recon Technical Report A, 76, 1976.

[21] J.P. Boris and D.L. Book. Flux-corrected transport. I. SHASTA, a fluid transport algorithm that works. Journal of computational physics, 11:38–69, 1973.

[22] J.P. Boris, F.F. Grinstein, E.S. Oran, and R.L. Kolbe. New insights into large eddy simulation. Fluid dynamics research, 10:199, 1992.

[23] S.T. Bose, P. Moin, and D. You. Grid-independent large-eddy simulation using explicit filtering. Physics of Fluids, 22:105103, 2010.

[24] D. Bouris and G. Bergeles. 2D LES of vortex shedding from a square cylinder. Journal of Wind Engineering and Industrial Aerodynamics, 80:31–46, 1999.

[25] M. Breuer. Large eddy simulation of the subcritical flow past a circular cylinder: numerical and modeling aspects. International Journal For Numerical Methods in Fluids, 28:1281–1302, 1998.

[26] W.L. Briggs, V.E. Henson, and S.F. McCormick. A multigrid tutorial. SIAM, 2000.

[27] S.L. Brunton and J.N. Kutz. Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press, 2022.

[28] W. Cao, J. Song, and W. Zhang. A solver for subsonic flow around airfoils based on physics-informed neural networks and mesh transformation. Physics of Fluids, 36, 2024.

[29] J. Capecelatro, O. Desjardins, and R.O. Fox. Numerical study of collisional particle dynamics in cluster-induced turbulence. Journal of Fluid Mechanics, 747, 2014.

[30] F.J. Carrillo and I.C. Bourg. A Darcy-Brinkman-Biot approach to modeling the hydrology and mechanics of porous media containing macropores and deformable microporous regions. Water Resources Research, 55:8096–8121, 2019.

[31] J. Cen and Q. Zou. Deep finite volume method for partial differential equations. Journal of Computational Physics, page 113307, 2024.

[32] Christophe Chalons, Damien Kah, and Marc Massot. Beyond pressureless gas dynamics: quadrature-based velocity moment models. arXiv preprint arXiv:1011.2974, 2010.

[33] X. Chen, L. Wang, F. Meng, and Z. Luo. Physics-informed deep learning for modelling particle aggregation and breakage processes. Chemical Engineering Journal, 426:131220, 2021.

[34] J. Cho, H. Kim, A. Gebreselassie, and D. Shin. Deep neural network and random forest classifier for source tracking of chemical leaks using fence monitoring data. Journal of Loss Prevention in the Process Industries, 56:548–558, 2018.

[35] P.Y. Chou. On velocity correlations and the solutions of the equations of turbulent fluctuation. Quarterly of Applied Mathematics, 3:38–54, 1945.

[36] J. Costa. Atmospheric flow over forested and non-forested complex terrain. PhD thesis, Universidade do Porto (Portugal), 2007.

[37] S. Cuomo, V.S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. Journal of Scientific Computing, 92:88, 2022.

[38] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In Proceedings of the 1969 24th national conference, pages 157–172, 1969.

[39] M.S. Darwish and F. Moukalled. Tvd schemes for unstructured grids. International Journal of heat and mass transfer, 46:599–611, 2003.

[40] J.J. Derksen and S. Sundaresan. Direct numerical simulations of dense suspensions: wave instabilities in liquid-fluidized beds. Journal of Fluid Mechanics, 587:303–336, 2007.

[41] J. Ding and D. Gidaspow. A bubbling fluidization model using kinetic theory of granular flow. AIChE Journal, 36:523–538, 1990.

[42] M.S. Dodd and A. Ferrante. On the interaction of Taylor length scale size droplets and isotropic turbulence. Journal of Fluid Mechanics, 806:356–412, 2016.

[43] D.A. Drew. Mathematical modeling of two–phase flow. Annual Review of Fluid Mechanics, 15:261–291, 1982.

[44] D.V. Dung, N.D. Song, P.S. Palar, and L.R. Zuhal. On the choice of activation functions in physics-informed neural network for solving incompressible fluid flows. In AIAA SCITECH 2023 Forum, page 1803, 2023.

[45] K. Duraisamy and P. Durbin. Transition modeling using data driven approaches. In Proceedings of the Summer Program, page 427, 2014.

[46] P.A. Durbin. Separated flow computations with the k-epsilon-v-squared model. AIAA journal, 33:659–664, 1995.

[47] V. Dwivedi and B. Srinivasan. Physics informed extreme learning machine (pielm)–a rapid method for the numerical solution of partial differential equations. Neurocomputing, 391:96–118, 2020.

[48] N. Epstein. On tortuosity and the tortuosity factor in flow and diffusion through porous media. Chemical Engineering Science, 44:777–779, 1989.

[49] J.L. Favero, A.R. Secchi, N.S.M. Cardozo, and H. Jasak. Viscoelastic fluid analysis in internal and in free surface flows using the software OpenFOAM. Computers & Chemical Engineering, 34:1984–1993, 2010.

[50] C.L. Fefferman. Existence and smoothness of the Navier-Stokes equation. The millennium prize problems, 57:67, 2000.

[51] J.H. Ferziger, M. Perić, and R.L. Street. Computational methods for fluid dynamics, volume 3. Springer, 2002.

[52] R.O. Fox. Large-eddy-simulation tools for multiphase flows. Annual Review of Fluid Mechanics, 44:47–76, 2012.

[53] K. Fukami, K. Fukagata, and K. Taira. Super-resolution reconstruction of turbulent flows with machine learning. Journal of Fluid Mechanics, 870:106–120, 2019.

[54] C. Fureby and F.F. Grinstein. Monotonically integrated large eddy simulation of free shear flows. AIAA Journal, 37:544–556, 1999.

[55] M. Gamahara and Y. Hattori. Searching for turbulence models by artificial neural network. Physical Review Fluids, 2:054604, 2017.

[56] Z. Gao, D. Li, A. Buffo, W. Podgórska, and D.L. Marchisio. Simulation of droplet breakage in turbulent liquid–liquid dispersions with CFD-PBM: Comparison of breakage kernels. Chemical Engineering Science, 142:277–288, 2016.

[57] M. Germano, U. Piomelli, P. Moin, and W.H. Cabot. A dynamic subgrid-scale eddy viscosity model. Physics of Fluids A: Fluid Dynamics, 3:1760–1765, 1991.

[58] S. Ghosal and P. Moin. The basic equations for the large eddy simulation of turbulent flows in complex geometry. Journal of Computational Physics, 118:24–37, 1995.

[59] S. Godunov and I. Bohachevsky. Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics. Matematičeskij sbornik, 47:271–306, 1959.

[60] B. Goyeau, D. Lhuillier, D. Gobin, and M.G. Velarde. Momentum transport at a fluid–porous interface. International Journal of Heat and Mass Transfer, 46:4071–4081, 2003.

[61] C.J. Greenshields and G. Weller. Notes on computational fluid dynamics: General principles. John Wiley & Sons, 2022.

[62] G.W. Griffiths. Numerical Analysis Using R. Cambridge University Press, 2016.

[63] X. Guo, W. Li, and F. Iorio. Convolutional neural networks for steady flow approximation. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 481–490, 2016.

[64] L. Han, S. Gong, Y. Li, Q. Ai, H. Luo, Z. Liu, and Y. Liu. A novel theoretical model of breakage rate and daughter size distribution for droplet in turbulent flows. Chemical Engineering Science, 102:186–199, 2013.

[65] D.M. Hargreaves and N.G. Wright. On the use of the k–ε model in commercial CFD software to model the neutral atmospheric boundary layer. Journal of wind engineering and industrial aerodynamics, 95:355–369, 2007.

[66] K. Hasegawa, K. Fukami, T. Murata, and K. Fukagata. CNN-LSTM based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different Reynolds numbers. Fluid Dynamics Research, 52:065501, 2020.

[67] L. He and D.K. Tafti. A supervised machine learning approach for predicting variable drag forces on spherical particles in suspension. Powder technology, 345:379–389, 2019.

[68] A. Hellsten. Some improvements in Menter's k-omega SST turbulence model. In 29th AIAA, Fluid Dynamics Conference, page 2554, 1998.

[69] C.W. Hirt and B.D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. Journal of computational physics, 39:201–225, 1981.

[70] P. Horgue, C. Soulaine, J. Franc, R. Guibert, and G. Debenest. An open-source toolbox for multiphase flow in porous media. Computer Physics Communications, 187:217–226, 2015.

[71] Y. Hu, Y. Li, and C. Wu. Rayleigh-Bénard convection of cold water near its density maximum in a cubical cavity. Physics of Fluids, 27, 2015.

[72] M. Ishii and S. Kim. Development of one-group and two-group interfacial area transport equation. Nuclear Science and Engineering, 146:257–273, 2004.

[73] H. Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. Imperial College London (University of London), 1996.

[74] W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng. Stiff-pinn: Physics-informed neural network for stiff chemical kinetics. The Journal of Physical Chemistry A, 125:8098–8106, 2021.

[75] G. Karniadakis, Z. Wang, and X. Meng. Solution multiplicity and effects of data and eddy viscosity on Navier-Stokes solutions inferred by physics-informed neural networks. Bulletin of the American Physical Society, 2023.

[76] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. Nature Reviews Physics, 3:422–440, 2021.

[77] S.E. Kim. Large eddy simulation using an unstructured mesh based finite-volume solver. In 34th AIAA fluid dynamics conference and exhibit, page 2548, 2004.

[78] Tilman Koblitz. CFD Modeling of non-neutral atmospheric boundary layer conditions. DTU Wind Energy, 2013.

[79] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. Proceedings of the National Academy of Sciences, 118:e2101784118, 2021.

[80] C. Kong, J. Chang, Z. Wang, Y. Li, and W. Bao. Data-driven super-resolution reconstruction of supersonic flow field by convolutional neural networks. AIP Advances, 11, 2021.

[81] J. Kutz, S.L. Brunton, B.W. Brunton, and J.L. Proctor. Dynamic mode decomposition: data-driven modeling of complex systems. SIAM, 2016.

[82] O.A. Ladyzhenskaya. The mathematical theory of viscous incompressible flow. Gordon & Breach, 1969.

[83] P.L.C. Lage. On the representation of QMOM as a weighted-residual method–The dual-quadrature method of generalized moments. Computers and Chemical Engineering, 35:2186–2203, 2011.

[84] C.K.G. Lam and K. Bremhorst. A modified form of the k-ε model for predicting wall turbulence. Journal of Fluid Engineering, 103:456–460, 1981.

[85] B.E. Launder, G.J. Reece, and W. Rodi. Progress in the development of a reynolds-stress turbulence closure. Journal of fluid mechanics, 68:537–566, 1975.

[86] B.E. Launder and B.I. Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. Letters In Heat And Mass Transfer, 1:131–137, 1974.

[87] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. In Numerical prediction of flow, heat transfer, turbulence and combustion, pages 96–116. Elsevier, 1983.

[88] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1:541–551, 1989.

[89] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86:2278–2324, 1998.

[90] S. Lee and D. You. Data-driven prediction of unsteady flow over a circular cylinder using deep learning. Journal of Fluid Mechanics, 879:217–254, 2019.

[91] Jean Leray. Sur le mouvement d'un liquide visqueux emplissant l'espace. Acta mathematica, 63:193–248, 1934.

[92] R.J. LeVeque. Finite volume methods for hyperbolic problems. Cambridge university press, 2002.

[93] D. Li, A. Buffo, W. Podgórska, D.L. Marchisio, and Z. Gao. Investigation of droplet breakup in liquid-liquid dispersions by CFD-PBM simulations: The influence of the surfactant type. Chinese Journal of Chemical Engineering, 25:1369–1380, 2017.

[94] D. Li, Z. Gao, A. Buffo, W. Podgorska, and D. Marchisio. Droplet breakage and coalescence in liquid–liquid dispersions: Comparison of different kernels with EQMOM and QMOM. AIChE Journal, 63:2293–2311, 2017.

[95] D. Li, Z. Li, and Z. Gao. Compressibility induced bubble size variation in bubble column reactors: Simulations by the CFD–PBE. Chinese Journal of Chemical Engineering, 26(10):2009–2013, 2018.

[96] D. Li, Z. Li, and Z. Gao. Quadrature-based moment methods for the population balance equation: An algorithm review. Chinese Journal of Chemical Engineering, 27:483–500, 2019.

[97] D. Li and D. Marchisio. Implementation of CHyQMOM in OpenFOAM for the simulation of non-equilibrium gas-particle flows under one-way and two-way coupling. Powder Technology, 396:765–784, 2022.

[98] D. Li, D. Marchisio, C. Hasse, and D. Lucas. Comparison of Eulerian QBMM and classical Eulerian–Eulerian method for the simulation of polydisperse bubbly flows. AIChE Journal, 65:e16732, 2019.

[99] D. Li, D. Marchisio, C. Hasse, and D. Lucas. twoWayGPBEFoam: An open-source Eulerian QBMM solver for monokinetic bubbly flows. Computer Physics Communications, page in press, 2019.

[100] D. Li, Y. Wei, and D. Marchisio. QEEFoam: A Quasi-Eulerian-Eulerian model for polydisperse turbulent gas-liquid flows. Implementation in OpenFOAM, verification and validation. International Journal of Multiphase Flow, 136:103544, 2021.

[101] J. Li and W. Huang. From multiscale to mesoscience: addressing mesoscales in mesoregimes of different levels. Annual review of chemical and biomolecular engineering, 9:41–60, 2018.

[102] F.S. Lien and G. Kalitzin. Computations of transonic flow with the v2–f turbulence model. International Journal of Heat and Fluid Flow, 22:53–61, 2001.

[103] M.J. Lighthill. On sound generated aerodynamically I. General theory. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 211:564–587, 1952.

[104] D.K. Lilly. A proposed modification of the Germano subgrid-scale closure method. Physics of Fluids A: Fluid Dynamics, 4:633–635, 1992.

[105] J. Ling, A. Kurzawski, and J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. Journal of Fluid Mechanics, 807:155–166, 2016.

[106] C. Liu, Y. Gao, S. Tian, and X. Dong. Rorte-A new vortex vector definition and vorticity tensor and vector decompositions. Physics of Fluids, 30, 2018.

[107] F. Liu. A thorough description of how wall functions are implemented in OpenFOAM. Proceedings of CFD with OpenSource Software, 34, 2016.

[108] Z. Liu and B. Li. Scale-adaptive analysis of euler-euler large eddy simulation for laboratory scale dispersed bubbly flows. Chemical Engineering Journal, 338:465–477, 2018.

[109] J.L. Lumley. The structure of inhomogeneous turbulent flows. Atmospheric turbulence and radio wave propagation, pages 166–178, 1967.

[110] N.K. Madavan, C.L. Merkle, and S. Deutsch. Numerical investigations into the mechanisms of microbubble drag reduction. Journal of Fluids Engineering, 107:370–377, 1985.

[111] J.F. Manwell, J.G. McGowan, and A.L. Rogers. Wind energy explained: theory, design and application. John Wiley & Sons, 2010.

[112] D.L. Marchisio and R.O. Fox. Computational models for polydisperse particulate and multiphase systems. Cambridge University Press, 2013.

[113] K. Masatsuka. I do Like CFD, vol. 1, volume 1. Lulu. com, 2009.

[114] R. Maulik, O. San, A. Rasheed, and P. Vedula. Subgrid modelling for two-dimensional turbulence using neural networks. Journal of Fluid Mechanics, 858:122–144, 2019.

[115] R. Maulik, H. Sharma, S. Patel, B. Lusch, and El. Jennings. A turbulent eddy-viscosity surrogate modeling framework for reynolds-averaged navier-stokes simulations. Computers & Fluids, 227:104777, 2021.

[116] M. Meier, G. Yadigaroglu, and B. Smith. A novel technique for including surface tension in PLIC-VOF methods. European Journal of Mechanics-B/Fluids, 21:61–73, 2002.

[117] C. Meneveau, T.S. Lund, and W.H. Cabot. A lagrangian dynamic subgrid-scale model of turbulence. Journal of Fluid Mechanics, 319:353–385, 1996.

[118] F. Menter. Zonal two equation k-w turbulence models for aerodynamic flows. In 23rd fluid dynamics, plasmadynamics, and lasers conference, page 2906, 1993.

[119] F.R. Menter, M. Kuntz, and R. Langtry. Ten years of industrial experience with the SST turbulence model. Turbulence, Heat And Mass Transfer, 4:625–632, 2003.

[120] M. Milano and P. Koumoutsakos. Neural network modeling for near wall turbulent flow. Journal of Computational Physics, 182:1–26, 2002.

[121] M. Milelli, B.L. Smith, and D. Lakehal. Large-eddy simulation of turbulent shear flows laden with bubbles. In Direct and Large-Eddy Simulation IV, pages 461–470. Springer, 2001.

[122] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[123] K.W. Morton and D. F. Mayers. Numerical solution of partial differential equations: an introduction. Cambridge university press, 2005.

[124] E. Mueller, W. Mell, and A. Simeoni. Large eddy simulation of forest canopy flow for wildland fire modeling. Canadian Journal of Forest Research, 44:1534–1544, 2014.

[125] A. Ozel, J. Kolehmainen, S. Radl, and S. Sundaresan. Fluid and particle coarsening of drag force for discrete-parcel approach. Chemical engineering science, 155:258–267, 2016.

[126] N. Panicker, A. Passalacqua, and R.O. Fox. On the hyperbolicity of the two-fluid model for gas–liquid bubbly flows. Applied Mathematical Modelling, 57:432–447, 2018.

[127] A. Parente, C. Gorlé, J. Van Beeck, and C. Benocci. Improved k–ε model and wall function formulation for the RANS simulation of ABL flows. Journal of wind engineering and industrial aerodynamics, 99:267–278, 2011.

[128] R. Picardi, L. Zhao, and F. Battaglia. On the ideal grid resolution for two-dimensional eulerian modeling of gas–liquid flows. Journal of Fluids Engineering, 138:114503, 2016.

[129] T.J. Poinsot and S.K. Lelef. Boundary conditions for direct simulations of compressible viscous flows. Journal of Computational Physics, 101:104–129, 1992.

[130] Y. Qi, J. Lu, R. Scardovelli, S. Zaleski, and G. Tryggvason. Computing curvature for volume of fluid methods using machine learning. Journal of Computational Physics, 377:155–161, 2019.

[131] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019.

[132] M. Raissi, A. Yazdani, and G.E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science, 367:1026–1030, 2020.

[133] D. Ramkrishna. Population balances: Theory and applications to particulate systems in engineering. Academic Press, 2000.

[134] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28, 2015.

[135] X. Ren, P. Hu, H. Su, F. Zhang, and H. Yu. Physics-informed neural networks for transonic flow around a cylinder with high reynolds number. Physics of Fluids, 36, 2024.

[136] P.J. Richards and R.P. Hoxey. Appropriate boundary conditions for computational wind engineering models using the k-ε turbulence model. Journal of wind engineering and industrial aerodynamics, 46:145–153, 1993.

[137] P.L. Roe. Approximate riemann solvers, parameter vectors, and difference schemes. Journal of Computational Physics, 43:357–372, 1981.

[138] J. Roenby, H. Bredmose, and H. Jasak. A computational method for sharp interface advection. Royal Society open science, 3:160405, 2016.

[139] I Roghair, M.V.S. Annaland, and J.A.M. Kuipers. An improved Front-Tracking technique for the simulation of mass transfer in dense bubbly flows. Chemical Engineering Science, 152:351–369, 2016.

[140] E. Roohi, A.P. Zahiri, and M. Passandideh-Fard. Numerical simulation of cavitation around a two-dimensional hydrofoil using VOF method and LES turbulence model. Applied Mathematical Modelling, 37:6469–6488, 2013.

[141] F. Sarghini, G. de Felice, and S. Santini. Neural networks based subgrid scale modeling in large eddy simulations. Computers & fluids, 32:97–108, 2003.

[142] V. Sekar, Q. Jiang, C. Shu, and B.C. Khoo. Fast flow field prediction over airfoils using deep learning approach. Physics of Fluids, 31, 2019.

[143] P.N. Shankar and M.D. Deshpande. Fluid mechanics in the driven cavity. Annual review of fluid mechanics, 32:93–136, 2000.

[144] T.H. Shih. A realizable Reynolds stress algebraic equation model, volume 105993. National Aeronautics and Space Administration, 1993.

[145] T.H. Shih, W.W. Liou, A. Shabbir, Z. Yang, and J. Zhu. A new k-epsilon eddy viscosity model for high reynolds number turbulent flows: Model development and validation. NASA Sti/recon Technical Report N, 95:11442, 1994.

[146] D. Shu, Z. Li, and A. Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. Journal of Computational Physics, 478:111972, 2023.

[147] R. Singh, B. Datta, and A. Jain. Identification of unknown groundwater pollution sources using artificial neural networks. Journal of water resources planning and management, 130:506–514, 2004.

[148] S.W. Sloan. An algorithm for profile and wavefront reduction of sparse matrices. International Journal for Numerical Methods in Engineering, 23:239–251, 1986.

[149] A. Sogachev, M. Kelly, and M.Y. Leclerc. Consistent two-equation closure modelling for atmospheric research: buoyancy and vegetation implementations. Boundary-layer meteorology, 145:307–327, 2012.

[150] A. Sogachev and O. Panferov. Modification of two-equation models to account for plant drag. Boundary-Layer Meteorology, 121:229–266, 2006.

[151] C. Soulaine and H.A. Tchelepi. Micro-continuum approach for pore-scale simulation of subsurface processes. Transport in porous media, 113:431–456, 2016.

[152] P. Spalart. An old-fashioned framework for machine learning in turbulence modeling. arXiv preprint arXiv:2308.00837, 2023.

[153] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. In 30th aerospace sciences meeting and exhibit, page 439, 1992.

[154] P.R. Spalart. Comments on the feasibility of LES for wings and on the hybrid RANS/LES approach. In Proceedings of the First AFOSR International Conference on DNS/LES, 1997, pages 137–147, 1997.

[155] D.B. Spalding. A single formula for the law of the wall. Journal of Applied Mechanics, 28:455–458, 1961.

[156] C.G. Speziale and S. Thangam. Analysis of an RNG based turbulence model for separated flows. International Journal of Engineering Science, 30:1379–1389, 1992.

[157] P. Srinivasan, L. Guastoni, H. Azizpour, P. Schlatter, and Ricardo Vinuesa. Predictions of turbulent shear flows using deep neural networks. Physical Review Fluids, 4:054603, 2019.

[158] A.I. Stamou and G. Papadonikolaki. Modeling the 3-D flow around a cylinder using the SAS hybrid model. Global Nest Journal, 16:901–918, 2014.

[159] C. Ströfer, J. Wu, H. Xiao, and E. Paterson. Data-driven, physics-based feature extraction from fluid flow fields. arXiv preprint arXiv:1802.00775, 2018.

[160] H. Struchtrup. Macroscopic transport equations for rarefied gas flows. Springer, 2005.

[161] C. Sun, S. Liu, Q. Wang, Z. Wan, and D. Sun. Bifurcations in penetrative Rayleigh-Bénard convection in a cylindrical container. Applied Mathematics and Mechanics, 40:695–704, 2019.

[162] S. Sundaresan, A. Ozel, and J. Kolehmainen. Toward constitutive models for momentum, species, and energy transport in gas–particle flows. Annual review of chemical and biomolecular engineering, 9:72–85, 2018.

[163] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. Journal of Computational physics, 114:146–159, 1994.

[164] T. Tao. Finite time blowup for an averaged three-dimensional Navier-Stokes equation. Journal of the American Mathematical Society, 29:601–674, 2016.

[165] S. Tenneti, R. Garg, and S. Subramaniam. Drag law for monodisperse gas–solid systems using particle-resolved direct numerical simulation of flow past fixed assemblies of spheres. International journal of multiphase flow, 37:1072–1092, 2011.

[166] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu. Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows. AIAA Journal, 58:25–36, 2020.

[167] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating eulerian fluid simulation with convolutional networks. In International Conference on Machine Learning, pages 3424–3433. PMLR, 2017.

[168] E.F. Toro. Riemann solvers and numerical methods for fluid dynamics: a practical introduction. Springer Science & Business Media, 2013.

[169] M. Torrilhon. Modeling nonequilibrium gas flow based on moment equations. Annual review of fluid mechanics, 48:429–458, 2016.

[170] B. Tracey, K. Duraisamy, and J. Alonso. Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. In 51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition, page 259, 2013.

[171] B.D. Tracey, K. Duraisamy, and J.J. Alonso. A machine learning strategy to assist turbulence model development. In 53rd AIAA aerospace sciences meeting, page 1287, 2015.

[172] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.J. Jan. A front-tracking method for the computations of multiphase flow. Journal of Computational Physics, 169:708–759, 2001.

[173] C. Tsouris and L.L. Tavlarides. Breakage and coalescence models for drops in turbulent dispersions. AIChE Journal, 40:395–406, 1994.

[174] O. Ubbink. Numerical prediction of two fluid systems with sharp interfaces. PhD thesis, University of London, 1997.

[175] Salih Ozen Unverdi and Grétar Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. Journal of computational physics, 100:25–37, 1992.

[176] Maarten Paul van der Laan, Mark Kelly, Rogier Floors, and Alfredo Peña. Rossby number similarity of atmospheric RANS using limited length scale turbulence closures extended to unstable stratification. Wind Energy Science, 83:75–98, 2019.

[177] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[178] H.K. Versteeg and W. Malalasekera. An introduction to computational fluid dynamics: the finite volume method. Pearson education, 2007.

[179] A. Vié, C. Chalons, R. Fox, F. Laurent, and M. Massot. A multi-gaussian quadrature method of moments for simulating high stokes number turbulent two-phase flows. Annual Research Brief of the Center for Turbulence Research-Stanford University, pages 309–320, 2012.

[180] H. Wang, L. Lu, S. Song, and G. Huang. Learning specialized activation functions for physics-informed neural networks. arXiv preprint arXiv:2308.04073, 2023.

[181] J. Wang and J. Liu. 基于 gpu 的并行 dilu 预处理技术. 计算机科学, 49:108–118–335, 2022.

[182] J. Wang, J. Wu, and H. Xiao. Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. Physical Review Fluids, 2:034603, 2017.

[183] L. Wang, L. Wang, Z. Guo, and J. Mi. Volume-averaged macroscopic equation for fluid flow in moving porous media. International Journal of Heat and Mass Transfer, 82:357–368, 2015.

[184] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris. An expert's guide to training physics-informed neural networks. arXiv preprint arXiv:2308.08468, 2023.

[185] D.C. Wilcox. Reassessment of the scale-determining equation for advanced turbulence models. AIAA journal, 26:1299–1310, 1988.

[186] D.C. Wilcox. Turbulence modeling for CFD, volume 2. DCW industries La Canada, CA, 1998.

[187] D.C. Wilcox. Formulation of the kw turbulence model revisited. AIAA journal, 46:2823–2838, 2008.

[188] H. Xiao, J. Wu, J. Wang, and E.G. Paterson. Physics-informed machine learning for predictive turbulence modeling: Progress and perspectives. Proceedings of the 2017 AIAA SciTech, 2017.

[189] X. Xiao, Y. Zhou, H. Wang, and X. Yang. A novel CNN-based Poisson solver for fluid simulation. IEEE transactions on visualization and computer graphics, 26:1454–1465, 2018.

[190] C. Xie, J. Wang, and W. E. Modeling subgrid-scale forces by spatial artificial neural networks in large eddy simulation of turbulence. Physical Review Fluids, 5:054606, 2020.

[191] K. Xu. Gas-kinetic schemes for unsteady compressible flow simulations. Computational Fluid Dynamics, Annual Lecture Series, 29 th, Rhode-Saint-Genese, Belgium, 1998.

[192] K. Xu. A gas-kinetic BGK scheme for the Navier–Stokes equations and its connection with artificial dissipation and Godunov method. Journal of Computational Physics, 171:289–335, 2001.

[193] V. Yakhot, S.A. Orszag, S. Thangam, T.B. Gatski, and C.G. Speziale. Development of turbulence models for shear flows by a double expansion technique. Physics of Fluids A: Fluid Dynamics, 4:1510–1520, 1992.

[194] C. Yang, X. Yang, and X. Xiao. Data-driven projection method in fluid simulation. Computer Animation and Virtual Worlds, 27:415–424, 2016.

[195] Y. Yang, M. Gu, S. Chen, and X. Jin. New inflow boundary conditions for modelling the neutral equilibrium atmospheric boundary layer in computational wind engineering. Journal of Wind Engineering and Industrial Aerodynamics, 97:88–95, 2009.

[196] Y. Yang, Z. Xie, and M. Gu. Consistent inflow boundary conditions for modelling the neutral equilibrium atmospheric boundary layer for the SST k-ω model. Wind and Structures, 24:465–480, 2017.

[197] Z. Yang, B. Lu, and W. Wang. Coupling artificial neural network with EMMS drag for simulation of dense fluidized beds. Chemical Engineering Science, 246:117003, 2021.

[198] M.E. Young and A. Ooi. Comparative assessment of LES and URANS for flow over a cylinder at a Reynolds number of 3900. In Proceedings of the 16th Australian Fluid Mechanics Conference, Gold Coast, Australia, 1063-1070, 2007.

[199] S.T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. Journal of computational physics, 31:335–362, 1979.

[200] S. Zhang and X. Zhao. General formulations for Rhie-Chow interpolation. In Heat Transfer Summer Conference, volume 46911, pages 567–573, 2004.

[201] S. Zhang, X. Zhao, and S. Bayyuk. Generalized formulations for the Rhie-Chow interpolation. Journal of Computational Physics, 258:880–914, 2014.

[202] L. Zhu, J. Tang, and Z. Luo. Machine learning to assist filtered two-fluid model development for dense gas–particle flows. AIChE Journal, 66:e16973, 2020.

[203] L. Zhu, W. Zhang, J. Kou, and Y. Liu. Machine learning methods for turbulence modeling in subsonic flows around airfoils. Physics of Fluids, 31, 2019.

[204] J. Zhuang, D. Kochkov, Y. Bar-Sinai, M.P. Brenner, and S. Hoyer. Learned discretizations for passive scalar advection in a two-dimensional turbulent flow. Physical Review Fluids, 6:064605, 2021.

[205] 同济大学数学系. 高等数学. 高等教育出版社, 2007.

[206] 宇波等. 数值传热学实训. 科学出版社, 2018.

[207] 李庆扬等. 数值分析. 华中科技大学出版社, 2007.

[208] 温凯杰, 郭力, 夏诏杰, and 陈建华. 一种耦合 CFD 与深度学习的气固快速模拟方法. 化工学报, 74:3775, 2023.

[209] 约翰. 安德森. 计算流体力学基础及其应用. 机械工业出版社, 2007.