



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

博士学位论文

GPU 加速的超大规模可压缩湍流直接数值模拟研究

作者姓名: 党冠麟

指导教师: 李新亮 研究员

中国科学院力学研究所

学位类别: 工学博士

学科专业: 流体力学

培养单位: 中国科学院力学研究所

2022 年 12 月

**Direct numerical simulation of very large-scale compressible
turbulence accelerated by graphics processing unit**

A dissertation submitted to
University of Chinese Academy of Sciences
in partial fulfillment of the requirement
for the degree of
Doctor of Engineering
in **Fluid Mechanics**
By
Dang Guanlin
Supervisor: Professor Li Xinliang

Institute of Mechanics, Chinese Academy of Sciences

December, 2022

中国科学院大学 学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。承诺除文中已经注明引用的内容外，本论文不包含任何其他个人或集体享有著作权的研究成果，未在以往任何学位申请中全部或部分提交。对本论文所涉及的研究工作做出贡献的其他个人或集体，均已在文中以明确方式标明或致谢。本人完全意识到本声明的法律结果由本人承担。

作者签名：
日期：2022年10月

中国科学院大学 学位论文授权使用声明

本人完全了解并同意遵守中国科学院大学有关收集、保存和使用学位论文的规定，即中国科学院大学有权按照学术研究公开原则和保护知识产权的原则，保留并向国家指定或中国科学院指定机构送交学位论文的电子版和印刷版文件，且电子版与印刷版内容应完全相同，允许该论文被检索、查阅和借阅，公布本学位论文的全部或部分内容，可以采用扫描、影印、缩印等复制手段以及其他法律许可的方式保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名： 导师签名：
日期：2022年10月 日期：2022年10月

摘要

可压缩湍流边界层在超/高超声速飞行器等航空航天应用中广泛存在。开展可压缩湍流的研究既是航空航天飞行器设计的迫切需求，又有重要的理论意义。直接数值模拟（Direct Numerical Simulation, DNS）能够提供精度和分辨率最高的湍流数据，而且不需要湍流模型，是研究可压缩湍流的最有力的手段之一。但当前，制约可压缩 DNS 发展的主要因素有以下两个，1. 直接数值模拟对计算机资源需求极大，需要与高性能计算技术深度结合；2. 在对高超声速、强激波、分离的流动进行模拟时，数值方法的低耗散与高鲁棒性的矛盾点被激化，需要进一步发展数值方法，以扩宽 DNS 的应用范围。本文致力于对这两个难题进行攻关。具体内容如下：

(1) 面向 CPU (Central Processing Unit) -GPU (Graphics Processing Unit) 异构超算体系，开发了一套适用于可压缩湍流高精度数值模拟的计算流体力学软件 OpenCFD-SCU (open computational fluid dynamic code for scientific computation with GPU system)，该软件基于有限差分方法，使用 GPU 加速。与基于相同算法和相同 MPI (Message Passing Interface) 进程数的 CPU 软件相比，能加速 200 倍以上。在升力体算例中，OpenCFD-SCU 仅使用 512 个 GPU 的运行速度就超过了使用 13 万个 CPU 核的 CPU 计算。本文使用 GPU-Stream 技术实现了计算和通信的重叠，在 24576 个 GPU 上达到了 98.7% 的并行弱可扩展性。该软件代码行数超过两万行，包括了多种高精度有限差分格式，大幅提升了可压缩 DNS 能力。该软件已开源，希望可以为可压缩湍流的研究提供新的数值工具，软件源代码可在<http://developer.hpccube.com/codes/danggl/opencfd-scu.git> 上获得。此外，本文介绍了针对 GPU 硬件的程序编写与优化方法，可以为其它学科的数值模拟软件开发提供参考。

(2) 在高超声速边界层流动中，存在激波，流动分离等复杂现象，在数值仿真中，计算的高鲁棒性与低耗散难以同时满足。针对该问题，本文在 Jameson 人工粘性系数的基础之上，改进得到了新的激波识别器，并基于此发展了三种有限差分格式混合的混合格式。在可压缩流动模拟时，流场中的大部分位置采用低耗散的 7 阶线性迎风格式；在激波和间断处，根据所需耗散的不同，采用 7 阶 WENO (Weighted Essentially Nonoscillatory) 或 5 阶 WENO 格式。该混合格式能够兼顾高精度与高鲁棒性，在多个大规模问题的模拟中得到了验证；结果表明，流场 90% 以上的部分都使用高精度的 7 阶迎风格式，且计算鲁棒性大幅提升。

(3) 利用开发的 OpenCFD-SCU 软件，本文进行了平板、钝锥、升力体及压缩折角等多个算例的 DNS，并建立了开放的壁湍流数据库。这些流动算例具有典型性，包含了超/高超声速多个来流马赫数，几何构型从简单的几何外形到接近真实飞行器外形。数据库中的算例具体为：来流马赫数为 2.9 的 24° 压缩折角，网格数为 76.8 亿；来流马赫数为 6 的 34° 压缩折角，网格数为 93 亿；来流马赫

数为 10 的冷壁平板湍流边界层，摩擦雷诺数为 1550，网格数为 45 亿；头部半径为 1 mm，攻角为 0° 的钝锥湍流边界层，网格数为 240 亿；来流马赫数为 6 的升力体模型，网格数为 111 亿。与同类的数值模拟结果相比，本文数据库中算例具有更大的计算域、更高的分辨率或更高的雷诺数。本文对这些不同情况的流动进行了验证、分析和比较。数据的获取方式可以在 OpenCFD-SCU 源代码的开源网站<http://developer.hpccube.com/codes/danggl/opencfd-scu.git> 上获得，用户可以下载数据，也可参考相关设置自行设计算例。希望该数据库能够给湍流机理研究，模型发展，人工智能训练等带来帮助。

关键词：软件，湍流，高超声速，直接数值模拟，图形处理器

Abstract

Compressible turbulent boundary layer flows exist widely in aerospace applications such as supersonic or hypersonic vehicles. The research on compressible turbulence is of great importance to improve the national defense, enhance the national competitiveness, support the engineering development and meet the national major demand. Direct numerical simulation (DNS) can provide turbulence data with the highest accuracy and resolution, with no turbulence model being required, and it has become a powerful tool for studying compressible turbulence. Nevertheless, there are two significant challenges that have hindered the development of DNS. The first challenge lies in the demand of DNS for huge computational resources, which needs to be deeply integrated with high-performance computing technology. The second concerns the numerical schemes, especially for compressible turbulence with hypersonic, strong shock waves and flow separation. The conflict between numerical stability and accurate resolution becomes acute, and hence the computational algorithms need to be further developed to broaden the scope of application of DNS. This paper is devoted to addressing these two challenges. The details are as follows:

(1) This paper develops an open-source computational fluid dynamics software named open computational fluid dynamic code for scientific computation with graphics processing unit (GPU) system (OpenCFD-SCU). This software is based on the finite difference method and is accelerated by the use of GPUs, which provides an acceleration by a factor of more than 200 compared with central processing unit (CPU) software based on the same algorithm and number of message passing interface (MPI) processes, and the running speed of OpenCFD-SCU with only 512 GPUs exceeds that of CPU software with 130 000 CPUs. GPU-Stream technology is used to implement overlap of computing and communication, achieving 98.7% parallel weak scalability with 24 576 GPUs. The software has more than 20,000 lines of code and includes a variety of high-precision finite difference schemes, and greatly improves our the ability of DNS for turbulent boundary layer. In addition, we open source the software, hoping to make further contributions to the development of DNS. The code is available and supported at <http://developer.hpccube.com/codes/danggl/opencfd-scu.git>. In this paper, the programming and optimization methods for GPU hardware are also introduced, which can provide a reference for numerical simulation in other disciplines.

(2) In order to deal with the simulation of hypersonic complex flow with shock wave, turbulence and separation. Based on the artificial dissipation coefficient of Jameson, we improve a shock wave recognizer and develop a hybrid scheme of three schemes. The shock wave recognizer that is based on the artificial dissipation coefficient of Jame-

son is improved. In addition, a hybrid scheme of three schemes is developed. In the compressible flow simulation, the low-dissipation linear seventh-order upwind scheme calculates most of the flow field. The fifth-order WENO (weighted essentially nonoscillatory) or seventh-order WENO is used to calculate some discontinuities in the flow field, depending on whether the discontinuity is strong or weak. The hybrid schemes have good numerical stability, and they can simultaneously provide high accuracy and low numerical dissipation. They are verified in the simulation of several large-scale cases, and the results indicate that more than 90% of the flow field uses the low-dissipation seventh-order upwind scheme;

(3) An open database of wall turbulence is established via DNS using OpenCFD-SCU with hybrid scheme. The database contains a variety of representative flows, from supersonic to hypersonic, and from relatively simple geometric pattern to that close to the actual aircraft. Cases in the database are a 24° compression ramp at Mach number (Ma) 2.9 with the length of the ramp being 200mm and the mesh number 7.68 billion; a 34° compression-ramp at Ma 6 with mesh number being 9.3 billion; a cold-wall flat plate at Ma 10 with the friction Reynolds number being 1550 and the mesh number 4.5 billion; a blunt cone with 1mm head radius and 0° attack angle at Ma 10, where the mesh number is 24 billion; a lifting body model at Ma 6 with the mesh number 11.1 billion. Compared with the reported DNS with similar conditions, these cases have a larger computational domain, higher resolution or higher Reynolds number, which demonstrate the simulation capability of OpenCFD-SCU and broaden the scope of DNS applications. Some analysis has been performed, which builds an open-access database to store these data; the source code of OpenCFD-SCU can be accessed at <http://developer.hpccube.com/codes/danggl/opencfd-scu.git>, this website also contains detailed database descriptions and data acquisition methods. It is expected that these efforts can be useful for turbulence mechanism study, turbulence model development and the training and verification of artificial intelligence models.

Key Words: Software, Turbulence, Hypersonic, Direct numerical simulations, Graphics processing unit

目 录	
摘 要	I
Abstract	III
符号列表	XIII
第 1 章 绪论	1
1.1 湍流	1
1.1.1 认识湍流	1
1.1.2 二十世纪的湍流研究	3
1.1.3 可压缩湍流研究	4
1.2 湍流的数值模拟	6
1.2.1 雷诺平均模拟方法	7
1.2.2 大涡模拟方法	7
1.2.3 直接数值模拟方法	7
1.3 高性能计算发展与异构并行计算	8
1.3.1 高性能计算发展历史	8
1.3.2 异构计算	9
1.3.3 GPU 与众核处理器	10
1.4 本文的研究内容和组织结构	12
第 2 章 控制方程及数值方法	15
2.1 引言	15
2.2 控制方程	15
2.3 无粘项数值离散方法	18
2.3.1 Steger-Warming 流通矢量分离	18
2.3.2 使用特征变量的通量重构	19
2.3.3 无粘项差分离散方法	21
2.3.4 边界差分离散方法	30
2.4 粘性项差分离散方式	30
2.5 时间推进格式	31
2.6 小结	32

第 3 章 高精度有限差分软件 OpenCFD-SCU 框架与结构	33
3.1 引言	33
3.2 程序总体框架	33
3.3 数据的三维并行分割与 MPI 通讯	35
3.4 程序流水线设计	39
3.5 MPI 并行 IO	39
3.6 小结	42
第 4 章 OpenCFD-SCU 在 CPU-GPU 异构体系下的优化方法	43
4.1 引言	43
4.2 硬件环境	43
4.3 CPU-GPU 系统异构编程方法与实现	47
4.4 OpenCFD-SCU 程序热点分析	50
4.5 优化方法	51
4.5.1 全局内存的合并访问	51
4.5.2 线程束内的数据交换 Warp-shuffle 与冗余计算设计	54
4.5.3 基于共享内存的线程重排	56
4.5.4 原子操作	58
4.5.5 页锁定内存与 CPU-GPU 通讯前的数据打包	59
4.5.6 多流水线并发	59
4.5.7 其它优化方法	60
4.6 小结	61
第 5 章 OpenCFD-SCU 性能与扩展性测试	63
5.1 引言	63
5.2 正确性测试	63
5.2.1 与 CPU 程序的结果对比	63
5.2.2 正确性的进一步测试	63
5.3 与 CPU 程序相比的加速效果与强扩展性测试	66
5.4 弱扩展性测试	67
5.5 性能评估	69
5.6 跨平台测试	70
5.7 大规模湍流直接数值模拟测试	70
5.8 小结	74

第 6 章 超大规模湍流直接数值模拟应用及数据库建设	75
6.1 引言	75
6.2 马赫 2.9 压缩折角流动的直接数值模拟	76
6.3 马赫 6 压缩折角流动的直接数值模拟	84
6.4 马赫 10 平板湍流边界层的直接数值模拟	90
6.5 马赫 6 钝锥湍流边界层的直接数值模拟	95
6.6 马赫 6 升力体转捩的直接数值模拟	103
6.7 小结	108
第 7 章 结论与展望	109
7.1 结论	109
7.2 本文工作的创新点	110
7.3 展望	111
参考文献	113
附录	121
.1 OpenCFD-SCU 控制文件与参数说明	121
.1.1 OpenCFD-SCU 参数定义模块	121
.1.2 OpenCFD-SCU 参数读取模块	122
.1.3 控制文件参数填写注意事项与说明	129
.1.4 OpenCFD-SCU 控制文件示例	130
.2 数据库说明	139
致谢	141
作者简历及攻读学位期间发表的学术论文与研究成果	143

图目录

图 1.1 文艺复兴时期, 达芬奇关于漩涡的画作	1
图 1.2 CPU 与 GPU 的架构差异 ^[1]	10
图 1.3 NVIDIA 费米 GPU 架构示意图 ^[1]	12
图 2.1 差分模板点	20
图 3.1 OpenCFD-SCU 程序框架	34
图 3.2 OpenCFD-SCU 中的 MPI 数据交换模型	36
图 3.3 三维 MPI 数据划分	38
图 3.4 一个 MPI 进程中的数据结构	38
图 3.5 OpenCFD-SCU 程序时间线设计	39
图 4.1 自动伸缩性 ^[1]	43
图 4.2 GPU 内存结构	45
图 4.3 异构编程 ^[1]	46
图 4.4 异构程序实现过程示意图	49
图 4.5 OpenCFD-SCU 中的三维线程块划分	50
图 4.6 程序热点分析	50
图 4.7 全局内存合并对齐访问	51
图 4.8 全局内存合并未对齐访问	51
图 4.9 全局内存对齐未合并访问 ^[1]	52
图 4.10 线程块获取数据过程	53
图 4.11 线程块之间的冗余区	55
图 4.12 线程束内的数据获取方式与线程重排	56
图 4.13 程序优化前后的时间线	58
图 4.14 开启多流水线后的程序时间线	60
图 5.1 OpenCFD-SC 与 OpenCFD-SCU 输出结果对比图	64
图 5.2 使用参考温度无量纲化的瞬时温度	64
图 5.3 Van Driest 变换后的平均速度剖面	65
图 5.4 平均壁面压力分布, DNS 与实验的差距约为 3%	65
图 5.5 OpenCFD-SC 和 OpenCFD-SCU 在模拟相同问题时的加速情况比较 (基本速度被选为 OpenCFD-SC 可以计算的最小 MPI 过程下的程序运行速度)	67
图 5.6 弱扩展性并行效率	68
图 5.7 OpenCFD-SCU 双精度浮点运算性能随进程数的变化	69
图 5.8 网格和计算域示意图: (a) 三维; (b) $x-y$ 截面	71

图 5.9 壁面摩擦阻力系数 C_f 分布情况	72
图 5.10 $X=44$ mm 处沿展向的速度分布: (A) 时间平均后; (B) 瞬时	73
图 5.11 三个不同小区域的瞬时速度和流线	73
图 6.1 马赫 2.9 压缩折角流直接数值模拟计算域示意图	77
图 6.2 马赫 2.9 压缩折角流不同区域的网格加密情况	78
图 6.3 马赫 2.9 压缩折角流混合格式分布情况	78
图 6.4 马赫 2.9 压缩折角流网格分辨率流向分布情况	79
图 6.5 马赫 2.9 压缩折角流展向中截面的瞬时温度分布, 使用参考温度进行无量纲化	79
图 6.6 马赫 2.9 压缩折角流展向中截面的瞬时温度分布的放大图。 (a) 平板区; (b) 角区; (c) 坡面区	80
图 6.7 马赫 2.9 压缩折角流根据壁面距离进行染色的湍流相干结构	81
图 6.8 马赫 2.9 压缩折角流不同位置经 Van Driest 变换后的平均速度剖面	81
图 6.9 马赫 2.9 压缩折角流壁面的瞬时速度分布	82
图 6.10 马赫 2.9 压缩折角流的流向分布的 (a) 摩擦阻力系数 (b) 壁面热流 (c) 壁面压力	83
图 6.11 马赫 6 压缩折角流计算域	85
图 6.12 马赫 6 压缩折角流网格分辨率的流向分布	85
图 6.13 马赫 6 压缩折角流混合格式分布情况	86
图 6.14 马赫 6 压缩折角流展向中截面的瞬时温度分布, 使用参考温度进行无量纲化。 (a) 全局分布情况。 (b) 折角区局部放大情况	86
图 6.15 马赫 6 压缩折角流利用根据壁面距离染色的湍流涡相干结构	87
图 6.16 马赫 6 压缩折角流不同位置经 Van Driest 变换后的平均速度剖面	87
图 6.17 马赫 6 压缩折角流的流向分布的 (a) 摩擦阻力系数 (b) 壁面热流 (c) 壁面压力	88
图 6.18 不同情况经时间平均后的流向平均速度和流线等值线: (a) 马赫 2.9 压缩折角 (参见小节 6.2) (b) 马赫 6 压缩折角	89
图 6.19 马赫 10 平板计算域) (b) 马赫 6 压缩折角	91
图 6.20 马赫 10 平板混合格式分布	91
图 6.21 马赫 10 平板边界层网格分辨率的流向分布	92
图 6.22 马赫 10 平板边界层无量纲化的瞬时密度、温度分布情况: (a) 无量纲化瞬时密度的全局分布情况。 (b) 无量纲化瞬时密度的局部放大情况。 (c) 无量纲化瞬时温度的全局分布情况。 (d) 无量纲化瞬时温度的局部放大情况	93
图 6.23 马赫 10 平板边界层利用根据壁面距离染色的湍流涡相干结构	94
图 6.24 马赫 10 平板边界层厚度沿流向变化	94
图 6.25 马赫 10 平板边界层不同位置经 Van Driest 变换后的平均速度剖面	95

图 6.26 马赫 6 钝锥边界层直接数值模拟计算域与网格示意图	96
图 6.27 马赫 6 钝锥流向截面瞬时温度分布	97
图 6.28 马赫 6 钝锥不同流向位置的周向截面瞬时温度分布	98
图 6.29 马赫 6 钝锥边界层利用根据壁面距离染色的湍流涡相干结构	99
图 6.30 马赫 6 钝锥边界层厚度沿流向变化	100
图 6.31 马赫 6 钝锥边界层不同流向位置经 Van Driest 变换后的平均速度 剖面	100
图 6.32 马赫 6 钝锥壁面摩擦阻力系数 C_f 分布	101
图 6.33 马赫 6 钝锥摩擦阻力系数 C_f 沿流向变化曲线	102
图 6.34 不同算例的脉动速度均方根	102
图 6.35 马赫 6 升力体计算域示意图	104
图 6.36 马赫 6 升力体不同流向位置的周向截面瞬时温度分布	105
图 6.37 马赫 6 升力体壁面摩擦阻力系数 C_f 分布	106
图 6.38 马赫 6 升力体壁温 T_w 以及壁面流线分布	107

表目录

表 1.1 截至 2022 年 5 月，超级计算机 TOP500 榜单前 10 名	11
表 2.1 差分格式表	21
表 4.1 MI60 (Vega 20 GL) 架构计算能力	46
表 4.2 MI60 (Vega 20 GL) 架构内存相关参数	47
表 5.1 OpenCFD-SC 和 OpenCFD-SCU 模拟相同问题所需时间的比较 ...	66
表 5.2 双精度浮点运算数	68
表 5.3 OpenCFD-SCU 在不同 GPU 上的运行时间	70
表 5.4 来流条件	71
表 6.1 数据库算例列表	75
表 6.2 模拟成本和时间	75
表 6.3 马赫 2.9 压缩折角流计算参数与参考位置边界层相关参数	77
表 6.4 马赫 6 压缩折角流计算参数与参考位置边界层相关参数	84
表 6.5 马赫 10 平板边界层的流计算参数与参考位置边界层相关参数	90
表 6.6 马赫 6 钝锥体计算参数与参考位置边界层相关参数	96
表 6.7 马赫 6 升力体计算参数与参考位置边界层相关参数	103
表 1 OpenCFD-SCU 控制文件中基础参数说明	133
表 2 OpenCFD-SCU 控制文件中差分选择参数说明	134
表 3 OpenCFD-SCU 控制文件中边界条件相关参数	135
表 4 OpenCFD-SCU 控制文件中滤波相关参数	136
表 5 OpenCFD-SCU 控制文件中后处理相关参数 (1)	137
表 6 OpenCFD-SCU 控制文件中后处理相关参数 (2)	138
表 7 OpenCFD-SCU 控制文件中混合格式相关参数	139
表 8 数据库文件说明	140

符号列表

字符

Symbol	Description	Unit
R	the gas constant	$\text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1}$
C_v	specific heat capacity at constant volume	$\text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1}$
C_p	specific heat capacity at constant pressure	$\text{m}^2 \cdot \text{s}^{-2} \cdot \text{K}^{-1}$
E	specific total energy	$\text{m}^2 \cdot \text{s}^{-2}$
e	specific internal energy	$\text{m}^2 \cdot \text{s}^{-2}$
h_T	specific total enthalpy	$\text{m}^2 \cdot \text{s}^{-2}$
h	specific enthalpy	$\text{m}^2 \cdot \text{s}^{-2}$
k	thermal conductivity	$\text{kg} \cdot \text{m} \cdot \text{s}^{-3} \cdot \text{K}^{-1}$
S_{ij}	deviatoric stress tensor	$\text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2}$
τ_{ij}	viscous stress tensor	$\text{kg} \cdot \text{m}^{-1} \cdot \text{s}^{-2}$
δ_{ij}	Kronecker tensor	1
I_{ij}	identity tensor	1

缩写

CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
GPU	Graphics Processing Unit
MPI	Message Passing Interface
WENO	Weighted Essentially Non-oscillatory
DNS	Direct Numerical Simulations

第1章 绪论

1.1 湍流

1.1.1 认识湍流

湍流又称紊流，一般指流体处于的一种混乱无序的运动状态。湍流在自然界和工程应用中无所不在。在自然界中，河流中广泛存在湍流，海洋表面下的水流一般是湍流，著名的墨西哥湾暖流是一种湍流的壁射流。地球大气的流动大多是湍流，大气对流层上层的急速流动是湍流，积云的运动也是湍流，大气中的输运现象主要由湍流控制，大气污染可以通过湍流扩散，大气湍流甚至会造成当地气候的剧烈变化^[2]。在更大的尺度上，太阳和类似恒星的光球层是湍流状态，太阳外层大气向外射出的太阳风是湍流，气态星云的运动状态也是湍流。工程应用上，湍流存在于输气/油管道里天然气和石油的流动中，还存在于内燃机和涡轮机等设备的流动中。大多数的燃烧过程也都涉及湍流，化学工程师利用湍流加速制造混合物，利用湍流加快某些化学反应的反应速度。湍流也几乎能在所有运动物体的周围观察到，如飞机、高铁、汽车，影响它们受到的阻力，乘飞机旅行的人可能会因飞机的剧烈颠簸而亲身感受湍流的威力^[3]。此外，在人体内血管中也有湍流的存在^[4]。



图 1.1 文艺复兴时期，达芬奇关于漩涡的画作

Figure 1.1 Leonardo da Vinci's paintings of swirls during the Renaissance

人们对湍流现象的细致描述，在文艺复兴时期的藝術作品中就已经体现。现存于法国文献研究所的一副意大利著名艺术家、科学家、工程师西奥多·达芬奇 (Leonardo da Vinci, 1452-1519) 的画作，如图 1.1 (图片来自于网络)，在该作品中，能够清晰地看到管道中的水流入池塘后激起的跨越多个尺度的大大小小的

涡，达芬奇还在他的手稿中使用过湍流这一术语 “turbolenza”。这是对湍流感性认识的较早记录。

对湍流的科学研究最初起源于人们对管道流动的研究。17 世纪，为满足太阳王路易十四对凡尔赛宫输水管道设计的要求，刚成立不久的法国皇家科学院把管道液压系统的研究列入了议程，在 1668 年由院士让 · 皮卡尔 (Jean Picard 1620–1682) 和埃梅德 · 马里奥 (Edme Mariotte 1620–1684) 负责相关研究工作^[5]。埃梅德 · 马里奥也主要由于这部分工作贡献被誉为法国实验方法之父。在 1804 年，法国水利学家加斯帕德 · 德普罗尼 (Gaspard de Prony 1755–1839) 提出了一个水流摩阻的经验公式，如式 $\frac{gdh}{4l} = au + bu^2$ 。这是较早的利用复杂数学描述经验水力学知识的记录。他在式中指出影响流体摩擦力主要有两种作用，一种是流体分子的相互作用，另一种是流体分子与壁面不规则的碰撞，前者导致摩擦力与流速成正比，后者产生的影响使摩擦与流速的平方成正比。德普罗尼的公式中 u 和 u^2 的差别可能预示着层流与湍流的区别^[6]。19 世纪 20 年代，克劳德 · 路易 · 纳维 (Claude-Louis Navier 1785–1836) 在流动基本方程中引入粘性的概念，在 19 世纪下半叶，管道公式中开始考虑流体粘性的影响^[6]。1839 年，德国水利学家哈根 (Gotthilf Hagen 1797–1884) 对管道流动做了大量的实验研究，并考虑了水流温度对管道摩擦的影响。哈根在他的管道研究中已经观察到了流动从层流向湍流转捩的过程，他在手稿中写到“我观察到，在较小压力下，木屑只随着管道方向移动，而在较强压力的时候，管道中的木屑会从一侧喷到另一侧，运动方式常常呈漩涡状”^[7]。他的这一观察随后也被其它水利学家注意到，法国水力学家亨利 · 达西 (Henry Darcy 1803–1858) 在几年后也进行了管道中木屑的实验，而且在 1850 年，达西利用其负责巴黎供水系统的机会，对粘性管道内流动沿程损失进行了系统的实验研究，进一步观察到两种不同性质的流动规律，他评论到“似乎在管道中水流存在两个定律，但至少在我测量的管道直径中，水流以 9-10 厘米/秒流速流动的情况下，这两个定律可以合并”^[7]。到了 19 世纪 60 年代，许多水利工程师的实验已经证明，狭窄管道中的流动与大型管道中的流动存在本质的不同，而且流速对流动状态也有影响。但是当时的流体力学理论未能很好地解释这个问题，1872 年，著名理论力学家阿德马尔 · 巴雷德 · 圣维南 (Adhémar Barré de Saint-Venant 1797–1886) 认为这个问题是“一个真正的谜”，它的学生布辛涅斯克 (Joseph Boussinesq 1842–1929) 将圣维南的理论观点与达西等人的工程知识相结合，推导并提出了涡粘的概念^[8]。1877 年，布辛涅斯克在研究水渠流动时在纳维-斯托克斯 (Navier-Stokes) 方程中引入了湍流波动，假设一个可变的由当地影响的摩擦系数，会对流动粘性造成额外的影响^[9]，这一理论尽管受到了一些批评，但也被一部分人认为是一种有希望让人理解水渠中湍流的方式。几年后，曼彻斯特的工程学教授奥斯本 · 雷诺 (Osborne Reynolds 1842–1912) 通过圆管流动实验，系统地研究了管道尺寸、流体性质、流动速度对流动状态的影响^[10]，它的研究工作在湍流研究史上具有里程碑式的意义。雷诺在 1895 年发表的一篇文章中提出了管道流体从层流转捩为湍流的临界速度为 $K\mu/D\rho$ 、其中 D 为管道直径、 ρ 为流体密度、 K 为一常数^[11]。这一结论后来引申出了湍流研究中

最重要的无量纲参数之一的雷诺数。在这篇文章中，雷诺还把湍流瞬时量分解成平均和脉动两部分，由此导出了统计平均满足的方程，这便是著名的雷诺方程，将雷诺方程与 Navier-Stokes 方程进行比较，便提出了雷诺应力的概念。

1.1.2 二十世纪的湍流研究

第一次世界大战时期，人们已经清楚地知道，湍流是流动雷诺数达到一定程度产生的现象，湍流的转捩和完全发展的湍流成为 20 世纪几代湍流研究者关注的问题。1904 年，普朗特 (Ludwig Prandtl) 给出了边界层的概念^[12]。20 世纪之前，人们或多或少已经有了溪流中固体表面与自由流动的水流之间存在一定距离的摩擦层的概念，但理论都没有进一步发展。普朗特将这种现象发展成了一套流动要脱离物体表面的理论，即边界层理论。在最早的形式中，这个概念由普朗特的第一个博士生布拉休斯 (Heinrich Blasius) 为层流所发展^[13]。湍流边界层的流动与脱离最初在哥廷根的风洞中被发现。普朗特的另一个学生冯·卡门 (Theodore von Kármán) 为湍流边界层的理论铺平了道路^[14,15]。

雷诺在 1895 年的文章中提出了雷诺方程^[11]，由于湍流的脉动，在平均流中产生了一种新的应力，除了分子粘性应力引起的动量交换外还增加了速度脉动对动量输运的贡献。但是雷诺方程是不封闭的，表现在控制方程的数目总是小于未知数的数目，这个问题一直困扰着流体力学的研究者们。为使雷诺方程封闭，普朗特在布辛涅斯克的涡粘理论基础上发展出用以建立雷诺应力与平均速度之间的关系的混合长理论^[16]，为湍流的半经验理论开辟了道路。半经验理论的目标是建立雷诺应力与流动基本量之间的关系，普朗特的混合长认为雷诺应力与平均速度有关，而不同科学家对此有不同的观点，如冯·卡门认为雷诺应力与湍流速度波动的特征尺度有关，剑桥大学的泰勒 (Geoffrey Ingram Taylor) 认为雷诺应力与涡量的交换有关，由此催生出了不同的湍流半经验理论，如动量输运理论，涡量输运理论和卡门相似理论^[17]。半经验理论的缺点在于，当平均速度梯度为 0 时，湍流应力也为 0，但目前的研究表明，湍流中存在大尺度的拟序结构，复杂的湍流中的雷诺应力不仅取决于局部速度梯度还与整体湍流的特征有关。而整体的湍流特征包含有湍流涡的倾斜角度和各项异性特征等^[18]。

自雷诺方程提出后，人们集中解决雷诺方程的封闭性问题，但是成效不大。20 世纪 20 年代，泰勒开始用统计学的方法对湍流“扩散作用”进行分析^[19]。这被称之为湍流统计理论，湍流统计理论的主要思想是借助概率论和数理统计的工具，以研究湍流脉动场的统计规律和流动特性。这一湍流研究方式在 20 世纪 30 年代达到高潮，这一阶段的统计概念主要由泰勒^[20]和冯·卡门^[21]提出，关键的测量在英国 (Simmons, Salter)^[22]、德国 (Prandtl, Reichardt)^[23] 和美国 (Dryden and collaborators)^[24,25] 进行。值得一提的是，泰勒率先提出了均匀各向同性湍流的概念，即流场中所有点在任意方向统计性质完全相同的湍流。尽管这是一种理想模型，但泰勒认为，可以通过此模型得到真实湍流中的一些共性结论，用来研究湍流的物质输运、能量输运等特性。均匀各向同性湍流在后来的湍流理论研究中有重要地位。冯·卡门和霍华斯 (Howarth) 从 NS 方程出发，引入正交坐标系的

张量运算，简化了泰勒从均匀各向同性湍流中推导的动力学微分方程，导出了卡门-霍华斯 (Karman-Howarth) 方程。1939 年，柯莫哥洛夫 (Kolmogorov) 开始了另一项研究工作，他提出小尺度湍流具有局部各向同性的假设，通过量纲分析，得到了著名的柯式湍谱 $-5/3$ 次率，这一研究成果在 1941 年发表^[26]，但直到第二次世界大战结束之前，在俄罗斯以外的地方一直鲜为人知，随后的实验测得的一维湍谱很好的证明了 $-5/3$ 率的正确性，柯氏局部各向同性湍流假设把统计理论引向实际应用，因而获得很高评价^[27]。在这之后的二三十年，湍流理论的大部分研究成果都出自统计理论，在 20 世纪 60 年代，湍流的相干结构被发现，为湍流研究带来了新的发展，大量学者开始对湍流拟序结构开展研究^[28-32]。此外，随着计算机的发展，人们也开始尝试采用高级湍流模式研究雷诺方程封闭性的问题^[33]。尽管人们对湍流问题的认识越来越深刻，但是鉴于湍流问题本身的复杂性，直到现在人们依然缺乏对湍流根本性的理解，湍流问题的本质是非线性问题，如果该问题能被解决，将为数学和物理学以及整个非线性科学的发展带来重大变革，加深人们对世界的认识。

1.1.3 可压缩湍流研究

可压缩流动指流体密度随流动状态变化不可忽略的流动，它对应 N-S (Navier-Stokes) 方程中速度的散度不为 0。可压缩流动广泛存在于航空航天的应用中。相比于不可压缩流动，可压缩流动的研究对工程实践具有更强的指导作用，而可压缩的湍流问题是可压缩流动中最引人关注的重点和难度问题之一。研究发现，在飞行器表面流动状态从层流转捩为湍流后会使壁面摩擦阻力系数和热流系数提升数倍^[34]，而可压缩湍流是高超声速飞行器表面主要流动状态。研究清楚可压缩湍流流动特性对于火箭、导弹、高超声速飞行器的减阻防热，超声速民用飞机降噪，超燃冲压发动机提高燃料利用率等有重要意义。我国在 2006 年发布的《国家中长期科学和技术发展规划纲要 (2006-2020)》中也把可压缩湍流列为了重点研究项目，因此开展可压缩湍流的相关研究是能够支撑工程发展、提高国防实力、增强国家核心竞争力的重大需求性工作。

目前，湍流研究的大部分成果都集中在不可压缩流动中。不可压缩流动重点关注雷诺数 (Reynolds 数) 效应，而对于可压缩流动，在关注雷诺数效应的同时还要考虑另一个重要的无量纲参数，马赫数 (Mach 数, Ma)，Ma 可以衡量流动压缩性的强弱，考虑压缩性效应是可压缩流动不同于不可压缩流动的最明显特征。此外，在可压缩流动中，密度变化一般由热力学方程给出，因此需要将热力学方程引入到流动方程中联立求解，流动方程中的能量方程便不能解耦于质量方程和动量方程。当可压缩流动的马赫数较大时，激波是普遍存在的现象，激波与边界层的相互干扰会造成流动的分离，还会产生具有各种频率特征的流动结构^[35]。综上所述，相对于不可压缩流动，可压缩流动尤其是可压缩湍流是更为复杂和难以研究的问题。

可压缩湍流研究起步较晚，研究成果相比于不可压缩湍流更少，此处简要回顾一些可压缩湍流研究中的重要进展。1962 年，Morkovin 提出当流动马赫数不

是很高时，流动主要以外压缩性为主，即湍流流场中热力学变量的脉动量相比于平均量为小量。因此，只需要考虑平均流动受可压缩效应的影响，该理论在可压缩湍流研究中具有里程碑的意义。在 Morkovin 假设下，可压缩湍流的一些统计特征将类似于不可压缩湍流，因而在经过一些修正后，在可压缩湍流中也可以使用不可压缩湍流中的理论^[36]。Van Dierst 曾在不可压缩流动中考虑粘性系数与温度变化关系，修正了平均速度剖面，使之能与对数曲线重合^[37]。Morkovin 假设使得 Van Dierst 能够在可压缩流动中得到应用。可压缩边界层的平均速度剖面经 Van Dierst 变换后也存在对数律和壁面律，其结果可与不可压流动的对数曲线重合。对 Morkovin 假设有效性的检查，是可压缩湍流研究中的一个重要问题。Coleman 等人用直接数值模拟验证了来流马赫数为 3 的等温槽道 Morkovin 假设的有效性^[38]。Duan 等人验证了来流马赫数为 5 时变壁温情况下 Morkovin 假设的有效性^[39]。1977 年，Bradshaw 将 Morkovin 假设的适用范围界定到马赫数小于 5 的情况^[40]，但目前的研究表明，Morkovin 假设可能比预期的适用范围更大^[41,42]。

强雷诺比拟理论 (Strong Reynolds Analogy, SRA) 是可压缩湍流研究过程中又一个重要成果，比拟理论是指当两个物理现象的控制方程相似时，通过测定一个物理现象能够推测另一个物理现象。Morkovin 给出了湍流速度脉动和温度脉动之间的联系^[36]。最初的强雷诺比拟理论假设总温基本不变，总温脉动接近于 0，主要适用于绝热壁的情况，但在之后的研究中发现，由于此假设的严苛性，强雷诺比拟理论的普适性并不好，总温脉动接近于 0 的条件在一般情况下很难满足。1974 年，Cebeci 等学者扩展了 SRA 的适用范围^[43]，将 SRA 从只适用于绝热壁推广到了可适用于非绝热壁情况。后续有其它学者不断发展各式 SRA 理论，SRA 的应用范围得到进一步扩展^[44-47]。

此外，湍流相干结构（拟序结构）的研究也是湍流研究中一个重要的研究领域。1967 年，Kline 等人首次在实验中发现了湍流边界层中的条带结构^[28]。在边界层近壁区，存在高、低速交替的速度条带，条带结构会不断上扬，之后破碎，这一过程被称之为湍流猝发，湍流猝发后来被认为是判断湍流边界层的标志性特征。湍流条带的上抛下扫，实际上也和湍流中的涡结构的运动密切相关。而且，这种上抛下扫的过程，贡献了 60%-80% 的雷诺应力，也被认为流体转换为湍流后会在壁面产生高摩阻的主要原因。Hamilton 等人利用直接数值模拟研究了近壁面湍流相干结构的动力学过程^[48,49]。他们认为，条带结构及流向涡与湍流的生成、维持密切相关，湍流发展过程中，准周期性地存在三个过程：条带结构破碎，流向涡生成，条带结构生成。此外，大量的研究发现，在三维流场中，湍流物理量会在远大于湍流最小空间、时间尺度的尺度上存在相比于比小尺度更大的空间或时间相关性，这表明了，在湍流结构运动过程中，大尺度的流动结构可能扮演了更关键的角色^[32,50,51]。相对于不可压缩湍流拟序结构，可压缩湍流拟序结构的研究相对较少，而且目前还存在一些争议。1989 年，Smits 等人通过热线流速计测量了 $Ma_{\infty}=0.1$ 和 $Ma_{\infty}=2.9$ 的流场信号^[52]，率先开展了可压缩湍流

拟序结构的研究。他们的研究发现，对于亚声速和超声速的情况，湍流流向结构的展向间距不变，但质量通量脉动的流向距离在亚声速时是超声速的两倍。随后他们更多的测量印证了随着马赫数和雷诺数的增加，湍流结构流向长度尺度明显减少，但展向尺度不受雷诺数和马赫数的影响^[53]。不过，Ganapathisubramani 等人^[54] 对 $Ma_{\infty}=2$, $Re_{\theta}=35000$ ($Re_{\tau}=5600$) 的高雷诺数流动开展粒子图像测速 (Particle Image Velocimetry, PIV) 的实验结果与 Smits 等人^[53] 的结论正相反，他们实验测得的流向速度脉动的两点相关性表明， $Ma_{\infty}=2$ 时湍流结构流向长度尺度比不可压缩情况的大四倍，在可压缩时展向尺度也略大于不可压缩情况，Ganapathisubramani 等人^[54] 认为产生不同结论的原因在于高雷诺数效应，以及速度脉动和质量通量脉动性质有所不同。Pirozzoli 等人^[55] 对 $Ma_{\infty}=2$ 的平板湍流边界层开展了直接数值模拟以分析可压缩湍流拟序结构，得出了更加不同的结论，与不可压缩情况相比，直接数值模拟测得的湍流结构流向长度尺度保持不变，而与实验数据相比，计算得到的展向长度尺度略大。最近的实验结果表明^[56]，相比于亚声速湍流边界层，超声速的情况下谱分析的最大能量频率具有更长的波长，而湍流结构的展向尺度也更大。可压缩湍流的拟序结构还有待更进一步的研究。

1.2 湍流的数值模拟

和其它自然学科一致，湍流的研究方法主要有实验、理论、计算。在湍流研究的早期，主要以实验研究为主，比如雷诺的圆管实验，普朗特的水槽实验等。随着数学工具的发展与完善，湍流的理论研究也蓬勃发展起来，比如 20 世纪 30 年代由泰勒引领的湍流统计理论研究热潮。流体力学和湍流的计算研究，从 20 世纪 50 年代，计算物理概念出现之后，也随之出现，并伴随着计算机的出现，通过计算来研究湍流问题逐渐变得火热。实验，理论，计算三种方法各有优缺点，可以相互支撑，共同促进解决湍流难题。目前的湍流实验研究主要借助风洞、水洞等实验设备，一般来说，结果较为可靠，所获的经验能够直接指导工程实践，但实验方法会受技术水平、模型尺寸等因素的影响，适用范围较窄，一次实验能获得的可用数据较少，且实验也可能会存在实验成本高昂、花费的时间周期长等问题。理论研究能够揭示出物理规律，可以让人们清晰地理解物理现象本质，但受限于数学工具的发展，目前只能在简单的模型和简化的情况下开展理论分析，对于复杂的流动情况，目前的数学发展水平还不足以支撑理论研究顺利开展。相比于实验与理论，湍流计算研究的地位随着计算机的出现以及计算机性能的不断提升而持续升高，目前已经成为了湍流研究的主要手段，计算能够求解理论无法分析的复杂流动问题，还可以模拟实验无法复现的流动问题。计算能提供更多的数据，可以让人们深入观察流动现象，既能够弥补实验的不足，又可以为理论研究的发展提供指导。目前，计算研究可压缩湍流的方法主要有三种，分别是 Reynolds 平均数值模拟 (Reynolds-Averaged Navier-Stokes Simulation, RANS)，大涡模拟 (Large Eddy Simulation, LES) 和直接数值模拟 (Direct Numerical

Simulation, DNS), 本小节对这三种方法分别进行简单的介绍。

1.2.1 雷诺平均模拟方法

RANS 方法是目前工程中最常用的一种流体力学数值模拟方法。它主要基于雷诺平均方程，将瞬时量分为平均量和脉动量两个部分，通过各种假设对脉动量贡献进行建模，使方程封闭。根据对脉动量建模方式的不同，RANS 方法又分为雷诺应力模型和涡粘模型。相比于雷诺应力模型，涡粘模型的计算量更小，在工程上运用更为广泛，涡粘模型根据建模所需方程数不同又分为代数模型、一方程模型、二方程模型等。常见的代数模型有 Baldwin-Lomax 模型，一方程模型有 Spalart-Allmaras 模型、Baldwin-Barth 模型等，二方程模型有 $k - \omega$ 模型、 $k - \epsilon$ 模型等。RANS 方法需要理论与经验的指导，而由于湍流理论本身发展不完善，所以 RANS 方法往往精度相对较低，建立的模型普适性一般也较差，对包含流动转捩、分离、激波的复杂情况难以处理。RANS 更适用于流动较为简单，对精度要求不高，并需要快速获得模拟结果的工程情况。

1.2.2 大涡模拟方法

LES 的主要理论基础是 Kolmogorov 的能量级串理论，Kolmogorov 认为大尺度的湍流结构包含着大部分的能量，小尺度的湍流结构的能量相对较小，湍流的能量传递是从大尺度结构传递到小尺度结构，即大涡破碎成小涡，而在 Kolmogorov 尺度，湍流结构表现出均匀各向同性的特征。因此，大尺度的流动结构通过方程直接求解，而小尺度的湍流脉动对大尺度湍流脉动的作用则利用建模的方式来实现，最终使方程封闭。在壁湍流的 LES 中，目前较常见的模型有 Smagorinsky 的 SGS 模型、尺度相似模型、动力模型等。LES 相对于 RANS，大尺度的湍流结果是通过直接求解方程得到的，因而对流动特征的反映较好，而且通过湍流模型也具有捕捉小尺度湍流脉动的能力，精度相比 RANS 更高，但与 RANS 相比，LES 的计算量更大。LES 方法也会存在模型误差，另外 LES 计算时，可通过人为调整滤波的方式来影响流场演算结果，而不同的人可能采用不同的滤波尺度和方式，导致不同人 LES 模拟结果可能存在差异，降低了 LES 的普适性。

1.2.3 直接数值模拟方法

直接数值模拟 (DNS) 只需要给定初始条件和边界条件，就能够捕捉到全部时间和空间尺度的湍流流动。而且它采用对流体力学的控制方程 Navier-Stokes 直接求解的方式，不需要任何模型，因而没有模型误差和人为参数干扰。DNS 能够获得全部的流动信息，捕捉到流场中所有的流动结构，通过这些数据信息，可以让人们对湍流的生成、维持、能量输运、耗散机理进行分析和研究，为湍流建模和湍流控制的研究提供依据。DNS 也可以获得很多实验中难以测量的湍流脉动信息和非定常数据，并能模拟一些难以开展实验的复杂或极端的流动情况。Pope 曾在书中指出“DNS 的描述水平和准确性是无可比拟 (unrivalled) 的” [57]，

DNS 也对于人们理解湍流做出了突出的贡献^[58]。

直接数值模拟最早被引入湍流研究是在 1972 年，Orszag 和 Patterson^[59] 采用 32^3 个网格对 $Re_\lambda = 35$ 的均匀各向同性湍流开展了 DNS，打开了湍流 DNS 研究的大门。Orszag 等人的计算结果在当时并不受到主流的认可，而且以现在的标准来看，他们的网格分辨率确实是不够的，不过这依然是一项里程碑式的工作。1981 年，Rogallo 等人^[60] 对 Orszag 等人的计算方法进行了改进，再次对均匀各向同性湍流开展了直接数值模拟，他们把 DNS 计算结果与实验和理论进行了对比，取得了很好的效果，他们的工作为后续均匀各向同性湍流模拟提供了标准，数据也被广泛用于湍流建模当中。1982 年，Moin 和 Kim^[61] 首次对不可压缩槽道湍流开展了 LES 研究，他们的计算结果与实验观察相吻合。1987 年，Kim 等人^[62] 首次对不可压缩槽道湍流开展了 DNS 研究，摩擦雷诺数 $Re_\tau = 180$ ，他们还给出了湍流的统计特性，开启了 DNS 对壁湍流机理研究的先河。在此过程中，DNS 也逐渐被主流科学界所认可，被当成一种湍流科学的研究手段。1988 年，Spalart^[63] 等人首次对不可压平板湍流边界层开展了 DNS 研究，在此之后，1994 年，Guo 等^[64] 首次对可压缩平板湍流边界层开展了时间模式 DNS 研究。1995 年，Rai 等人^[65] 最早对来流马赫数为 2.25 的可压缩平板边界层开展了空间模式的 DNS。时间模式 DNS 相较于空间模式的 DNS 需要添加能够把湍流时间增长与边界层空间增长联系起来的假设，因此具有一定的模型误差，难以精确反映边界层转换过程，但计算量更小。众多的研究者已经采用时间模式 DNS 对壁面湍流边界层做了系统的研究^[39,41,66,67]，随着计算机计算能力的增强，越来越多的研究者选择使用空间模式的 DNS 进行湍流边界层研究，比如李新亮^[68–70]，Pirozzoli^[55,71–73] 等学者。

DNS 的缺点是计算量太大。由于湍流的尺度跨度随雷诺数 (Re) 的增加增长很快，DNS 计算时需要将所有尺度的湍流涡都识别出来，这意味着网格尺度需要达到最小湍流结构对应的尺度量级，计算域尺度也要大于湍流场中最大的特征尺度，因而在雷诺数较高时将需要非常多网格点，有人估计，DNS 计算需要的网格量与雷诺数的关系为 $Re_L^{9/4}$ ^[74]，考虑到真实飞行器飞行时的雷诺数往往在 10^7 以上，对应的网格数将是海量的。此外，湍流模拟时还需正确分辨湍流脉动随时间变化的多尺度特征，对时间步长也提出了较高要求，预估 DNS 计算完成需要的时间步数与雷诺数的关系为 $Re_L^{1/2}$ ^[74]，Spalart 等人^[75] 在 2000 年以当时的计算机发展水平推测，到 2080 年人们才有能力对大型飞机整机开展直接数值模拟。因此制约 DNS 发展的最大瓶颈之一便是计算能力的欠缺。

1.3 高性能计算发展与异构并行计算

1.3.1 高性能计算发展历史

第二次世界大战期间，美国主导设立了著名的曼哈顿计划用以研制核武器，计算物理的概念在这一时期也被提出。计算物理主要研究如何通过数值方法来分析可量化的物理问题，在人类第一颗核武器的研制过程中，计算物理发挥了重

要的作用^[76]。后来计算物理衍生出了众多分支，包括计算化学（Computational Chemistry, CC）、计算电动力学（Computational Electromagnetics, CE）、计算流体力学（Computational Fluid Dynamics, CFD）等。计算物理与计算紧密相连，它的发展也伴随着计算机技术的不断发展。在曼哈顿计划期间，1946年，冯·诺依曼等人（John von Neumann）发明了人类历史上第一台通用计算机 ENIAC（Electronic Numerical Integrator And Computer）^[77]。ENIAC 由 17468 根电子管，7200 根二级管，七万余个电阻器，一万余个电容器，一千五百个继电器，六千余个开关组成，重达 30 吨，占地约 170 平方米，拥有每秒 5000 次的计算能力。冯·诺依曼也由于在通用计算机架构理论上的贡献被称之为“现代计算机之父”。20世纪 60 年代末已经出现了小型计算机。我国第一台电子管计算机是中国科学院计算技术研究所研制的 103 机，主要用于两弹一星事业发展。人类第一台超级计算机为 CDC6600^[78]，最初也是服务于核武器研制，CDC6600 由 Cray 公司的创始人西蒙·克雷（Seymor Cray）领导研制，克雷在 20 世纪 60 年代受美国原子能委员会的委托，开始研发巨型计算机，他在 1964 年成功研发了 CDC6600 巨型计算机，该机由四十万个晶体管组成，拥有 3MFlops 的计算能力。CDC6600 在当时被高校、研究所以及核武器研制单位采购了超过 100 台。西蒙·克雷也由于在巨型计算机研制上的贡献被后人称之为“巨型机之父”。20 世纪 80 年代初个人计算机开始出现，1971 年 Kenbak 公司推出的 Kenbak-1^[79] 被计算机历史博物馆认为是世界历史上第一台个人计算机。Kenbak-1 由小型集成电路制成，没有使用微处理器。20 世纪 80 年代，由于美苏冷战的升级和军备竞赛的加剧，1983 年，美国制定了用于建立反弹道导弹系统的“星球大战计划”（Strategic Defense Initiative, SDI），在这个过程中，制定了“战略计算机计划”（Strategic Computer Program, SCP），该计划研制出了人类历史上第一台多 CPU 并行机，达到了 10 亿次的计算能力。1985 年 Cray 公司发布了具有 8 个处理器核心的 Cary-2 计算机^[80]，它的计算能力达到 1.9GFlops，标志着高性能计算机发展进入 G 级机时代。1996 年 Intel 公司成功研发出 ASCI-RED 超级计算机^[81]，计算能力达到 1.453Tflops，标志着高性能计算机发展进入 T 级机时代。2008 年 IBM 公司研制出超级计算机“走鹃”（ReadRunner）^[82]，浮点运算能力达到 1.026Pflops，带领高性能计算机进入 P 级机时代，2008 年美国 DARPA 信息处理技术办公室（IPTO）公布了 E 级机的计算研究报告，2022 年 5 月 30 日美国橡树岭国家实验室公布了 CPU（Central Processing Unit）-GPU（Graphics Processing Unit）异构超算“前沿”（Frontier）^[83]，其浮点运算性能超过了每秒百亿亿次，标志着超级计算发展进入 E 级机时代，实际上中华人民共和国中央人民政府网站早在 2018 年就刊文表示我国三个 E 级超算原型机系统完成交付^[84]，目前个别机器已经验收，我国的超级计算机发展水平已经可以与世界一流水平一较高下。

1.3.2 异构计算

从 1985 年 Gray 公司 Gray-2 带领超级计算进入 GFlops 的 G 级机时代以来，超级计算机的运算能力基本每 10 左右年提升一个代级，从 G 级机，T 级机，P

级机，到如今的 E 级机。不过在进入 P 级机时代之后，伴随着摩尔定律所描述的芯片上的晶体管每 2 年提升一倍的速度的减缓，人们开始思考新的高性能计算机发展模式，异构、众核的思想为计算机带来了新的性能提升。异构计算是指计算节点分主机端和设备端，分别注重逻辑处理和浮点计算，而众核处理器是指有几十、上百核心以上的处理器，而它往往成为异构计算时设备端的用以浮点运算的加速卡。

异构计算设备端的加速卡，与主机端有不同专长的处理能力，合理的异构程序可以使两者优势互补，形成合力，体现出比单一芯片更好的性能功效比。异构计算在最近几年受到的关注越来越多，相比于追求全能的同构计算，让更擅长某些工作的芯片去做对应的工作，显然更具收益^[85–88]。

1.3.3 GPU 与众核处理器

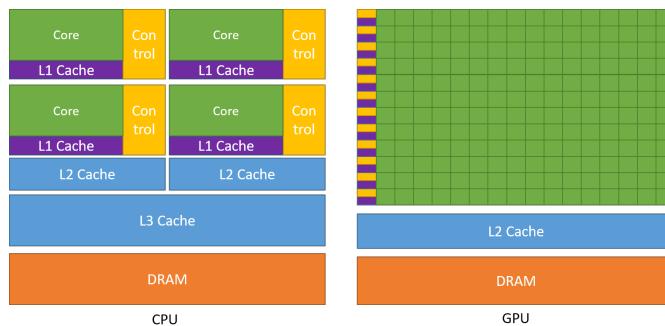


图 1.2 CPU 与 GPU 的架构差异^[1]

Figure 1.2 Architectural differences between CPU and GPU^[1]

众核处理器比如 GPU (Graphics Processing Unit)，一般会成为异构计算时设备端负责浮点计算的加速卡，这主要由于，GPU 把几乎所有的晶体管都用于计算，而 CPU 把大量晶体管用于高速缓存和逻辑控制，如图 1.2，常见的众核处理器除了 GPU 之外，还有 Intel 公司的 MIC 加速卡，以及国产的神威系列众核芯片。而目前最主流的加速卡为 GPU。表 1.1展示了 2022 年 5 月的 TOP500 超级计算机排行榜前十名，排名前十的超级计算机中有八台采用了 GPU 作为硬件加速器。根据 Green500 榜单^[89]，排名前十的超级计算机中只有一台没有应用 GPU 加速。GPU 名为“图形处理器”，正如其名，它最早是被设计用于计算机图像显示的专用芯片，图形显示操作过程中需要对大量像素点进行密集的浮点运算，为满足这种需求，GPU 设置了大量的计算核心，尽管每个计算核心的时钟频率远不及 CPU 核心的频率，但针对图形计算这种像素点可以大规模并行计算的应用，GPU 架构可以显著提高总体的浮点计算速度^[1]，GPU 最早在图形显示这个应用领域发力，但随后它在科学计算和人工智能训练等应用领域的潜力也被发现，此外随着晶圆制造水平的提升，GPU 受益于最初设计时的众核架构思想，只要芯片越大，就能容纳更多的核心，GPU 性能提升不完全依赖于晶体管制程提升。目前，GPU 的浮点运算能力和带宽相比 CPU 有量级的提高，如今成为了最主流的

高性能计算加速卡。但并不是所有的应用都适用于 GPU，鉴于 CPU 和 GPU 硬件特点的不同，它们的适用环境存在差异，对于一些对缓存要求高，逻辑操作繁复的应用，如计算机操作系统，更适合在 CPU 上运行，而对于一些计算时不存在大量数据依赖且需要密集浮点运算的应用程序则更适合 GPU 的硬件环境。

表 1.1 截至 2022 年 5 月，超级计算机 TOP500 榜单前 10 名

Table 1.1 Top 10 from the TOP500 list of supercomputers, as of May 2022^[90]

Rank	Name	GPU used?	GPU model
1	Frontier	Yes	AMD MI250X
2	Fugaku	No	/
3	LUMI	Yes	AMD MI250X
4	Summit	Yes	NVIDIA GV100
5	Sierra	Yes	NVIDIA GV100
6	Sunway TaihuLight	No	/
7	Perlmutter	Yes	NVIDIA A100
8	Selene	Yes	NVIDIA A100
9	Tianhe-2A	No	/
10	Adastra	Yes	AMD MI250X

GPU 的大部分晶体管被用来组成数量极多的计算核心，如图 1.3，这些核心在 NVIDIA 的硬件中被称之为 CUDA (Compute Unified Device Architecture) Core，GPU 的硬件采用多级的管理形式，或许可以从一个类比中理解其组织形式。GPU 由于核心数众多，想管控好每个核心是很复杂的，如果把 GPU 的每个核心比作一个学生，那么 GPU 就是一所拥有数千学生的学校，如果让校长一个人管理所有的学生显然是不现实的，所以要把学校的学生分成很多的班级，校长只需要对班主任下达命令，班主任就可以管理班级里的学生，在 GPU 里，SM (流多处理器，Stream Multiprocessor) 就起到了班主任的角色，它控制了 Block (线程块) 内的线程行为，即便如此，每个班的学生依然过多，班主任还是很难管理好每个学生，所以班级里的学生又被分为了很多小组，这些小组的成员在同一时间进行同样的操作。在 GPU 里，Warp (线程束) 就可以被看成这些小组，Warp 是 GPU 最小的执行单元，Warp 内的线程以 SIMT (Single Instruction Multiple Thread) 的方式执行命令。也可以把 GPU 加速卡看作是一台大规模并行计算机，其上的大量算术单元能够同时执行指令。在程序设计时，也需要根据 GPU 特殊的硬件特点，针对性地设计数值算法和程序结构，平衡通信、计算以及数据访问，知道“学生”能做什么事，并让“学校”中的每一个“老师”知道应该让班级里的“学生”做什么事情，才能发挥出 GPU 硬件最佳的性能。

图 1.3 NVIDIA 费米 GPU 架构示意图^[1]Figure 1.3 NVIDIA Fermi GPU architecture diagram^[1]

1.4 本文的研究内容和组织结构

本文致力于扩宽可压缩湍流直接数值模拟的应用范围，并为可压缩湍流机理研究提供数值工具和具有代表性的湍流数据。本文主要针对困扰可压缩湍流直接数值模拟的两个难题开展探索，其一是可压缩湍流直接数值模拟对计算量需求极大的难题。本文开发了一套利用 GPU 加速的异构有限差分方法软件 OpenCFD-SCU，这款软件是开源的，本文详细介绍了该软件的软件架构，计算通讯重叠方法，结合硬件的程序优化技巧等，并展示了大规模测试的结果。经测试，在相同 MPI 进程下 OpenCFD-SCU 比 CPU 端的成熟软件 OpenCFD-SC 加速 200 倍以上，弱扩展性在 24,576 个 GPU 上达到 98.7% 的并行效率，性能达到 17.1Pflops。第二个难题是对高超声速复杂问题开展直接数值模拟时存在计算鲁棒性与计算精度相矛盾的问题。本文利用 Jameson 的人工粘性系数设计了激波识别器，并基于此开发了一套 7 阶迎风格式、7 阶 WENO (Weighted Essentially Nonoscillatory) 格式与 5 阶 WENO 格式混合的混合格式，该格式大幅的提升了计算鲁棒性，在对高超声速压缩折角流动这类包含激波、湍流、流动分离的复杂问题开展高分辨率直接数值模拟时也能保证计算全程不发散，而且在对多个算例开展大规模直接数值模拟时，发现大部分网格点都采用 7 阶迎风格式进行计算，证明该格式兼具了高精度和低耗散的需求。本文利用 GPU 加速的高性能程序和混合格式建立了一个开放的湍流数据库，包含多个典型算例，本文对这些算例进行了初步的分析和比较。

本文的组织结构如下：

第一章：绪论部分，主要对湍流这一物理现象，湍流研究的数值模拟方法，高性能计算发展进行了介绍。

第二章：控制方程及数值方法。主要介绍了 OpenCFD-SCU 所涉及到的数值方法，包括 Steger-Warming 流通矢量分裂，特征重构，无粘项差分格式及混合格式，边界差分格式，粘性项差分格式，时间推进方法等。

第三章：高精度有限差分软件 OpenCFD-SCU 框架与结构。主要介绍了 OpenCFD-SCU 的程序总体框架设计，流水线设计，MPI 三维剖分方式等。

第四章：OpenCFD-SCU 在 CPU-GPU 异构体系下的优化方法。主要介绍了 OpenCFD-SCU 针对 AMD Vega20 架构 GPU 的优化方法，包括硬件环境介绍，程序计算热点分析，对核函数的若干优化方法。

第五章：OpenCFD-SCU 性能与扩展性测试。展示 OpenCFD-SCU 大规模测试的结果，与 CPU 相比的加速效果，强扩展性，弱扩展性，性能估计，跨平台测试。

第六章：展示了使用 OpenCFD-SCU 和混合格式建立的一个开放的湍流数据库，包括从超声速到高超声速的几个典型问题：马赫 2.9 压缩折角流动；马赫 6 压缩折角流动；马赫 10 平板湍流边界层；马赫 6 钝锥湍流边界层；马赫 6 升力体转捩。展示了这些算例的初步分析结果，并进行了算例数据之间的比较。

第七章：结论部分，总结了本文的研究结果，提出了对未来工作的展望。

附录：包含了 OpenCFD-SCU 的使用指南，控制文件参数含义，以及湍流数据库文件说明。

第 2 章 控制方程及数值方法

2.1 引言

这一部分详细介绍了可压缩湍流直接数值模拟求解的控制方程及数值方法。方程为曲线坐标下的可压缩的三维 N-S 方程组，求解方法采用有限差分方法。本章介绍了几个常用的有限差分方法，如 2 阶的无波动、无自由参数的耗散差分格式 (NND 格式)，5 阶与 7 阶的加权基本无振荡格式 (WENO 格式)，WENO-SYMBO 格式，6 阶优化保单调格式 (OMP6 格式)，线性差分格式，以及混合差分格式，还对差分计算中常用的 Steger-Warming 流通矢量分裂、特征重构等可以提升差分求解的质量与求解稳定性的技术进行了介绍，最后介绍了边界格式与 Runge-Kutta 时间推进方法。

2.2 控制方程

曲线坐标下的无量纲化的三维 N-S 方程组形式如下：

$$\frac{\partial \widetilde{\mathbf{U}}}{\partial \tau} + \frac{\partial(\widetilde{\mathbf{F}} - \widetilde{\mathbf{F}}_v)}{\partial \xi} + \frac{\partial(\widetilde{\mathbf{G}} - \widetilde{\mathbf{G}}_v)}{\partial \eta} + \frac{\partial(\widetilde{\mathbf{H}} - \widetilde{\mathbf{H}}_v)}{\partial \zeta} = 0, \quad (2.1)$$

该方程由连续方程、动量方程、能量方程组成。采用曲线坐标下的 N-S 方程，可以对任意曲面计算域的算例进行求解，增强计算程序的通用性，方程中的 $\widetilde{\mathbf{F}}$ 、 $\widetilde{\mathbf{G}}$ 和 $\widetilde{\mathbf{H}}$ 表示曲线坐标系 ξ 方向、 η 方向和 ζ 方向的无粘通量， $\widetilde{\mathbf{F}}_v$ 、 $\widetilde{\mathbf{G}}_v$ 和 $\widetilde{\mathbf{H}}_v$ 表示 ξ 方向、 η 方向和 ζ 方向的粘性通量。

尽管计算区域的物理空间可以在曲线坐标系表示，但在差分离散时依然需要物理量在笛卡尔坐标系下进行，物理空间的曲线坐标系 (ξ, η, ζ, τ) 与计算空间的笛卡尔坐标系 (x, y, z, t) 的坐标变换关系如下：

$$\begin{aligned} \xi &= \xi(x, y, z, t), \\ \eta &= \eta(x, y, z, t), \\ \zeta &= \zeta(x, y, z, t), \\ \tau &= t. \end{aligned} \quad (2.2)$$

假设 τ 只是 t 的函数而不依赖于 x 、 y ，和 z 。那么坐标变换将满足以下关系：

$$\begin{aligned} \xi_x &= J(y_\eta z_\zeta - z_\eta y_\zeta), & \zeta_x &= J(y_\xi z_\eta - z_\xi y_\eta), & \eta_x &= J(y_\zeta z_\xi - z_\zeta y_\xi), \\ \xi_y &= J(z_\eta x_\zeta - x_\eta z_\zeta), & \zeta_y &= J(z_\xi x_\eta - x_\xi z_\eta), & \eta_y &= J(z_\zeta x_\xi - x_\zeta z_\xi), \\ \xi_z &= J(x_\eta y_\zeta - y_\eta x_\zeta), & \zeta_z &= J(x_\xi y_\eta - y_\xi x_\eta), & \eta_z &= J(x_\zeta y_\xi - y_\zeta x_\xi). \end{aligned} \quad (2.3)$$

其中 J 表示坐标变换时的 Jacobian 行列式，其值为：

$$J = \left| \frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} \right| = \left| \frac{\partial(x, y, z)}{\partial(\xi, \eta, \zeta)} \right|^{-1}. \quad (2.4)$$

因此，在(2.1)中，守恒变量 \mathbf{U} 就可以表示为：

$$\tilde{\mathbf{U}} = J^{-1} \mathbf{U} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{bmatrix}. \quad (2.5)$$

无粘通量可以表示为：

$$\tilde{\mathbf{F}} = J^{-1} \begin{bmatrix} \rho \tilde{u} \\ \rho u \tilde{u} + \xi_x p \\ \rho v \tilde{u} + \xi_y p \\ \rho w \tilde{u} + \xi_z p \\ (E + p) \tilde{u} \end{bmatrix}, \quad \tilde{\mathbf{G}} = J^{-1} \begin{bmatrix} \rho \tilde{v} \\ \rho u \tilde{v} + \eta_x p \\ \rho v \tilde{v} + \eta_y p \\ \rho w \tilde{v} + \eta_z p \\ (E + p) \tilde{v} \end{bmatrix}, \quad \tilde{\mathbf{H}} = J^{-1} \begin{bmatrix} \rho \tilde{w} \\ \rho u \tilde{w} + \zeta_x p \\ \rho v \tilde{w} + \zeta_y p \\ \rho w \tilde{w} + \zeta_z p \\ (E + p) \tilde{w} \end{bmatrix}, \quad (2.6)$$

这里的 \tilde{u} , \tilde{v} , \tilde{w} , 为物理空间曲线坐标系下沿坐标轴三个方向的速度，其与计算空间速度的变换关系为：

$$\begin{aligned} \tilde{u} &= \xi_t + \xi_x u + \xi_y v + \xi_z w, \\ \tilde{v} &= \eta_t + \eta_x u + \eta_y v + \eta_z w, \\ \tilde{w} &= \zeta_t + \zeta_x u + \zeta_y v + \zeta_z w \end{aligned} \quad (2.7)$$

式(2.6)中的能量 E 由内能和动能组成：

$$E = \frac{p}{\gamma - 1} + \rho \frac{u^2 + v^2 + w^2}{2}. \quad (2.8)$$

粘性通量可以表示为：

$$\begin{aligned} \tilde{\mathbf{F}}_v &= J^{-1} \begin{bmatrix} 0 \\ \xi_x \tau_{xx} + \xi_y \tau_{xy} + \xi_z \tau_{xz} \\ \xi_x \tau_{xy} + \xi_y \tau_{yy} + \xi_z \tau_{yz} \\ \xi_x \tau_{xz} + \xi_y \tau_{yz} + \xi_z \tau_{zz} \\ \xi_x \Theta_x + \xi_y \Theta_y + \xi_z \Theta_z \end{bmatrix}, \quad \tilde{\mathbf{G}}_v = J^{-1} \begin{bmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} + \eta_z \tau_{xz} \\ \eta_x \tau_{xy} + \eta_y \tau_{yy} + \eta_z \tau_{yz} \\ \eta_x \tau_{xz} + \eta_y \tau_{yz} + \eta_z \tau_{zz} \\ \eta_x \Theta_x + \eta_y \Theta_y + \eta_z \Theta_z \end{bmatrix}, \\ \tilde{\mathbf{H}}_v &= J^{-1} \begin{bmatrix} 0 \\ \zeta_x \tau_{xx} + \zeta_y \tau_{xy} + \zeta_z \tau_{xz} \\ \zeta_x \tau_{xy} + \zeta_y \tau_{yy} + \zeta_z \tau_{yz} \\ \zeta_x \tau_{xz} + \zeta_y \tau_{yz} + \zeta_z \tau_{zz} \\ \zeta_x \Theta_x + \zeta_y \Theta_y + \zeta_z \Theta_z \end{bmatrix}. \end{aligned} \quad (2.9)$$

其中 ς_x , ς_y 和 ς_z 表示流体的应力与热传递在三个方向的贡献:

$$\begin{aligned}\Theta_x &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + k \frac{\partial T}{\partial x}, \\ \Theta_y &= u\tau_{xy} + v\tau_{yy} + w\tau_{yz} + k \frac{\partial T}{\partial y}, \\ \Theta_z &= u\tau_{xz} + v\tau_{yz} + w\tau_{zz} + k \frac{\partial T}{\partial z}.\end{aligned}\quad (2.10)$$

流体应力可由牛顿切应力公式得出:

$$\tau_{ij} = \begin{cases} \mu \left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j} \right), & i \neq j, \\ \mu \left(2 \frac{\partial u_i}{\partial x_i} - \frac{2}{3} \operatorname{div} \mathbf{V} \right), & i = j. \end{cases} \quad (2.11)$$

热传递系数可以表示为:

$$k = \frac{C_p \mu}{\operatorname{Pr} \operatorname{Re}} = \frac{\mu}{(\gamma - 1) \operatorname{Ma}^2 \operatorname{Pr} \operatorname{Re}} \quad (2.12)$$

其中 μ 表示无量纲的粘性系数, 在温度不超过 2000k 时, 它可以通过 Sutherland 公式得到:

$$\tilde{\mu} = \mu_0 \left(\frac{\tilde{T}}{\tilde{T}_c} \right)^{3/2} \frac{\tilde{T}_c + T_S}{\tilde{T} + T_S} \quad (2.13)$$

式中 μ_0 为一个大气压下、0°C 时的流体粘性系数, T_S 为 Sutherland 常数, 与气体性质有关。用参考粘性系数无量纲后, Sutherland 公式形式为:

$$\mu = (T)^{3/2} \frac{1 + T_S}{T + T_S} \quad (2.14)$$

无量纲声速 C 可以表示为:

$$C^2 = \frac{T}{\operatorname{Ma}^2}. \quad (2.15)$$

此外, C_p 表示定压比热, Pr , Re 和 Ma 为无量纲参数普朗特数, 雷诺数, 马赫数, 它们根据无量纲时的参考值选取不同会有不同的结果。

一般而言, 无量纲化采用以下方式:

$$\begin{aligned}x &= x_{\text{local}}/L_{\text{ref}}, & u &= u_{\text{local}}/U_{\text{ref}}, \\ t &= t_{\text{local}}U_{\text{ref}}/L_{\text{ref}}, & \rho &= \rho_{\text{local}}/\rho_{\text{ref}}, \\ T &= T_{\text{local}}/T_{\text{ref}}, & p &= p_{\text{local}}/(\rho_{\text{ref}} U_{\text{ref}}^2).\end{aligned}\quad (2.16)$$

那么, 雷诺数 Re , 马赫数 Ma , 普朗特数 Pr 可以表示为:

$$\operatorname{Re} = \frac{\rho_{\text{ref}} U_{\text{ref}} L_{\text{ref}}}{\mu_{\text{ref}}}, \quad \operatorname{Ma} = \frac{U_{\text{ref}}}{C_{\text{ref}}}, \quad C_{\text{ref}}^2 = \gamma R T_{\text{ref}}. \quad (2.17)$$

在计算时，只需要根据实际的参考量物理值给出对应的雷诺数 Re ，马赫数 Ma ，普朗特数 Pr ，程序就可以开始计算，考虑到计算域在几十个毫米到几百个毫米之间，雷诺数 Re 一般采用每单位毫米的雷诺数，对于理想气体而言，比热比 γ 一般为 1.4，普朗特数 Pr 一般为 0.7。

2.3 无粘项数值离散方法

2.3.1 Steger-Warming 流通矢量分离

流通矢量分裂可将流场中的特征波根据传播方向不同分为两个部分，针对两个部分分别使用对应迎风性质的差分格式进行离散，这可以有效增强计算稳定性，抑制混淆误差，削减数值求解过程中存在的非物理振荡^[91]。

Steger-Warming 流通矢量分裂^[92] 是广泛用于超声速流动差分求解的流通矢量分裂方式，其数值耗散相对较小，对流场微结构的捕捉能力更好。它是根据流通矢量 Jacobian 矩阵的特征值将流通矢量分解为一个正流通矢量 \mathbf{f}^+ 和一个负流通矢量 \mathbf{f}^- ，可分别采用后差分格式与正差分格式进行离散。

计算空间任一方向的流通矢量可以被表示为物理空间对应的曲线坐标系下三个方向的流通矢量贡献之和：

$$\mathbf{f} = \alpha_1 \mathbf{f}_1 + \alpha_2 \mathbf{f}_2 + \alpha_3 \mathbf{f}_3. \quad (2.18)$$

那么，在(2.18)中，计算空间 x 方向的流通矢量对应的系数就为：

$$\alpha_1 = \xi_x/J, \alpha_2 = \xi_y/J, \alpha_3 = \xi_z/J$$

y 方向对应的系数为：

$$\alpha_1 = \eta_x/J, \alpha_2 = \eta_y/J, \alpha_3 = \eta_z/J$$

z 方向对应的系数为：

$$\alpha_1 = \varsigma_x/J, \alpha_2 = \varsigma_y/J, \alpha_3 = \varsigma_z/J$$

对于流通矢量的 Jacobian 矩阵 \mathbf{A} ，即 $\mathbf{A} = D(\mathbf{f})/D(\mathbf{U})$ 而言，特征值为：

$$\lambda_1 = \lambda_2 = \lambda_3 = \tilde{V}, \quad \lambda_4 = \tilde{V} - c\sigma, \quad \lambda_5 = \tilde{V} + c\sigma, \quad (2.19)$$

这里的：

$$\tilde{V} = \alpha_1 u + \alpha_2 v + \alpha_3 w, \quad (2.20)$$

$$\sigma = \sqrt{\alpha_1^2 + \alpha_2^2 + \alpha_3^2} \quad (2.21)$$

在分别对 x, y, z 方向进行流通矢量分裂时，其中的系数分别对应于式(2.3.1)，(2.3.1)，(2.3.1)。

把特征值分为正值 λ_k^+ 和负值 λ_k^- ，方式如下：

$$\lambda_k = \lambda_k^+ + \lambda_k^- \quad (k = 1, \dots, 5), \quad (2.22)$$

$$\lambda_k^+ = \frac{\lambda_k + |\lambda_k|}{2}, \quad \lambda_k^- = \frac{\lambda_k - |\lambda_k|}{2}. \quad (2.23)$$

那么在正特征值 λ_k^+ 时, 对应的流通矢量便为正流通矢量 \mathbf{f}^+ , 在负特征值 λ_k^- 时, 对应的流通矢量便为负流通矢量 \mathbf{f}^- , 最终表现形式为:

$$\mathbf{f} = \mathbf{f}^+ + \mathbf{f}^-. \quad (2.24)$$

对于三维流动情况, 用特征值表示的流通矢量具体表现形式为:

$$\mathbf{f} = \frac{\rho}{2\gamma} \begin{bmatrix} 2(\gamma-1)\tilde{\lambda}_1 + \tilde{\lambda}_4 + \tilde{\lambda}_5 \\ 2(\gamma-1)\tilde{\lambda}_1 u + \tilde{\lambda}_4(u - c\tilde{k}_1) + \tilde{\lambda}_5(u + c\tilde{k}_1) \\ 2(\gamma-1)\tilde{\lambda}_1 v + \tilde{\lambda}_4(v - c\tilde{k}_1) + \tilde{\lambda}_5(v + c\tilde{k}_1) \\ 2(\gamma-1)\tilde{\lambda}_1 w + \tilde{\lambda}_4(w - c\tilde{k}_1) + \tilde{\lambda}_5(w + c\tilde{k}_1) \\ (\gamma-1)\tilde{\lambda}_1 V + W + \tilde{\lambda}_4 v_{c1} + \tilde{\lambda}_5 v_{c2} \end{bmatrix}, \quad (2.25)$$

其中:

$$\begin{aligned} v_{c1} &= \frac{\tilde{\lambda}_4}{2} [(u - c\tilde{k}_1)^2 + (v - c\tilde{k}_1)^2 + (w - c\tilde{k}_1)^2], \\ v_{c2} &= \frac{\tilde{\lambda}_5}{2} [(u + c\tilde{k}_1)^2 + (v + c\tilde{k}_1)^2 + (w + c\tilde{k}_1)^2]. \end{aligned} \quad (2.26)$$

这里:

$$V = \frac{1}{2}(u^2 + v^2 + w^2), \quad (2.27)$$

$$W = \frac{(3-\gamma)(\tilde{\lambda}_4 + \tilde{\lambda}_5)c^2}{2(\gamma-1)}, \quad (2.28)$$

$$\tilde{k}_1 = \frac{\alpha_1}{\sigma_1}, \quad \tilde{k}_2 = \frac{\alpha_2}{\sigma_2}, \quad \tilde{k}_3 = \frac{\alpha_3}{\sigma_3}. \quad (2.29)$$

2.3.2 使用特征变量的通量重构

特征重构是将流通量转换到特征空间, 在特征空间对特征流通变量进行重构。使用特征重构可以明显增强计算鲁棒性, 减小非物理振荡, 但由于变换过程需要进行矩阵运算, 会导致求解时的计算量增大很多。在对一些计算鲁棒性要求较高的算例求解时可以添加此方法。

由于差分计算时使用的差分格式一般都为守恒型差分格式, 即可以表示为以下形式的差分格式: $\left. \frac{\partial f}{\partial x} \right|_j = \frac{\hat{f}_{j+1/2} - \hat{f}_{j-1/2}}{\Delta x}$ 。所以本部分介绍特征变换时, 以其中

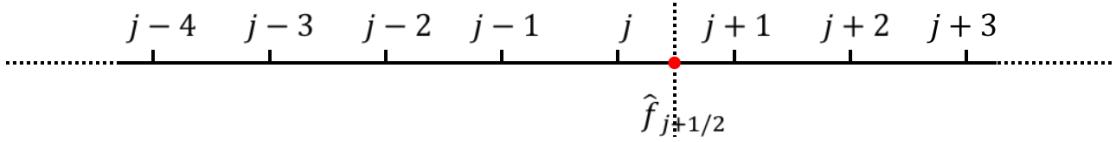


图 2.1 差分模板点

Figure 2.1 Difference scheme stencil point

$j + 1/2$ 的一支进行介绍。如图 2.1 展示了 7 阶线性迎风格式算正方向 $j + 1/2$ 的一支的差分 $\frac{\partial f}{\partial x}|_{j+1/2}$ 时所用的模板点。守恒型的差分格式能够保证整个差分区域的积分守恒律严格满足，只要边界上没有误差，总积分就不会有任何误差，对于求解质量提升很有帮助。

虽然无粘通量 $f(\mathbf{U})$ 在整个计算域并不能表示为守恒变量 \mathbf{U} 的常系数方程，但在差分模板点上，可以认为局部冻结系数。那么在如图 2.1 的差分模板点上，无粘通量的导数可以表示为：

$$\hat{f}_{j+1/2} = S^{-1} F(S f_{j-4}, S f_{j-3}, \dots, S f_{j+3}). \quad (2.30)$$

其中 S^{-1} 和 S 分别为 $A(U_{j+1/2})$ 的左、右特征矩阵，即： $A(U_{j+1/2}) = S^{-1} \Lambda S$ 这里的 $\mathbf{U}_{j+1/2}$ 可以通过简单平均的方式得到 (Roe 格式效果更好)：

$$\mathbf{U}_{j+1/2} = \frac{\mathbf{U}_j + \mathbf{U}_{j+1}}{2}. \quad (2.31)$$

其中，特征值 Λ 可以表示为 (这里的特征值实际上与(2.19)一致)：

$$\Lambda = \sigma \text{diag}(u_n, u_n, u_n, u_n - c, u_n + c). \quad (2.32)$$

特征矩阵 \mathbf{S} 有很多种表示方法，这里采用的构造形式为：

$$\mathbf{S}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ u & l_1 & m_1 & u - cn_1 & u + cn_1 \\ v & l_2 & m_2 & v - cn_2 & v + cn_2 \\ w & l_3 & m_3 & w - cn_3 & w + cn_3 \\ V & u_l & u_m & H - cu_n & H + cu_n \end{bmatrix}, \quad (2.33)$$

$$\mathbf{S} = \begin{bmatrix} 1 - \kappa V & \kappa u & \kappa v & \kappa w & -\kappa \\ -u_l & l_1 & l_2 & l_3 & 0 \\ -u_m & m_1 & m_2 & m_3 & 0 \\ \kappa_1 V + \chi u_n & -\chi n_1 - \kappa_1 u & -\chi n_2 - \kappa_1 v & -\chi n_3 - \kappa_1 w & \kappa_1 \\ \kappa_1 V - \chi u_n & \chi n_1 - \kappa_1 u & \chi n_2 - \kappa_1 v & \chi n_3 - \kappa_1 w & \kappa_1 \end{bmatrix}. \quad (2.34)$$

其中：

$$H = \frac{E + p}{\rho} = V + \frac{c^2}{\gamma - 1}, \quad (2.35)$$

((2.35)中 V 可由式(2.27)给出)

$$\kappa = \frac{\gamma - 1}{c^2}, \quad \kappa_1 = \frac{\kappa}{2}, \quad \chi = \frac{1}{2c}, \quad (2.36)$$

$$\begin{aligned} u_n &= un_1 + vn_2 + wn_3, \\ u_1 &= ul_1 + vl_2 + wl_3, \\ u_m &= um_1 + vm_2 + wm_3 \end{aligned} \quad (2.37)$$

$\mathbf{l} = (l_1, l_2, l_3)$, $\mathbf{m} = (m_1, m_2, m_3)$, 和 $\mathbf{n} = (n_1, n_2, n_3)$ 是三个正交的单位向量。这些向量也有很多种可给出形式, 这里采用的形式是:

$$(l_1, l_2, l_3) = \begin{cases} (-n_2, n_1, 0)/\sqrt{n_1^2 + n_2^2} & \text{if } |n_3| \leq |n_2|, \\ (-n_3, 0, n_1)/\sqrt{n_1^2 + n_3^2} & \text{otherwise,} \end{cases} \quad (2.38)$$

$$\mathbf{m} = \mathbf{n} \times \mathbf{l} = (n_2 l_3 - n_3 l_2, n_3 l_1 - n_1 l_3, n_1 l_2 - n_2 l_1). \quad (2.39)$$

特征重构可以严格保证局部的特征方向, 对提升数值解质量, 增强计算稳定性很有帮助。但正如上述推导, 其正、逆特征变换增加了很多计算量, 会使整体程序计算时间变慢 30% ~ 100%。

2.3.3 无粘项差分离散方法

无粘项的差分离散一般采用迎风形式的差分, 即对流通矢量分裂后的正负通量, 分别采用具有正向迎风性质的后差分格式和逆向迎风性质的前差分格式。

本部分对常见的迎风型差分进行介绍, 如表 2.1:

表 2.1 差分格式表

Table 2.1 Difference schemes list.

Scheme No.	UDwind difference scheme
1	Second-order NND ^[93]
2	Fifth-order WENO
3	Sixth-order OMP6 ^[94]
4	Seventh-order WENO ^[95]
5	Seventh-order WENO-SYMBO ^[96]
6	Seventh-order UDwind

2.3.3.1 二阶 NND 格式

NND 格式^[93,97]是一种满足熵条件且总变差减小 (TVD 型) 的差分格式，它在空间上具有二阶精度。而且由于它是由二阶中心格式和二阶迎风格式结合而来，具有耗散特征，能够很好的抑制振荡，是一种兼具极强鲁棒性和激波捕捉能力的差分格式，被广泛应用在工程问题的计算中。NND 格式的具体形式如下：

$$f_j'^{NND2} = \frac{1}{h} (f_{j+1/2}^{NND2} - f_{j-1/2}^{NND2}) \quad (2.40)$$

其中正通量为：

$$f_{j+1/2}^{NND2} = f_j + \frac{1}{2} \min \text{mod} (f_{j+1} - f_j, f_j - f_{j-1}) \quad (2.41)$$

对于负通量，把正通量中全部的下标 $j+m$ 换成 $j-m$, $j+1/2$ 换成 $j-1/2$ 就能得到负通量的表达形式。使用 NND 格式可以很好地处理激波区的数值振荡，保持计算稳定。但它过高的耗散与过低的精度，也意味着 NND 较难满足湍流的高分辨率模拟要求。

2.3.3.2 五阶 WENO 格式

1994 年 Liu 等^[98] 在高精度的 ENO(essentially non-oscillatory) 格式的基础上，利用加权平均的思想提出了 WENO(weighted ENO 格式)，之后，Jiang 等人^[95] 年构造了新的光滑度量因子，进一步提高了计算的效率。WENO 格式由于其在保持高精度的同时具有非常好的鲁棒性，在高超声速湍流中得到广泛使用。近年来，人们在 Jiang 和 Shu 的基础上对 WENO 格式进行了大量的改进和优化^[96,99–105]。五阶 WENO 格式正通量的形式如下：

$$f_j'^{\text{WENO5}} = \frac{1}{h} (f_{j+1/2}^{\text{WENO5}} - f_{j-1/2}^{\text{WENO5}}) \quad (2.42)$$

$j+1/2$ 的一支为：

$$f_{j+1/2}^{\text{WENO5}} = \omega_1 q_1 + \omega_2 q_2 + \omega_3 q_3 \quad (2.43)$$

其中：

$$\begin{aligned} q_1 &= (2f_{j-2} - 7f_{j-1} + 11f_j)/6 \\ q_2 &= (-f_{j-1} + 5f_j + 2f_{j+1})/6 \\ q_3 &= (2f_j + 5f_{j+1} - f_{j+2})/6 \end{aligned} \quad (2.44)$$

对上式三个三阶精度数值通量进行加权平均，就得到五阶精度的 WENO 格式的数值通量，其权函数的计算如下：

$$\omega_k = \frac{\alpha_k}{\alpha_1 + \alpha_2 + \alpha_3} \quad (2.45)$$

$$\alpha_k = \frac{C_k}{(\varepsilon + IS_k)^2} \quad (2.46)$$

式(2.46)中的 ε 为一个小量, 在计算时一般取 $\varepsilon = 10^{-6}$ 。

式(2.46)中的其余系数:

$$C_1 = 3/10, C_2 = 3/5, C_3 = 1/10 \quad (2.47)$$

光滑度量因子为:

$$IS_k = \frac{13}{12}f_k'^2 + \frac{1}{4}f_k''^2 \quad (2.48)$$

其中, 第一个模板:

$$\begin{aligned} f_1' &= f_{j-2} - 2f_{j-1} + f_j \\ f_1'' &= f_{j-2} - 4f_{j-1} + 3f_j \end{aligned} \quad (2.49)$$

第二个模板:

$$\begin{aligned} f_1' &= f_{j-1} - 2f_j + f_{j+1} \\ f_1'' &= f_{j-1} - f_{j+1} \end{aligned} \quad (2.50)$$

第三个模板:

$$\begin{aligned} f_1' &= f_j - 2f_{j+1} + f_{j+2} \\ f_1'' &= 3f_j - 4f_{j+1} + f_{j+2} \end{aligned} \quad (2.51)$$

以上 WENO5 格式的表达式是针对正通量构造出来的, 类似地, 对于负通量, 根据对称性, 把正通量中全部的下标 $j+m$ 换成 $j-m$, $j+1/2$ 换成 $j-1/2$ 就可以得到负通量的差分表达式了。

2.3.3.3 七阶 WENO 格式

对于可压缩湍流的直接数值模拟, 采用五阶精度的 WENO 格式显得耗散偏大, 一般使用七阶精度的 WENO 格式, 对于正通量, WENO7 差分格式如下:

$$f_j^{\text{WENO7}} = \frac{1}{h} \left(f_{j+1/2}^{\text{WENO7}} - f_{j-1/2}^{\text{WENO7}} \right) \quad (2.52)$$

$j+1/2$ 的一支为:

$$f_{j+1/2}^{\text{WENO7}} = \omega_1 q_1 + \omega_2 q_2 + \omega_3 q_3 + \omega_4 q_4 \quad (2.53)$$

其中:

$$\begin{aligned} q_1 &= (-3f_{j-3} + 13f_{j-2} - 23f_{j-1} + 25f_j)/12 \\ q_2 &= (f_{j-2} - 5f_{j-1} + 13f_j + 3f_{j+1})/12 \\ q_3 &= (-f_{j-1} + 7f_j + 7f_{j+1} - f_{j+2})/12 \\ q_4 &= (3f_j + 13f_{j+1} - 5f_{j+2} + f_{j+3})/12 \end{aligned} \quad (2.54)$$

对上式四个四阶精度数值通量进行加权平均，就得到七阶精度的 WENO 格式的数值通量，其权函数的计算如下：

$$\omega_k = \frac{\alpha_k}{\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4} \quad (2.55)$$

$$\alpha_k = \frac{C_k}{(\epsilon + IS_k)^2} \quad (2.56)$$

式(2.72)中的 ϵ 为一个小量，在计算时一般取 $\epsilon = 10^{-6}$ 。

式(2.72)中的其余系数：

$$C_1 = 1/35, C_2 = 12/35, C_3 = 18/35, C_4 = 4/35 \quad (2.57)$$

光滑度量因子为：

$$IS_k = f_k'^2 + \frac{13}{12}f_k''^2 + \frac{1043}{960}f_k'''^2 + \frac{1}{12}f_k'f_k''' \quad (2.58)$$

其中，第一个模板上：

$$\begin{aligned} f_1' &= (-2f_{j-3} + 9f_{j-2} - 18f_{j-1} + 11f_j)/6 \\ f_1'' &= -f_{j-3} + 4f_{j-2} - 5f_{j-1} + 2f_j \\ f_1''' &= -f_{j-2} + 3f_{j-1} + 3f_j + f_{j+1} \end{aligned} \quad (2.59)$$

第二个模板上：

$$\begin{aligned} f_2' &= (f_{j-2} - 6f_{j-1} + 3f_j + 2f_{j+1})/6 \\ f_2'' &= f_{j-1} - 2f_j + f_{j+1} \\ f_2''' &= -f_{j-2} + 3f_{j-1} + 3f_j + f_{j+1} \end{aligned} \quad (2.60)$$

第三个模板上：

$$\begin{aligned} f_3' &= (f_{j-1} - 6f_j + 3f_{j+1} + 2f_{j+2})/6 \\ f_3'' &= f_{j-1} - 2f_j + f_{j+1} \\ f_3''' &= -f_{j-1} + 3f_j + 3f_{j+1} + f_{j+2} \end{aligned} \quad (2.61)$$

第四个模板上：

$$\begin{aligned} f_4' &= (-11f_j + 18f_{j+1} - 9f_{j+2} + 2f_{j+3})/6 \\ f_4'' &= 2f_j - 5f_{j+1} + 4f_{j+2} - f_{j+3} \\ f_4''' &= -f_j + 3f_{j+1} + 3f_{j+2} + f_{j+3} \end{aligned} \quad (2.62)$$

类似地，以上是 WENO7 格式的正通量构造形式，对于负通量，把正通量中全部的下标 $j+m$ 换成 $j-m$ ， $j+1/2$ 换成 $j-1/2$ 可得到负通量的差分表达式。

2.3.3.4 7阶 WENO-SYMBO 格式

WENO-SYMBO^[96] 是在传统的 WENO 格式基础之上增加一个模板的改进型，与 WENO 格式相比，模板增加会使差分离散时增加了一个所需的基架点，这也使得 WENO-SYMBO 格式的基架点类似于中心型差分格式，因此耗散特性得到了改善，此外，Martin 等人^[96] 在格式中给出了一个限制器，在较光滑区时，使 WENO 格式完全退化为线性格式，可以进一步降低由于 WENO 格式误判与降阶带来的额外耗散，并提升精度，更利于计算湍流问题。WENO-SYMBO 格式在提出之初被用于超声速压缩折角构型激波边界层干扰的直接数值模拟研究。对于正通量，7 阶 WENO-SYMBO 差分格式如下：

$$f_j' \text{ WENO_SYMBO } = \frac{1}{h} \left(f_{j+1/2}^{\text{WENO_SYMBO}} - f_{j-1/2}^{\text{WENO_SYMBO}} \right) \quad (2.63)$$

$j + 1/2$ 的一支为：

$$f_{j+1/2}^{\text{WENO7}} = \omega_1 q_1 + \omega_2 q_2 + \omega_3 q_3 + \omega_4 q_4 + \omega_5 q_5 \quad (2.64)$$

其中：

$$\begin{aligned} q_1 &= (-3f_{j-3} + 13f_{j-2} - 23f_{j-1} + 25f_j) / 12 \\ q_2 &= (f_{j-2} - 5f_{j-1} + 13f_j + 3f_{j+1}) / 12 \\ q_3 &= (-f_{j-1} + 7f_j + 7f_{j+1} - f_{j+2}) / 12 \\ q_4 &= (3f_j + 13f_{j+1} - 5f_{j+2} + f_{j+3}) / 12 \\ q_5 &= (25f_{j+1} - 23f_{j+2} + 13f_{j+3} - 3f_{j+4}) / 12 \end{aligned} \quad (2.65)$$

WENO-SYMBO 与 WENO 相比只是增加了一个模板，与其类似，对上式五个四阶数值通量进行加权平均，就得到 WENO-SYMBO 格式的数值通量，其权函数的计算与 WENO 格式也类似：

$$\omega_k = \frac{\alpha_k}{\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5} \quad (2.66)$$

$$\begin{aligned} C_1 &= 0.040195483373, C_2 = 0.249380000671, C_3 = 0.480268625626, \\ C_4 &= 0.200977547673, C_5 = 0.029178342658 \end{aligned} \quad (2.67)$$

限制器形式为：

$$TV_j = |f_{j-2} - f_{j-3}| + |f_{j-1} - f_{j-2}| + |f_j - f_{j-1}| \quad (2.68)$$

$$\begin{aligned} TV_{max} &= \max(TV_j, TV_{j+1}, TV_{j+2}, TV_{j+3}, TV_{j+4}) \\ TV_{min} &= \min(TV_j, TV_{j+1}, TV_{j+2}, TV_{j+3}, TV_{j+4}) \end{aligned} \quad (2.69)$$

$$\begin{aligned} WENO_T V_Limiter &= 5.0 \\ WENO_T V_M AX &= 0.2 \end{aligned} \quad (2.70)$$

当 $TVmax < TVmin * TVLimiter$ 并且 $TVmax < TVLimitermax$ 时，认为当地处于光滑区，格式退化为线性格式：

$$\alpha_k = C_k \quad (2.71)$$

否则，光滑度量因子的形式通过下述方式取得：

$$\alpha_k = \frac{C_k}{(\varepsilon + IS_k)^{\frac{1}{2}}} \quad (2.72)$$

$$IS_k = f_k'^2 + \frac{13}{12}f_k''^2 + \frac{781}{720}f_k'''^2 \quad (2.73)$$

其中，第一个模板上：

$$\begin{aligned} f_1' &= (-2f_{j-3} + 9f_{j-2} - 18f_{j-1} + 11f_j) / 6 \\ f_1'' &= -f_{j-3} + 4f_{j-2} - 5f_{j-1} + 2f_j \\ f_1''' &= -f_{j-3} + 3f_{j-2} + 3f_{j-1} + f_j \end{aligned} \quad (2.74)$$

第二个模板上：

$$\begin{aligned} f_2' &= (f_{j-2} - 6f_{j-1} + 3f_j + 2f_{j+1}) / 6 \\ f_2'' &= f_{j-1} - 2f_j + f_{j+1} \\ f_2''' &= -f_{j-2} + 3f_{j-1} + 3f_j + f_{j+1} \end{aligned} \quad (2.75)$$

第三个模板上：

$$\begin{aligned} f_3' &= (-2f_{j-1} - 3f_j + 6f_{j+1} - f_{j+2}) / 6 \\ f_3'' &= -f_j - 2f_{j+1} + f_{j+2} \\ f_3''' &= -f_{j-1} + 3f_j + 3f_{j+1} + f_{j+2} \end{aligned} \quad (2.76)$$

第四个模板上：

$$\begin{aligned} f_4' &= (-11f_j + 18f_{j+1} - 9f_{j+2} + 2f_{j+3}) / 6 \\ f_4'' &= 12f_j - 30f_{j+1} + 24f_{j+2} - 6f_{j+3} \\ f_4''' &= -f_j + 3f_{j+1} + 3f_{j+2} + f_{j+3} \end{aligned} \quad (2.77)$$

第五个模板上：

$$\begin{aligned} f_5' &= (-26f_{j+1} + 57f_{j+2} - 42f_{j+3} + 11f_{j+4}) / 6 \\ f_5'' &= 3f_{j+1} - 8f_{j+2} + 7f_{j+3} - 2f_{j+4} \\ f_5''' &= -f_{j+1} + 3f_{j+2} + 3f_{j+3} + f_{j+4} \end{aligned} \quad (2.78)$$

上式依旧只给出了正通量的表达式，对于负通量，把正通量中的下标 $j + m$ 换成 $j - m$, $j + 1/2$ 换成 $j - 1/2$ 即为负通量的差分表达式。

2.3.3.5 OMP6 格式

在高超声速复杂流动的直接数值模拟中，高精度激波捕捉格式是目前 CFD 技术的研究热点。WENO7 格式在激波捕捉格式中取得了很大的成功，但是计算量大，在某些计算条件下，由于其耗散较大，对小尺度结构有抹平的作用，对此，Suresh 和 Huynh^[106] 发展了一系列的保单调格式 (MP schemes)。并且被广泛使用^[107]，这类格式与传统的 WENO 格式相比具有较高的分辨率和计算效率，保单调格式由高阶线性部分和保单调限制器构成。李新亮等人^[94] 提出了非线性谱分析方法，并基于该方法对保单调格式进行了色散及耗散优化，构造了 6 阶优化保单调格式 (OMP6)，OMP6 格式也具有一定的激波捕捉能力，而且可以人为对格式色散和耗散性质进行控制，可以拥有比相同阶传统 WENO 格式更低的耗散，6 阶保单调格式的形式如下所示：

$$f_j'^{\text{OMP6}} = \frac{1}{h} (f_{j+1/2}^{\text{OMP6}} - f_{j-1/2}^{\text{OMP6}}) \quad (2.79)$$

当

$$(f_{j+1/2}^{\text{linear}} - f_j) (f_{j+1/2}^{\text{linear}} - f_{j+1/2}^{\text{MP}}) \leq \varepsilon, (\varepsilon = 10^{-10}) \quad (2.80)$$

时，则：

$$f_{j+1/2} = f_{j+1/2}^{\text{linear}} \quad (2.81)$$

否则：

$$f_{j+1/2} = f_{j+1/2}^{\text{linear}} + \min \operatorname{mod} (f_{j+1/2}^{\min} - f_{j+1/2}^{\text{linear}}, f_{j+1/2}^{\max} - f_{j+1/2}^{\text{linear}}) \quad (2.82)$$

在上述式中：

$$\begin{aligned} f_{j+1/2}^{\text{MP}} &= f_{j+1/2}^{\text{linear}} + \min \operatorname{mod} [(f_{j+1} - f_j), 4(f_j - f_{j-1})] \\ f_{j+1/2}^{\max} &= \min [\max (f_j, f_{j+1}, f_{j+1/2}^{\text{MD}}), \max (f_j, f_{j+1/2}^{\text{UL}}, f_{j+1/2}^{\text{LC}})] \\ f_{j+1/2}^{\min} &= \max [\min (f_j, f_{j+1}, f_{j+1/2}^{\text{MD}}), \min (f_j, f_{j+1/2}^{\text{UL}}, f_{j+1/2}^{\text{LC}})] \end{aligned} \quad (2.83)$$

$$\begin{aligned} f_{j+1/2}^{\text{UL}} &= f_j + 4(f_j - f_{j-1}) \\ f_{j+1/2}^{\text{MD}} &= 1/2(f_j + f_{j+1}) - 1/2d_{j+1/2}^M \\ f_{j+1/2}^{\text{LC}} &= 1/2(3f_j + f_{j-1}) + 3/4d_{j-1/2}^M \end{aligned} \quad (2.84)$$

$$d_{j+1/2}^M = \min \operatorname{mod} (4d_j - d_{j+1}, 4d_{j+1} - d_j, d_j, d_{j+1}) \quad (2.85)$$

$$d_j = f_{j-1} + f_{j+1} - 2f_j \quad (2.86)$$

对于线性部分：

$$\begin{aligned} f_{j+1/2}^{\text{linear}} = & \frac{\xi + \eta}{2} f_{j+4} + \left(\frac{1}{60} - \frac{7\xi + 5\eta}{2} \right) f_{j+3} + \\ & \left(-\frac{2}{15} + \frac{21\xi + 9\eta}{2} \right) f_{j+2} + \left(\frac{37}{60} - \frac{35\xi + 5\eta}{2} \right) f_{j+1} + \\ & \left(\frac{37}{60} + \frac{35\xi - 5\eta}{2} \right) f_j + \left(-\frac{2}{15} - \frac{21\xi - 9\eta}{2} \right) f_{j-1} + \\ & \left(\frac{1}{60} + \frac{7\xi - 5\eta}{2} \right) f_{j-2} - \frac{\xi - \eta}{2} f_{j-3} \end{aligned} \quad (2.87)$$

通过调整式(2.87)中的 ξ 和 η 可分别对格式的耗散与色散特性进行优化，这里选取 $\xi = 0.015$, $\eta = 0$ 可使格式退化为中心型差分格式，以保持 OMP6 格式的最佳耗散特性，可用于湍流的精细模拟研究。类似的，这里仅给出了正通量的表达式，对于负通量，根据对称性，把正通量中的下标 $j+m$ 换成 $j-m, j+1/2$ 换成 $j-1/2$ 就可以得到负通量的差分表达式了。

2.3.3.6 七阶迎风格式

高阶格式可看做子模板低阶格式的线性组合，而 WENO 格式一般存在对应的高阶线性格式^[108]。与一般的 7 阶 WENO 格式相比，7 阶线性迎风格式 (UD7) 具有较低的耗散和高精度，但处理间断的能力差。对于正通量，UD7 差分格式如下：

$$f_j'^{\text{UD7}} = \frac{1}{h} \left(f_{j+1/2}^{\text{UD7}} - f_{j-1/2}^{\text{UD7}} \right) \quad (2.88)$$

$j+1/2$ 的一支为：

$$\begin{aligned} f_{j+1/2}^{\text{UD7L}} = & (-3.0f_{j-3} + 25.0f_{j-2} - 101.0f_{j-1} + 319.0f_j \\ & + 214.0f_{j+1} - 38.0f_{j+2} + 4.0f_{j+3}) / 420.0. \end{aligned} \quad (2.89)$$

类似地，把正通量中全部的下标 $j+m$ 换成 $j-m$, $j+1/2$ 换成 $j-1/2$ 可得到负通量的差分表达式。

2.3.3.7 混合差分格式

在提高计算鲁棒性方面。混合格式是一个很好的方法，混合格式一般由两种不同的格式混合形成，两种格式一种具有较强激波捕捉能力，但耗散偏大，而另一种则耗散较低但激波捕捉能力不强，间断处使用具有激波捕捉能力但耗散较高的格式，如 ENO/WENO，流场光滑处使用低耗散格式，流场间断与光滑处则通过设计的开关函数来判定。早在 20 世纪 70 年代，Harten 就提出了混合格式的

概念^[109]。但是由于之前缺少理论较为完善的高精度激波捕捉格式，因此真正意义上的类谱 ENO/WENO 混合格式只有随着 ENO 格式和 WENO 的提出和完善才逐渐出现。

1996 年, Adams 和 Shariff^[110] 提出了紧致迎风-ENO 混合格式, 并使用这个格式首次对 Mach 3 的 18° 压缩折角构型进行了 DNS 研究, 紧致迎风与 ENO 的混合策略使计算在激波区能够顺利处理间断, 且在光滑区保持了较高精度。Pirozzoli 在 2002 年构造了紧致差分与 WENO 混合的守恒型混合格式, 并用于入射激波边界层干扰的直接数值模拟计算中^[111]。国内学者也在混合格式研究上做出过大量的工作, Sun 和 Ren 等^[112] 提出一种含有 2 个自由参数的线性格式, 其中一个参数控制格式的色散误差, 而另外一个参数控制格式的耗散误差, 在此基础上基于 WENO 的加权思想, 构建了 MDCD-WENO 格式, 该格式可以达到色散最小且耗散可控。Zhang 等^[113] 研究了间断有限元方法与有限体积方法的混合效率。验证了所构造的混合格式可以达到 3 阶精度, 并提出了一种激波探测器用于捕捉激波, 数值抑制振荡。Sun 和 Ren 等^[114] 将 MDCD-WENO 格式进一步拓展到了六阶精度, He 和 Li 等^[115,116] 发现如果采用网格点过多, 则会激发 WENO 格式, 带来耗散, 而采用较少网格点, 则会带来误差, 引起数值振荡, 为此提出了加权群速度格式与 WENO 格式的混合 WGVC-WENO, 并将该格式成功运用到激波-湍流边界层相互干扰研究中。Zhao 和 Qiu 等^[117], 提出一种基于有限体积法的 HWENO, 该格式一维情况下达到五阶精度, 二维情况下达到四阶精度, 利用 DG 方法中的限制器思想修正间断附近单元的一阶矩, 对光滑区域直接使用线性近似进行数值重构, 既有效地抑制了抑制数值振荡, 又提高了格式的计算效率。

本文使用三种差分格式构造了一种混合格式。第一种是七阶迎风格式, 其形式及特点在小节 2.3.3.6 中已进行了介绍, 它主要用来承担流场中除间断外, 大部分区域的计算。第二种格式为七阶 WENO 格式, 参见小节 2.3.3.3, WENO 格式作为混合格式中的一种中间格式, 主要对存在间断但间断强度较弱的位置和区域进行计算。最后一种混合格式为五阶 WENO 格式, 参见小节 2.3.3.2。5 阶 WENO 耗散较高, 但其处理的间断的能力最强, 主要负责强激波位置的流场计算。

激波识别器采用改进的 Jameson 激波识别器^[118]。Jameson 在使用有限体积方法求解欧拉方程时, 发现激波或驻点位置会出现严重的数值抖动, 而添加人工耗散能够抑制抖动, 他在 1981 年, 提出了一种人工耗散的形式^[118], 其系数 $\nu_{i,j}$ 取决于局部压力梯度:

$$\nu_{i,j} = \frac{|p_{i+1,j} - 2p_{i,j} + p_{i-1,j}|}{|p_{i+1,j}| + 2|p_{i,j}| + |p_{i-1,j}|} \quad (2.90)$$

当系数 $\nu_{i,j}$ 越大时, 对应的人工耗散也越大。本文认为 $\nu_{i,j}$ 本质是一种激波识别器, 当其值越大的地方, 意味着此处需要更大的人工粘性来抑制振荡。所以 $\nu_{i,j}$ 也可以用在有限差分法当中, 作为控制混合格式的开关函数。当 $\nu_{i,j}$ 越大, 就使用高耗散的数值格式对此处的进行压制。

本文对 Jameson 的原始形式进行了简单的修正：

$$\begin{aligned}\phi_i &= \frac{|-p_{i-1} + 2p_i - p_{i+1}|}{p_{i-1} + 2p_i + p_{i+1}} \\ \phi_j &= \frac{|-p_{j-1} + 2p_j - p_{j+1}|}{p_{j-1} + 2p_j + p_{j+1}} \\ \phi_k &= \frac{|-p_{k-1} + 2p_k - p_{k+1}|}{p_{k-1} + 2p_k + p_{k+1}}\end{aligned}\quad (2.91)$$

$$\theta = \phi_i + \phi_j + \phi_k \quad (2.92)$$

此外由于本文考虑的问题大多是三维流动问题，为此在三个方向都进行判别，将三个方向判别结果之和，作为修正的 Jameson 激波识别器。

在使用混合格式进行计算时，设置阈值 θ_1 和 θ_2 ，当 $\theta \leq \theta_1$ 时，使用七阶迎风格式进行计算，当 $\theta_1 < \theta \leq \theta_2$ 时，使用七阶 WENO 格式进行计算，当 $\theta_2 \leq \theta$ 时，使用五阶 WENO 格式进行计算，阈值 θ_1 和 θ_2 设置的越大，混合格式整体的耗散也越大。在对流场不是特别复杂的问题计算时，可将阈值设为 0.02 和 0.1。

经过大量算例的测试，该混合格式具有很好的数值稳定性，见章节 6。当与特征变量重构结合使用时，如果时间步长适当，可以保证章节 6 中介绍的所有算例，模拟全过程不发散。此外，在模拟过程中，90% 以上的位置都采用低耗散七阶迎风格式，只有少数有激波或流动分离的位置跳转为七阶或五阶 WENO 格式。由于三种混合格式精度较高（五阶以上），只有耗散特性不同，格式切换后可能产生的伪波相对较小。即使产生了这些伪波，它们也会被局部激波、湍流或分离淹没，不会对流场产生明显影响。

2.3.4 边界差分离散方法

对于非周期边界，需要在边界附近降阶。本程序采用五阶 WENO 格式计算边界附近的数值通量。当 WENO 格式的子模板使用边界外的网格点时，子模板的权重被强制为 0，屏蔽边界外模板，五阶 WENO 格式参见小节 2.3.3.2。由于 5 阶 WENO 本身具有一定的激波捕捉能力，比线性格式鲁棒性好，因此采用 5 阶 WENO 降模板的方式对边界降阶相比于线性格式降阶能让计算具有更好的鲁棒性。尽管格式降阶后，其分辨率下降了，但由于对边界层开展 DNS 时，边界附近往往布有最密的网格，且降阶计算的网格点数只有几个，物理跨度只有百分之几毫米甚至是千分之几毫米，其损失的精度通过网格的密度而获得了补偿，总体分辨率依然很高。

2.4 粘性项差分离散方式

粘性项的差分离散一般采用中心差分格式，为匹配无粘项计算格式的精度，粘性项主要采用六阶中心和八阶中心两种格式。对于非周期边界，边界计算采用

降阶方法。在最外层边界点采用二阶单边差分格式，在第二、三、四个边界点分别采用二阶、四阶和六阶中心差分格式。

六阶中心格式的形式如下^[119]:

$$f'_j = (45(f_{j+1} - f_{j-1}) - 9(f_{j+2} - f_{j-1}) + (f_{j+3} - f_{j-3})) / 60h \quad (2.93)$$

八阶中心格式的形式如下^[119]:

$$\begin{aligned} f'_j = & (a_1(f_{j+1} - f_{j-1}) + a_2(f_{j+2} - f_{j-1}) + a_3(f_{j+3} - f_{j-3}) \\ & + a_4(f_{j+4} - f_{j-4})) / h \end{aligned} \quad (2.94)$$

其中:

$$\begin{aligned} a_1 &= 0.8 & a_2 &= -0.2 & a_3 &= 3.80952380952 \times 10^{-2} \\ a_4 &= -3.571428571428 \times 10^{-3} \end{aligned} \quad (2.95)$$

2.5 时间推进格式

差分的时间推进方法一般分为显式和隐式方法。显式方法指利用当前时间点各位置值直接求解待求时间点上各位置上的值。而隐式方法则一般要用到待求时间点值，将各位置的值联立起来进行求解。相比于隐式方法，显式方法对时间步长有所限制，但单步计算量一般更小，而且由于显格式不需要联立求解方程组，在大规模并行计算时，其需要的数据通讯量更小，因此并行效率更高。在求解大尺度宏观定常流动问题时，使用隐式方法可以选择较大的时间步长，往往比显式方法会更有效率。而对于湍流的直接数值模拟，流动本身是非定常的，流动变化在空间上也表现出多种尺度，捕捉湍流脉动要求时间格式步长不能太大，因此显式方法更有优势。

Runge-Kutta 方法是常微分方程里比较成熟的显式求解方法，广泛应用于半离散方法的时间推进，半离散方程的形式:

$$\frac{\partial U}{\partial t} = N(U) + L(U) = R(U) \quad (2.96)$$

其中 $N(U)$ 为方程中的非线性部分，对应于 N-S 方程中的对流项， $L(U)$ 对应于 Navier-Stokes 方程中的粘性项。具有三阶精度的三步 TVD R-K 方法^[92,120] 为:

$$\begin{aligned} U^{(1)} &= U^n + \Delta t R(U^n) \\ U^{(2)} &= \frac{3}{4}U^n + \frac{1}{4}[U^{(1)} + \Delta t R(U^{(1)})] \\ U^{n+1} &= \frac{1}{3}U^n + \frac{2}{3}[U^{(2)} + \Delta t R(U^{(2)})] \end{aligned} \quad (2.97)$$

2.6 小结

本章介绍了使用有限差分方法开展可压缩湍流直接数值模拟时涉及到的数值算法，本章中介绍的所有算法都在 OpenCFD-SCU 中被采用，包括 Steger-Warming 流通矢量分裂、特征重构，常见的无粘项差分方法，5 阶或 7 阶 WENO，OMP6，WENO-SYMBO，7 阶迎风格式；粘性项差分方法，6 阶或 8 阶中心格式；时间推进方法，3 步 3 阶 Runge-Kutta 方法。本章重点介绍了利用 Jameson 人工粘性系数所开发的修正的 Jameson 激波识别器，并以此搭建的将 7 阶迎风，7 阶 WENO，5 阶 WENO 混合的混合格式。

第3章 高精度有限差分软件 OpenCFD-SCU 框架与结构

3.1 引言

本章介绍一套 CPU/GPU 异构平台的有限差分法计算软件, 软件名为 OpenCFD-SCU (open computational fluid dynamic code for scientific computation with graphics processing unit (GPU) system)。该软件是在中国科学院力学研究所李新亮研究员自主开发的高精度计算流体力学软件 OpenCFD-SC (Open Computational Fluid Dynamic code for Scientific Computation) 基础之上进行的异构移植, 并进行了进一步的程序框架优化。

OpenCFD-SCU 具有以下特点, 它针对可压缩 NS 方程采用有限差分法离散求解 (上一部分已对可压缩湍流直接数值模拟求解的控制方程及本程序涉及到的数值方法进行了介绍), 该方法数值精度高, 经过三维雅克比 (Jacobian) 变换后能对曲线坐标系下的问题进行计算。无粘项的差分离散除了常规的线性差分格式与常见的激波捕捉格式之外, 还集成了自主构造的带有迎风性质的混合格式, 兼具高精度、低耗散、高鲁棒性; 粘性项离散采用中心型差分格式; 时间推进方法采用显式 TVD 型三步三阶龙格库塔 (Runge-Kutta) 方法, 可以最大程度保证计算程序的并行可扩展性。

本部分详细地对 OpenCFD-SCU 的程序框架与并行实现方式进行介绍, 包括程序总体框架设计, 数据三维分割与 MPI 通讯的设计, CPU/GPU 异构计算的程序流水线设计, 和针对大规模计算的并行 IO 设计。

3.2 程序总体框架

OpenCFD-SCU 基于中国科学院力学所李新亮研究员自主开发高精度计算流体力学软件 OpenCFD-SC, 该软件的特点在于精度高, 通过高精度的数值模拟, 既可以得到流动的整体特性, 又能获得流动的细节特征, 用来研究湍流中的物理机制, 揭示新的物理现象^[119]。OpenCFD-SC 使用高精度的有限差分方法来求解流体运动的基本方程, 利用曲线坐标系下的三维雅克比变换, 可求解任意曲线坐标系下的可压缩 Navier-Stokes 方程, 该软件可以用于各种复杂流动机理研究。如各种流动非稳定性特征、湍流及其转捩的机理问题 (包括混合层, 槽道流, 射流, 边界层等湍流及其转捩, 均匀各向同性湍流等), 各种复杂的非定常分离流动等。这类问题的研究往往更关心流动结构与细节, 采用常规的 CFD 软件很难达到精度的需求。OpenCFD-SC 和 OpenCFD-SCU 中也为用户提供了各种可自定义的函数接口, 用户可以根据自己的需要添加边界条件, 后处理程序, 数值方法等, 并已经开源^[119], OpenCFD-SC 目前已经得到国内外超过 200 家研究组的广泛使用。

OpenCFD-SCU 的基础算法功能与 OpenCFD-SC 保持一致, 但在算法实现时,

为了进一步优化程序框架 OpenCFD-SCU 将无数据依赖的计算放到了不同的流水线上并发执行。图 3.1 展示了 OpenCFD-SCU 的程序流程图，从中可以看出程序的框架设计：

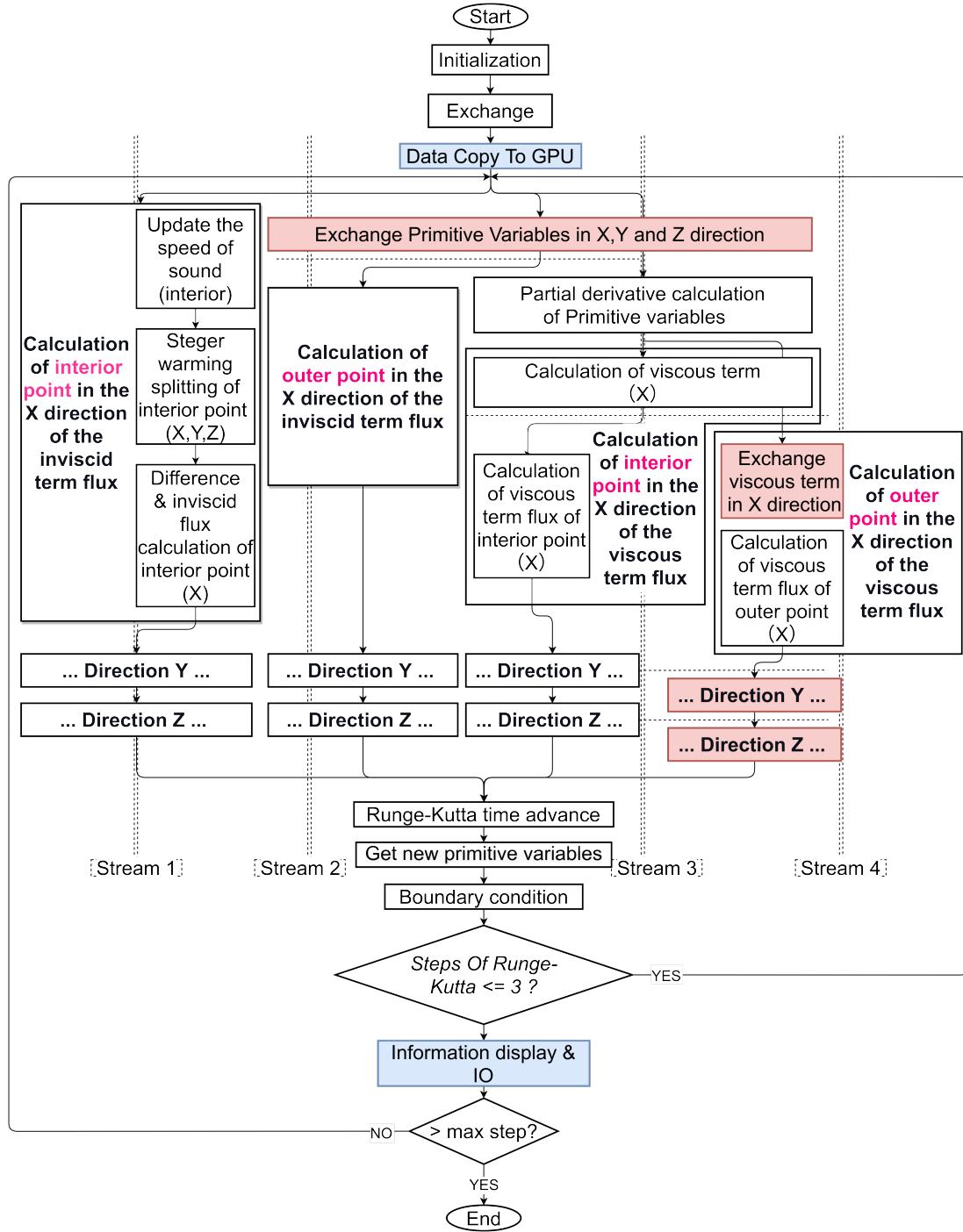


图 3.1 OpenCFD-SCU 程序框架

Figure 3.1 Program framework of OpenCFD-SCU

程序通过读入外部控制参数可对其进行初始化，CPU 和 GPU 上各自分配内存，外部数据先通过 MPI 并行 IO 读入到各 CPU 进程中，然后传递到 CPU 绑定的 GPU 上去。之后开始计算，在图 3.1 中红框内的操作表示该操作涉及到进程间

的数据通讯，蓝框中的操作表示需要进行 CPU 与 GPU 之间数据交换。程序把无粘项内、外区，粘性项内、外区通量计算与更新的操作放在了四条流水线上。在对每个通量进行计算与更新时，都是先计算与更新 x 方向的通量，再计算与更新 y 和 z 方向的通量，之后程序进行时间推进，通过守恒变量算出新的原始变量，再调用边界条件，当计算的时间步达到设置算达的时间步时，程序会终止计算，在之前可以告知程序在何时进行数据的保存和显示。

3.3 数据的三维并行分割与 MPI 通讯

直接数值模拟需要的计算量非常大，并行计算对于直接数值模拟是必不可少的，并行计算一般分为共享式并行计算和分布式并行计算，共享式并行计算基于共享式存储，并行计算时并行核心以多线程的方式相对独立地处理同一块内存中的数据和执行各自任务，过程中可以通过指令来操纵各线程的同步与等待，常见的共享式并行编程方式有 OpenMP 和 Pthread，共享式并行计算往往由于共享存储大小难以扩大而受到限制。分布式并行计算基于分布式存储，是指各进程之间拥有独立的存储，并行计算时进程间的数据共享通过网络通讯实现，分布式并行计算可以通过网络连接成千上万个计算节点，可扩展性强，是大规模并行计算采取的主要方式，但网络通讯会带来额外的开销，常见的节点间互连网络规范为 InfiniBand net (IB 网)，它由 Intel 公司主导开发。分布式并行计算编程方式有 MPI (Message Passing Interface)、ZeroMQ (0MQ)、Hadoop 等。其中，MPI 是目前最流行的并行编程消息传递规范，得到了大多数超级计算平台的支持。采用并行计算开展直接数值模拟，其中一个原因是大规模的并行计算可以利用更多的计算核心加快模拟速度，另一个原因是单个计算节点的内存很难满足直接数值模拟的内存需求，需要通过分布式的并行存储系统分散存储数据。

MPI 常见的应用程序编程接口函数有 (OpenCFD-SCU 程序中使用的到 MPI 函数)：

`MPI_Init(int *argc, char ***argv)`: MPI 初始化。

`MPI_Init_thread(int *argc, char ***argv, int required, int *provided)`: 多线程时的 MPI 初始化。

`MPI_Comm_rank(MPI_Comm comm, int *myid)`: 获取通讯域中的进程 ID。

`MPI_Comm_split(MPI_Comm comm, int color, int key, MPI_Comm *newcomm)`: 将通讯域划分为若干个子通讯域。

`MPI_Comm_size(MPI_Comm comm, int *size)`: 获取通讯域中的进程数信息。

`MPI_Bcast(void *buffer, int count, MPI_Datatype datatype, int root, MPI_Comm comm)`: 将数据传到通讯域的所有进程中。

`MPI_Buffer_attach(void *buffer, int size)`: 设置阻塞通讯时的缓冲区大小。

`MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)`: 利用缓冲区进行阻塞式的数据发送

`MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)`: 利用缓冲区进行阻塞式的数据接收

`MPI_Sendrecv(void *sendbuf, int sendcount, MPI_Datatype sendtype, int dest, int sendtag, void *recvbuf, int recvcount, MPI_Datatype recvtype, int source, int recvtag, MPI_Comm comm, MPI_Status *status)`: 利用缓冲区同时进行发送与接收数据的操作。

`MPI_Allreduce(const void *sendbuf, void *recvbuf, int count, MPI_Datatype datatype, MPI_Op op, MPI_Comm comm)`: 集合函数，通讯域中的所有进程执行 `MPI_Op` 中规定的操作，并返回到所有进程。

`MPI_Barrier(MPI_Comm comm)`: 同步通讯域中的所有进程。

`MPI_Finalize()`: MPI 结束。

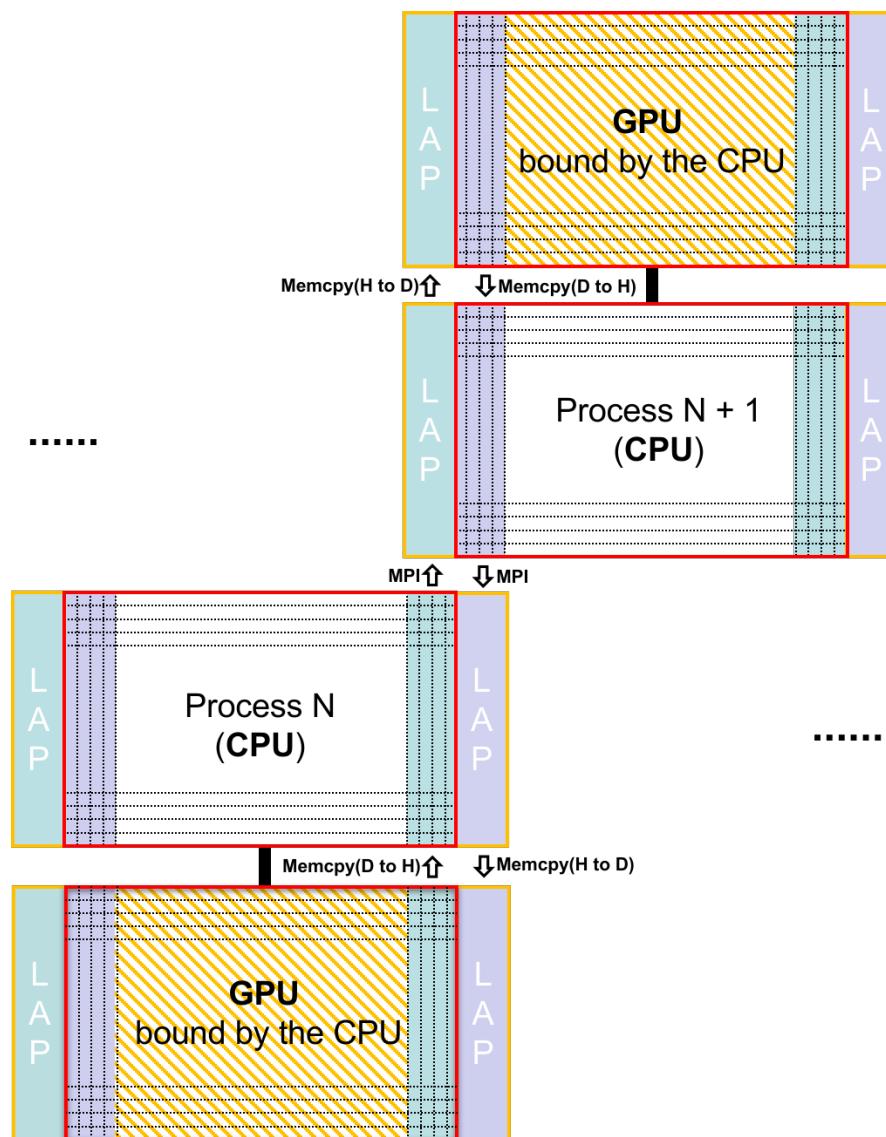


图 3.2 OpenCFD-SCU 中的 MPI 数据交换模型

Figure 3.2 MPI interprocess data exchange mode of OpenCFD-SCU

差分方法的 MPI 通讯相对简单，由于差分模板往往只有几个基架点，所以差分方法只需要通过 MPI 通讯获取到临近进程薄薄几层数据，数据通讯量相对较少。与同构的纯 CPU 间 MPI 通讯不同，在 CPU/GPU 异构并行计算时，由于数据是在 GPU 上存储的，对于一些没有 GPU 直连支持的超级计算机，需要先将数据从 GPU 传到 CPU，然后 CPU 间通过 MPI 交换数据，获得数据后的 CPU 再将数据传回 GPU，图 3.2 展示了这一过程。

OpenCFD-SCU 采用三维并行分割，数据划分方式如图 3.3，每个块表示每个进程上处理的数据，三个方向的索引信息由 `MPI_Comm_split()` 函数产生。图中橘黄色框住的区域表示进程间的重叠区（LAP 区），进程将自己边界的几层数据传入到临近进程的重叠区中，再从临近进程获取数据存到自己的重叠区中，粉红色的区域表示数据块的外区，在计算时，只有外区计算需要通过 MPI 通讯获取临近进程数据，内区只需要在本地获取数据，因此内区的计算可以和外区的计算、外区 MPI 通讯重叠起来。

图 3.4 能更清楚的看到单个进程的数据划分方式，在内存申请时，为了更为方便地调用 API 函数申请内存，每个进程申请的内存实际上是 $(nx + 2 \text{ LAP}) \times (ny + 2 \text{ LAP}) \times (nz + 2 \text{ LAP})$ 的长方体，但实际有用的内存如图 3.4(a) 所示，并非只是一个完整的长方体（长方体加上外面包有的重叠区的几层数据），在本程序中由于最高支持的差分格式为 7 阶格式，所以重叠区一般设置为 4，而外区的厚度被设置为了 2 倍重叠区，这是为了 GPU 数据访问时的合并与对齐（详见章节 4），图 3.4(b) 为图 3.4(a) 的一个中截面，可以更直观地看到 LAP 区、外区和内区。图 3.4(c) 为数据结构的拆分图，可以看出程序中 x, y, z 方向外区的尺寸并不完全一致， x 方向的尺寸为 $\text{LAP} \times (ny - 4 \text{ LAP}) \times (nz - 4 \text{ LAP})$ ， y 方向的尺寸为 $nx \times \text{LAP} \times (nz - 4 \text{ LAP})$ ， z 方向的尺寸为 $nx \times ny \times \text{LAP}$ ，它们组合在一起正好可以将内区点包裹。

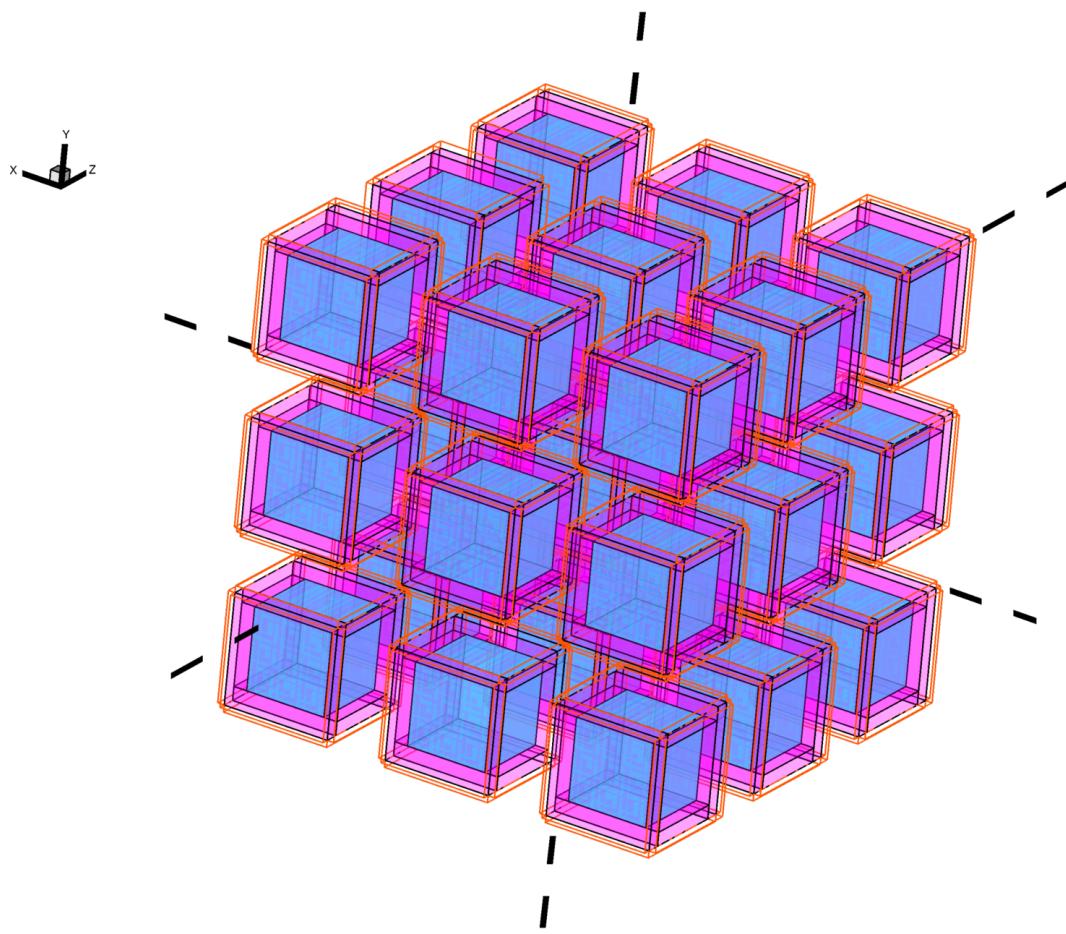


图 3.3 三维 MPI 数据划分
Figure 3.3 Data partition of 3D MPI process

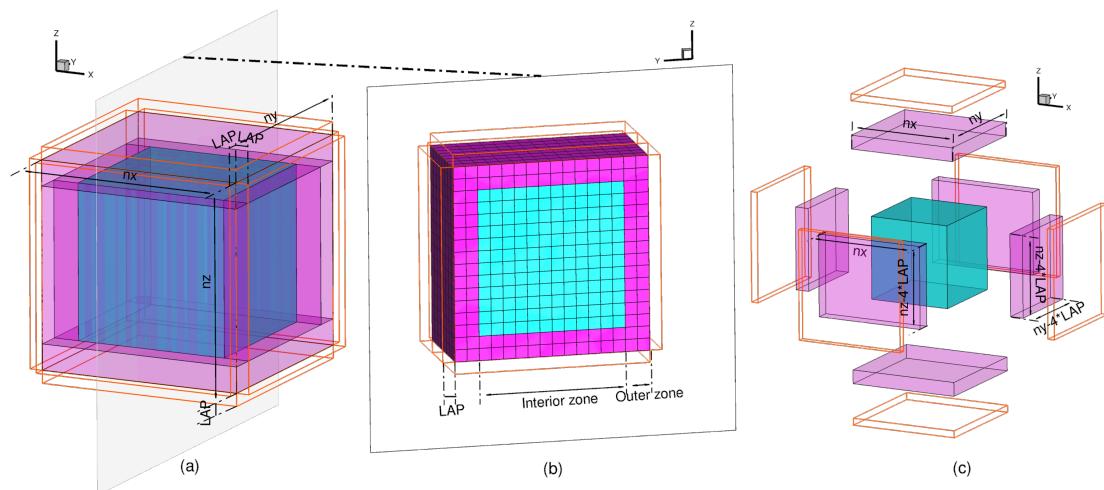


图 3.4 一个 MPI 进程中的数据结构
Figure 3.4 A data structure within an MPI process

3.4 程序流水线设计

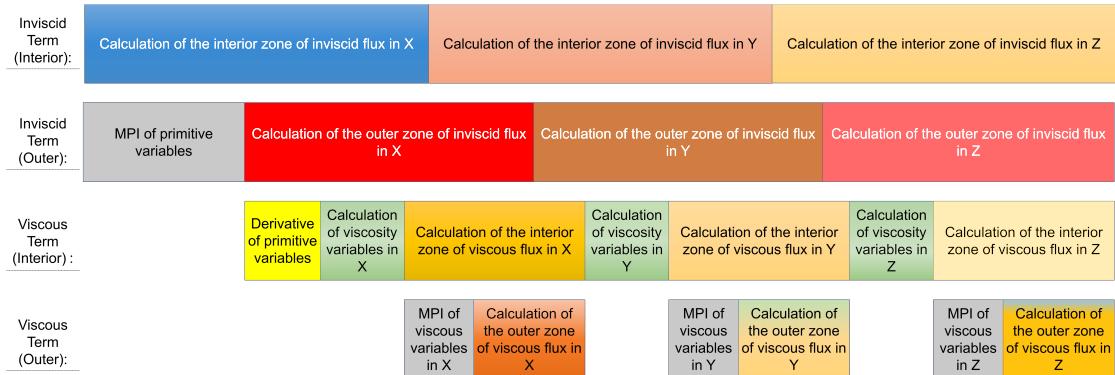


图 3.5 OpenCFD-SCU 程序时间线设计

Figure 3.5 Program timeline design of OpenCFD-SCU

计算与通讯的重叠对提升程序扩展性至关重要，在 CPU 上，MPI 通讯与计算的重叠通常使用多线程或者 MPI 的非阻塞指令完成，而在 CPU-GPU 的异构架构中，由于 CPU 与 GPU 上执行的指令是异步的，CPU 上执行的指令并不会干扰 GPU 上操作。只需 GPU 上承担足够的计算量，则不管 CPU 上的 MPI 通讯是否阻塞，CPU 都会在 GPU 获取数据之前完成 MPI 数据交换。此时程序的计算通讯重叠只需要考虑 GPU 上的计算与 CPU-GPU 间通讯的重叠，这通常使用 CUDA-stream 技术实现（详见章节 4），这是一种 GPU 上进行任务级并发的技术。在本文的设计中，GPU 上共设置了四条流水线，在通量计算开始之后，无粘项内区与外区的计算被放到两条流水线上，只有外区的计算需要获取到临近进程的数据，它们可以并发执行，待无粘项外区关于原始变量的数据交换完成之后，粘性项内区和外区在另外两条流水线上开始各自的计算，粘性项外区在进行计算前需要进行进程间粘性变量的数据交换，此时的数据通讯与其它流水线上的计算也是重叠的。

3.5 MPI 并行 IO

在大规模并行计算中，由于数据量大，输入输出 (IO) 可能成为程序的瓶颈。许多高性能计算应用程序允许每个进程分别读写一个数据文件，这是 IO 设计时的选项之一。但是采用这种方式，如果调整 MPI 进程的数量，就会给继续计算带来麻烦。MPI 提供并行 IO 接口函数，允许所有 MPI 进程同时对文件进行读写。本文使用 MPI 并行 IO 函数来设计 OpenCFD-SCU 的 IO 部分。当读取数据时，本文使用应用程序接口 (API) 函数 `MPI_File_write_at()` 等。同一行的 MPI 进程通过集体通信获取文件中某一行的数据，然后根据自身的索引信息选择保留哪一部分。当写出数据时，每个进程将数据传输到该行中的第一个进程，该进程汇总并调用 API 函数 `MPI_File_write_at()` 来写出数据，数据汇总的设计可以减少了 `MPI_File_write_at()` 函数调用的次数，也可以避免 MPI 进程在写入同一行

文件时的等待。过往使用 CPU 程序尝试计算百亿网格的问题时，仅数据读取就要花费数小时，如果需要调试大规模算例，任务需要频繁提交时，这些低效的数据读取速度，让完成任务目标变得异常困难。IO 的优化让本文可以在十分钟内读入数百亿网格规模直接数值模拟算例的流场和网格数据（文件大小 1T 以上）。

关于并行 IO 的使用，此处给出一个使用一维并行 IO 读取 OpenCFD-SCU 网格示例代码：

```

1 void read3d_MPI(double *data, int num, int NZ){ /*  

    data 是数据存储位置的地址指针，num 是每一行读取的变  

    量个数，NZ 是本进程读取的数据行数（每个进程的行数可能  

    不同）  

    MPI_Status status;  

3     int num_byte;  

5         for(int k = 0; k < NZ; k++){ // 循环NZ次把一个变  

    量读进来  

        MPI_File_read(tmp_file, &num_byte, 1, MPI_INT  

        , &status); // 由于程序兼容 fortran 中的 WRITE 函数，因此  

        每一行首尾要多读4字节  

7         MPI_File_read(tmp_file, data+num*k, num,  

        MPI_DOUBLE, &status); // 每次读取num个双精度浮点型变  

        量，从 data+num*k 作为起始位置写入内存  

        MPI_File_read(tmp_file, &num_byte, 1, MPI_INT  

        , &status); // 结尾多读4字节  

9     }  

    }  

11  

13     int num = nx * ny; // 每一行读 nx*ny 个数据（即把 nx*ny*nz  

    的三维块看成每行 nx*ny 个变量，共 nz 行的二维块）  

15 MPI_File tmp_file; // 声明一个并行 IO 文件属性的变量  

15 MPI_File_open(MPI_COMM_WORLD, "OCFD3d-Mesh.dat",  

    MPI_MODE_RDONLY, MPI_INFO_NULL, &tmp_file); // 使用  

    MPI_File_open 函数打开名为 "OCFD3d-Mesh.dat" 的文  

    件，文件打开方式是只读  

17 MPI_File_seek(tmp_file, 0, MPI_SEEK_SET); // 文件指针定  

    位到，文件 0 字节的开头位置

```

```

19 if (my_id == 0) printf("READ X3d ...\\n"); // 屏幕打印信息
    MPI_File_seek(tmp_file, NP[my_id]*(num*sizeof(double)
        + 2*sizeof(int)), MPI_SEEK_CUR); // 把文件指针定位到
    NP[my_id]*(num*sizeof(double) + 2*sizeof(int)) 的位
    置。数组NP存储的是各个进程处理数据起始的位置，此位
    置对于nz等于多少。文件指针内填入的是字节数
21 read3d_MPI(x3d, num, NZ); // 开始调用子函数读取数据，读
    取的数据块大小是nx*ny*NZ, NZ是每个进程各自行数（全
    部数据的总行数是nz, c语言区分大小写）

23 if (my_id == 0) printf("READ Y3d ...\\n"); // 屏幕打印信息
    MPI_File_seek(tmp_file, (nz-NZ)*(num*sizeof(double) +
        2*sizeof(int)), MPI_SEEK_CUR); // 将文件指针位置向后
    移动(nz-NZ)*(num*sizeof(double) + 2*sizeof(int))字
    节
25 read3d_MPI(y3d, num, NZ); // 继续读取nx*ny*NZ个变量，存
    到首地址指针名为y3d的内存中

27 if (my_id == 0) printf("READ Z3d ...\\n"); // 屏幕打印信息
    MPI_File_seek(tmp_file, (nz-NZ)*(num*sizeof(double) +
        2*sizeof(int)), MPI_SEEK_CUR); // 再将数据指针向后移
    动(nz-NZ)*(num*sizeof(double) + 2*sizeof(int))字节
29 read3d_MPI(z3d, num, NZ); // 读取变量，存入内存

31 MPI_File_close(&tmp_file); // 关闭文件

```

上例为采用一维并行 IO 读取 OpenCFD-SCU 网格文件的程序，网格文件存有 x, y, z 方向的格点信息，因此调用三次 read3d_MPI() 将其读入。在读取数据文件时，需要注意，数据文件的头部有 20 个字节的文件信息，存储了计算步数和计算的无量纲时间，可用以下代码先将文件头部信息读入。

```

1 MPI_Status status;
2 int tt; double dt;
3 MPI_File_seek(tmp_file, sizeof(int), MPI_SEEK_CUR);
4 MPI_File_read_all(tmp_file, &tt, 1, MPI_INT, &status);
    // 并行文件读取的集合函数，所有进程都获取到所读信息
5 MPI_File_read_all(tmp_file, &dt, 1, MPI_DOUBLE, &
    status);

```

```
MPI_File_seek( tmp_file , sizeof(int) , MPI_SEEK_CUR );
```

由于数据文件存储了五个原始变量（密度、速度、温度），因此之后要调用五次 `read3d_MPI()` 函数。

以上演示为一维的 MPI 并行 IO 数据读取方式，OpenCFD-SCU 采用了 MPI 三维剖分，因此程序实现时让每个进程一次读取 $nx*NY$ （而不是 $nx*ny$ ）数据，然后在其中截取自己所需的数据段。详情可见本文已经开源的代码。

OpenCFD-SCU 写文件的实现方式要比读文件更复杂，主要是要兼容 Fortran 的 `WRITE` 函数，因此每次写出前要行前后要填一个 INT 型变量，存储此行的字节数，而且如果写文件的函数调用过多会极其影响效率，因此先通过 `MPI_Gatherv` 函数将数据汇总，再并行写出。

并行 IO 的设计让本文在几分钟内就能读入百亿算例的网格和流场数据，有效支持了超大规模直接数值模拟的顺利开展。

3.6 小结

本章介绍了利用 GPU 加速的异构有限差分法软件 OpenCFD-SCU 的软件框架，三维 MPI 剖分方式，流水线设计等。为了进一步提升程序性能并能够实现计算通讯重叠，软件设计时将无粘项内、外区、粘性项内、外区放到了不同的流水线，利用 GPU 上的多线程技术（Stream 技术）进行并发执行。MPI 通讯时采用三维数据剖分方式，临近进程仅交换数据块外薄薄几层数据，而且只有外区计算需要用到这些数据，因此可以实现外区通讯与内区计算的重叠。本部分还介绍了 MPI 并行 IO 的实现方式，让 MPI 三维剖分时的所有进程可以有序、高效地从一个文件中获取自身进程计算所需的数据，破解了超大规模直接数值模拟时的 IO 瓶颈。

第4章 OpenCFD-SCU 在 CPU-GPU 异构体系下的优化方法

4.1 引言

本节介绍 OpenCFD-SCU 的硬件环境、GPU 加速的实现和优化技术。GPU 编程和 CPU 编程最大的区别在于前者采用了细粒度的多线程编程方法，要求程序员更熟悉硬件结构才能获得更好的性能。在 OpenCFD-SCU 的设计中，让每个 GPU 线程负责计算一个网格点，遍历多个块来完成所有网格点的计算。优化部分主要集中在 GPU 数据访问方面。通过使用共享内存、Warp Shuffle 等技术，内核函数的性能在优化前后提高了一倍多。程序针对 CPU 与 GPU 数据传输和核函数内的计算也做了优化。

4.2 硬件环境

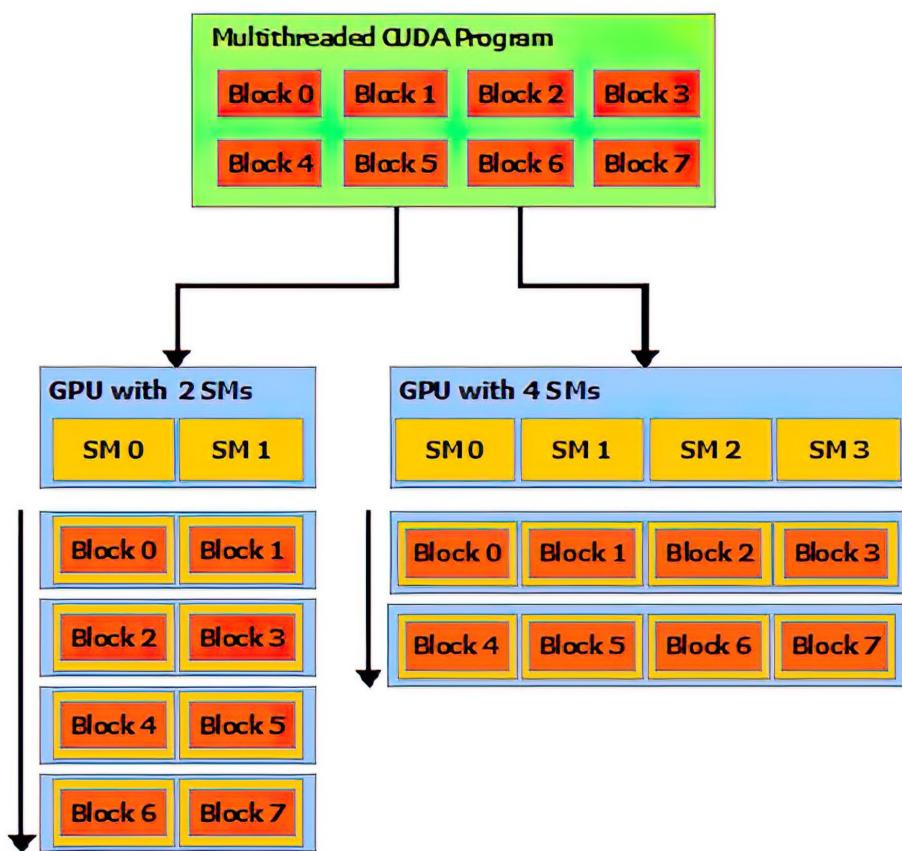


图 4.1 自动伸缩性^[1]

Figure 4.1 Automatic Scalability^[1]

在本文的第一章中已经对 GPU 硬件进行了简单的介绍，本节对 GPU 硬件结构进行进一步介绍。目前主要的 GPU 厂商有 NVIDIA 和 AMD 等，本文软件部署和开展大规模计算时的 CPU/GPU 异构超级计算机为中科曙光系列计算机，其

上安装的 GPU 是海光系列 GPU，或称之为海光协处理器（DCU）。第一代 DCU 是在 AMD Vega 20 架构的基础之上进行的研发与国产化。其技术参数对标 AMD Radeon Instinct M160 GPU。

AMD GPU 的架构与 NVIDIA GPU 并不完全相同，实际上即便是 NVIDIA 自己的 GPU 产品，每代产品的架构也会有一定差异，但不管是不同厂家的产品还是相同厂家不同代系的产品，到目前为止，GPU 在基础架构上都有着一定的共同点。GPU 架构的共性之处便是组织核心的方式与组织内存的方式。对于 GPU 核心的组织，由于 GPU 的核心数量非常多，为了便于管理，采取了一种特殊的组织架构，即单指令多线程（Single Instruction Multiple Threads，SIMT）。在物理层，GPU 分布着若干个核心，在逻辑层，这些核心每一个都是一个执行任务的单元，被称之为线程（Thread）。在物理层，GPU 核心被包裹在一个叫流式多处理器（SM，AMD GPU 称之为 CU）的结构中，在 SM 中有一个硬件结构叫线程束调度器（Warp Scheduler），控制着 SM 中的一组核心。在逻辑层，多个线程被组成一个线程束（Warp），线程束是 GPU 上的最小执行单元，在任何时刻一个线程束里的所有线程都会同时执行同一条指令，当然，它们可以在同一个指令的指示下处理不同位置的数据，逻辑层和物理层具有某种程度的对应关系。就像 SM 内有多个 Warp Scheduler 一样，在逻辑层上多个 Warp 还会被包装成一个线程块（Block）。Block 内的所有线程是资源共享的，此处的资源主要指 SM 内一些硬件资源如寄存器（Register），共享内存（Shared memory），缓存（Cache）等。Block 的大小是可以人为设定的，所以，当单个线程内用的寄存器、共享内存等资源多了以后，就会导致 Block 的大小受限，会影响程序并发性的提升。当然，物理层与逻辑层也不是一一对应的，比如 GPU 上设置的 Block 数量可以远远多于 SM 个数，而且对于版本较新的 GPU，同一时间一个 SM 上也可能有多个 Block 在工作。在逻辑层，Block 的数量由 Block 的大小和网格（Grid）的大小来决定，Grid 即为执行一个任务需要的总线程数，它一般由提交的任务规模决定，比如让一个 GPU 计算网格规模为 $nx \times ny \times nz$ 的任务，如果一个线程计算一个点位，则 Grid 大小便为 $nx \times ny \times nz$ 。GPU 硬件上的每个 SM，遍历执行完每一个 Block 的计算，即完成了整个 Grid 的计算后，GPU 上的一个任务就执行完成了。以上所述的便是目前大部分 GPU 核心调度的基本逻辑。

图 4.1 展示了应用程序与 GPU 硬件的关系，这从一定程度上反应了逻辑层与物理层之间的对应关系，GPU 程序被分为多个 Block（图示为 8 个）后，它们被发射到 GPU 上去，对于只有两个 SM 的 GPU 硬件，需要 4 次遍历才能完成计算，而对于 SM 数量更多，计算能力更强的带有 4 个 SM 的 GPU 硬件，只需要 2 次遍历就能完成计算，显然，SM 数量少的硬件完成计算需要更多的时钟周期。

此外，相似于 GPU 核心的多层次组织形式，GPU 内存也被组织成了多层次。图 4.2 展示 GPU 的内存组织形式。GPU 的内存分为 SM 内部的具有本地属性的内存和 SM 外部的具有全局属性的内存，外部的内存主要是 GPU 的全局内存（Global memory）又称之为显存，它是 GPU 上空间最大的存储结构，但同时也是

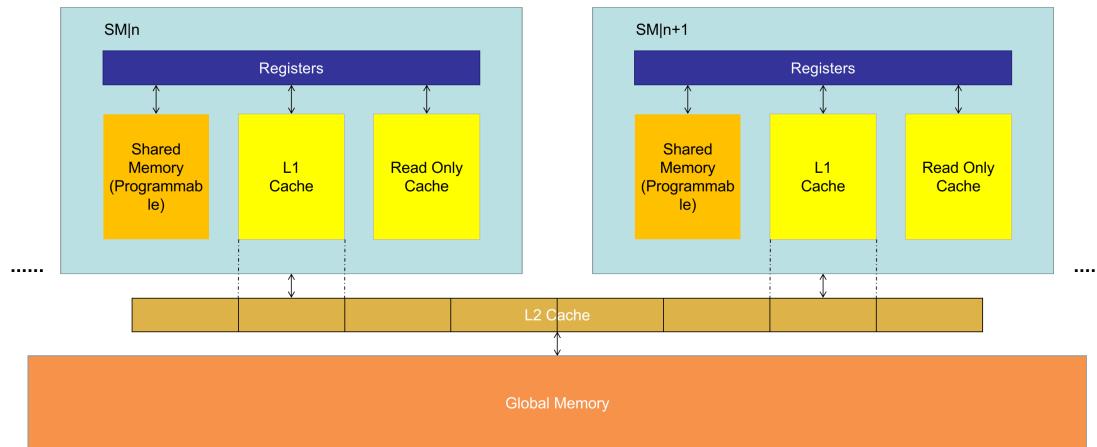


图 4.2 GPU 内存结构

Figure 4.2 Memory structure of the GPU

GPU 上访问延迟最高的内存，Block 内的线程通过 2 级缓存（L2 cache）对全局内存进行访问，访问全局内存时一次内存事务会获取到一组具有很宽位宽的连续数据，GPU 的高的内存访问带宽便是通过这样高的访问位宽实现的。SM 内部的内存对于不同架构的 GPU 不尽然相同，一般有一级缓存（L1 cache），这是 SM 内部的缓存，延迟比 L2 cache 小；共享内存（Shared memory），是一种可编程的缓存，延迟较小，共享内存由许多个容器（Bank）组成，不同线程访问同一个 Bank 会导致访问冲突；在有些架构中 SM 内部还有只读缓存用以加速访问一些只读变量，如常量缓存（Constant memory）和纹理缓存（Texture memory）也是只读缓存；此外每个线程还有自己本地独享的存储结构，称之为寄存器（Register），寄存器的延迟最低，但其在 SM 内的总数量是固定的，一般而言，线程内的一个局部变量会自动分配一个寄存器，若定义的局部变量过多，寄存器的欠缺会导致 Block 的大小将无法扩大，程序的并发性会受到影响。图 4.3 更清晰地展示了 GPU 内存与逻辑层的对应关系，不同的 Grid 和 Block 可以访问相同的 Global memory，但 Shared memory 只能由同一个 Block 内的线程访问（在新版本 GPU 中，一个 SM 可以同时执行不超过线程束调度器个数的多个 Block，同时在同一 SM 上的多个 Block 会被组成一个线程块簇（Cluster），同一个 Cluster 内的 Block 也可以访问相同的 Shared memory，寄存器只能由一个线程访问）。

本文已经对 GPU 物理层和逻辑层都进行了介绍，尽管 GPU 的硬件组织形式相似，但不同 GPU 的 SM 数量、全局内存大小、缓存大小与位宽、Warp 包含的线程数量、Shared Memory 的大小和 Bank 数量都不尽相同。对于 AMD Radeon Instinct MI60 GPU，其需要关心的指标被列举在表 4.1 和表 4.2 中。表 4.1 列出的是核心相关参数，表 4.2 列出的是内存相关参数。

Radeon Instinct MI60 (Vega 20 GL) 由 13.2×10^9 个晶体管组成，芯片尺寸为 331 mm^2 ，双精度浮点计算能力为 6.144 TFLOPS，OpenCFD-SCU 采用双精度对物理问题求解，因此本文只关心芯片双精度的计算能力。该芯片共有 4096 个物理核心，被放在 64 个 SM 内，而它每个 Warp 包含的线程束也为 64，意味着此架

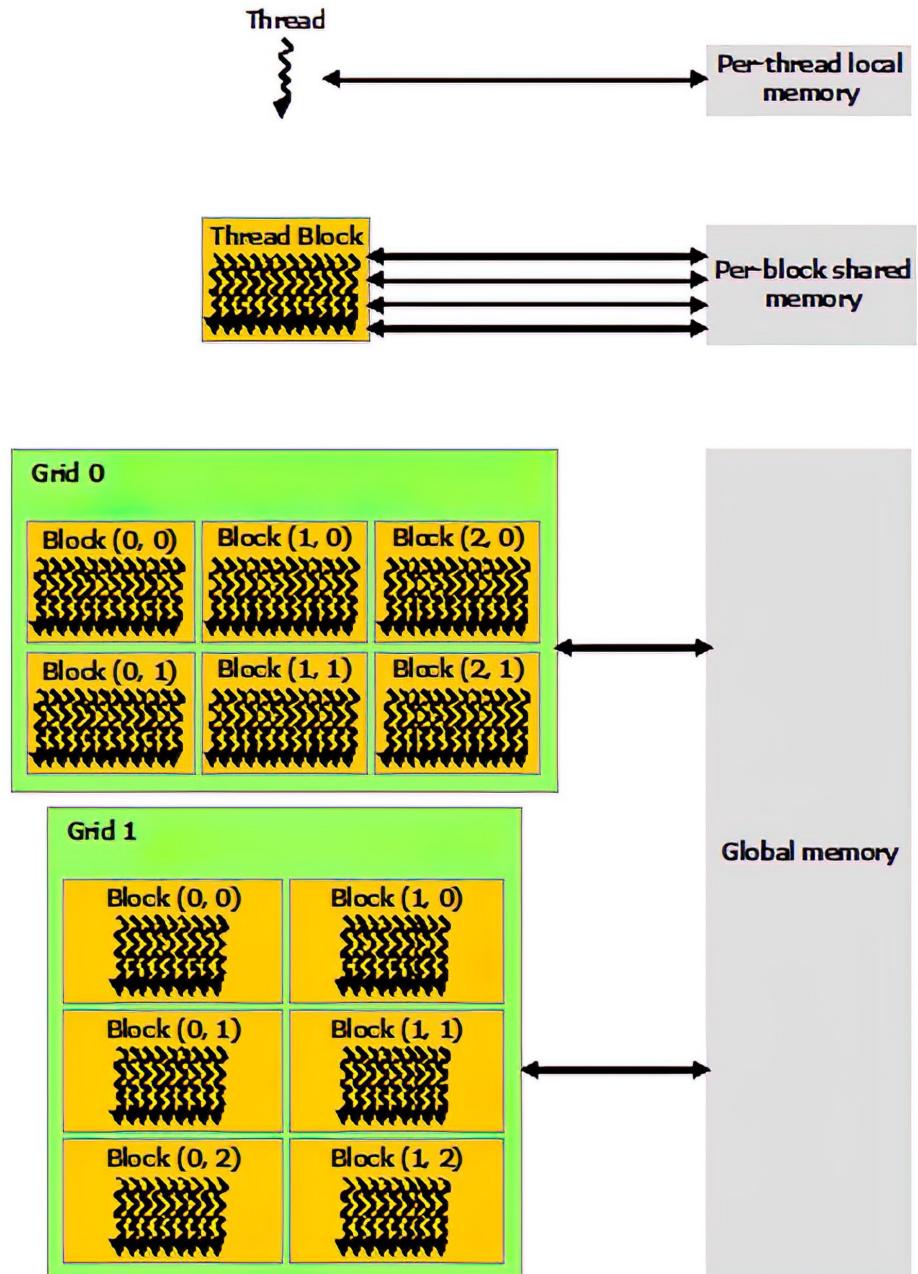
图 4.3 异构编程^[1]Figure 4.3 Heterogeneous Programming^[1]

表 4.1 MI60 (Vega 20 GL) 架构计算能力

Table 4.1 Computing capability of MI60 (Vega 20 GL)

Model	Threads per multiprocessor	Threads per warp	Warp per multiprocessor
Radeon Instinct M160 (Vega 20 GL)	4096	64	64

表4.2 MI60(Vega 20 GL)架构内存相关参数

Table 4.2 Memory-related parameters of MI60 (Vega 20 GL)

Model	Memory bus type	Memory bus width	Memory size	Shared memory bus width
Radeon Instinct MI60 (Vega 20 GL)	HBM2	4096 bits	16 GB	128 bits

构的 GPU 芯片每个 SM 内只有一个线程束调度器。此芯片的内存采用了 HBM2 类型的内存，该内存类型标准是由三星电子、AMD、海力士半导体共同发起制定，采用 3D 堆叠工艺具有高位宽，高能效的特点，Radeon Instinct MI60 GPU 采用的 HBM2 内存具有 4096 比特的内存总线位宽，考虑到 HBM2 由 8 个通道 (Channel) 组成，则全局内存每次内存事务访问的内存位宽为 512 比特。本文所使用的 MI60 具有 16G 的全局内存容量（这决定了每张卡能够计算的问题规模），和 128 比特的共享内存位宽。有关此 GPU 参数的更多详细信息，请参阅^[121]。

4.3 CPU-GPU 系统异构编程方法与实现

GPU 编程方法有多种，比如 OpenCL (Open Computing Language)、CUDA (Compute Unified Device Architecture)、HIP (Heterogeneous-compute Interface Portability) 等。其中 OpenCL 是一种具有通用性的 GPU 编程标准，它最早由苹果公司开发，随后与业界的其它企业如 AMD, IBM, 英特尔, NVIDIA 等合作进行了完善，OpenCL 可以支持跨平台的异构程序部署，但是此种编译方法并不支持各 GPU 平台特有的硬件结构，而且后期缺乏主要 GPU 厂商的支持，导致其性能发挥和可移植性都受到了挑战。CUDA 是 NVIDIA 开发一种 CPU/GPU 异构平台的程序编程模型，NVIDIA 公司作为 GPU 业界的龙头厂商，它所制定的 CUDA 编程标准受到了最多异构程序开发者的使用，软件生态最为完善。HIP 是 AMD 公司开发的一套 GPU 程序编程模型，它的函数调用接口与 CUDA 类似，AMD 还提供了一套可将 CUDA 代码替换为 HIP 代码的软件工具。

本文选择 CUDA 作为 GPU 编程的主要模型，是考虑到 CUDA 是目前最主流的 GPU 编程方法，且 CUDA 程序可以通过转码工具转变为 AMD GPU 所支持的 HIP 程序，这使 CUDA 编程方式也具有一定的通用性。

在 NVIDIA 系列显卡上，对 CUDA 程序编译时使用的编译器为 NVCC 编译器，它可以识别文件后缀名为 “.cu” 的程序文件。使用 CUDA 编程模型进行编程时，把只在 CPU 上运行的函数称为主机函数；只在 GPU 上运行的函数称之为核函数；连接 CPU 与 GPU，在 CPU 上调用但用以启动核函数的函数称之为控制函数。在 OpenCFD-SCU 中主机函数主要包含一些控制程序计算流程的函数，负责节点间数据通讯的函数，以及负责 CPU 与 GPU 内存分配的函数等。控制函数也可以看作主机函数的一种，但它只有 API 的调用而没有计算，控制函数通过设置 Grid 大小、Block 大小，静态声明的 Shared memory 大小，计算起始位置等，

对核函数进行调用与发起。核函数则负责 GPU 上的计算任务。在.cu 文件中，识别主机函数需要通过“`_host_`”这一特殊标识符进行声明，而核函数则需要用“`_global_`”或“`_device_`”标识符进行声明，“`_global_`”声明的核函数可以在主机端被调用，而“`_device_`”声明的核函数只能在设备端被调用，即只能被核函数调用。

在控制函数中，对核函数进行调用时，利用“`<<< ... >>>`”语法来告诉核函数 Grid 和 Block 的信息，函数调用方式如下：

```
func <<< griddim, blockdim, 0, * stream >>> (&para) (4.1)
```

`griddim` 和 `blockdim` 即为启用核函数时设置的 Grid 和 Block 的大小，它们的变量类型都为 `dim3` 型，这是一种 CUDA 中特有的数据类型，它可以看成一个最大维度为三，最小维度为一的可变维度的结构体，式4.1中第三个变量用于静态声明共享内存的大小，此处不使用共享内存，所以设为 0，第四个变量设置核函数在哪条流上运行，流的数据类型为“`cudaStream_t`”，它也是 CUDA 中特有的数据类型，在使用流之前，需要先使用运行时 API 函数“`cudaStreamCreate`”对流进行创建，如果不填第四个变量，则核函数会在默认流上运行，默认流不需要创建。核函数被启动之后，它能够通过 `dim3` 型的内嵌变量 `Block.dim` 和 `threadIdx.x` 获得 Block 维度索引的信息以及 Block 内线程索引的信息。

在 CUDA 程序编写时，经常会遇到 CUDA 与其它语言混编的情况，比如与 C 语言的混编。在 Linux 环境下可以使用“`ar`”指令将其它语言的编译后文件打包成静态链接库，最后与 CUDA 编译后文件链接。其它语言间的混编也可使用此方法，需要注意函数声明时添加“`extern`”标识符。

在 AMD GPU 环境下部署 CUDA 程序时，需要对 CUDA 函数进行转码，CUDA 代码被转化为 AMD 平台支持的 HIP 代码后，程序才能在 AMD GPU 上编译并运行。AMD 提供了一个名为 HIPIFY 的转码工具，HIPIFY 的具体使用方法可参见官方提供的转码指导手册^[122]，其使用方法类似于编译器的使用方法。在使用 HIPIFY 转码时，90% 的基本函数都可以转化为能在 AMD GPU 上直接运行的 HIP 函数，但对于个别 CUDA API 函数，采用 HIPIFY 转码后不能直接运行，如“`CUDAMemcpyToSymbol`”，原因是在 CUDA 和 AMD 平台上此函数的形参类型不完全一致，因而需要进行简单的调整。用户可参见 AMD 官方手册进行手动修改^[122]。对于一些较新的 CUDA API，AMD 平台可能没有对应的 API 支持，因而函数转码后可能不能运行或运行无效，对于 NVIDIA GPU 和 AMD GPU 上不通用的硬件结构，利用此硬件的函数在转码后可能会失效，比如在常量内存的使用中，常量缓存是 NVIDIA GPU 伏特（Volta）架构 GPU 的 SM 内硬件结构，可以加速常量变量的读取速度，对使用常量缓存的变量可使用“`_constant_`”标识符进行声明，但对于 AMD Vega 20 架构，并不存在常量缓存这种硬件结构，因而在转码后所有使用常量内存的变量会被放置到全局内存中，加载速度将会变得很慢。纹理内存也有类似情况，在 AMD GPU 上不存在纹理内存这一硬件结

构，CUDA中绑定纹理内存的代码在AMD GPU上不会生效，在AMD GPU运行此类代码不会让程序发挥出预想中的性能，反而纹理内存绑定过多时还可能导致程序启动失败。



图 4.4 异构程序实现过程示意图

Figure 4.4 Flow diagram of heterogeneous program implementation

图 4.4 展示了 CPU/GPU 程序的大致运行流程，CPU 和 GPU 上各自申请内存，然后 CPU 把数据上传到 GPU，之后 CPU 上调用主机端的控制函数，GPU 上的核函数开始执行计算，最后计算得到的数据被传回 CPU。

在 Block 设置时，可以被设置为一维、二维或三维，在 OpenCFD-SCU 中，Block 被以三维的形式组织起来。如图 4.5，Block 中的每一个线程只负责一个网格点的计算，在程序中，本文把 Block 的大小设置为了 $8 \times 8 \times 8$ 或 $8 \times 8 \times 4$ ，即每一个 Block 只负责 $8 \times 8 \times 8$ 或 $8 \times 8 \times 4$ 这么大的一个三维小区域的计算，Block 的维度这样设置是有原因的，在前一小节，本文已经提到了本文所使用的超算上的 AMD Vega 20 架构 GPU 的 Warp 尺寸为 64，这样的 Block 设置就可以让一个 Warp 正好符合 Block 上一层网格点的计算，Warp 内的数据也可以比较容易的共享，全局内存的带宽可以被更好的利用。目前，在本文所使用的 16G 显存的 GPU 上，本文最大进行了网格规模为 280^3 的计算，即一张 16G 显存的 GPU 显卡能够计算约 2.2 千万网格的任务。OpenCFD-SCU 的所有计算都由 GPU 完成，CPU 只负责数据交换和逻辑控制。

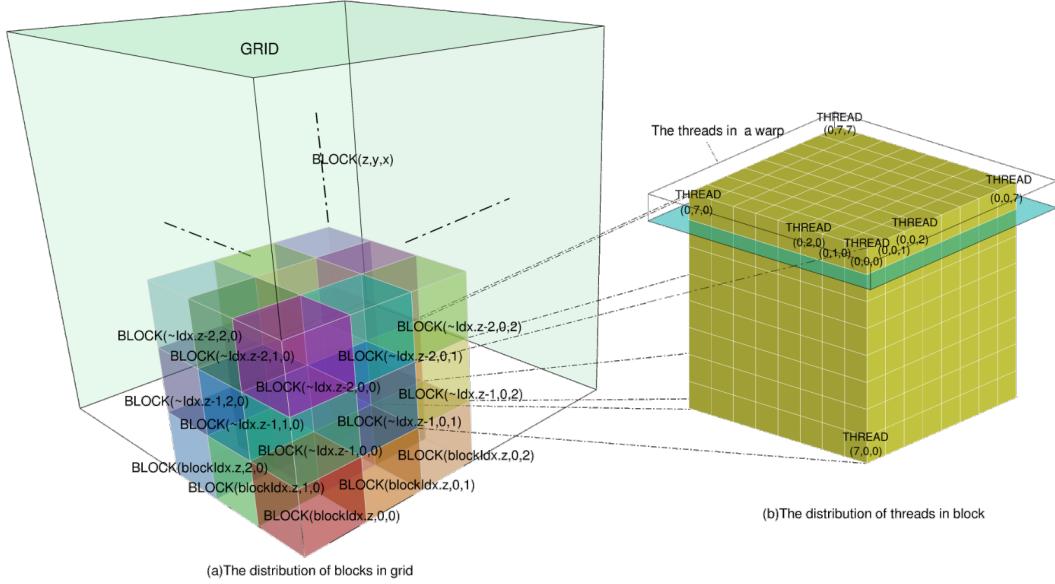


图 4.5 OpenCFD-SCU 中的三维线程块划分

Figure 4.5 Three-dimensional block used by OpenCFD-SCU

4.4 OpenCFD-SCU 程序热点分析

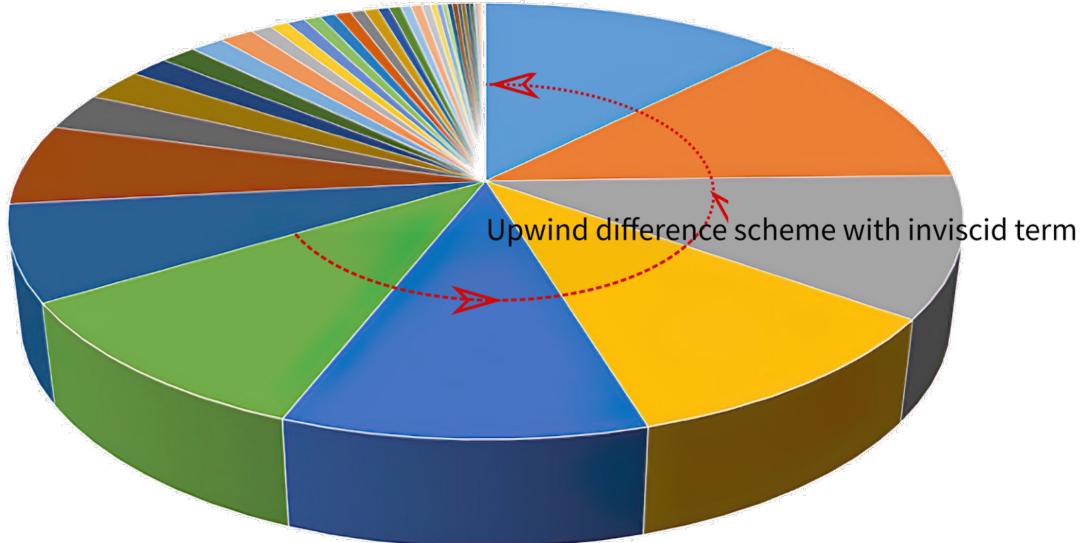


图 4.6 程序热点分析

Figure 4.6 Hot spot analysis

本节之前的两个小节，本文对 GPU 硬件信息和 CPU/GPU 异构程序的编程与实现方法进行了介绍。后续部分会对 GPU 程序的优化方法进行介绍，在此之前，需要了解本文所要优化的程序的瓶颈与热点。

图 4.6展示了未优化版本的 OpenCFD-SCU 在运行时各部分的时间占比图。测试时采用的数值方法为，无粘项先采用 Steger-Warming 流通矢量分裂，后使用 WENO-SYMBIO 格式离散，粘性项采用 6 阶中心格式离散。从图中可以看出，

OpenCFD-SCU作为一个差分程序，差分计算占程序总时间的绝大部分，尤其是无粘项的WENO-SYMBO差分格式计算占了程序总时间的70%，所以程序优化重点放在差分计算上，特别是无粘项差分。差分计算属于典型的模板计算问题，模板计算问题的瓶颈普遍是访存受限，OpenCFD-SCU中差分计算的瓶颈主要也是访存受限。访存优化的总体思路是保证全局内存的合并访问以保证全局内存带宽的高效利用，并尽可能减少全局内存的内存事务次数。

4.5 优化方法

4.5.1 全局内存的合并访问

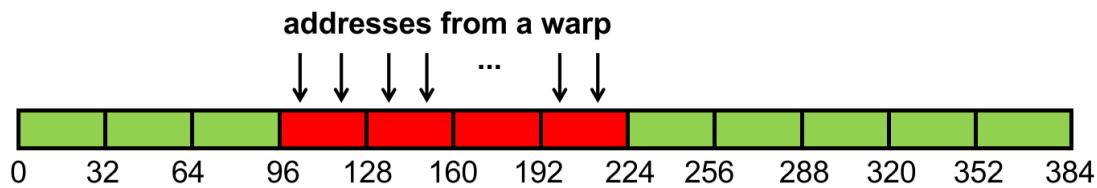


图 4.7 全局内存合并对齐访问

Figure 4.7 Global memory coalesced aligned access

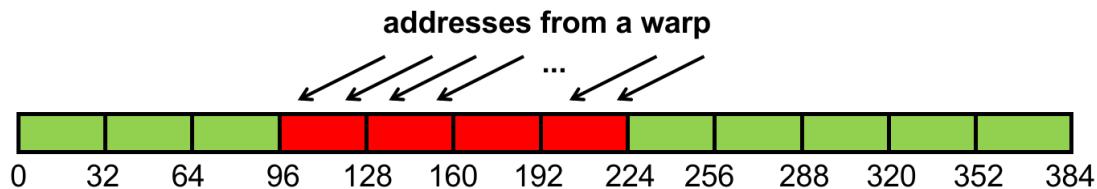


图 4.8 全局内存合并未对齐访问

Figure 4.8 Global memory coalesced unaligned access

访存优化首先要考虑的是全局内存合并与对齐访问，何为合并与对齐访问？在之前的小节中，本文已经介绍了GPU的内存的高带宽是由高位宽实现的，即一次内存事务可以获取到内存中地址连续的一行的数据。若一个Warp中的所有线程所需的数据都在这次内存事务获取到的连续数据段中，那么一次内存事务就完成了所有访问，如果Warp中线程所需的数据是内存中连续的数据，而且需要的第一个数据正好是内存事务访问的首地址的数据，需要的最后一个数据是内存事务访问数据段中的最后一个数据，那么数据访问就是合并与对齐的，此时全局内存带宽可以到达100%的利用率。如图4.7所示便是合并与对齐访问的情况，假设一次内存事务的访问位宽为32字节，Warp需要从全局内存中获取128字节的连续数据，它需要的第一个数据地址与内存事务访问的第一个内存地址对齐，只需要4次内存事务，就把128字节的数据读到了Warp中。图4.8展示了合并但不对齐的情况，Warp仍然需要从全局内存中获取128字节的连续数据，但它需要的第一个数据不是内存事务获取到的第一个数据，而是第一次内

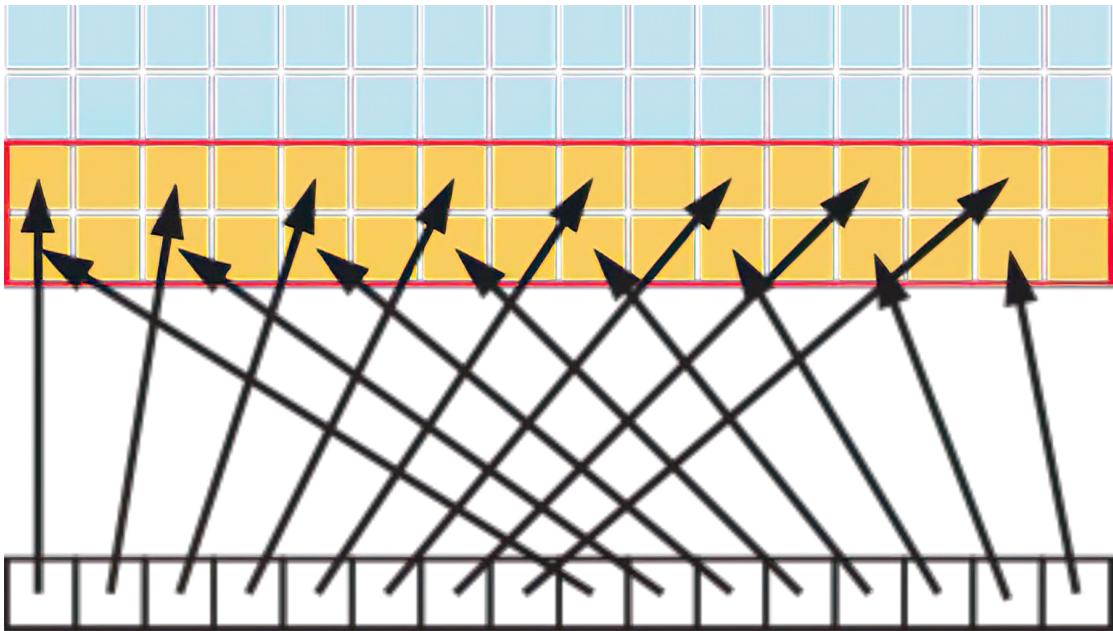


图 4.9 全局内存对齐未合并访问^[1]
Figure 4.9 Global memory uncoalesced unaligned access^[1]

存事务访问到的数据段中间的某个数据，在此情况下，同样读取 128 字节，需要 5 次内存数据，全局内存带宽利用率为 80%。全局内存访问同时满足合并与对齐是全局内存访问优化的目标，但合并访问远比对齐访问重要，使程序对齐访问的难度也远大于合并访问。图 4.9 展示对齐但不合并的情况，尽管 Warp 需要的第一个数据正是内存事务访问首地址的数据，但由于需要获取的数据不连续，仍然需要两次内存事务，全局内存带宽利用率只有 50%。

以上介绍主要基于全局内存的内存事务位宽为 128 比特（32 字节）的情况，NVIDIA 系列的大多数 GPU 全局内存访问位宽都是 32 字节。而如之前所介绍的（见小节 4.2），本文使用的 AMD Vega 20 架构 GPU 全局访问位宽为 512 比特（128 字节），这是需要注意的不同之处。在小节 4.3 中，本文提到 OpenCFD-SCU 的 Block 设置为 $8 \times 8 \times x$ ，一方面是让一个 Warp 正好处理 Block 的一层，另一方面，如图 4.10，当 BlockIDx.x 为 8 时，对于本文常用的 7 阶差分格式，连续 8 个的差分计算需要的模板正好是连续 16 个数据，由于 OpenCFD-SCU 计算采用双精度，因此 16 个数据正好对应 128 字节的全局内存访问位宽。在对齐的情况下一次内存事务即可获取到全部的数据。

另外，需要注意的是对于二维和三维内存块，为了保证每一行第一个数据地址正好对齐于内存事务访问的首地址，需要在申请内存时使用特殊的 API 函数，在内存块为三维时可使用 `cudaMalloc3d` 函数申请对齐的内存，它通过填充一些数据，从而保证每行内存的起始位置是内存事务的开端。

在 OpenCFD-SCU 中，正如章节 3 所介绍的，为了能够实现程序计算与通讯的重叠，三维数据块被分成了内区和外区。为了保证外区与内区计算的合并与对齐，外区的宽度被设置为了 8 字节，对应一次内存事务可以访问的数据宽度。

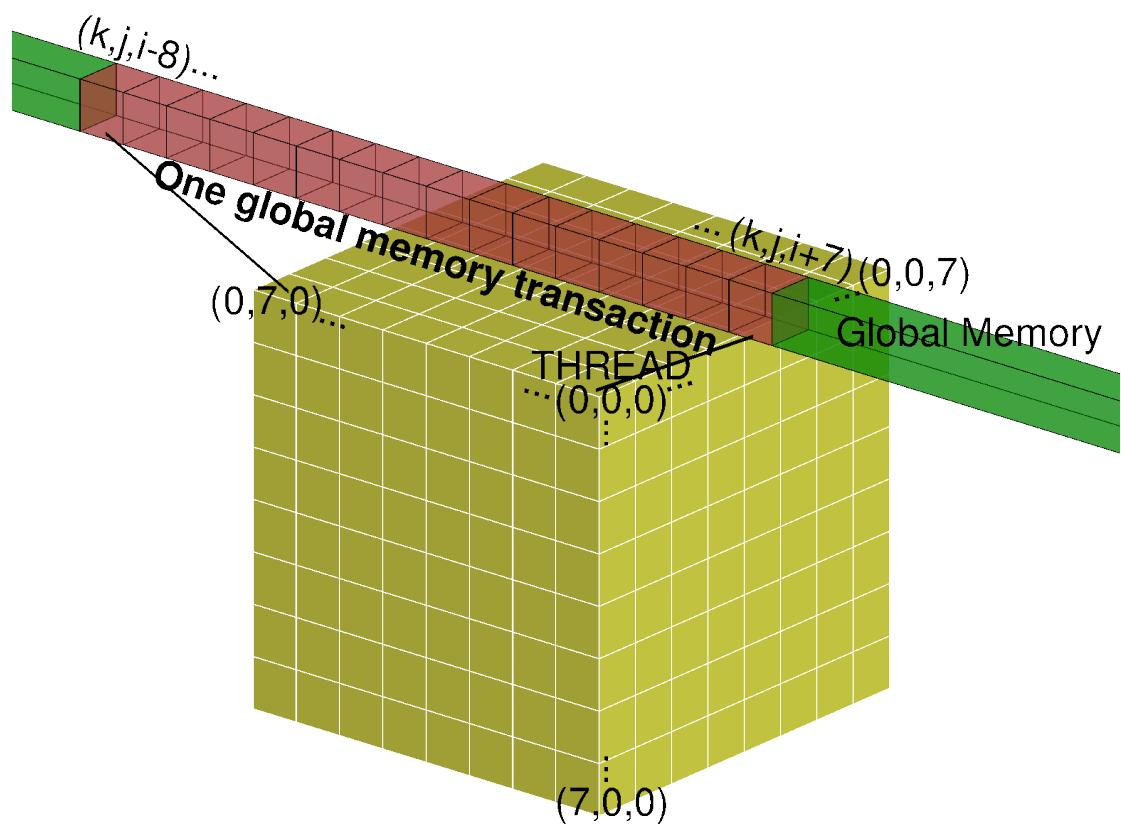


图 4.10 线程块获取数据过程

Figure 4.10 Intra-block thread acquires data

在差分计算时不同的差分格式对应的模板宽度是不同的，很难保证每个格式都能让每个 Block 中的 Warp 全局内存访问对齐，但不完全对齐对计算效率的影响也并不大，因为 L2 cache 的存在，不同 Block 间获取的重复数据很大程度可以在 L2 cache 中命中。因此，全局内存访问的合并比对齐更重要。

4.5.2 线程束内的数据交换 Warp-shuffle 与冗余计算设计

NVIDIA 和 AMD 的 GPU 提供了一种用于 Warp 内线程交换本地变量的技术，称之为线程束洗牌（Warp shuffle），这是一种不通过共享内存，开销极小的 Warp 内共享数据的方式。常见的 Warp-shuffle 函数有：

`int __shfl(int var, int srcLane, int width=warpSize)`: 调用此函数的线程将变量 var 的值传递到同一个 Warp 中索引为 srcLane 的线程。

`int __shfl_up(int var, unsigned int delta, int width=warpSize)`: 调用此函数的线程将变量 var 的值传递到同一个 Warp 中索引比自身小 delta 的线程。

`int __shfl_down(int var, unsigned int delta, int width=warpSize)`: 调用此函数的线程将变量 var 的值传递到同一个 Warp 中索引比自身大 delta 的线程。

`int __shfl_xor(int var, int laneMask, int width=warpSize)`: 调用此函数的线程将变量 var 的值传递到索引为自身索引与 laneMask 做异或运算的线程。

NVIDIA 和 AMD 原生的 Warp-shuffle 只支持 4 字节的变量交换，由于 OpenCFD-SCU 采用双精度计算，因此本文先将变量通过强制类型转换，转换为两个 int 型的 4 字节变量，然后通过 2 次 Warp-shuffle 将数据传递到临近线程，再将两个 4 字节 int 型数据强制类型转换回原来的双精度数据，参考代码如下：

```
#define __shfl_up_double(val, delta, width)
    __shfl_up_double_(*(int2 *)&val), delta,
    width)
2 __device__ __forceinline__ double __shfl_up_double(
    int2 &val, unsigned char delta, unsigned char
    width){
    int2 out = *(int2 *)&val;
4    out.x = __shfl_up(out.x, delta, width);
    out.y = __shfl_up(out.y, delta, width);
6    return (*((double *)&out));
}
```

OpenCFD-SCU 中所有的迎风差分都为守恒形式的差分，在对其计算时，为了减少计算量，每个线程只计算守恒形式差分的一支，再进行重组。即相邻线程分别计算 $\hat{f}_{j-1/2}$ 和 $\hat{f}_{j+1/2}$ ，再进行相邻线程间计算结果的通讯，之后就可以重组出导数 $\partial f / \partial x = (\hat{f}_{j+1/2} - \hat{f}_{j-1/2}) / \Delta x$ 。由于 Warp 大小为 64，在 Block 尺寸为

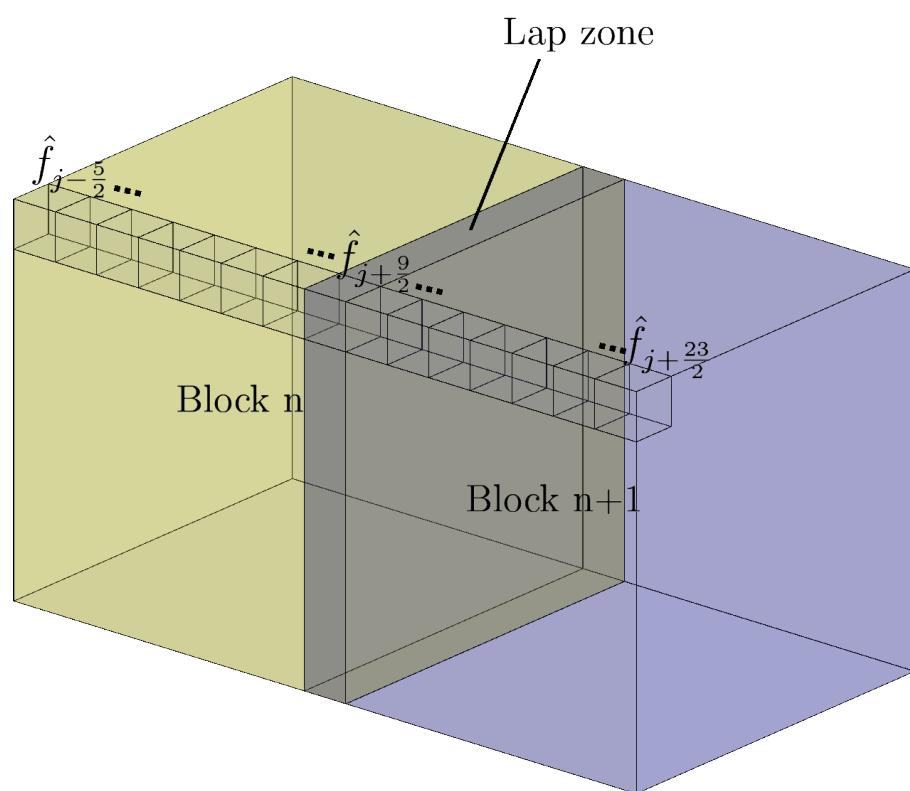


图 4.11 线程块之间的冗余区

Figure 4.11 Redundancy zone between blocks

$8 \times 8 \times x$ 时，一个 Warp 可以计算出 7×8 个点的差分，则对于一个 Block 来说，就能计算 $7 \times 8 \times x$ 个点的差分。对于 Warp 内线程间通讯的过程，本文使用 Warp-shuffle 来完成，对于 Block 间，通过设计冗余计算使之不存在通讯，如图 4.11，Block(n) 计算从 $\hat{f}_{j-5/2}$ 到 $\hat{f}_{j+9/2}$ 的数据，Block($n + 1$) 计算从 $\hat{f}_{j+9/2}$ 到 $\hat{f}_{j+23/2}$ 的数据，尽管所有 Block 都计算了 $\hat{f}_{j+9/2}$ ，但避免了 Block 之间的数据交换。这样做的原因是 Warp-shuffle 只适用于 Warp 内的通讯，不能用于 Block 间线程的数据通讯，而 Block 首或尾线程也需要临近线程传递来的值算差分，其所需的值位于临近 Block 的线程中。如果将数据写回全局内存中，进行线程同步，再从全局内存中读取临近 Block 写入的数据，这样做的开销很大。冗余计算的设计，让相邻 Block 重复计算一个点，尽管增大了计算量，但相比于过多低效的全局内存读写操作依然是更优的选择。

4.5.3 基于共享内存的线程重排

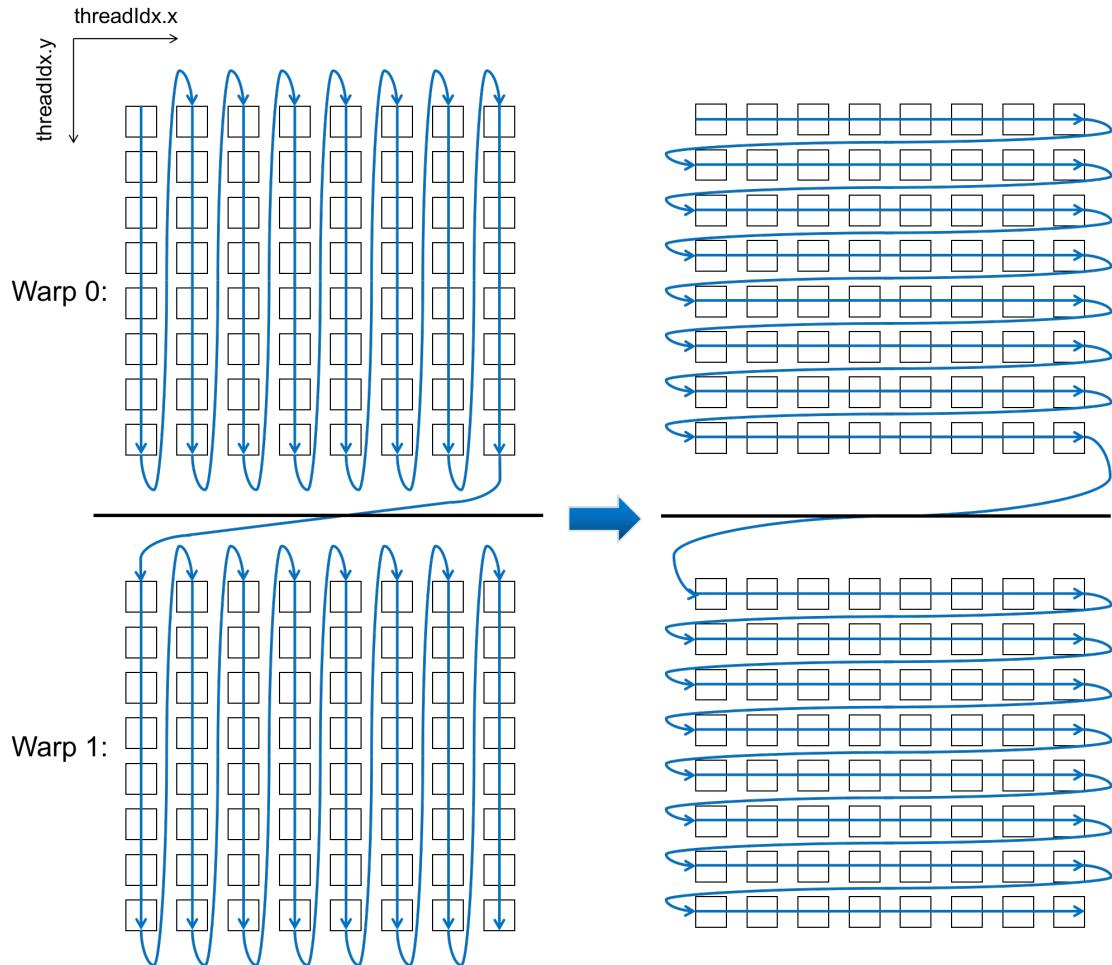


图 4.12 线程束内的数据获取方式与线程重排

Figure 4.12 Data acquisition mode and thread rearrangement in a warp

全局内存的合并访问能够极大地提高全局内存带宽的利用率，基于 Warp-shuffle 线程间数据交换能够以极低的开销完成 Warp 内数据交换从而减少计算

量，这些都是重要的优化方法，在x方向的计算时，由于内存中连续的数据是以x连续的，因此合并访问和Warp-shuffle能够很容易地同时实现，但对于y, z方向的差分计算，当y, z方向全局内存合并访问时，Warp中连续的线程获取的数据实际上并不是y, z方向连续的数据，因此Warp-shuffle就会变得异常的复杂。为了解决这个问题，提升y, z方向的差分计算速度，本文利用共享内存设计了线程的重排。其原理是，一个Warp通过合并访问的形式将计算所需的数据读取到线程中，利用Shared Memory将数据重排，使相邻线程获取到y, z方向的连续数据，进行守恒形式差分一支的计算，然后使用和x方向计算相同的Warp-shuffle交换计算结果，重组出差分值，关于线程重排的示意图参见图4.12，由于本文设计的Block维度为 $8 \times 8 \times x$ ，因此线程重排就相当于将线程中的值传递到索引矩阵转置后的对应线程中去。而且由于AMD Vega 20 GL架构的Warp尺寸是64，所以一个重排操作就在一个Warp内，也不需要进行线程同步，避免了额外开销。线程重排的具体代码如下：

```

1 unsigned int x = coords->x = blockDim.x * blockIdx.x +
    threadIdx.x;
unsigned int y = coords->y = (blockDim.y - 1) * blockIdx
    .y + threadIdx.y;
3 unsigned int z = coords->z = blockDim.z * blockIdx.z +
    threadIdx.z;

5 unsigned int ID1 = 128 * threadIdx.z + 16 * threadIdx.x +
    threadIdx.y;
unsigned int ID2 = 128 * threadIdx.z + 16 * threadIdx.y +
    threadIdx.x;
7
shared_memory[ID1] = get_data(f, x, y-LAP+1, z, num,
    offset);
9
if(x < (job.end.x-job.start.x) && y < (job.end.y-job.
    start.y-1) &&
11 z < (job.end.z-job.start.z))
    shared_memory[ID1+8] = get_data(f, x, y+LAP+1, z,
        num, offset);
13
for(int i = ka1; i <= kb1; i++){
15     stencil[i-ka1] = shared_memory[ID2+i+3];
    }
17

```

```

x = coords->x = blockDim.x * blockIdx.x + threadIdx.y;
19 y = coords->y = (blockDim.y - 1) * blockIdx.y +
      threadIdx.x;

```

图 4.13展示了程序在经过线程重排前后一个 Runge-kutta 子步的运行时间线，测试的网格规模为 280^3 。从中可以看出 z 方向的差分在优化前计算所需时间为 45.024ms，经过线程重排后的计算时间是 18ms，计算速度变快了 60%，且与 x 方向差分计算时间基本相同，计算一个 Runge-kutta 子步的总时间也从 380.381ms 减少到 291.615ms，优化后的运行时间比优化前加快 23.4%。

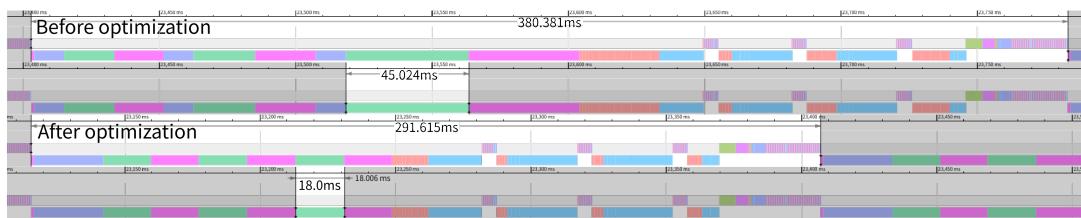


图 4.13 程序优化前后的时间线

Figure 4.13 Program timeline before and after optimization

4.5.4 原子操作

原子操作是 CUDA 和 HIP 中提供的一种多线程可以共同处理某一地址值的指令，它可用于避免多个不同线程处理同一个内存位置时产生的错误，GPU 的不同 Warp 中线程在访问同一个内存地址时，会产生访存冲突，在获取数据时，访存冲突不会导致错误，但是不同线程同一时间向同一个地址写数据时，比如同一时间向同一内存地址上加一些值，访存冲突可能导致最终结果错误。原子操作指令能够让不同线程的读、写、算连贯地执行，但它并不能提升速度。

常见的原子操作指令有：

`atomicAdd(double* address, double val)`: 原子加，向同一地址 address 累加某个值 val。

`atomicSub(int* address, int val)`: 原子减，某一地址 address 的值减去某个值 val。

`atomicExch(int* address, int val)`: 原子替换，用 val 值替换某一地址 address 的值。

`atomicMin(int* address, int val)`: 原子最小值，将自身值 val 于某一地址 address 的值比较，将最小值写到地址 address 中。

`atomicMax(int* address, int val)`: 原子最大值，将自身值 val 于某一地址 address 的值比较，将最大值写到地址 address 中。

在 OpenCFD-SCU 中，采用了多流水线用以完成计算通讯重叠和加快计算速度，无粘项、粘性项的计算被放到了不同流上，它们在计算完各自的通量之后，会向同一个内存地址添加自己贡献的通量值。此处使用原子加 “atomicAdd()” 来避免有可能产生的计算结果错误。

4.5.5 页锁定内存与 CPU-GPU 通讯前的数据打包

在计算机操作系统中，为了合理的调度内存资源，内存地址往往是不固定的，比如在内存使用紧张时，一些的计算机操作系统可能会把后台的应用从内存放入磁盘或虚拟内存中去，把更多的物理内存交给前台优先级更高的应用，这在单纯的 CPU 应用运行时可能是好的调度策略，但对于 GPU 应用，主机内存位置的不固定可能会影响 CPU 与 GPU 间的数据拷贝。CUDA 的 API 函数提供了一个命令，可以在主机端申请内存时保证内存固定，即主机上的操作系统不会对这块内存进行分页和交换操作，确保这块内存始终驻留在物理内存上。页锁定内存申请的 API 函数是 “cudaHostAlloc()”。

在 CPU 与 GPU 数据通讯时，页锁定内存可能比 “malloc” 申请的普通主机内存有更快的传输速度，更重要的是，在开辟多条流水线时，只有主机端是页锁定内存的情况下，才能保持某些流上正在执行计算，而某些流可以进行 CPU 与 GPU 的数据通讯。而在 OpenCFD-SCU 中，没有采用 GPU 直连技术，所以 CPU-GPU 数据通讯是需要关注的，在进程间进行 MPI 数据交换时，需要先将数据从 GPU 传到 CPU，交换完成后再将交换得到的数据从 CPU 传到 GPU。页锁定内存保证了计算通讯重叠的可行性。

在 CPU 与 GPU 进行数据通讯时，还有一个需要关注的地方，在差分法的 MPI 通讯时，需要传输的数据仅为每个进程的数据块外薄薄的几层，它们在内存中的存储位置是不连续的，测试发现，针对不连续的数据，CPU 与 GPU 数据传输效率极低。为解决这一麻烦，本文在 GPU 上申请了一块一维的缓存区内存，在数据传输前，在 GPU 上将数据打包进缓存区，再对内存地址连续的缓存区内存进行数据传输，MPI 交换完成后，将获得的临近进程的打包数据进行解包。这一操作能让 CPU 与 GPU 间数据传输速度有量级的提升。

4.5.6 多流水线并发

本文已经在章节 3 中介绍了 OpenCFD-SCU 中流水线的设计。它的主要目的是保证计算与通信的重叠，提升程序并行可扩展性和并行效率，但是，它也可以提升程序在单进程运行时的运算速度，主要原理是，GPU 上负责的访存的硬件和负责计算的硬件可以同时工作，在单流时，对于有明显计算受限或访存受限特点的任务，负载的不均衡会让硬件资源并不能充分利用，多流的设计使得 GPU 硬件综合利用率获得了提升。

本文在 4.14 中展示了开启多流水线后实测的程序时间线，可以看出实际流与原始设计中的流大致相同，流上的计算可以将其它流上的 MPI 通信重叠，达成了预先的设计目的，然而，当多个流同时执行计算任务时，每个流上的内核函

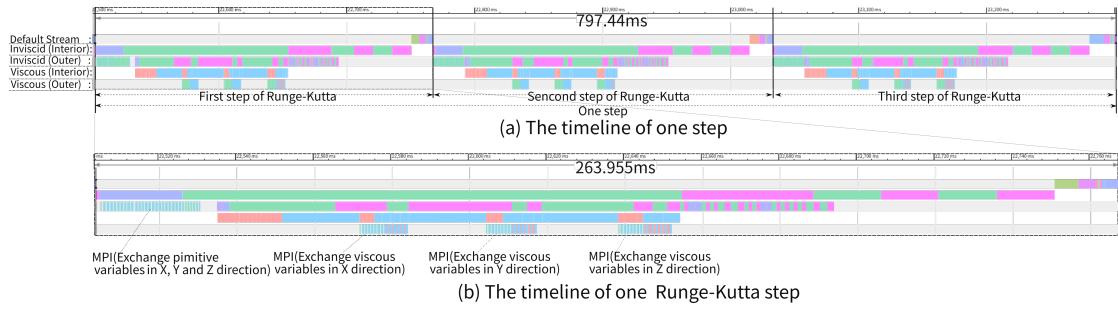


图 4.14 开启多流水线后的程序时间线

Figure 4.14 Program timeline after opening multi-pipeline

数都会变慢，这可能是由于 GPU 的负载过大。但即便如此，多流和单流相比仍然提高了程序的计算速度。一个 Runge-kutta 子步程序的运行时间由图 4.13 中展示的 291 ms 缩短到了 265 ms。

4.5.7 其它优化方法

除了上述的优化方法外，程序在编写时还针对 GPU 硬件特点进行了一些其它方面的优化。

在算法层面，为了减少全局内存访问量，数据块内区三个方向的 Steger-Warming 流通矢量函数被写到了一个核函数内。尽管此举需要多开辟一些存储空间，但也减少了两次原始变量从全局内存中的读取，对于计算速度的提升是有益的。在粘性项计算时，也采用了类似的操作，利用 `_global_` 声明的主机端可调用的核函数包装多个负责不同功能的 `_device_` 声明的设备端核函数，把不同设备端核函数的中间变量放到寄存器中，也可减少全局内存的读写次数。

由于 GPU 并不擅长进行逻辑操作，因此在程序实现时，尽量将判断放到核函数外，这能够避免 Warp 内产生分支（GPU 一个 Warp 内的线程在同一时间会执行同一个操作，如果 Warp 内的一些线程处于判断的一侧，而另一些线程处于判断的另一侧，则 Warp 内会在执行完判断一侧的任务后，再执行判断另一侧的任务，此即为分支）。实际上 GPU 在执行不会产生分支的判断时也会影响效率，不会产生分支的判断是指 Warp 内所有线程都会执行判断的一侧，实际测试发现，GPU 执行任何判断操作的效率都很低，但有些时候把所有判断放置到核函数外，可能会使代码行数大大增加，因此需要在程序可读性与效率之间进行取舍。但在某些情况下判断也可能会带来好处，比如添加少数的判断能够减少全局内存的访问时，在计算边界格式时，边界格式被融合进了差分格式里，本文在程序中增加了判断，通过判断确定线程计算的点是否位于边界，之后选择使用差分格式还是边界格式，这一优化手段在 GPU 上的负担的计算网格规模大时非常有效。

寄存器是 SM 内的稀有资源，它的数量有限。本文在实现核函数时，尽量减少每个线程寄存器的使用量，显式声明的寄存器也尽量复用，尽管编译器有时可以对寄存器进行优化，但通过算法设计依然可以减少一些寄存器的使用，从而提升程序并发性，针对寄存器可以在汇编层级做更多的优化，但由于本文程序的瓶

颈主要是访存，这里只在高级语言层面做了一些优化。

最后，GPU 内部计算浮点数除法的开销远远大于乘法，本文在程序实现时尽量减少了除法运算，对于难以避免的除法，本文把除法计算合并，如 7 阶 WENO 函数中差分计算的除法运算被提到了最外层，虽然增加了很多次乘法运算，但整个差分运算的除法只有一个，经测试发现，核函数的运行速度也得到了提升更快了，7 阶 WENO 函数在 280*280*280 网格的情况下能有 10% 的加速效果。

4.6 小结

OpenCFD-SCU 采用了线程级、任务级、进程级的多级并行。本章对线程级、任务级并行及优化方式进行介绍。线程级并行即为 CUDA 的细粒度并行方式，在本文的设计中，让每个点的差分计算放在一个 CUDA 线程里，采用合并对齐的方式从全局内存中获取数据，最大程度利用全局内存访存带宽，在 Warp 内部，通过使用 Warp_shuffle 等方式，增加数据共享程度，设计出对全局内存的访问需求次数最小的计算方式。在 Block 内部，调动各种资源最大化提升程序并发性。 y , z 方向通过数组的转置，让 y , z 方向的 Warp 内线程访问连续数据，确保 y , z 方向的合并访问。

任务级并行采用 CUDA-stream 进行，将无粘项与粘性项不相互依赖的计算进行并发，并对每个进程的数据块进行分割，将计算分为数据块内部和数据块边缘的计算，边缘计算根据全局内存的 Bank 宽度和数值格式的模板宽度来设置，在保证正确性的前提下，保证内外区在访问全局内存时都可以对齐。由于内区的计算在 GPU 的全局内存中就可以获取到计算所需的全部数据，在外区获取邻近进程的数据时，GPU 的计算可以与进程间的 MPI 通讯重叠。

第 5 章 OpenCFD-SCU 性能与扩展性测试

5.1 引言

本章主要对程序进行测试。测试内容包括以下六个部分：程序正确性、与 CPU 程序相比的加速情况、并行强扩展性和弱扩展性、性能估计、跨平台测试、大规模算例实际 DNS 测试。测试最大 GPU 数量达 24576 块 GPU，测试最大计算网格数量超过 5000 亿，实际算例模拟的网格数量达到 312 亿。

5.2 正确性测试

5.2.1 与 CPU 程序的结果对比

满足正确性要求是程序能够在实际计算中使用的先决条件。OpenCFD-SCU 是由中科院力学所李新亮研究员研发的 OpenCFD-SC 软件移植而来，这是一款已经被广泛使用的成熟的 CPU 端有限差分法软件，它在许多不同情况的湍流直接数值模拟问题中得到验证^[69,123–125]，OpenCFD-SC 目前发出的使用许可证已经超过 250 个，它在湍流直接数值模拟领域内具有一定影响力。本文首先考虑的是与 OpenCFD-SC 计算结果进行对比。

正确性对比测试的方法是，使用 OpenCFD-SC 和 OpenCFD-SCU 对同一个平板算例进行计算，参用相同的格式，此算例的网格设置为 $250 \times 240 \times 90$ ，总网格数 5.4×10^6 ，在不施加任何扰动的情况下，两款程序各自运行 1000 步并保存流场，流场保存的精度为 f16.9，随后使用 linux 下的 diff 工具比较两个保存的流场文件，diff 工具的使命命令是：

```
1 diff -W 240 -y --suppress-common-lines flow3d-cpu.dat
      flow3d-gpu.dat
```

会以列的形式输出文件中不同的行，将输出信息重定向到一个新文件里，用文本编辑器打开观察，如图 5.1。在 1000 步的计算后，程序输出流场数据与初始场已经很大不同。而对于两款程序输出的流场文件，540 万行文件中只有 56 行不同的行，检查发现这 56 行都包含有 0.00，而它们的不同之处在于 0.00 前面的正、负号，这可以证明在 f16.9 的保存精度下，GPU 程序与 CPU 程序的计算结果是完全一致。

5.2.2 正确性的进一步测试

此外本文还通过与实验、他人 DNS、理论结论的对比，进一步证明程序的正确性。测试方法是，本文使用 OpenCFD-SCU 软件对经典的直接数值模拟算例

2	53,74953000	3,20136912	25,34144602	0,95846351	-0,00000000	-0,00000000	53,74953000	3,20136912	25,34144602	0,95846351	-0,00000000	0,00000000	0,00000000
3	53,74953000	4,77991016	25,50244662	0,95112021	-0,00000000	0,00000000	9,00000000	53,74953000	4,77991016	25,50244662	0,95112021	-0,00000000	-0,00000000
4	53,74953000	7,99016040	24,31405919	0,93289279	-0,00000000	0,00000000	9,00000000	53,74953000	7,99016040	24,31405919	0,93289279	-0,00000000	-0,00000000
5	53,74953000	8,15455883	25,09652671	0,93292988	-0,00000000	-0,00000000	9,00000000	53,74953000	8,15455883	25,09652671	0,93292988	-0,00000000	0,00000000
6	53,74953000	9,43311141	23,90109112	0,92372225	-0,00000000	-0,00000000	9,00000000	53,74953000	9,43311141	23,90109112	0,92372225	-0,00000000	0,00000000
7	53,74953000	11,04331661	23,46804238	0,91201062	-0,00000000	0,00000000	9,00000000	53,74953000	11,04331661	23,46804238	0,91201062	-0,00000000	-0,00000000
8	53,74953000	16,80526322	22,02935368	0,87266481	-0,00000000	-0,00000000	9,00000000	53,74953000	16,80526322	22,02935368	0,87266481	-0,00000000	0,00000000
9	53,74953000	16,41075827	22,58748475	0,87067387	-0,00000000	0,00000000	9,00000000	53,74953000	16,41075827	22,58748475	0,87067387	-0,00000000	-0,00000000
10	53,74953000	17,77107141	21,48153518	0,85784076	-0,00000000	-0,00000000	9,00000000	53,74953000	17,77107141	21,48153518	0,85784076	-0,00000000	0,00000000
11	53,74953000	18,20043178	22,00054271	0,85566363	-0,00000000	0,00000000	9,00000000	53,74953000	18,20043178	22,00054271	0,85566363	-0,00000000	-0,00000000
12	53,74953000	19,61084135	20,88342493	0,84277787	-0,00000000	-0,00000000	9,00000000	53,74953000	19,61084135	20,88342493	0,84277787	-0,00000000	0,00000000
13	53,74953000	23,54928397	19,48166198	0,81548409	-0,00000000	0,00000000	9,00000000	53,74953000	23,54928397	19,48166198	0,81548409	-0,00000000	-0,00000000
14	53,74953000	25,65109278	18,64387522	0,80309845	-0,00000000	0,00000000	9,00000000	53,74953000	25,65109278	18,64387522	0,80309845	-0,00000000	0,00000000
15	53,74953000	26,17996565	19,02685843	0,80065590	-0,00000000	0,00000000	9,00000000	53,74953000	26,17996565	19,02685843	0,80065590	-0,00000000	-0,00000000
16	53,74953000	28,42129770	18,03925777	0,79007218	-0,00000000	0,00000000	9,00000000	53,74953000	28,42129770	18,03925777	0,79007218	-0,00000000	0,00000000
17	53,74953000	32,54107022	15,32166532	0,78304646	-0,00000000	0,00000000	9,00000000	53,74953000	32,54107022	15,32166532	0,78304646	-0,00000000	-0,00000000
18	53,74953000	33,13574596	15,59249848	0,77827211	-0,00000000	-0,00000000	9,00000000	53,74953000	33,13574596	15,59249848	0,77827211	-0,00000000	0,00000000
19	53,74953000	37,35759521	12,13821849	0,78782409	-0,00000000	-0,00000000	9,00000000	53,74953000	37,35759521	12,13821849	0,78782409	-0,00000000	0,00000000
20	53,74953000	39,68875956	10,19034145	0,79565874	-0,00000000	-0,00000000	9,00000000	53,74953000	39,68875956	10,19034145	0,79565874	-0,00000000	-0,00000000
21	53,74953000	40,40453249	10,37412070	0,79833786	-0,00000000	0,00000000	9,00000000	53,74953000	40,40453249	10,37412070	0,79833786	-0,00000000	-0,00000000
22	53,74953000	42,65631788	8,18159711	0,78000000	-0,00000000	0,00000000	9,00000000	53,74953000	42,65631788	8,18159711	0,78000000	-0,00000000	0,00000000
23	53,74953000	43,78985936	5,53310024	0,78128577	-0,00000000	-0,00000000	9,00000000	53,74953000	43,78985936	5,53310024	0,821577	-0,00000000	0,00000000
24	53,74953000	44,67167497	5,06433493	0,83749585	-0,00000000	0,00000000	9,00000000	53,74953000	44,67167497	5,06433493	0,83479985	-0,00000000	-0,00000000
25	53,74953000	45,18948674	2,84380152	0,84251518	-0,00000000	0,00000000	9,00000000	53,74953000	45,18948674	2,84380152	0,84251518	-0,00000000	-0,00000000
26	53,74953000	46,14502512	2,49031980	0,84488464	-0,00000000	0,00000000	9,00000000	53,74953000	46,14502512	2,49031980	0,84488464	-0,00000000	-0,00000000
27	53,74953000	46,68863054	0,00000000	0,85603493	-0,00000000	0,00000000	9,00000000	53,74953000	46,68863054	0,00000000	0,85603493	-0,00000000	-0,00000000
28	53,74953000	45,18943467	-2,84307825	0,83857358	-0,00000000	-0,00000000	9,00000000	53,74953000	45,18943467	-2,84307825	0,83857358	-0,00000000	0,00000000
29	53,74953000	47,78771230	-5,53167448	0,81866773	-0,00000000	-0,00000000	9,00000000	53,74953000	47,78771230	-5,53167448	0,81866773	-0,00000000	0,00000000
30	53,74953000	47,52573590	-8,11221783	0,79507519	-0,00000000	-0,00000000	9,00000000	53,74953000	47,52573590	-8,11221783	0,79507519	-0,00000000	0,00000000
31	53,74953000	39,35634047	-10,10732465	0,76753537	-0,00000000	-0,00000000	9,00000000	53,74953000	39,35634047	-10,10732465	0,76753537	-0,00000000	0,00000000
32	53,74953000	36,78288011	-11,9523936	0,74671785	-0,00000000	-0,00000000	9,00000000	53,74953000	36,78288011	-11,9523936	0,74671785	-0,00000000	0,00000000
33	53,74953000	37,35621776	-13,5214892	0,74550333	-0,00000000	-0,00000000	9,00000000	53,74953000	37,35621776	-13,5214892	0,74550333	-0,00000000	-0,00000000
34	53,74953000	34,1557530	-13,5214892	0,72863308	-0,00000000	-0,00000000	9,00000000	53,74953000	34,1557530	-13,5214892	0,72863308	-0,00000000	-0,00000000
35	53,74953000	34,64521542	-13,71701115	0,72815868	-0,00000000	-0,00000000	9,00000000	53,74953000	34,64521542	-13,71701115	0,72815868	-0,00000000	-0,00000000
36	53,74953000	39,21219963	10,45658073	0,70577538	-0,00000000	-0,00000000	9,00000000	53,74953000	39,21219963	10,45658073	0,70577538	-0,00000000	-0,00000000
37	53,74953000	39,31036836	-16,15784017	0,70466905	-0,00000000	-0,00000000	9,00000000	53,74953000	39,31036836	-16,15784017	0,70466905	-0,00000000	-0,00000000
38	53,74953000	36,59977413	-16,88072998	0,6997654	-0,00000000	-0,00000000	9,00000000	53,74953000	36,59977413	-16,88072998	0,6997654	-0,00000000	-0,00000000
39	53,74953000	36,92539334	-17,08623189	0,69983352	-0,00000000	-0,00000000	9,00000000	53,74953000	36,92539334	-17,08623189	0,69983352	-0,00000000	-0,00000000
40	53,74953000	24,9883763	-17,63581198	0,699607356	-0,00000000	-0,00000000	9,00000000	53,74953000	24,9883763	-17,63581198	0,699607356	-0,00000000	-0,00000000
41	53,74953000	24,58817660	-17,88762277	0,69453987	-0,00000000	-0,00000000	9,00000000	53,74953000	24,58817660	-17,88762277	0,69453987	-0,00000000	-0,00000000
42	53,74953000	20,06161563	-18,83911189	0,68895479	-0,00000000	-0,00000000	9,00000000	53,74953000	20,06161563	-18,83911189	0,68895479	-0,00000000	-0,00000000
43	53,74953000	20,23847643	-19,04745220	0,6880095	-0,00000000	-0,00000000	9,00000000	53,74953000	20,23847643	-19,04745220	0,6880095	-0,00000000	-0,00000000
44	53,74953000	18,31310565	-19,49955456	0,688598923	-0,00000000	-0,00000000	9,00000000	53,74953000	18,31310565	-19,49955456	0,688598923	-0,00000000	-0,00000000
45	53,74953000	12,85224814	-20,25190251	0,68477083	-0,00000000	-0,00000000	9,00000000	53,74953000	12,85224814	-20,25190251	0,68477083	-0,00000000	-0,00000000
46	53,74953000	11,25528652	-20,4729017	0,68433781	-0,00000000	-0,00000000	9,00000000	53,74953000	11,25528652	-20,4729017	0,68433781	-0,00000000	-0,00000000
47	53,74953000	9,81876174	-20,85621042	0,68199903	-0,00000000	-0,00000000	9,00000000	53,74953000	9,81876174	-20,85621042	0,68199903	-0,00000000	-0,00000000
48	53,74953000	8,23830737	-20,80758923	0,68386266	-0,00000000	-0,00000000	9,00000000	53,74953000	8,23830737	-20,80758923	0,68386266	-0,00000000	-0,00000000
49	53,74953000	8,32108999	-21,0165153	0,68153568	-0,00000000	-0,00000000	9,00000000	53,74953000	8,32108999	-21,0165153	0,68153568	-0,00000000	-0,00000000
50	53,74953000	5,39836119	-21,02522866	0,68319108	-0,00000000	-0,00000000	9,00000000	53,74953000	5,39836119	-21,02522866	0,68319108	-0,00000000	-0,00000000
51	53,74953000	4,6644471	-21,3065531	0,68155408	-0,00000000	-0,00000000	9,00000000	53,74953000	4,6644471	-21,3065531	0,68155408	-0,00000000	-0,00000000
52	53,74953000	1,34554742	-21,3863635	0,68054232	-0,00000000	-0,00000000	9,00000000	53,74953000	1,34554742	-21,3863635	0,680542		

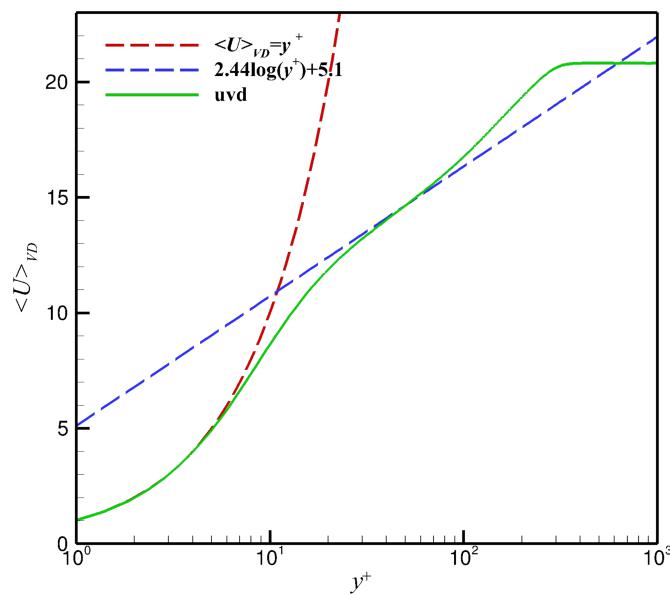


图 5.3 Van Driest 变换后的平均速度剖面

Figure 5.3 Van Driest transformed mean velocity profiles

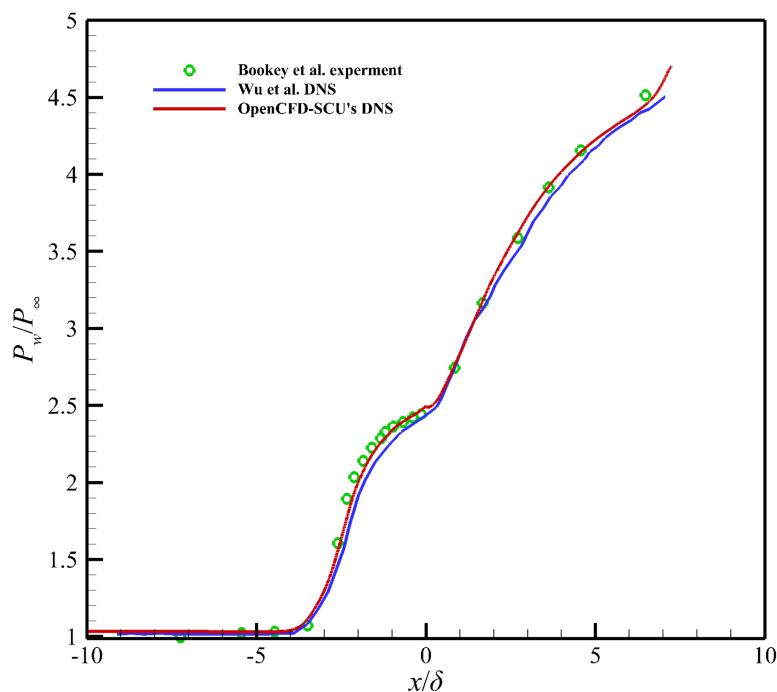


图 5.4 平均壁面压力分布，DNS 与实验的差距约为 3%

Figure 5.4 Mean wall-pressure distribution from DNS and experimental data, with 3% error bars

进行计算。算例采用来流马赫数为 2.9，压缩折角坡度为 24° 的模型。在他人的计算中^[96]，无粘项采用 WENO-SYMBO 格式，粘性项采用 6 阶中心格式，在本文的计算中无粘项的计算格式与他人相同，粘性项采用精度更高的 8 阶中心差分格式。采用比之更密的网格，网格设置为 $16\,100 \times 240 \times 200$ ，总网格数约为 7 亿。在湍流生成的方式上，Wu 等人采用了循环回收的方式发展出的湍流，而本文采用在平板上添加吹吸扰动的方式^[71]发展湍流。

图 5.2 展示了展向中截面上的采用无量纲温度进行无量纲化的瞬时温度云图，可以清晰地看到 $X-Y$ 截面上的湍流发展过程。图 5.3 中展示了参考位置 $x = -35\text{ mm}$ 处经 Van-Driest 变换的平均速度剖面与理论的对比情况，可以看到平均速度剖面在近壁区与对数区都与理论的预测吻合的较好。在图 5.4 展示了壁面压力沿流向分布曲线与实验测量^[126] 和 Wu 等人 DNS 计算结果的对比，壁面压力曲线与实验吻合的更好，综合误差不超过 3%。以上测试结果从另一个方面证明了程序的正确性。

5.3 与 CPU 程序相比的加速效果与强扩展性测试

表 5.1 OpenCFD-SC 和 OpenCFD-SCU 模拟相同问题所需时间的比较

Table 5.1 Comparison of times taken by OpenCFD-SC and OpenCFD-SCU when simulating the same problem

Program	Number of MPI processes	Time per step (s)
OpenCFD-SC	1280	10.66
	9120	1.93
	36 000	0.77
	72 576	0.54
	131 712	0.39
OpenCFD-SCU	128	0.5174
	256	0.2708
	512	0.1387
	1 024	0.0856
	2 048	0.0627

本文对 OpenCFD-SCU 和 OpenCFD-SC 相比的加速情况进行了测试，同时测试 OpenCFD-SCU 与 OpenCFD-SC 的强扩展性。强扩展性是指，在不增加问题规模的情况下，只增加进程数量，考虑程序的并行加速情况。测试时考虑的算例模型为升力体外形，几何形状为中国空气动力学研究开发中心设计的高超声速转捩研究飞行器（Hypersonic Transition Research Vehicle, HyTRV）^[127–129]，算例测试网格为 $4350 \times 2000 \times 160$ ，总网格量约为 14 亿，测试时 OpenCFD-SCU 与 OpenCFD-SC 采用相同的算法，无粘项先通过 Steger-Warming 流通矢量分裂，后利用 WENO-SYMBO 格式进行差分离散，粘性项采用 6 阶中心格式进行离散计算，时间推进方法为三步三阶 Runge-Kutta 方法。测试平台的 GPU 硬件参数已

已经在章节4.2中进行了介绍，测试平台的CPU采用海光系列X86架构CPU，基础频率为2.4GHz。表5.1展示了OpenCFD-SCU与OpenCFD-SC针对升力体开展直接数值模拟的计算花费时间，每步时间为程序运行100步的平均时间，从中可以对两款程序的计算速度进行比较。可以发现，在512块GPU运行程序时，OpenCFD-SCU的运行速度已经超过了OpenCFD-SC在13万个CPU核心上的运行速度。对于相同数量的MPI进程的情况下，OpenCFD-SCU的计算速度也可以相比OpenCFD-SC提升200倍以上。图5.5反应了两者的强扩展性和加速效果比较情况，以OpenCFD-SC最小的并行规模下的程序计算速度为基准速度，比较两者随MPI进程数增加的加速情况。可以发现，OpenCFD-SCU在MPI进程数较少的情况下，加速效果超过了OpenCFD-SC使用很多MPI进程。两者都在测试的最大进程数下保持了强可扩展性，OpenCFD-SCU在基准进程数增加16倍时，保持了60%的强扩展性并行效率。OpenCFD-SCU能够使本文的数值模拟能力提高了两个数量级。

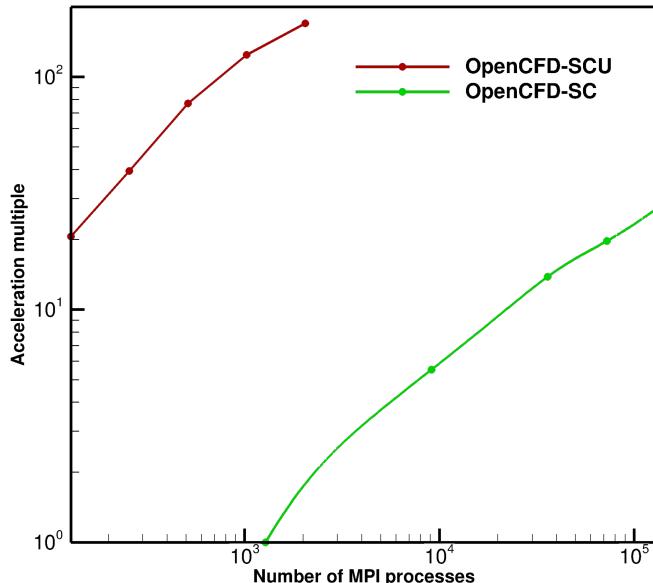


图5.5 OpenCFD-SC和OpenCFD-SCU在模拟相同问题时的加速情况比较（基本速度被选为OpenCFD-SC可以计算的最小MPI进程下的程序运行速度）

Figure 5.5 Comparison of calculation speed between OpenCFD-SC and OpenCFD-SCU when simulating the same problem. The base speed is selected as the speed of the minimum MPI process that OpenCFD-SC can calculate for this problem.

5.4 弱扩展性测试

本文测试了OpenCFD-SCU软件的弱扩展性，弱扩展性是指增加进程的同时，以相同的倍率增加问题规模，考虑程序的并行加速情况。即固定每个处理器上问题大小，看程序的计算时间随处理器数量变化。在测试时，算例模型为升力体外形，差分离散方法为无粘项先通过Steger-Warming流通矢量分裂，后利用WENO-SYMBO格式进行差分离散，粘性项采用6阶中心格式进行离散计算，时

表 5.2 双精度浮点运算数
Table 5.2 Number of double-precision floating-point operations

Kernel function	Kernel's floating-point operations	Calls of kernel	Grids calculated (including redundancy)	Total floating-point operations of kernel
WENO-SYMB0	216	90	25 088 000	4.877×10^{11}
Steger-Warming splitting	138	9	22 579 200	2.80×10^{10}
Sixth-order	9	72	21 952 000	1.42×10^{10}
Viscous term	128	9	22 579 200	2.60×10^{10}
Runge-Kutta (1st + 2nd + 3rd)	62	1	21 952 000	1.36×10^9
Total floating point operations		5.5726×10^{12}		
Time		792 ms		
Performance		705.4 GFLOPS		

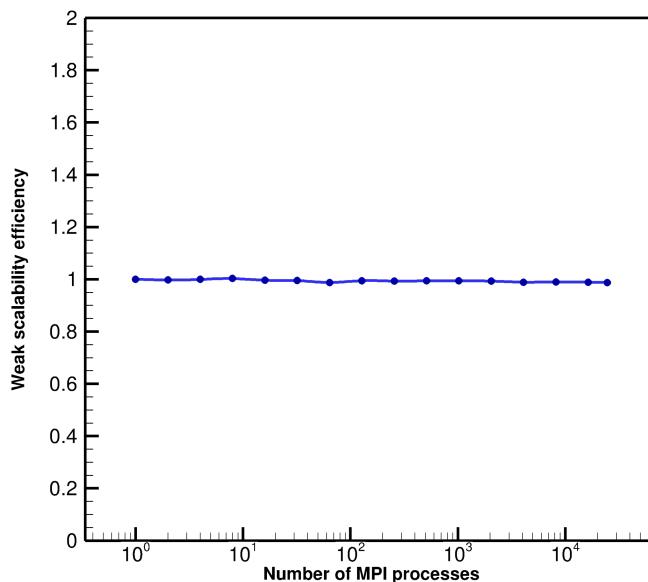


图 5.6 弱扩展性并行效率
Figure 5.6 Weak scaling parallel efficiency

间推进方法为三步三阶 Runge-Kutta 方法。本文让每个进程执行网格大小为 280^3 的计算任务，统计随 MPI 进程增加，程序每步计算时间的变化，每步计算时间为程序运行 100 步计算时间的平均值，MPI 进程数从 1 个进程测试到 24576 个进程，测试发现，程序并行弱扩展性良好，在不同 MPI 进程下，每步计算时间基本一致，在 MPI 进程为 1 时，程序每步计算时间为 0.78s，在 MPI 进程为 24576 时，程序每步计算时间为 0.791s，图 5.6 展示了弱扩展性并行效率随 MPI 进程变化的曲线，并行效率在 24576 进程时可以达到 98.7%，而此时对应的总计算网格点数超过 5000 亿个。弱扩展性的高并行效率从侧面证明了程序计算通讯重叠设计的成功。

5.5 性能评估

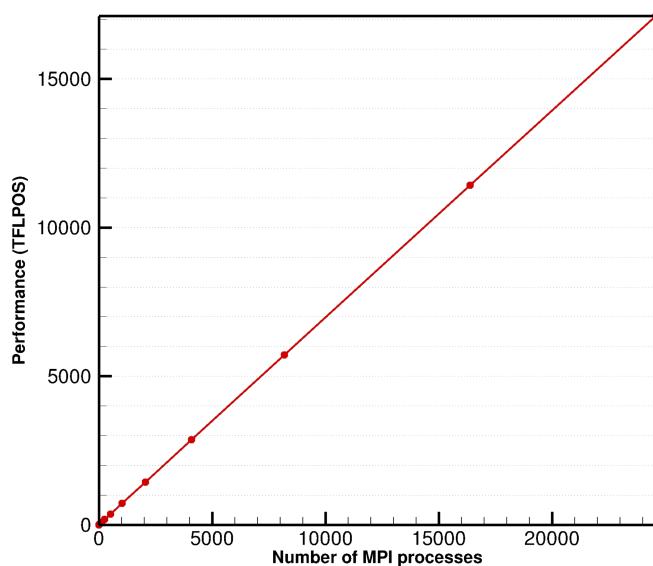


图 5.7 OpenCFD-SCU 双精度浮点运算性能随进程数的变化

Figure 5.7 Variation of double-precision floating-point operation performance of OpenCFD-SCU with number of processes

根据以上的测试结果，本文对程序的双精度浮点运算能力进行了估算。表 7 展示了手动计数的程序主要函数双精度浮点运算次数，并根据每步运行中函数的调用次数计算出其双精度浮点运算能力。手动计数只统计了上层编程语言可见的双精度浮点数的加法、减法、乘法和除法运算。寻址和其它计算不进行统计。对于一些使用到的库函数，如求根，在 GPU 没有专门处理这种库函数的硬件结构，因而此类库函数也是由软件实现的，实现过程的浮点运算次数不止一次，但由于其实现方式的不可见，在统计时也按作一次浮点运算计数。手动计数实际上统计的是程序主要函数浮点运算次数的最小值。以 WENO-SMYBO 格式为例，本文手动统计的双精度浮点运算次数为 262 次，而使用软件计数器 ROC PROF 来统计 GPU 上运行 WENO-SYMBO 内核函数时的 Value INSTS(用于衡量调用 GPU 算术单元的次数的参数) 显示结果为 388 次。但本文依然采用更为保守的手动计数获

得的浮点运算次数来估算程序性能。根据计算, OpenCFD-SCU 在单进程中(即单 GPU 下)可以达到 705.4GFLOPS 的计算能力。在 24567 进程时, OpenCFD-SCU 可以达到 17.1P 的性能, 占对应节点超算 GPU 性能双精度峰值的 11.4%。图 5.7 展示了 OpenCFD-SCU 双精度性能随 MPI 进程数的变化曲线。可以看到由于良好的并行弱扩展性, OpenCFD-SCU 的性能随 MPI 进程数量几乎呈线性增长。

5.6 跨平台测试

表 5.3 OpenCFD-SCU 在不同 GPU 上的运行时间

Table 5.3 Running times of OpenCFD-SCU on different GPUs

GPU	Time per step (s)
NVIDIA Tesla V100	4.702
NVIDIA RTX3090	4.038
NVIDIA Tesla A100	3.036
AMD Radeon M160	0.315

在章节 4 中本文已经介绍, OpenCFD-SCU 采用 CUDA 编程模型编写程序, 通过 HIPIFY 工具转码为 HIP 代码, 因而 OpenCFD-SCU 可以既可以在 NVIDIA 的 GPU 上运行又可以在 AMD 的 GPU 上运行。前文的测试都是基于 AMD Vega 20 架构 GPU 搭建的超级计算机开展的。本文也将程序部署到了 NVIDIA 平台的 GPU 上开展了测试, 由于资源受限, 只测试单进程(单 GPU)下 OpenCFD-SCU 的运行情况。表 6.2 展示了这些测试结果。测试算例为平板模型, 算法方面, 无粘项先进行 Steger-Warming 流通矢量分裂, 后利用 WENO-SYMBO 格式进行差分离散, 粘性项采用 6 阶中心格式进行离散计算, 时间推进方法为三步三阶 Runge-Kutta 方法。测试时的网格设置为 $200 \times 200 \times 200$, 每步计算时间为程序运行 100 步的平均时间。由于本文的程序针对 AMD Vega 20 架构做了大量针对性优化, 而 AMD Vega 20 架构与 NVIDIA Tesla A100 和 Gforce RTX 3090 采用的安培 (Ampere) 架构以及 NVIDIA Tesla V100 采用的伏特 (Volta) 架构有很大差异, 如 AMD Vega 20 架构的全局内存访问位宽为 512bit 而 NVIDIA GPU 一般为 128bit, AMD GPU 的 Warp 尺寸为 64 线程而 NVIDIA 系列 GPU 一般为 32 线程, 等, 因而 OpenCFD-SCU 在 AMD GPU 上表现出的性能远远高于在 NVIDIA GPU 上所表现的性能, 尽管某些 NVIDIA GPU 的理论性能高于 AMD 平台。这也说明了, 对 GPU 硬件的透彻了解和针对性优化对于让程序充分发挥其性能至关重要。

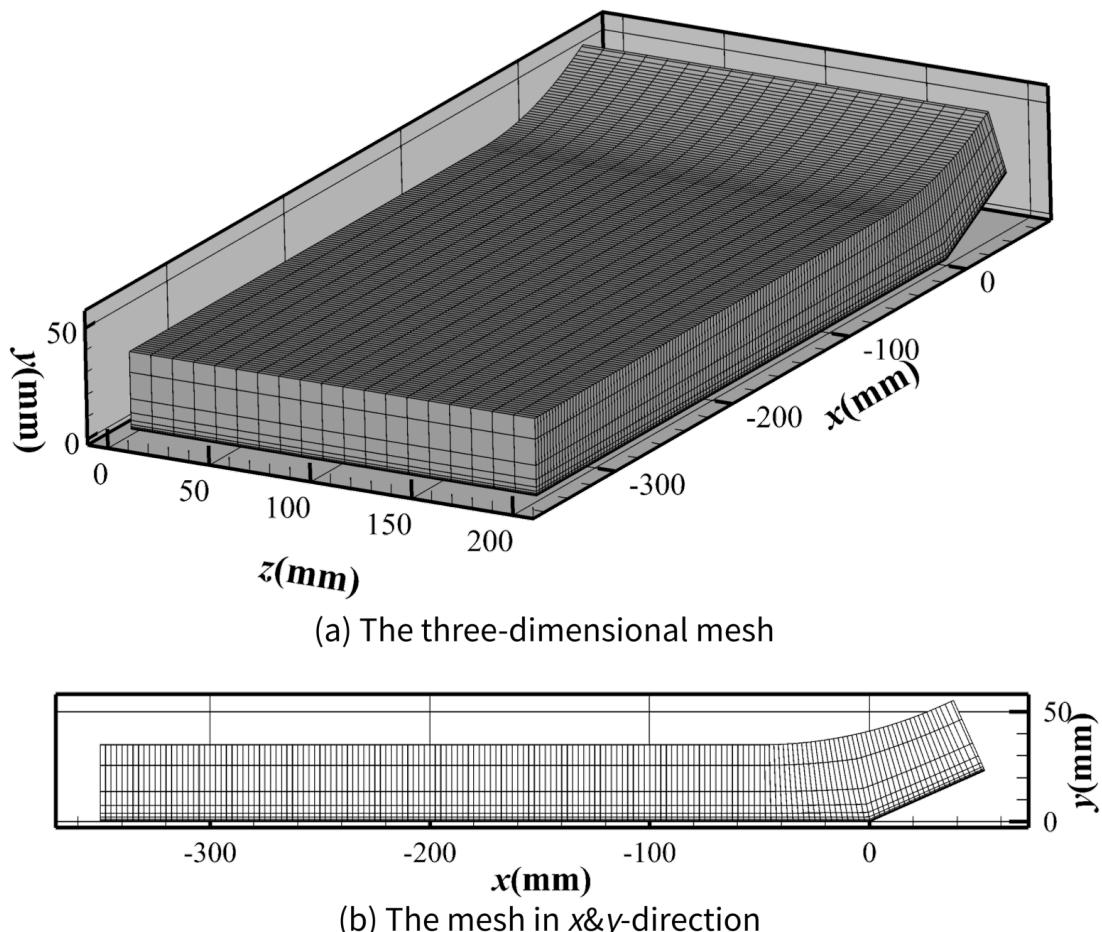
5.7 大规模湍流直接数值模拟测试

以上测试主要是针对程序的可扩展性、性能发挥、可移植性。最后为了测试 OpenCFD-SCU 应对真实的大规模湍流直接数值模拟的能力, 本文将小节 5.2.2 中

表 5.4 来流条件

Table 5.4 Conditions of incoming flow

Ma_{∞}	Re_{mm}	T_{∞}	T_w
2.9	5581.4	108.1 K	307 K

图 5.8 网格和计算域示意图: (a) 三维; (b) $x-y$ 截面Figure 5.8 Schematics of grid and computational domain: (a) three dimensions; (b) $x-y$ section

测试正确性时使用的算例扩大其展向计算域，使计算网格规模达到 312 亿（网格设置为 $16,250 \times 240 \times 8000$ ），测试 OpenCFD-SCU 实际开展超大规模直接数值模拟计算的能力。此算例的展向计算域为 200 mm，足以包涵 25 个以上的边界层厚度 δ ，计算域和网格如图 5.8。数值方法与正确性测试时算例的数值方法相同，无粘项先进行 Steger-Warming 流通矢量分裂，后利用 WENO-SYMBO 格式进行差分离散，粘性项采用 6 阶中心格式进行离散计算，时间推进方法为三步三阶 Runge-Kutta 方法。在流向出口和流动进入拐角前的平板段上边界，使用无反射边界条件，在流动进入拐角之后的上边界使用来流边界条件。转捩方式为壁面吹吸扰动诱发的转捩^[71]。扰动带布置位置为 $x = -350 \text{ mm}$ to -330 mm 。在建立流场的过程中，采用自适应滤波技术来保持计算的稳定性，在流场建立后将其关闭。为了消除滤波的影响，在关闭滤波后，保证流动再次流过整个计算域。计算参数设置可见表 6.1。以 $x = -35 \text{ mm}$ 处为参考位置，此处已经位于充分发展湍流区。参考位置流向、法向、展向的网格分辨率分别为 $\Delta x^+ = 1.2$, $\Delta y^+ = 0.41$, $\Delta z^+ = 1.2$ 。

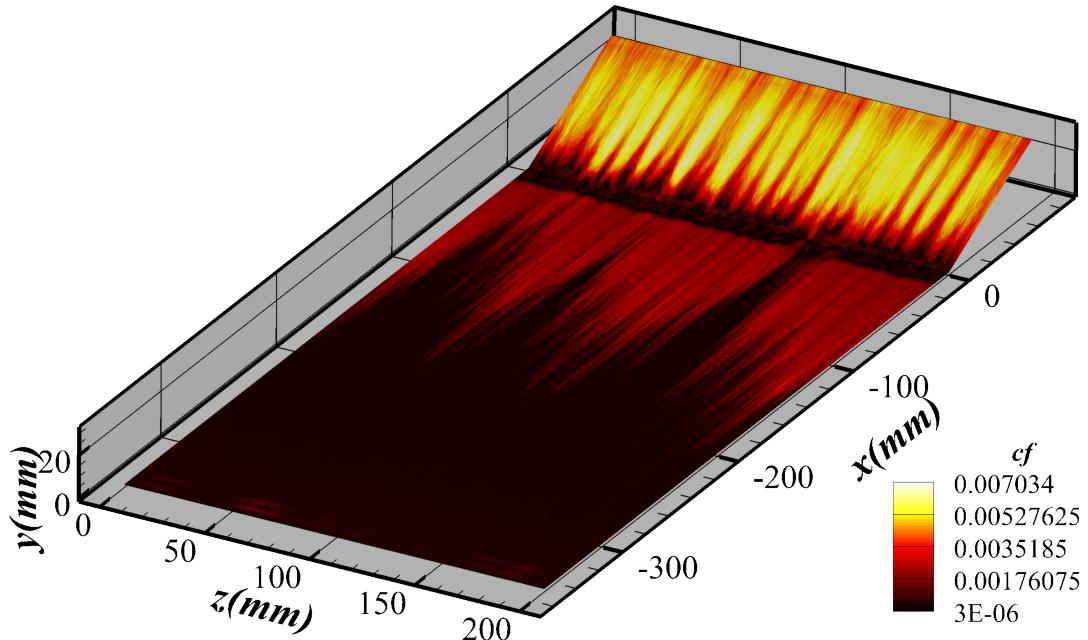


图 5.9 壁面摩擦阻力系数 C_f 分布情况

Figure 5.9 Distribution of skin friction coefficient C_f on wall

图 5.9 展示了瞬时壁面摩擦阻力系数 C_f 的分布，在上游平板区可以观察到沿展向分布的条纹结构，而在经过激波后的下游位置这些展向结构更加显著，这与 Taylor-Görtler 涡的形成密切相关。

图 5.10 展示了激波后 $x-z$ 截面上的速度分布情况，图 5.10(a) 显示的是时间平均后的流场速度，图 5.10(b) 展示了瞬时的流场速度分布。可以更明显地看到激波后沿展向分布的 Taylor-Görtler 涡结构。

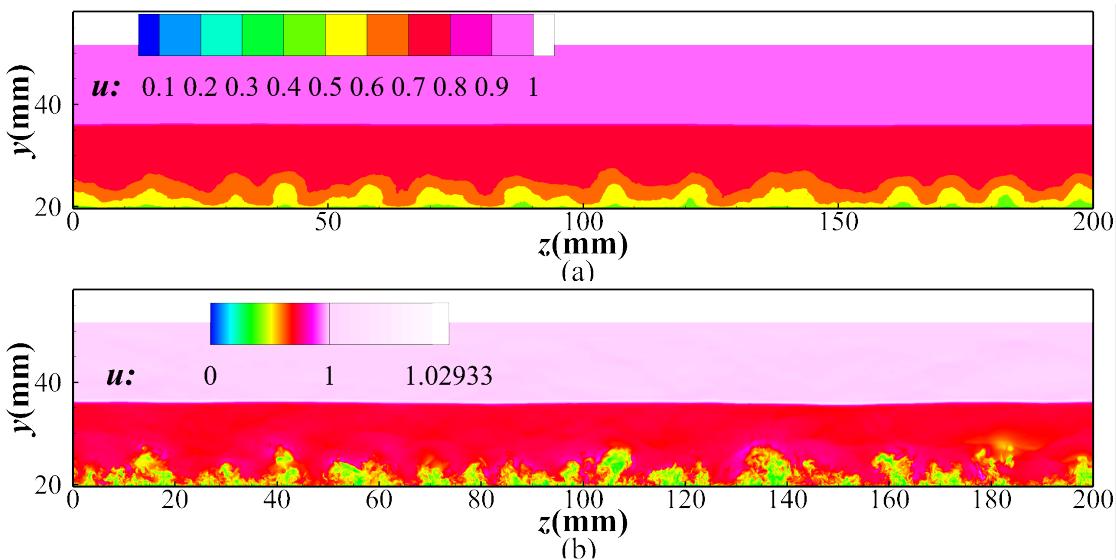


图 5.10 $X=44$ mm 处沿展向的速度分布: (A) 时间平均后; (B) 瞬时

Figure 5.10 Velocity distribution along spanwise direction at $x = 44$ mm: (a) after time averaging; (b) instantaneous.

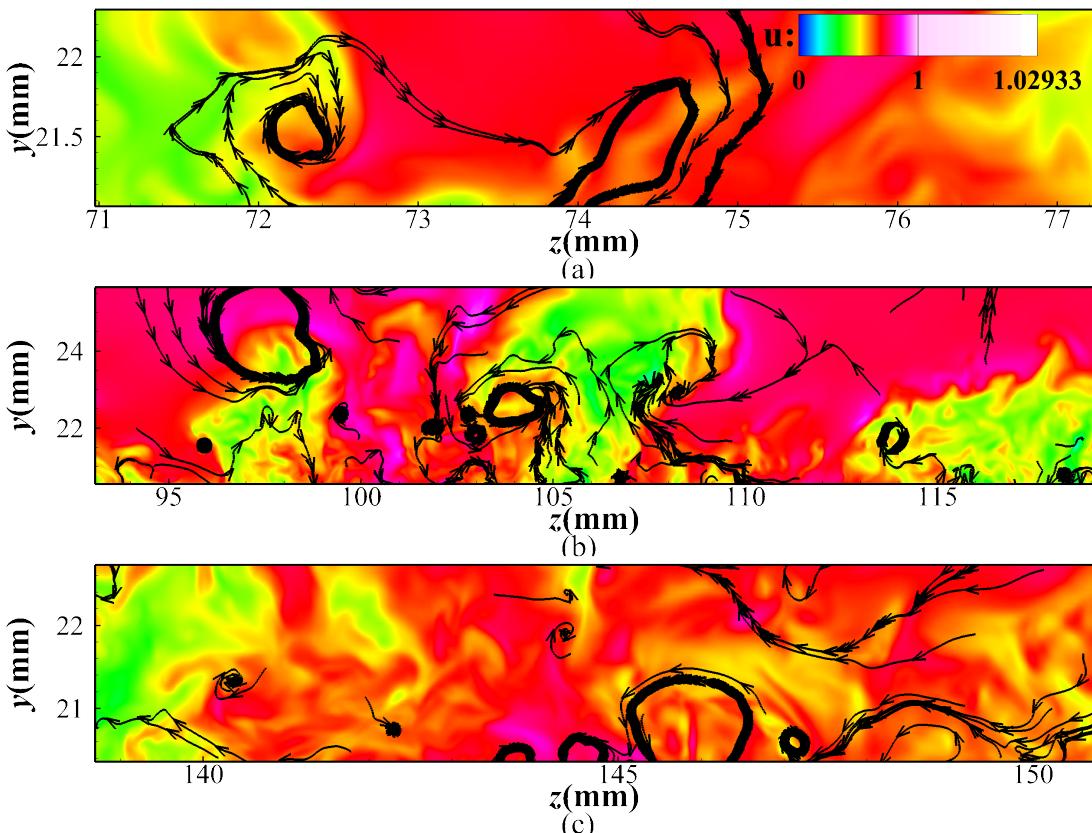


图 5.11 三个不同小区域的瞬时速度和流线

Figure 5.11 Instantaneous velocity and streamtraces in three different small areas

图 5.11展示了几个涡区的流线，在一个大尺度的 Taylor-Görtler 涡中，还分布着几个同向旋转或异向旋转的小尺度涡旋。

5.8 小结

本章介绍了 OpenCFD-SCU 的大规模测试结果，本文首先进行了正确性测试，从与 CPU 程序计算结果对比和与实验数据对比两方面证明了 OpenCFD-SCU 计算的正确性。与 CPU 程序相比，以 f16.9 格式保存文件时，两者结果完全相同（有符号浮点格式，存在 0.0 与 -0.0 的差别）。与实验壁压数据误差不超过 3%。与 CPU 程序比较了计算速度，相同 MPI 进程下可有 200 倍的速度提升。强扩展性在进程数增加 16 倍时，强扩展并行效率可以保持 60%。弱扩展性在 24576 个 GPU 上保持了 98.7% 的并行效率。在最大的测试规模下，对程序所发挥的性能进行了评估，发现程序性能可以达到 17.1PFLOPS，占对应节点 GPU 双精度峰值性能的 11.4%，还测试了程序跨平台部署的情况，分别在 NVIDIA Geforce RTX1660 Ti 显卡，NVIDIA Tesla V100 显卡以及 NVIDIA Tesla A100 显卡进行了部署，程序可跨平台地正确运行，但由于对 AMD Vega 20 架构针对性优化的原因，反而使其在 NVIDIA 平台性能发挥较差，这也证明了 GPU 程序根据硬件平台架构特点开展针对性优化的重要性。最后，对一个网格规模达到 312 亿的超大规模直接数值模拟算例开展了程序综合性测试，计算结果符合预期，证明了程序有对超大规模计算问题开展实际 DNS 计算的能力。

第6章 超大规模湍流直接数值模拟应用及数据库建设

6.1 引言

在之前的部分，本文介绍了 GPU 加速的高精度有限差分软件 OpenCFD-SCU，对其算法，软件结构，实现方式，异构加速优化手段进行了详细介绍。它能够带来比 CPU 版本同样算法代码两个数量级的性能提升。这让本文能够对一些更具挑战性的问题进行直接数值模拟研究。这一部分，本文介绍一个公开的湍流数据库，数据库中包含了几个使用 OpenCFD-SCU 软件模拟的典型可压缩流动直接数值模拟算例，所有算例都是更具真实性的壁湍流情况，比均匀各项同性湍流等理想情况具有更大的模拟难度。数据库算例简要描述可参见表 6.1。其中的算例包含了各种具有代表性的流动，从超声速到高超声速，从相对简单的几何外型到接近实际飞行器的几何构型。所有算例中，网格量最小的达到 45 亿，最大网格量高达 240 亿。使用 CPU 端程序开展如此大规模的模拟是有一定难度的，相比之下，通过使用 GPU 加速的软件 OpenCFD-SCU，能够利用较小的成本（包括时间成本和金钱成本）完成这些计算。表 6.2 显示了这些算例的模拟时间，从中可以看到完成其中任何一个算例所花费的计算时间都没有超过两周，使用的最大 GPU 数量为 1280 个。

表 6.1 数据库算例列表
Table 6.1 List of cases in the database

No.	Ma _∞	Geometry model	Grid number in three directions	Total grid number
Case 1	2.9	24° compression ramp	18 300 × 700 × 600	7.7 billion
Case 2	6	34° compression ramp	16 800 × 720 × 768	9.3 billion
Case 3	10	Zero pressure gradient flat-plate boundary layer	8800 × 640 × 800	4.5 billion
Case 4	6	Blunt cone	16 650 × 4500 × 320	24 billion
Case 5	6	Hypersonic Transition Research Vehicle.	8700 × 4000 × 320	11.1 billion

表 6.2 模拟成本和时间
Table 6.2 The simulation cost and time

No.	GPU numbers	Time-step	Total steps	Time per step(s)	Characteristic variables reconstruction used?	Total time(days)
Case 1	800	0.005	500000	0.89	Yes	5.1
Case 2	1200	0.005	1140000	0.92	Yes	12.1
Case 3	384	0.002	760000	1.20	No	10.5
Case 4	1200	0.005	320000	0.96	No	3.5
Case 5	1200	0.005	400000	0.71	Yes	3.2

在模拟此数据库中的所有算例时，针对无粘项的计算先使用 Steger-Warming 流通矢量分裂，然后使用 OpenCFD-SCU 中集成的混合格式进行差分离散（数

据库中的所有算例都是使用混合格式开展的计算)。它由七阶迎风格式、七阶 WENO 格式、五阶 WENO 格式组成。该格式具有较低的耗散而且能够很大增强计算的鲁棒性。在对数据库中的五个计算时, 当设置合理的时间步长并采用特征变量重构后, 能够保证在整个计算过程中不发散, 而且展示的五个算例所使用的 7 阶迎风格式占比都超过了 90%。线性的迎风格式比常用的激波捕捉格式(如 WENO 格式)具有更小的格式耗散, 更利于湍流模拟使用^[108]。粘性项的离散都使用了八阶中心差分格式。在算例 1、算例 2 和算例 5 的模拟时, 使用了特征变量重构来增强计算鲁棒性, 而其它算例只使用守恒变量重构。有关数值方法的更详细介绍, 请参阅本文的章节 2。

数据库中的算例与相似流动条件与工况下的直接数值模拟报道相比, 要么具有更大的计算域(算例 1 和算例 4)、要么具有更高的雷诺数(算例 3)或者更大的模拟难度(指更强的激波和更高的马赫数, 算例 2)。而且每个算例都具有很高的分辨率, 通过高分辨率的模拟, 还可以观察的一些新的现象, 如在算例 5 中, 本文能够清晰捕捉他人模拟时由于分辨率欠缺而难以捕捉到的附着线不稳定性(这在实验中已经被观察到)。以上算例能够充分证明 OpenCFD-SCU 的计算模拟能力, 而且算例数据本身极具价值, 本部分对这些算例进行了检验与比较, 并建立了一个公开的壁湍流数据库, 让其它研究者也能够下载这些数据从而进行进一步的流动分析, 或者参考算例数据计算时的控制文件, 自行设计新的算例来研究自己感兴趣的新问题。希望这些努力能对湍流研究、湍流模型开发以及人工智能的训练和验证提供帮助。数据获取地址与方式可在 OpenCFD-SCU 软件代码的开源网站<http://developer.hpccube.com/codes/danggl/opencfd-scu.git>上获得。

6.2 马赫 2.9 压缩折角流动的直接数值模拟

第一个算例是超声速压缩折角流动, 这一构型通常用以研究激波湍流边界层(shock wave/turbulent boundary layer interaction, STBLI)干扰问题^[130–132]。激波湍流边界层干扰在飞行器表面、发动机进气道处和发动机舱内部普遍存在, 压缩折角构型的实验研究过往也已有大量记录^[133–135]。然而, 实验很难提供详细的流动信息以帮助理解 STBLI 的内在机理。作为对实验的补充, 直接数值模拟可以提供流动全部的空间与时间信息, 能够帮助人们深入研究此类问题。过往大量研究者已经开展了超声速压缩折角流动的直接数值模拟^[136–138]。本文用 OpenCFD-SCU 也对一个自由流马赫数为 2.9 的 24 度坡面角的压缩斜坡问题开展了直接数值模拟, 该算例具有极高的网格分辨率, 而且与他人文献中的记录相比, 本文的算例具有更大的计算域, 压缩折角坡面的长度比以往报道长的多。

图 6.1展示了压缩折角构型直接数值模拟的计算域。该计算域具有 95mm 的极高的法向高度, 这是由于在计算时拐角之后的上边界采用了来流边界条件(平板段上边界采用无反射边界条件), 所以激波必须从尾部出口流出(出口为无反射边界条件), 为保证在较长坡面的情况下, 激波仍能从尾部流出, 所以设置了如此高的法向计算域。不过, 为节省计算资源, 计算域的法向实际上被分成了 2

表 6.3 马赫 2.9 压缩折角流计算参数与参考位置边界层相关参数

Table 6.3 The calculation parameters and the boundary layer parameters of compression ramp flow at Ma 2.9

Ma_{∞}	Re_{mm}	T_{∞}	T_w	δ	δ^*	θ
2.9	5581.4	108.1K	307K	6.22mm	2.25mm	0.44mm

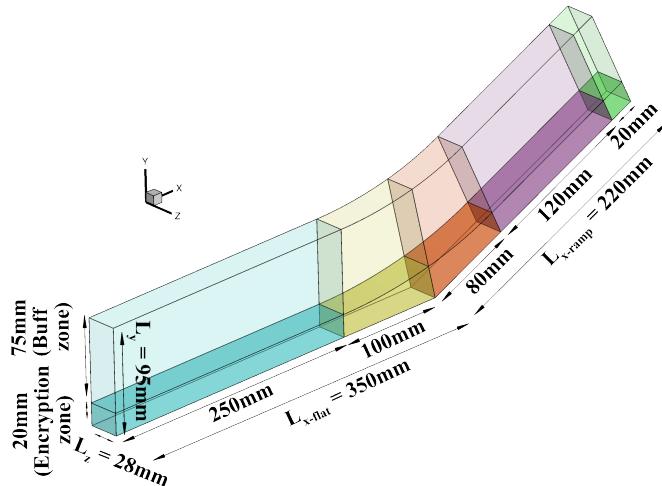


图 6.1 马赫 2.9 压缩折角流直接数值模拟计算域示意图

Figure 6.1 Schematic of computational domain of compression ramp flow DNS at Ma 2.9

部分，图 6.1 中颜色较浅的上半部区域采用了很多的网格，两个区域网格间距都由指数函数控制，网格沿法向被急速拉伸，确保大部分网格集中于壁面区域。流向在保证关键区（充分发展端、激波干扰段，流动再附段）网格分辨率足够的前提下，尽可能减少计算量，流向网格被设置为五段，蓝色平板段具有长度较长用以保证平板边界层具有足够的转捩距离，网格间距在此段从稀逐渐加密，黄色激波干扰段采用较密的均匀网格，以保证捕捉充分发展湍流与激波的质量，之后是橙色角区分离泡的位置，网格密度与激波干扰段相同，紫色再附段拥有最密的网格，这是由于激波过后在再附段的湍流脉动增强，而且由于流动在再附区与壁面会产生更强的摩擦作用导致更大的当地摩擦雷诺数，再附段也采用了均匀网格，再附段与橙色角区段通过几层网格进行线性过度，尾部绿色部分是网格拉伸段，采用指数函数对网格急速拉伸。无反射条件的出口位置网格都被指数拉伸到一个很大的网格间距，一是为了减少网格数量从而减少计算量，更关键的是需要通过更大的网格耗散保证流动在边界上不会产生反射（由于无反射条件为简单无反射）。

图 6.2展示了各部分的网格加密情况。无粘项计算采用混合格式和特征变量重构。粘性项采用八阶中心差分格式离散。在这一设置下，计算时不需要采用任何诸如滤波之类的增加稳定性手段。层流到湍流的转捩是通过改进了 Pirozzoli 等人^[71]提出的壁法向吹吸力扰动触发的，本文将扰动流向的周期增加到 10 个，并消除了影响流向和展向扰动的随机数，这样做的好处是扰动位置的剪切流更

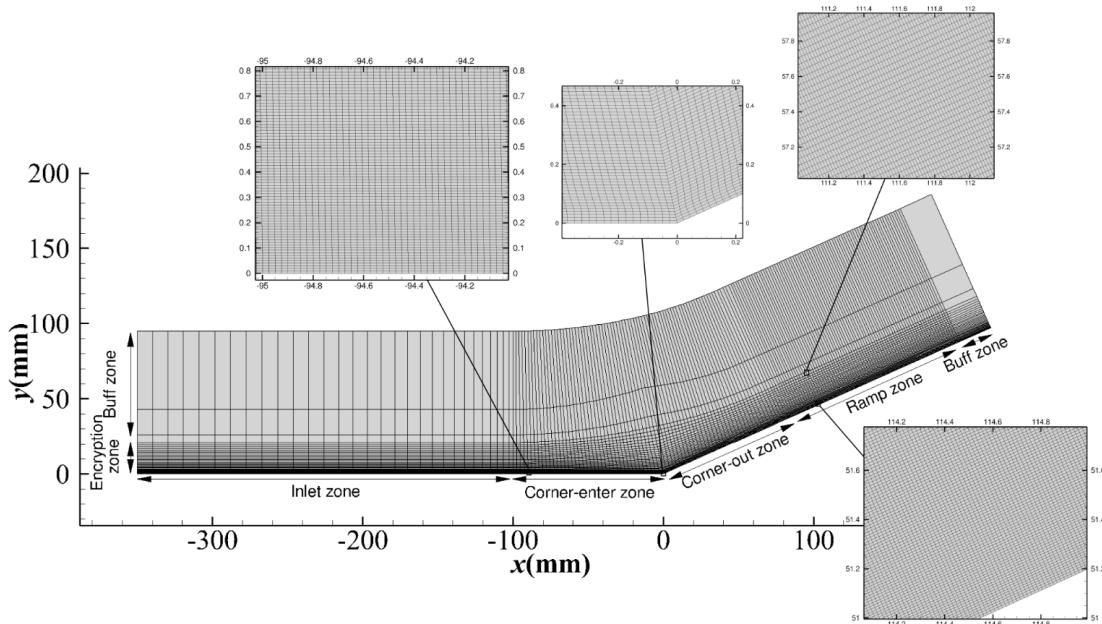


图 6.2 马赫 2.9 压缩折角流不同区域的网格加密情况

Figure 6.2 The grid densities are different in the different zones, about compression ramp flow at Ma 2.9

大，能够更容易引发转捩，而且消除了随机数，使任何情况下扰动形式都一样，避免了续算或者改变网格再次计算时扰动产生变化对流场的影响。这些扰动施加在平板区，范围从 $x = -330$ mm 到 -310 mm。计算参数与边界层参数展示在表 6.3 中，参考点的位置在 $x = -35$ mm 处。

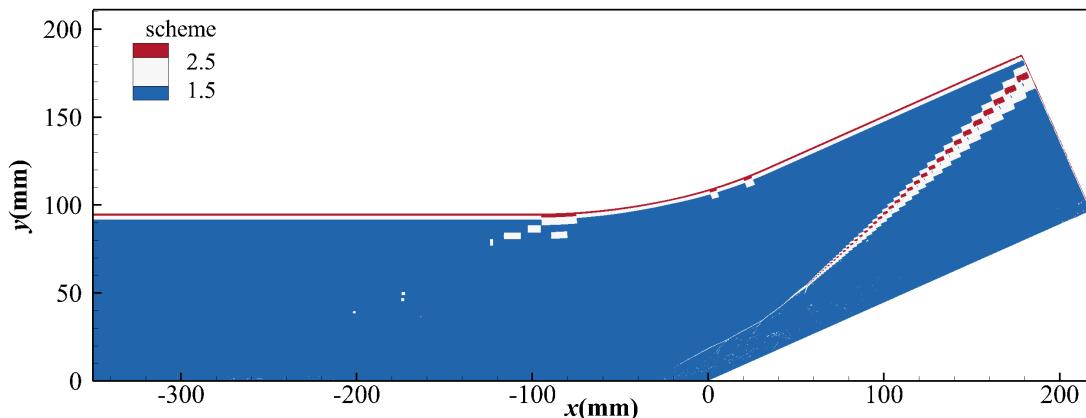


图 6.3 马赫 2.9 压缩折角流混合格式分布情况

Figure 6.3 Distribution of hybrid scheme, about compression ramp flow at Ma 2.9

计算时流向、法向、展向的网格设置分别为 $18\ 300 \times 700 \times 600$ ，计算网格总量大约 76.8 亿，图 6.3 展示了使用混合格式对此问题模拟时的格式分布情况，其中蓝色区域采用的是七阶迎风格式，白色区域采用的是七阶 WENO 格式，红色区域采用的是五阶 WENO 格式，它们的使用率占比分别为 99.27%、0.53% 和 0.2%。可以看到，绝大部分区域都采用线性的七阶迎风格式进行计算，只有在激

波位置和上边界网格极度拉伸的位置，格式跳转为了五阶 WENO 格式。

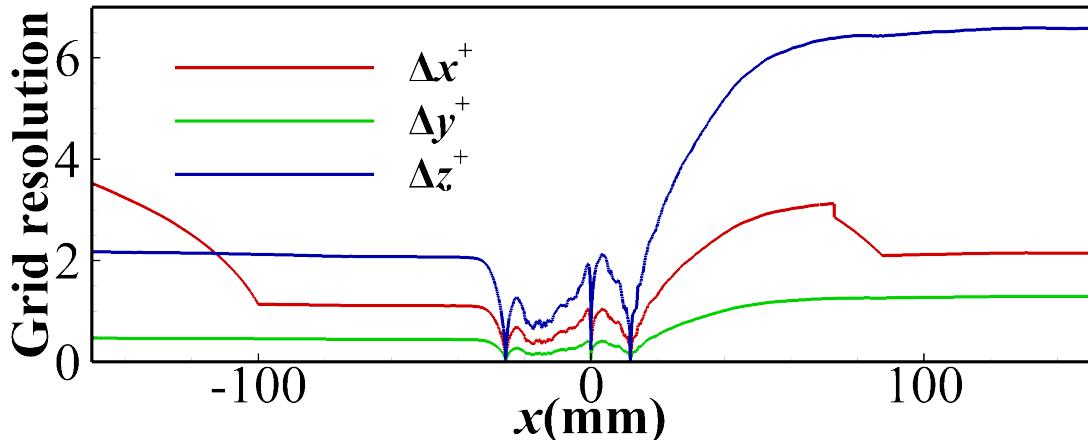


图 6.4 马赫 2.9 压缩折角流网格分辨率流向分布情况

Figure 6.4 Streamwise distribution of grid resolution, about compression ramp flow at Ma 2.9

图 6.4展示了网格分辨率沿流向的分布，从中可以看出由于网格是分段的，因此在不同段具有不同的网格分辨率。在湍流已经充分发展的平板段，网格延流向、法向、展向的分辨率分别达到 $\Delta x^+ = 1.1$, $\Delta y_w^+ = 0.43$, $\Delta z^+ = 2.07$ ，在坡面上的位置网格分辨率为 $\Delta x^+ = 2.14$, $\Delta y_w^+ = 1.28$, $\Delta z^+ = 6.57$ ，坡面上的分辨率相较于平板段有所降低，这是由于激波之后湍流增强，与壁面的剪切作用增大造成的。

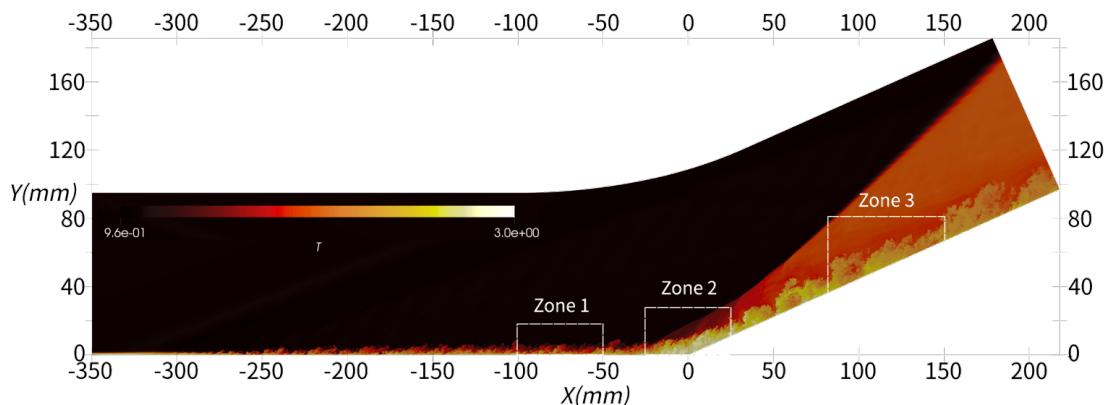


图 6.5 马赫 2.9 压缩折角流展向中截面的瞬时温度分布，使用参考温度进行无量纲化
Figure 6.5 Instantaneous temperature distribution in the middle section, nondimensionalized by the reference temperature, about compression ramp flow at Ma 2.9

图 6.5展示了展向中截面的瞬时温度分布，其中 Zone1 为平板段充分发展湍流区、Zone2 为折角位置流动分离区、Zone3 为斜坡湍流区，本文选择这三个区域在图 6.6进行了放大，可以借助温度场看到不同位置湍流脉动的情况，由于较高的分辨率，非常细小的脉动结构也清晰可见，与平板湍流区相比，激波后的斜坡湍流区明显具有更厚的边界层厚度，且受激波的影响流动温度较高。而在角区位

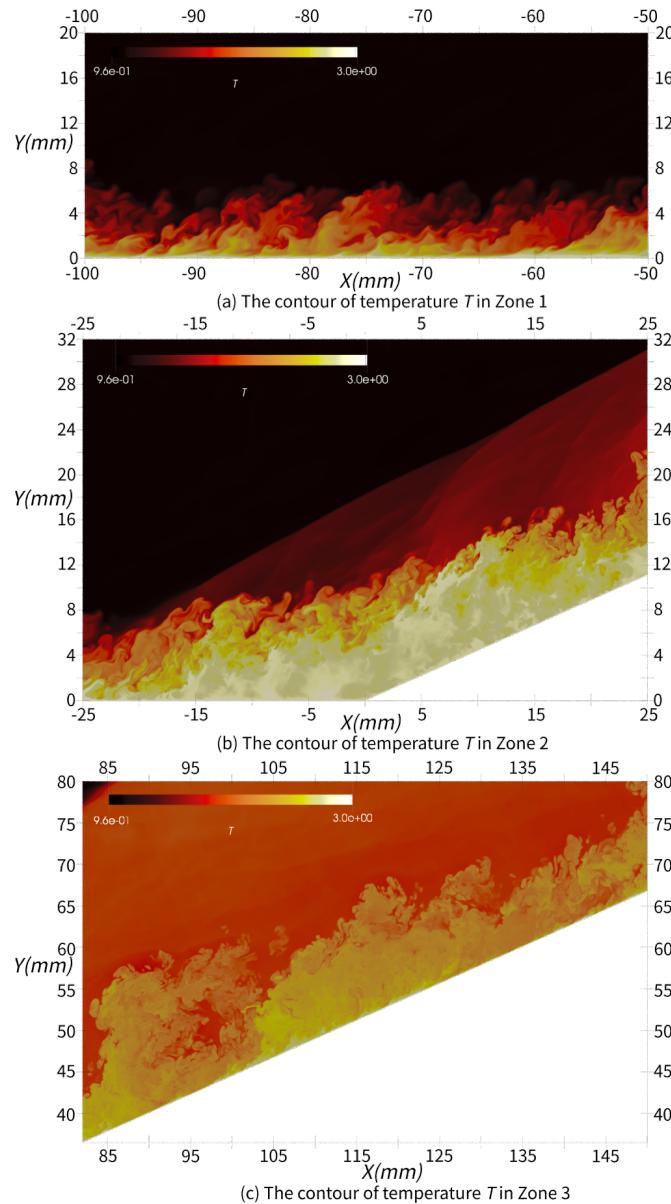


图 6.6 马赫 2.9 压缩折角流展向中截面的瞬时温度分布的放大图。(a) 平板区; (b) 角区; (c) 坡面区

Figure 6.6 Magnified views of the instantaneous temperature distribution, about compression ramp flow at Ma 2.9: (a) flat plate zone; (b) corner zone; (c) ramp zone

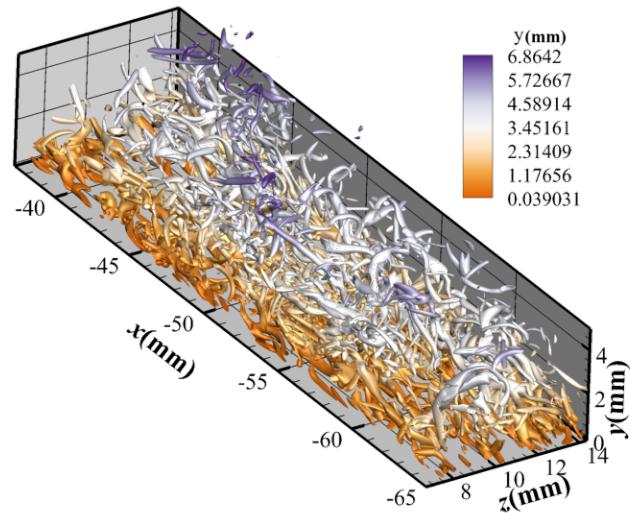


图 6.7 马赫 2.9 压缩折角流根据壁面距离进行染色的湍流相干结构

Figure 6.7 Turbulent coherent structures colored by distance from the wall, about compression ramp flow at Ma 2.9

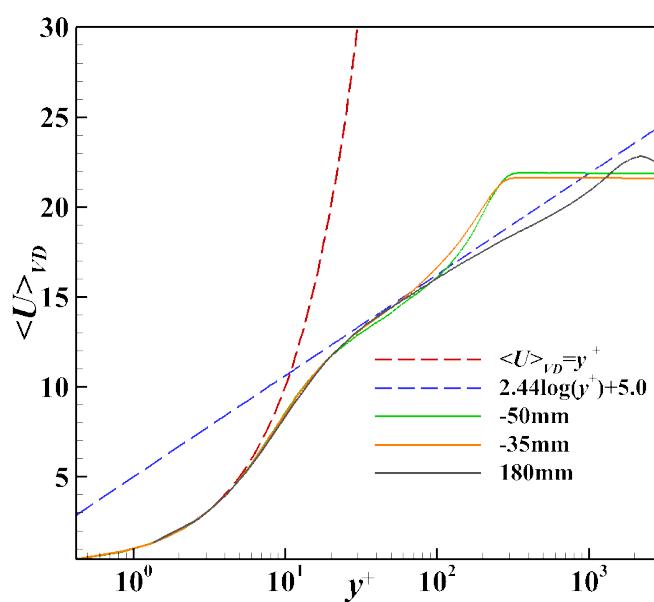


图 6.8 马赫 2.9 压缩折角流不同位置经 Van Driest 变换后的平均速度剖面

Figure 6.8 Van Driest transformed mean velocity profiles at different positions, about compression ramp flow at Ma 2.9

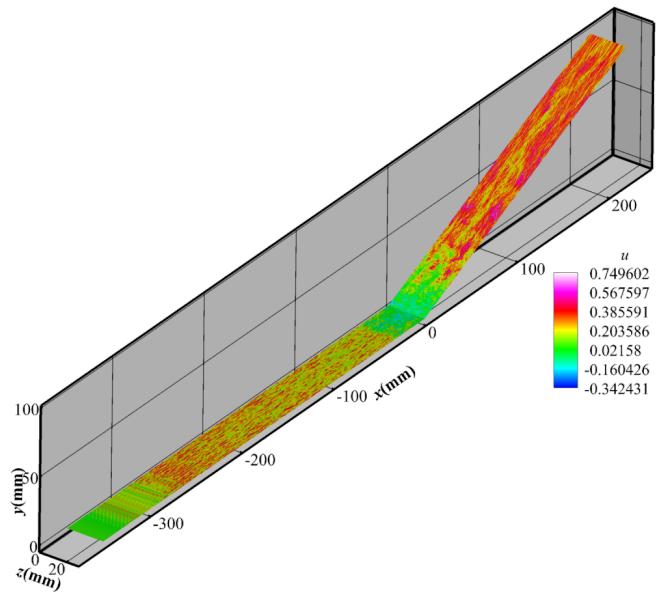


图 6.9 马赫 2.9 压缩折角流壁面的瞬时速度分布

Figure 6.9 Instantaneous temperature distribution in the wall, about compression ramp flow at Ma 2.9

置，由于分离泡造成的流动滞止作用，温度不容易通过物质交换传递，因而角区分离泡的温度普遍高于其它位置的湍流边界层。图 6.7 展示了湍流涡的相干结构，由 Q 判据的等值面得出，并根据壁面距离进行染色。由于计算网格较密，很难无失真地展示整体计算域的三维涡结构，因此此处只截取了 $30 \text{ mm} \times 5 \text{ mm} \times 6 \text{ mm}$ 很小一个区域的涡结构进行展示，但依然能够看出平板湍流区丰富的涡结构。

图 6.8 展示了 $x = -50 \text{ mm}, x = -35 \text{ mm}, x = 180 \text{ mm}$ 三个不同流向位置的经 Van-Driest 变换的平均速度剖面，可以看到在 $x = -35 \text{ mm}$ 位置平均速度剖面的斜率与截距已经与 Van-Driest 的理论吻合的很好， $x = -50 \text{ mm}$ 位置截距略低于 5.0 证明湍流还在发展过程中，而在坡面上 $x = 180 \text{ mm}$ 的位置，湍流的平均速度剖面也与 Van-Driest 的理论曲线有了很好的重合，但在 $\Delta y^+ > 100$ 距离壁面较远的位置，平均速度剖面的形状并不像平板区域一样进入了平台段，这可能是由于激波的作用。

图 6.9 展示了某一时刻壁面瞬时速度，可以看到扰动区由于吹吸作用的速度条带，流动通过扰动区之后，迅速发生转捩，体现为流向速度的条带状分布，转捩的位置沿展向也非常均匀。

图 6.10 展示了壁面摩擦系数 C_f 、热流密度 S_t 和壁面压力 P_w 。从图中可以看出，对于马赫数为 2.9，坡面角度为 24 度的压缩折角流动，壁面摩擦系数 C_f 和壁面压力 P_w 在经过激波后，不会瞬间达到峰值，而需要在坡面发展相当长的距离才能达到峰值，达到峰值的距离大概需要 15 个边界层厚度，而且它们在达到峰值后会保持在峰值一段距离，而壁面热流 S_t 则只需要 7 个边界层厚度左右就能达到峰值，与 C_f 和 P_w 不同， S_t 在达到峰值后会迅速下降，而不是长期保

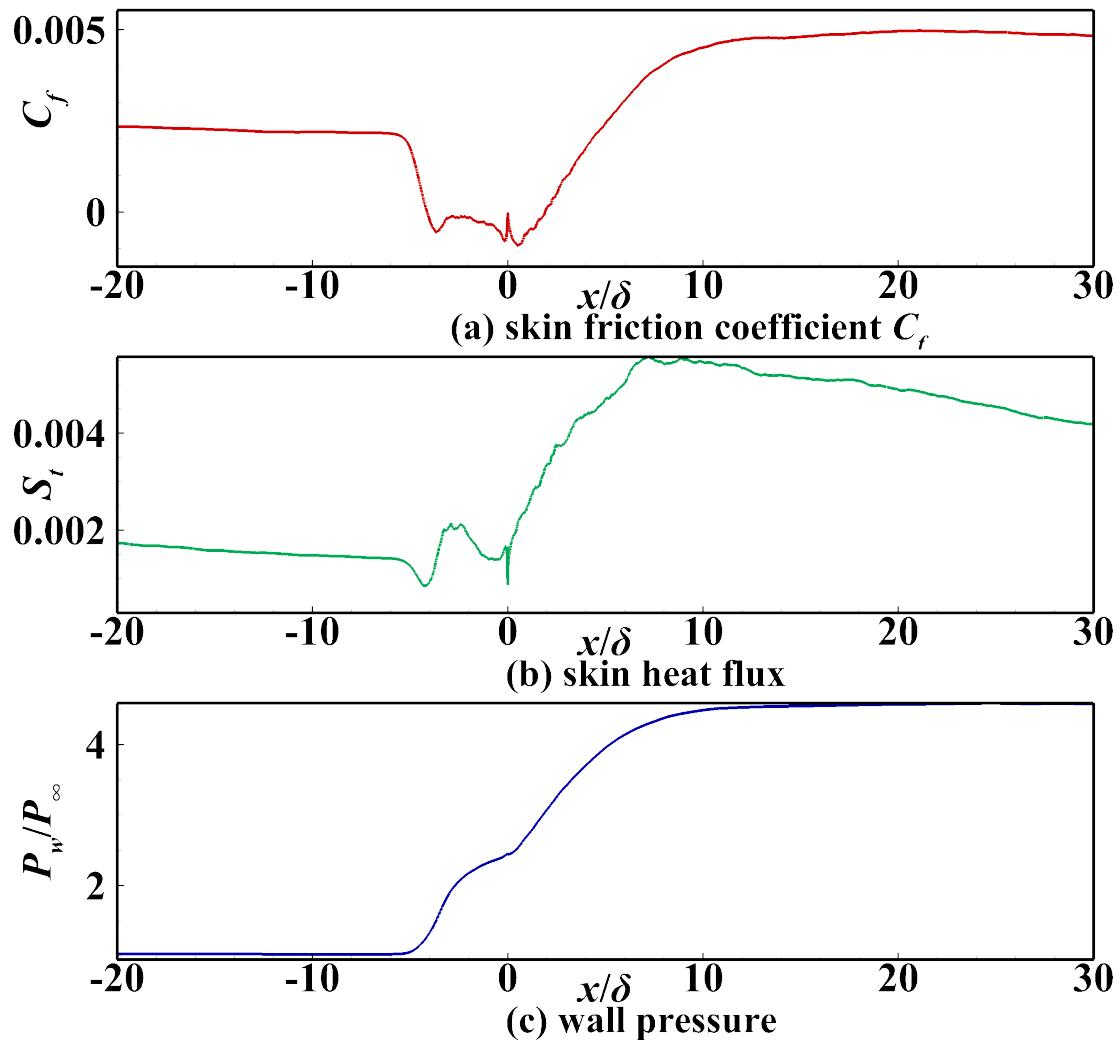


图 6.10 马赫 2.9 压缩折角流的流向分布的 (a) 摩擦阻力系数 (b) 壁面热流 (c) 壁面压力

Figure 6.10 Streamwise distributions of (a) skin friction coefficient C_f , (b) heat flux S_t , and (c) wall pressure P_w , about compression ramp flow at Ma 2.9

持这一值。在峰值位置的 St 大约为平板段的 4 倍, P_w 约为平板段的 4.5 倍, 而 C_f 大约为平板段的 2.5 倍。

6.3 马赫 6 压缩折角流动的直接数值模拟

与超声速和跨声速流动相比, 高超声速流动研究对于实验和数值模拟都是一个挑战。目前高超声速的压缩折角研究还较少, 采用直接数值模拟研究此问题也具有相当的难度, 在高超声速的情况下, 由于流体动能更高, 而且激波边界层干扰现象中存在各种复杂的流动结构, 强激波、湍流、流动分离、再附的相互作用很容易造成计算发散, 对数值方法要求较高, 但同时也是检验数值方法和程序可靠性的很好实验。目前的文献中还几乎没有对高超声速压缩折角开展很好的直接数值模拟的报道, 最近 Priebe 和 Martin^[139] 开展了一个来流马赫数为 7.2, 压缩折角角度为 8° 模型的直接数值模拟。但在他们的工况中, 由于折角的角度较小, 流动在角区并没有出现分离, 因而模拟难度较低。本文利用 OpenCFD-SCU 开展了一个来流马赫数为 6, 压缩折角角度达到 34° 的压缩折角流动直接数值模拟, 计算网格接近百亿, 采用混合格式和特征重构, 在如此高分辨率的情况下依然能够克服发散问题, 填补了高超声速大压缩角度压缩折角流缺乏高质量数值模拟数据的空白。

表 6.4 马赫 6 压缩折角流计算参数与参考位置边界层相关参数

Table 6.4 The calculation parameters and the boundary layer parameters of compression ramp flow at Ma 6

Ma_∞	Re_{mm}	T_∞	T_w	δ	δ^*	θ
6	10 000	79 K	293.88 K	10.17 mm	4.5 mm	0.49 mm

表 6.4展示了模拟时的计算参数与参考位置边界层相关参数。在高马赫数的情况下明显具有了比低马赫数更厚的边界层厚度。图 6.11展示对此构型展开直接数值模拟时的计算域尺寸, 平板段长 450 mm, 折角后的坡面长 170 mm, 展向计算域宽度 24 mm (满足 1000 以上粘性尺度的要求), 法向计算域高度 55 mm。在对马赫 6 压缩折角进行模拟时, 网格类似于上一小节 6.2 马赫 2.9 中的介绍, 但由于计算域法向并不如上一小节算例的法向计算域高, 且坡面长度也较短, 因而法向网格并不存在第二个拉伸区, 流向网格也不存在再附段, 而是与角区合为一段。流向、法向、展向的网格设置分别为 $16\,800 \times 720 \times 768$, 总网格量达到 93 亿。图 6.12展示了网格分辨率沿流向的分布情况, 在 $x = -50$ mm 的位置, 网格过渡为加密的均匀网格, 均匀加密区沿流向网格间距为 0.014 mm, 壁面法向第一层网格的高度为 0.01 mm。扰动添加方式也与马赫 2.9 类似, 扰动添加位置位于平板段 $x = -430$ mm 到 -400 mm 处。

在针对马赫 6, 34° 度折角角度的压缩折角流进行模拟时, 由于对数值算法鲁棒性要求很高, 因而在对此算例模拟时调整了混合格式的阀值, 阀值设置为 0.005 和 0.15, 这样相当于增加了高耗散格式的占比, 从而增强鲁棒性。但在这

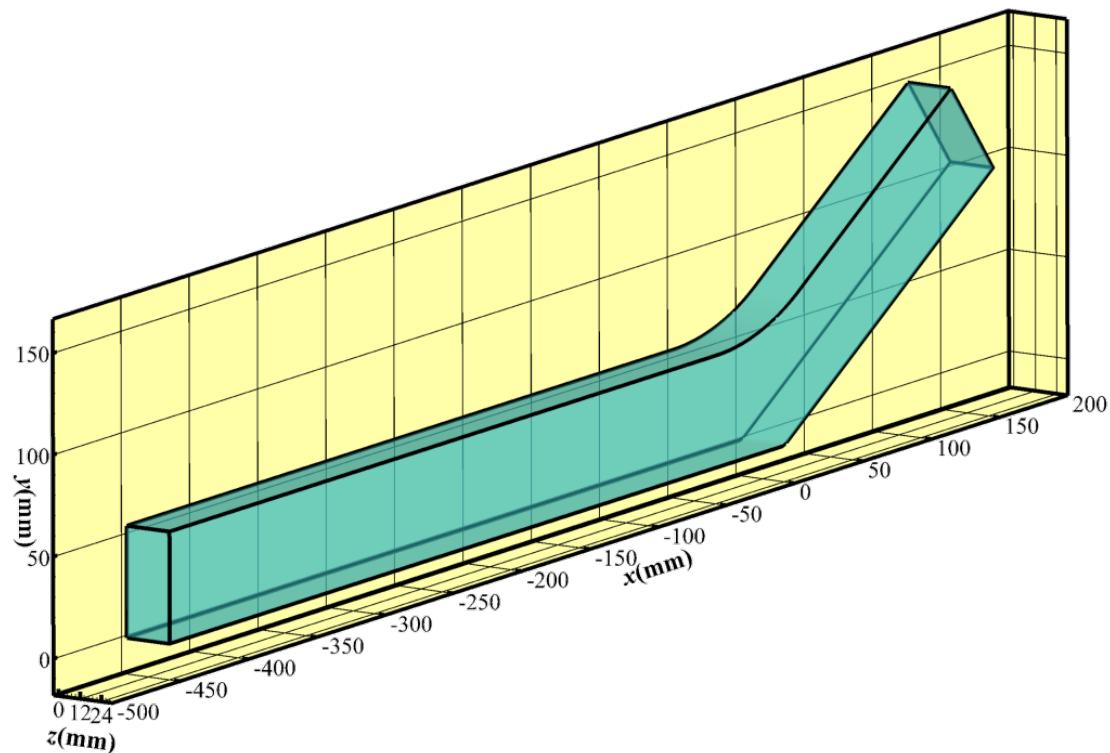


图 6.11 马赫 6 压缩折角流计算域

Figure 6.11 Schematic of computational domain, about compression ramp flow at Ma 6.

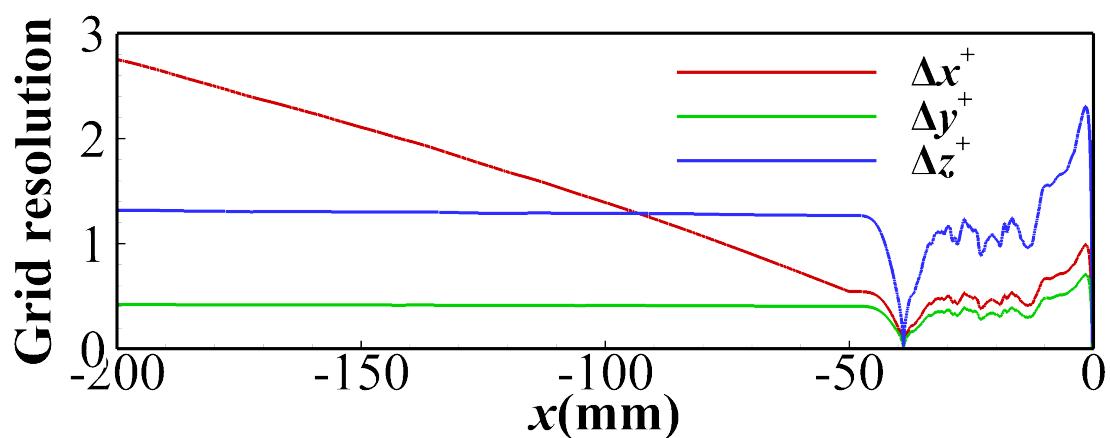


图 6.12 马赫 6 压缩折角流网格分辨率的流向分布

Figure 6.12 Streamwise distribution of grid resolution, about compression ramp flow at Ma 6

种情况下，混合格式中低耗散的线性七阶迎风格式占比依然能够超过 90%，七阶迎风格式、七阶 WENO 格式和五阶 WENO 格式的占比分别为 93%、6.8% 和 0.2%。混合格式位置分布情况如图 6.13。

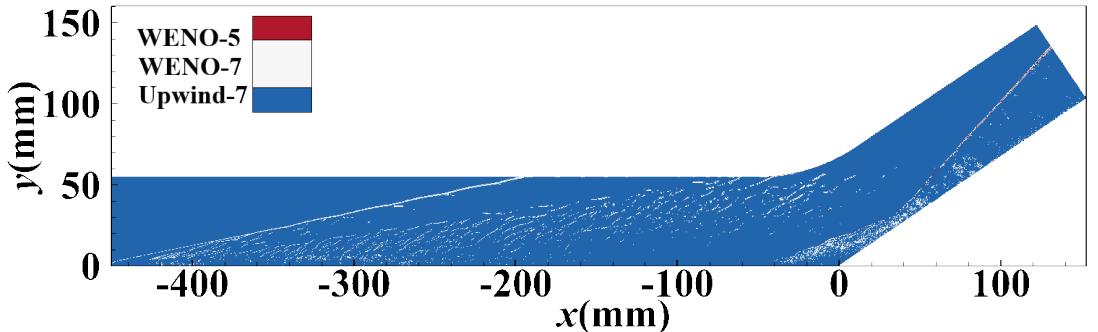


图 6.13 马赫 6 压缩折角流混合格式分布情况

Figure 6.13 Distribution of hybrid scheme, about compression ramp flow at Ma 6

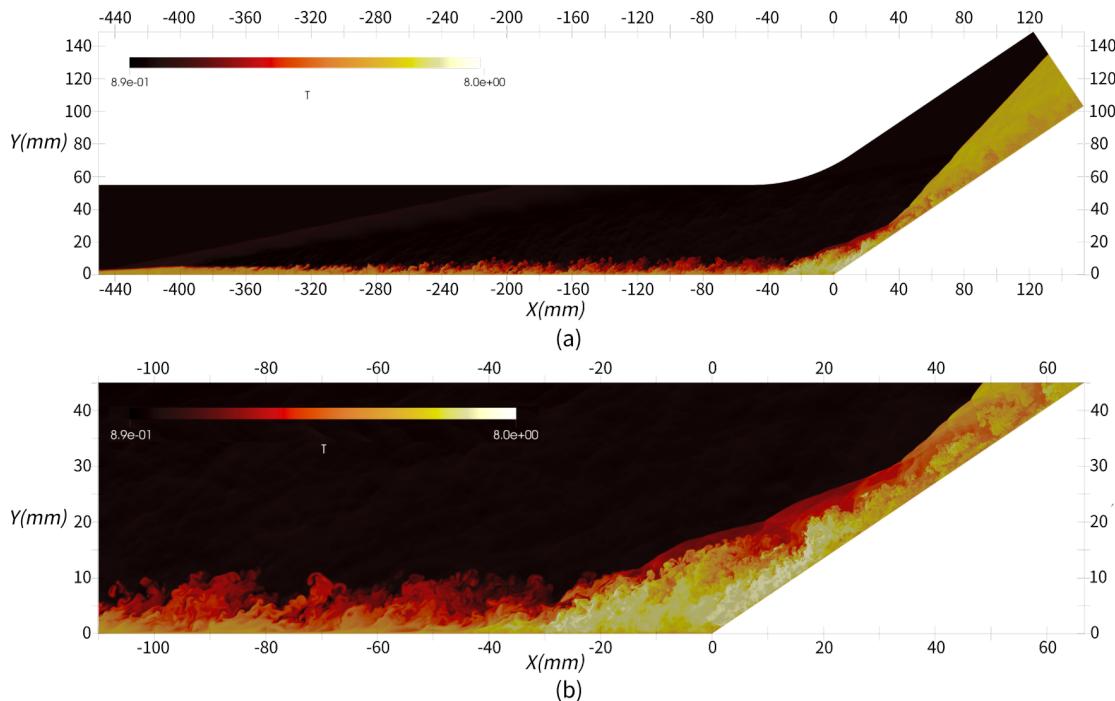


图 6.14 马赫 6 压缩折角流展向中截面的瞬时温度分布, 使用参考温度进行无量纲化。(a) 全局分布情况。(b) 折角区局部放大情况

Figure 6.14 Instantaneous temperature distribution in the middle section, nondimensionalized by the reference temperature, about compression ramp flow at Ma 6: (a) overall distribution; (b) magnified view of corner zone

图 6.14展示了展向中截面的瞬时温度分布情况，图 6.14(a) 展示了总体的瞬时温度云图。6.14(b) 对角区进行了放大，以便更清楚的观察。可以看到，由于添加扰动的影响，在平板段 $x = -350$ mm 左右位置就出现了类似湍流的结构，在转捩开始后，有 350 mm 的距离流动才接触到拐角，这让流动能够有足够的发展距离在进入角区前发展为充分发展湍流。从图中也可以观察到，与马赫数为

2.9 的压缩折角流类似，由于分离泡区域流动的滞止作用，使得当地温度很高。图 6.15 也展示了利用 Q 判据得到的局部湍流相干结构，根据与壁面距离进行染色，可以看到，由于选取的展示位置靠近拐角区，当地涡结构呈现一种上抛的趋势。

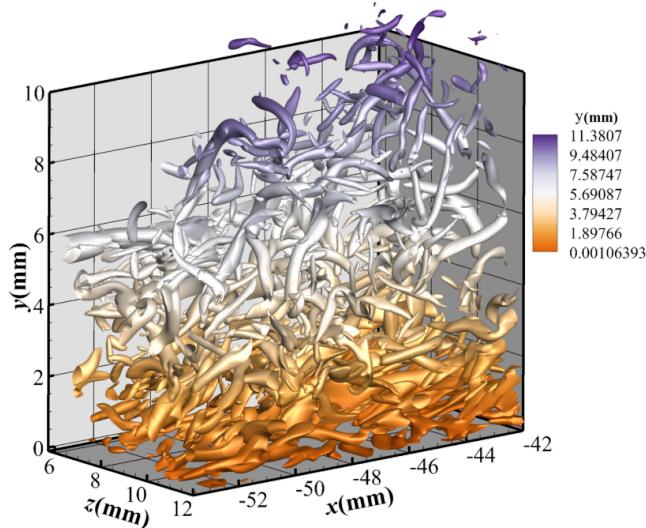


图 6.15 马赫 6 压缩折角流利用根据壁面距离染色的湍流涡相干结构

Figure 6.15 Turbulent coherent vortical structures colored by distance from wall, about compression ramp flow at Ma 6

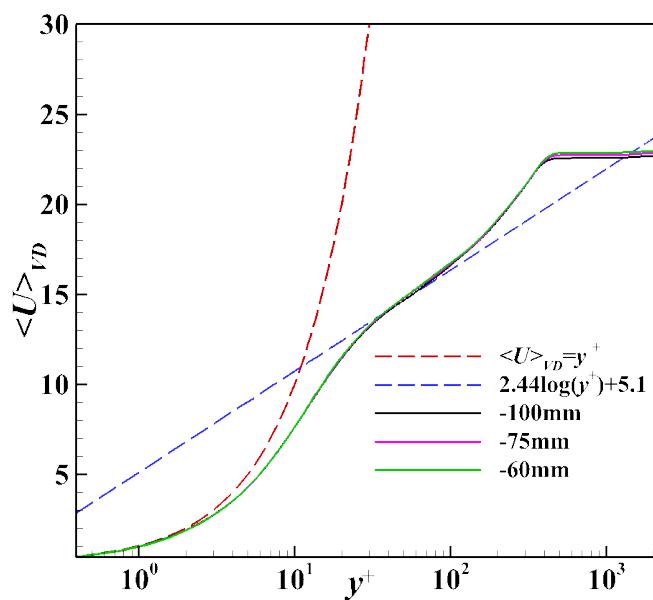


图 6.16 马赫 6 压缩折角流不同位置经 Van Driest 变换后的平均速度剖面

Figure 6.16 Van Driest transformed mean velocity profiles at different positions, about compression ramp flow at Ma 6

图 6.16 绘制了 $x = -100 \text{ mm}$ 、 -75 mm 和 -60 mm 三个不同位置的经过 Van Driest 变换的平均速度剖面图。可以看到三个位置的平均速度剖面几乎重合，但

也可以看出位于下游的位置平均速度剖面的平台段越高，这是由于边界层不断发展造成的，三个位置的剖面都与 Van Driest 的理论吻合的很好。但与马赫 2.9 不同的是，在壁面附近壁面率 $\langle U \rangle_{VD} = y^+$ 的关系满足的并不如前者好。

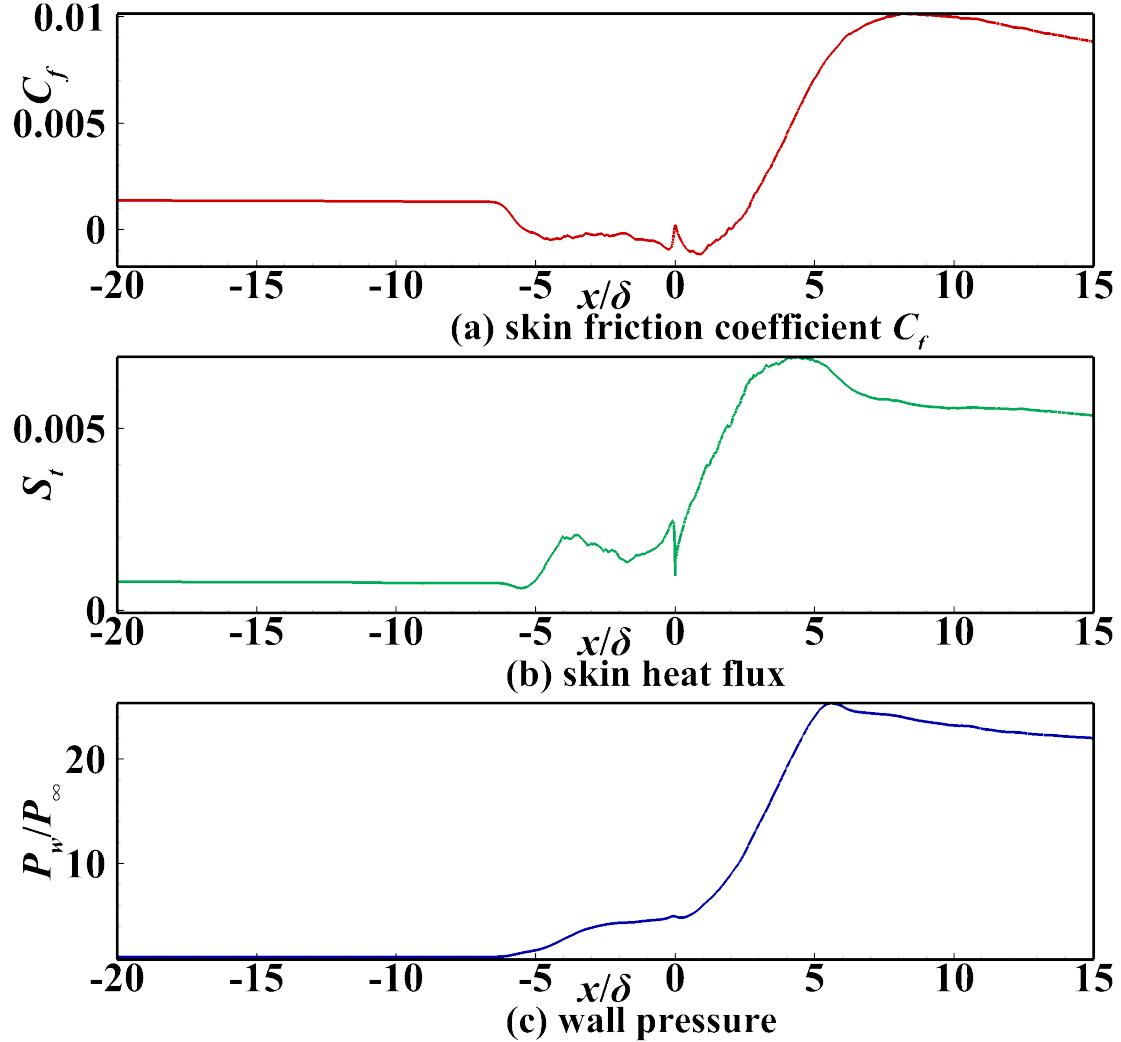


图 6.17 马赫 6 压缩折角流的流向分布的 (a) 摩擦阻力系数 (b) 壁面热流 (c) 壁面压力

Figure 6.17 Streamwise distributions of (a) skin friction coefficient C_f , (b) heat flux S_t , and (c) wall pressure P_w , about compression ramp flow at Ma 6

图 6.17 展示了马赫 6 压缩折角流的壁面摩擦系数 C_f 、热流密度 S_t 和壁面压力 P_w 分布情况。与马赫 2.9 坡面角度为 24° 的情况相比，马赫 6 坡面角度为 34° 的情况呈现出了较大的不同，在高超声速情况下， C_f 和 P_w 不需要那么长的距离就能较快达到峰值，达到峰值之后也不能长距离地维持此峰值，而是立即开始下降， S_t 表现的规律与马赫 2.9 也有不同之处， S_t 达到峰值之后先以一个较快的速度下降之后又以一个较慢的速度继续下降，在高超声速情况下 C_f ， S_t 和 P_w 的峰值也更高，与平板段相比 C_f 在坡面段的峰值高了约 8 倍， S_t 的峰值高了约 10 倍，而壁压 P_w 的峰值可以高出 25 倍。产生这些不同的原因是在高马赫数、更大斜坡角度情况下，流动会对壁面产生一定的冲击作用。图 6.18 展示了马赫 2.9， 24° 的斜坡角度和马赫 6， 34° 斜坡角度的经时间平均后的流向平均速度

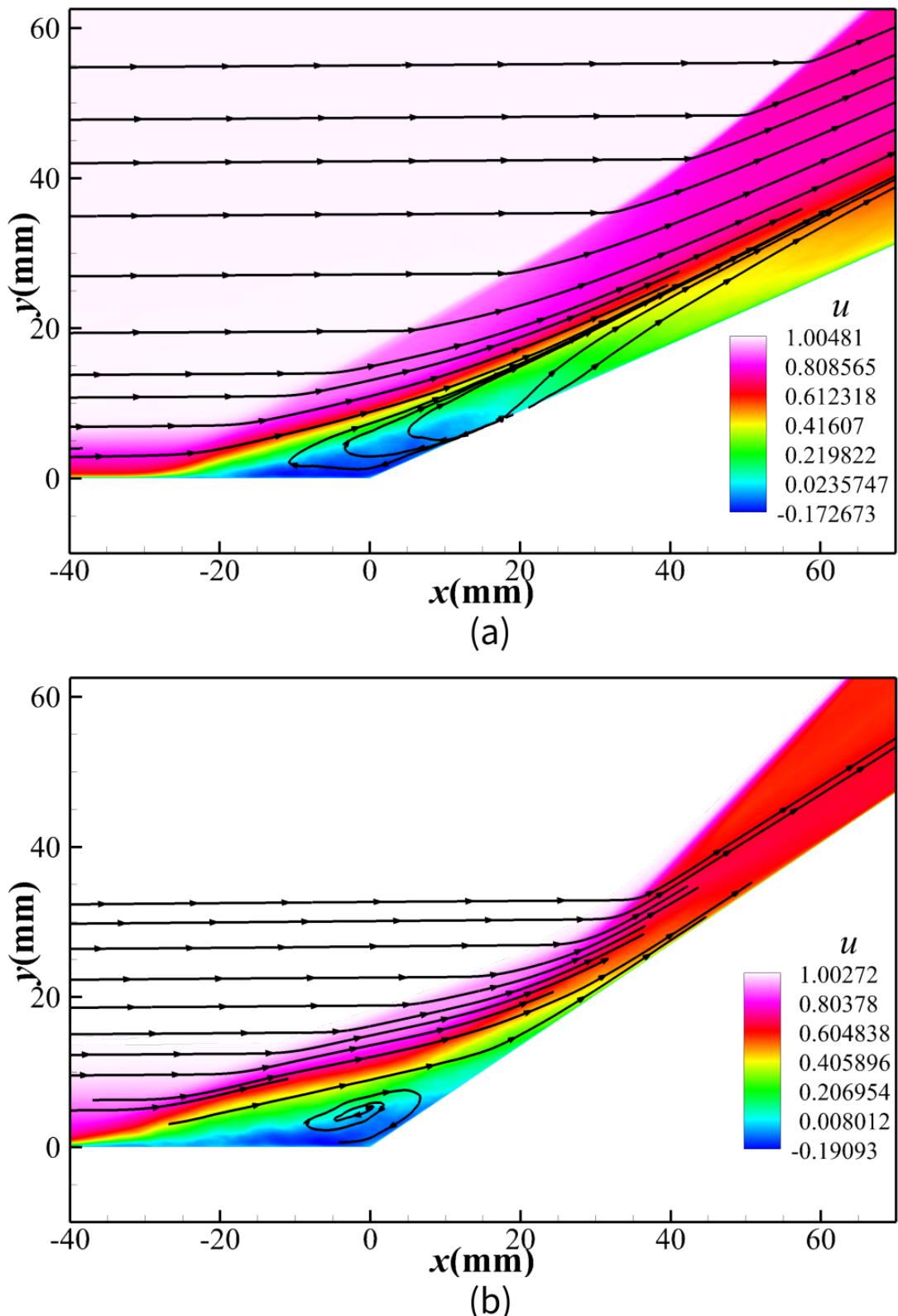


图 6.18 不同情况经时间平均后的流向平均速度和流线等值线: (a) 马赫 2.9 压缩折角 (参见小节 6.2) (b) 马赫 6 压缩折角

Figure 6.18 Contours of time-averaged streamwise velocity and streamtraces: (a) Mach 2.9 compression ramp (see Subsection 6.2; (b) Mach 6 compression ramp

和流线等值线。在马赫 6, 34° 斜坡角度的情况下, 可以看到激波的形状呈明显的弧形, 激波角的位置并不在分离区之前, 而更靠近分离区之后, 流动会在坡面上产生更大的折转角, 经过激波后流动方向并不立即与壁面平行, 而会产生对坡面的冲击作用。而 S_t 和 P_w 的峰值也正好出现在流动冲击点附近, 大约在距离折角 40 mm 到 50 mm 的位置, 在经过冲击点之后 S_t 和 P_w 都开始减少, 但由于冲击点位置的流向速度并不是最大值, 因而此处并不是 C_f 的峰值, 在经过冲击之后的一段距离, 流向速度增加, 导致流动与壁面剪切继续增强, C_f 峰值出现在下游。

6.4 马赫 10 平板湍流边界层的直接数值模拟

数据库中的第三个算例是马赫数达到 10, 摩擦雷诺数 $Re_\tau \approx 1550$ 的冷壁平板湍流边界层空间模式的直接数值模拟。该算例的网格数量达到了 40 亿。平板由于几何外型简单, 对其开展直接数值模拟的历史较早。1988 年, Spalart^[63,140]最早开展了对平板湍流边界层的直接数值模拟研究。1994 年, Adam 首次对可压缩的平板湍流边界层开展了时间模式的直接数值模拟研究 (TDNS)^[64], Duan 和 Matrin 利用 TDNS 对于高超声速的平板湍流边界层做过系统地参数研究, 包括壁温效应的影响, 马赫数效应的影响等。然而, 利用时间模式直接数值模拟需要添加能够把湍流时间增长与边界层空间增长联系起来的假设, 而且时间模式无法获得从层流到转捩过程的流场信息, 因此具有一定的弊端。空间模式的直接数值模拟 (SDNS) 能够更真实地模拟边界层演化过程。1995 年, Rai 等人^[65]最早对来流马赫数为 2.25 的可压缩平板边界层开展了 SDNS, 随后 Pirozzoli 等人^[71]使用高精度的数值格式和更密集的计算网格复现了这项工作。对于高超声速的情况, 李新亮等学者^[69]最早对来流马赫数为 6 的高超声速平板开展了 SDNS。研究发现高超声速平板的涡结构多为准流向涡, 而非超声速时的发卡涡。近几年, Zhang 等人^[141]利用 SDNS 建立了一个平板湍流边界层数据库, 包含了来流马赫数从 2.5 到 14, 壁温与恢复温度比在 0.18 到 1.0 之间的多种情况。然而对于高雷诺数的高超声速冷壁平板湍流边界层的 SDNS 还较少。在 2022 年最近的一篇文献中, Duan 和 Choudhari 对来流马赫数达到 10.9 的冷壁平板开展了 SDNS, 他们的 $Re_\tau \approx 1300$, 在同类型问题的模拟中属于较高水平, 在本文的工作中, 进一步扩展了高超声速平板边界层的雷诺数范围, $Re_\tau \approx 1550$ 。

表 6.5 马赫 10 平板边界层的流计算参数与参考位置边界层相关参数

Table 6.5 The calculation parameters and the boundary layer parameters of flat plate boundary layer flow at Ma 10

Ma_∞	Re_{mm}	T_∞	T_w	δ	δ^*	θ
10	100 000	79 K	293.88 K	6.33 mm	3.91 mm	0.248 mm

表 6.4 显示了本算例的具体参数设置与边界层信息。来流马赫数为 10, 单位毫米雷诺数达到了 100000。直接数值模拟的计算域如图 6.19 所示, 计算域的长

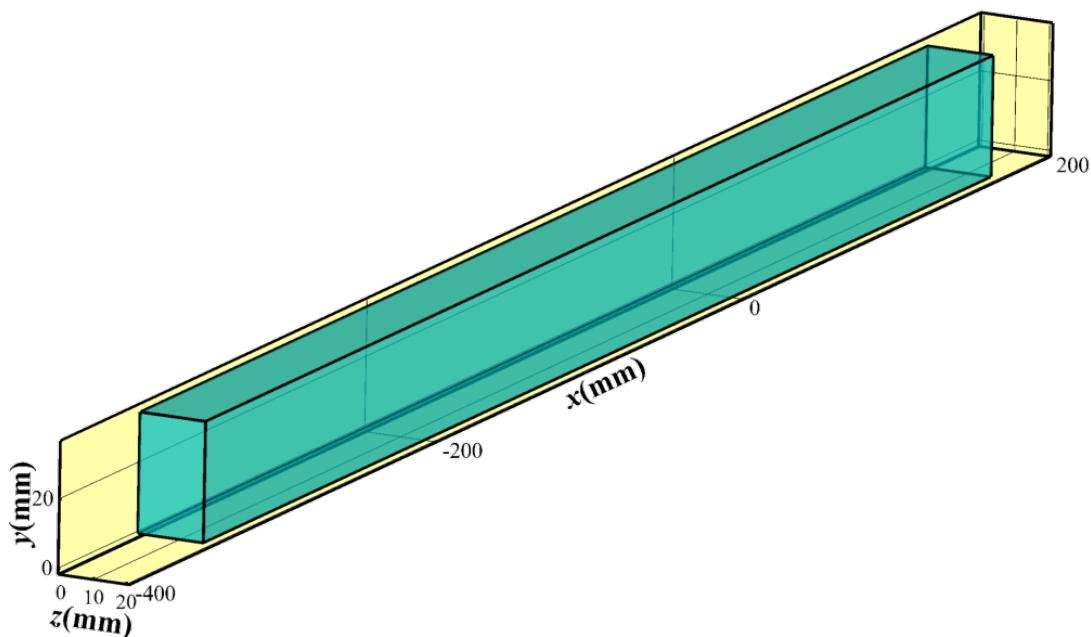


图 6.19 马赫 10 平板计算域) (b) 马赫 6 压缩折角

Figure 6.19 Schematic of computational domain, about flat plate boundary layer flow at Ma 10

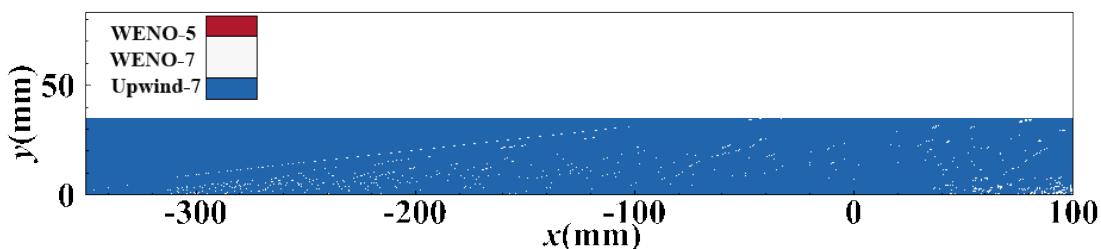


图 6.20 马赫 10 平板混合格式分布

Figure 6.20 Distribution of the hybrid scheme, about flat plate boundary layer flow at Ma 10

度、宽度、高度分别为 370 mm、20 mm 和 35 mm。网格流向、法向、展向分别设置为 $8800 \times 640 \times 800$ ，总网格点数约 45 亿。尽管对此算例开展直接数值模拟时的网格总量不及之前介绍的两个算例，但是由于高雷诺数、高马赫数时的壁面速度梯度更大，为能准确地捕捉壁面的流动，第一层网格只为前两个算例的五分之一，这意味着模拟时需要对应减小时间步长，对此算例模拟时的时间步长为每步 0.002 个无量纲时间，每一个无量纲时间对应着流动空间发展 1 mm 的距离，可见模拟成本并不小。不过由于外形相对简单，在计算时不需要开启特征重构即可保证模拟时的稳定，该算例的无粘项计算依然采用混合格式，粘性项计算仍为八阶中心格式。在使用混合格式时，阈值设置为 0.02 和 0.1，在此设置之下，七阶迎风格式、七阶 WENO 格式和五阶 WENO 格式的使用率占比分别为 99.3%、0.5% 和 0.2%。混合格式的分布情况见图 6.20，与之前相同，蓝色区域意味着使用的是线性的七阶迎风格式，白色区域使用的是七阶 WENO 格式，红色区域使用的是五阶 WENO 格式。激发转捩的扰动形式也与之前介绍的相同，只不过根据 $Ma_\infty = 10$ 对应的第二模态扰动波频率调整了扰动的时间频率，控制频率的参数设置为 0.08，扰动添加位置位于平板段 $x = -320$ mm 到 -300 mm 处。

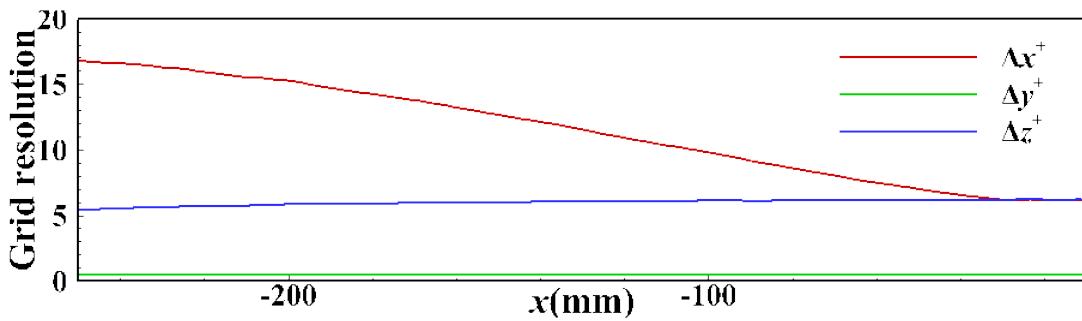


图 6.21 马赫 10 平板边界层网格分辨率的流向分布

Figure 6.21 Streamwise distribution of grid resolution, about flat plate boundary layer flow at Ma 10

图 6.21 展示了网格分辨率沿流向的分布情况。和前面介绍的算例类似，平板的流向网格也分为多段，分别是平板段，网格逐渐变密；加密段，采用均匀网格，对应充分发展的湍流位置；拉伸段，网格指数拉伸从而保证流动顺利流出。这些网格的变化也反应在图 6.21 描述的网格分辨率的变化中，分辨率随着平板段网格的变密而增高，网格在加密段是均匀，所以分辨率也几乎保持水平，此区域网格间距为 $x = 0.2\text{mm}$ ，长度为 50 mm。在加密段，流向、法向、展向的分辨率分别为 $\Delta x^+ = 6.19$ 、 $\Delta y_w^+ = 0.49$ 和 $\Delta z^+ = 6.19$ 。

图 6.22 展示了使用参考密度和参考温度无量纲化的展向中截面的瞬时密度、温度分布情况，图 6.22(a) 和图 6.22(b) 为密度的分布情况，图 6.22(c) 和图 6.22(d) 为温度的分布情况，其中图 6.22(b) 和图 6.22(d) 分别为密度和温度的放大图。图 6.22(a) 和图 6.22(c) 可以看出，流动在经过扰动带后，迅速发生转捩，而且下游的边界层厚度比上游相比更厚，从图 6.22(c) 和图 6.22(d) 可以看到细密的湍流脉动结构。从图 6.23 能更直观地看到湍流相干结构，Q 判据等值面依然根据与壁

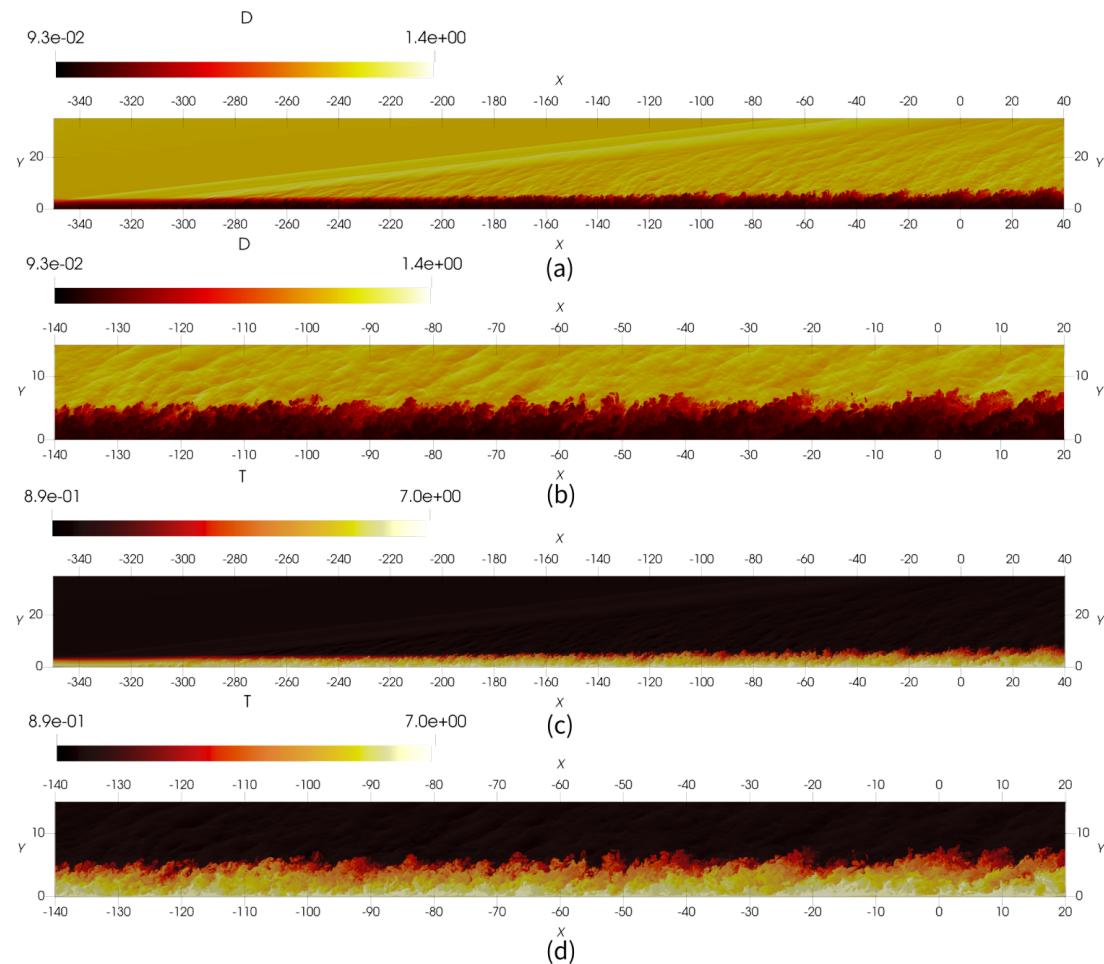


图 6.22 马赫 10 平板边界层无量纲化的瞬时密度、温度分布情况: (a) 无量纲化瞬时密度的全局分布情况。(b) 无量纲化瞬时密度的局部放大情况。(c) 无量纲化瞬时温度的全局分布情况。(d) 无量纲化瞬时温度的局部放大情况

Figure 6.22 Dimensionless instantaneous density and temperature, about flat plate boundary layer flow at Ma 10: (a) overall view of instantaneous density, nondimensionalized by the reference density; (b) magnified view of instantaneous density; (c) overall view of instantaneous temperature, nondimensionalized by the reference temperature; (d) magnified view of instantaneous temperature

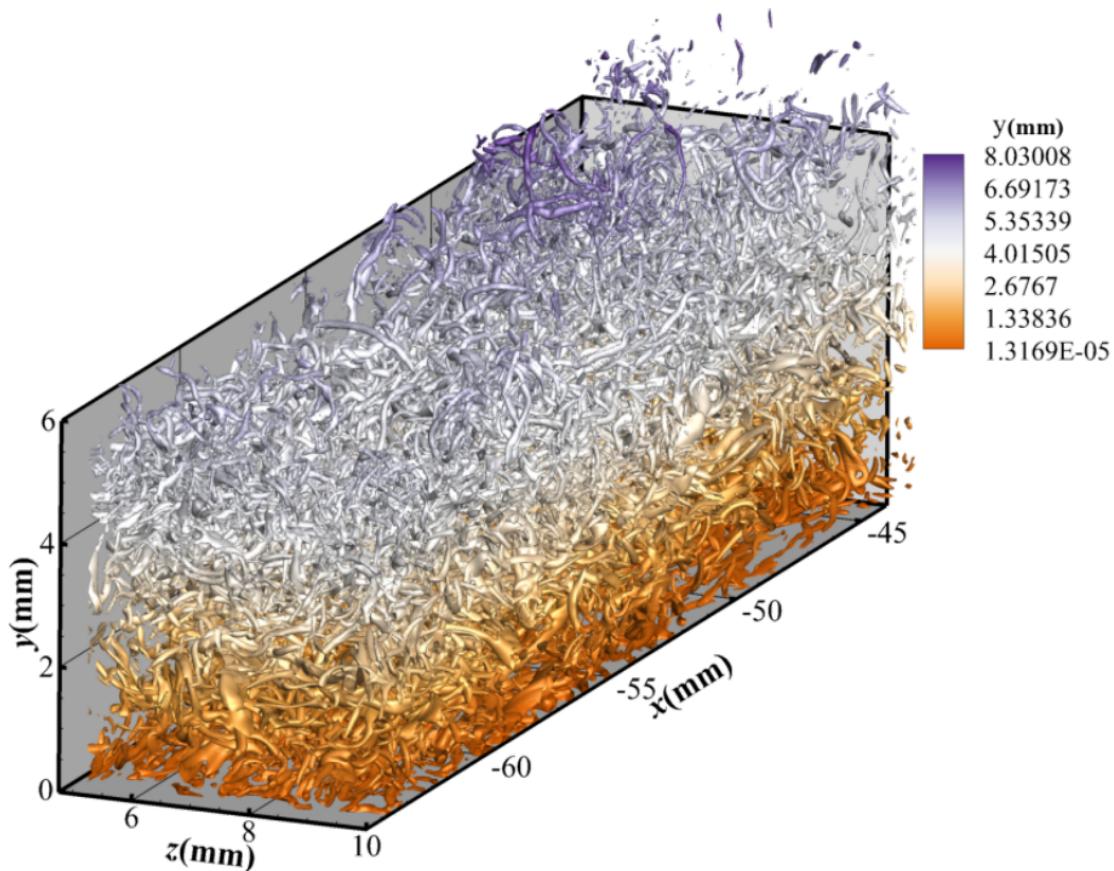


图 6.23 马赫 10 平板边界层利用根据壁面距离染色的湍流涡相干结构

Figure 6.23 Turbulent coherent vortical structures colored by distance from wall, about flat plate boundary layer flow at Ma 10

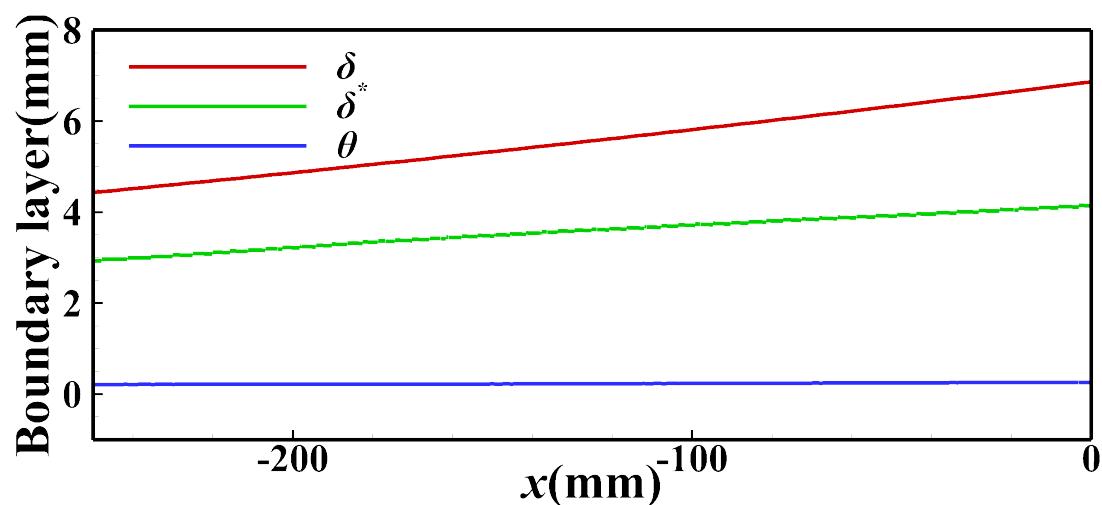


图 6.24 马赫 10 平板边界层厚度沿流向变化

Figure 6.24 Streamwise distribution of boundary layer thickness, about flat plate boundary layer flow at Ma 10

面距离进行染色，与之前的展示相比，高雷诺数的情况下，湍流涡的尺度更小，这从某一方法证明了对高雷诺数流动模拟时为捕捉这些更小尺度的湍流脉动需要尺度更小的计算网格。图 6.24 展示了边界层厚度沿流向的变化情况，随着流动向下游发展，速度边界层 δ 变化最明显，厚度从 4.6 mm 增加到 7.2 mm，而动量边界层 θ 一直维持在 2.1 mm 几乎保持不变。

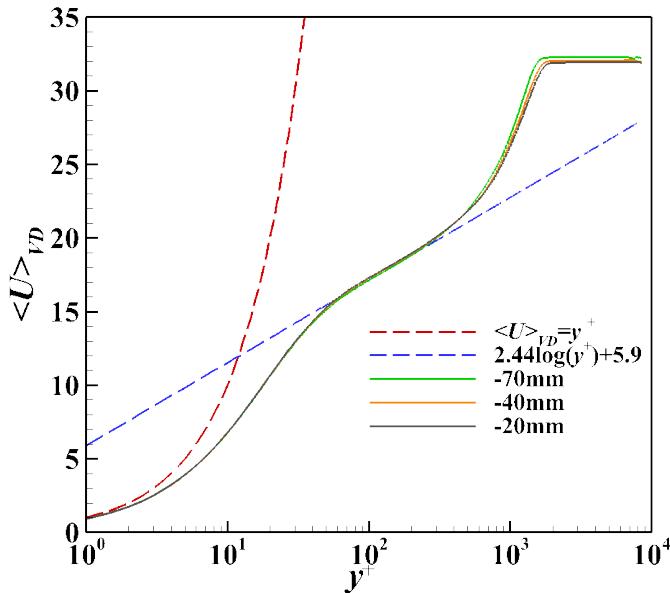


图 6.25 马赫 10 平板边界层不同位置经 Van Driest 变换后的平均速度剖面

Figure 6.25 Van Driest transformed mean velocity profiles at different positions, about flat plate boundary layer flow at Ma 10

图 6.25 展示了 $x = -70 \text{ mm}$ 、 -40 mm 和 -20 mm 三个不同位置的经过 Van Driest 变换的平均速度剖面图。由于都处于充分发展湍流段，这三个位置的平均速度剖面重合的比较好，但与低马赫数的情况相比，尽管 Van Driest 的斜率与 2.44 的理论标准值吻合的比较好，但截距却明显高于其它情况，此处的截距达到了 5.9，截距的差异可能和冷壁效应有关^[39]。此外在靠近壁面的位置平均速度剖面与理论的壁面率差别较大，这在 $\text{Ma}_{\infty} = 6$ 时已经有轻微的体现，在 $\text{Ma}_{\infty} = 10$ 的情况下展现出了更大的差异。

6.5 马赫 6 钝锥湍流边界层的直接数值模拟

钝锥体外形是被广泛用于火箭、导弹、飞机头部等飞行器的常见几何形状，因而也常被选为研究超声速/高超声速转捩与湍流的典型模型。在过去大多数研究中，由于受限于计算能力，很难对较完整的钝锥开展整体模拟计算，部分研究人员选择全尺寸锥体上的一部分区域进行直接数值模拟从而减少计算量^[142–144]，也有人采用加密网格和粗化网格相结合的方式对钝锥体进行模拟^[145,146]。在最近的工作中，Liu 对一个来流马赫数为 8 的钝锥体开展了 360° 度的全角度直接数值模拟^[147]，尽管他们已经使用了 30 亿个网格点，但也只解析了锥体上 $x=14$

mm 到 $x=51$ mm 长度仅为 37 mm 的区域，对完整的钝锥展开全角度 DNS 研究依然是一个挑战。本文借助 GPU 极强的计算能力，使用 OpenCFD-SCU 进行了计算域长度达到 400 mm 的全角度加密的钝锥的直接数值模拟， $Ma_\infty = 6$ ，计算网格点数达到了 240 亿。表 6.4 显示了本算例的具体参数设置与参考点处边界层信息，参考点位于 $x=100$ mm 处。

表 6.6 马赫 6 钝锥体计算参数与参考位置边界层相关参数

Table 6.6 The calculation parameters and the boundary layer parameters of blunt cone flow at Ma 6

Ma_∞	Re_{mm}	T_∞	T_w	δ	δ^*	θ
6	10 000	79 K	293.88 K	10.17 mm	4.5 mm	0.49 mm

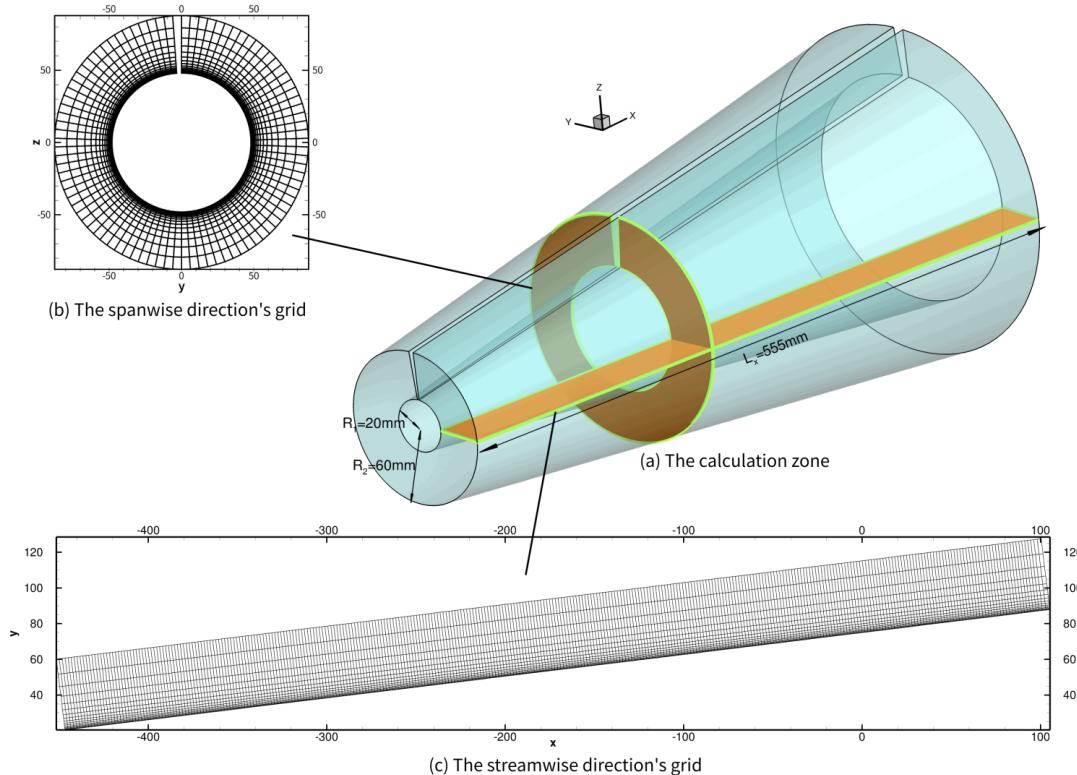


图 6.26 马赫 6 钝锥边界层直接数值模拟计算域与网格示意图

Figure 6.26 Schematic of computational domain and grid, about blunt cone flow at Ma 6

钝锥直接数值模拟的计算域和截面网格展示如图 6.26。DNS 时并不包含钝锥头部的区域，这是由于有限差分法的限制，无法对极点处进行处理，因而计算域的起始位置位于锥体半径达到 20 mm 处，在此位置的流向截面信息由有限体积方法的程序给出。即先使用有限体积方法的程序计算出钝锥层流场，如使用 OpenCFD-EC 程序，然后截取某一流向截面（锥体半径达到 20 mm 处）作为 DNS 计算的入口条件。DNS 计算网格的流向和周向都采用了均匀网格，网格流向、周向、法向分别设置为 $16\,650 \times 4\,500 \times 320$ ，总网格量约为 240 亿。差分离

散格式与前述相同，无粘项使用混合格式，粘性项使用八阶中心格式，针对此算例计算时，并不需要开启特征变量重构。混合格式采用的阈值同马赫 10 平板计算一致，阈值设置为 0.02 和 0.1，其中七阶迎风格式、七阶 WENO 格式和五阶 WENO 格式在计算时的格式占比分别为 99.33%，0.34%，和 0.33%。引发转捩的扰动形式采用了与之前三个算例类似的吹吸扰动，扰动强度为 0.2，时间频率参数设置为 0.1，与之前不同的是，周向波数设置为了 50 个，这是为了让钝锥体表面的转捩起始位置位于相同的流向位置。

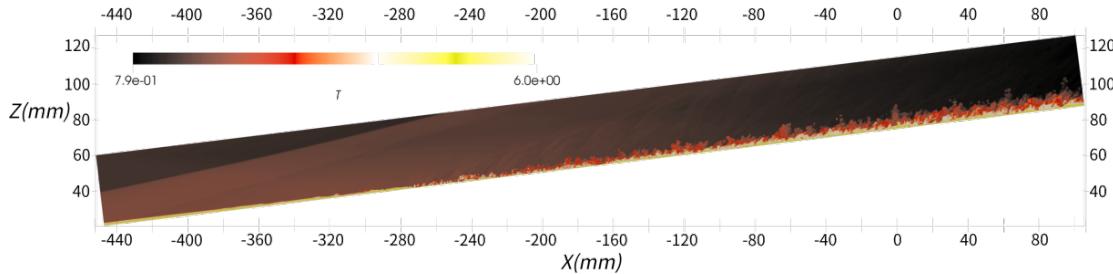


图 6.27 马赫 6 钝锥流向截面瞬时温度分布

Figure 6.27 Instantaneous temperature on a streamwise section, about blunt cone flow at Ma 6

图 6.27展示了使用参考温度无量纲化的流向截面的瞬时温度云图，可以看到有限体积方法提供的层流入口并没有完全位于激波之后，在此情况下激波需要从上边界打出，因而上边界采用了无反射边界条件。从瞬时温度云图来看，钝锥湍流边界层大体形状与平板湍流边界层并没有什么本质性的不同。图 6.28给出了不同流向位置的周向截面瞬时温度云图，从中可以看出湍流边界层沿流动方向不断变厚，而且湍流结构在周向并不非常均匀。图 6.29给出了根据 Q 判据等值面展示的湍流涡结构，它是根据离壁面距离进行染色。能更直观地看出钝锥表面湍流沿周向方向分布有多个涡系结构。

本文在图 6.30 中给出了钝锥体表面边界层厚度沿流向变化情况。可以看出，与平板类似，速度边界层 δ 沿流向变化最明显，它在 200 mm 的距离内增加 2.1 mm， δ 变厚的速度也与 $Ma_\infty = 10$ 平板的情况很接近， $Ma_\infty = 10$ 平板在 200mm 的距离 δ 增加了 2 mm。

图 6.31展示了钝锥体表面 $x = -220$ mm、 -210 mm、 -200 mm、 -180 mm 和 -115 mm 五个不同位置的经过 Van Driest 变换的平均速度剖面图。可以看到，钝锥体上游的边界层平均速度剖面比较接近 Van Driest 的理论情况，随着流动的发展，平均速度剖面的斜率在变得倾斜，截距也在变大，平台区的高度也随着流动方向在逐渐增加。

图 6.32展示了钝锥体表面的摩擦阻力系数 C_f 分布。可以明显的看到在经过头部扰动带之后，流动经过一定距离引发了均匀的转捩，转捩之后的表面摩阻呈条带状。在图 6.33中展示了摩阻 C_f 沿流向发展的曲线。可以看到，流动在 $x = -280$ mm 附近的位置开始转捩，转捩之后 C_f 急剧升高，在 $x = -250$ mm

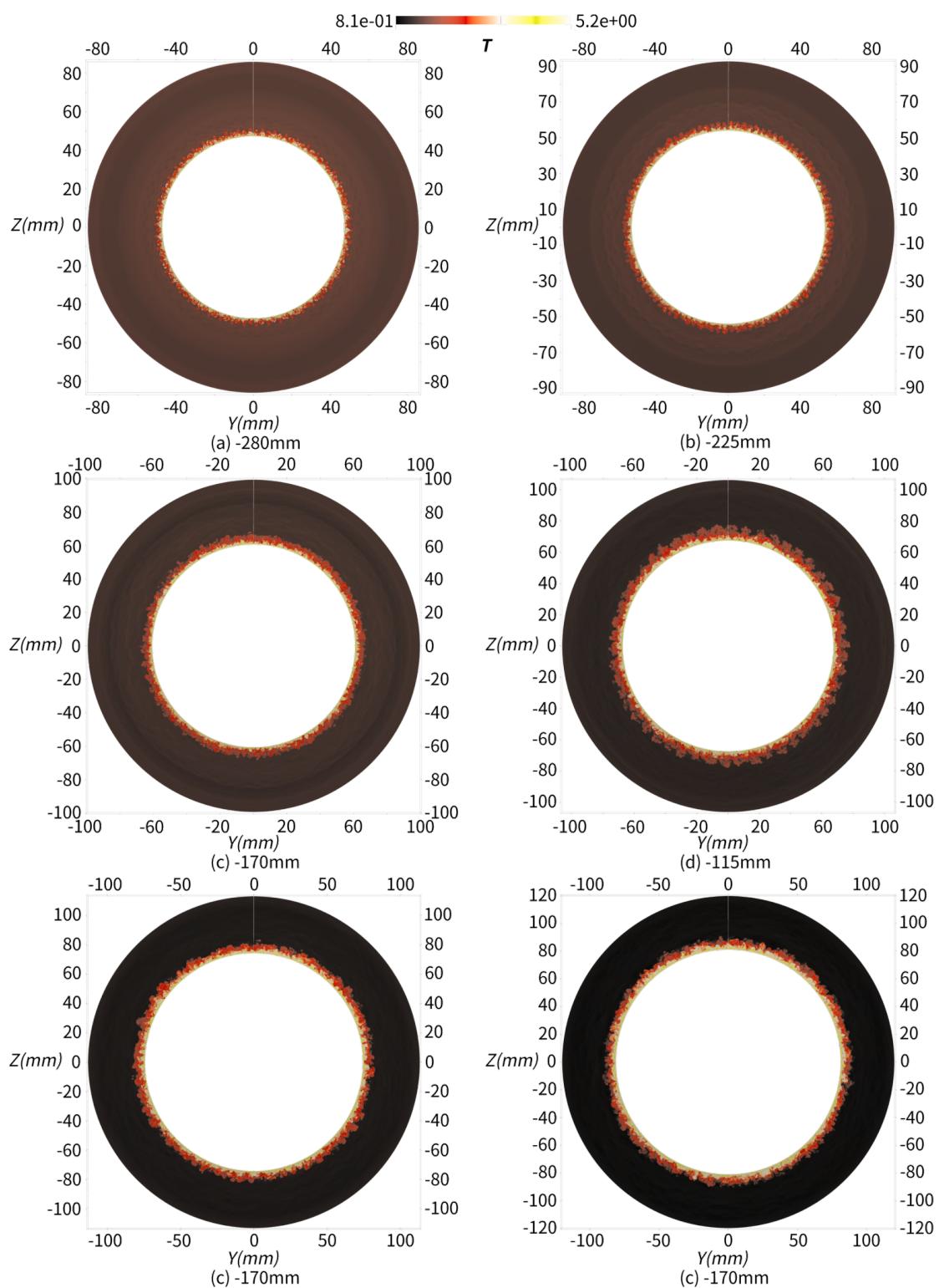


图 6.28 马赫 6 钝锥不同流向位置的周向截面瞬时温度分布

Figure 6.28 Instantaneous temperature on a spanwise section at different streamwise positions, about blunt cone flow at Ma 6

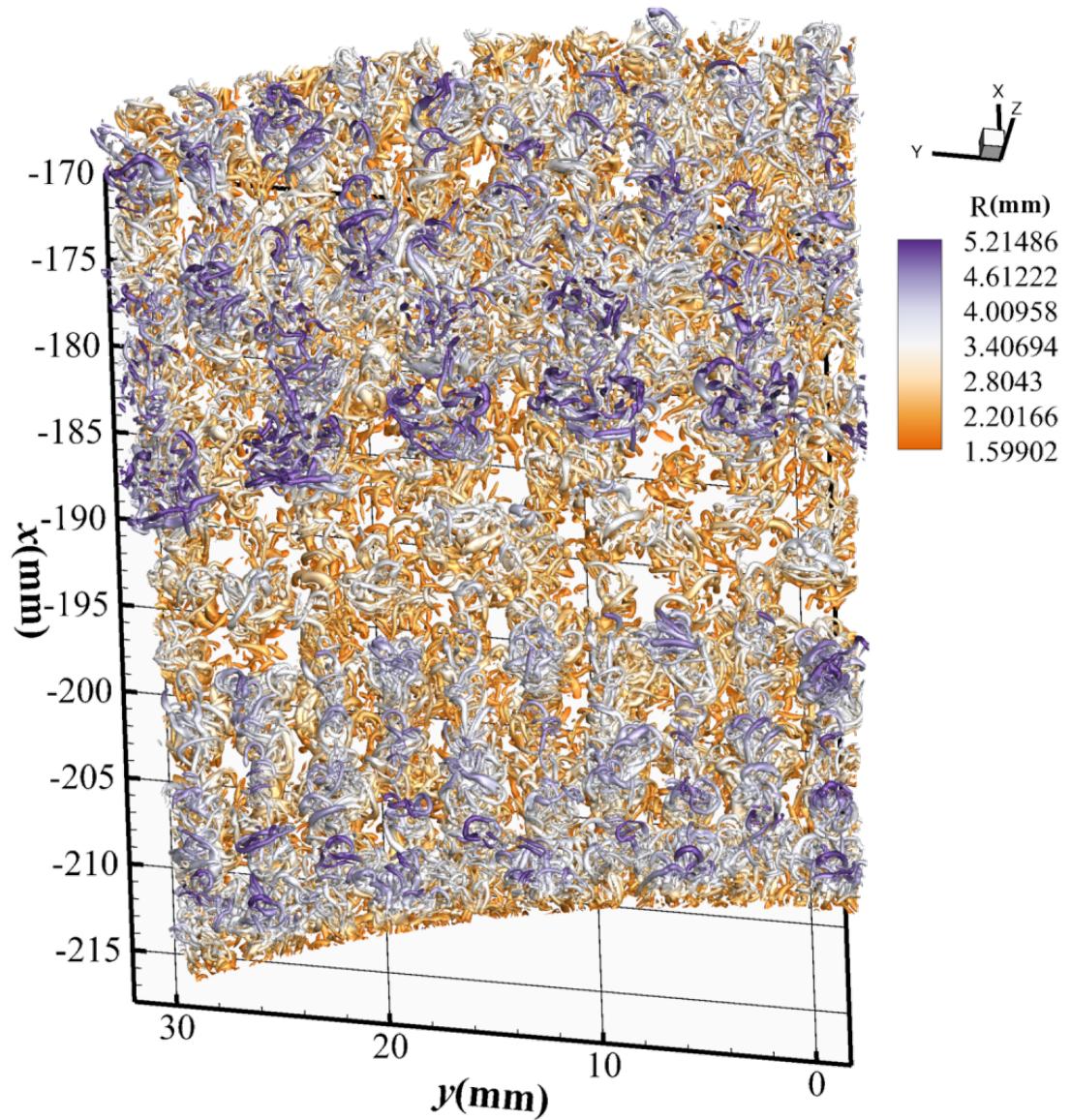


图 6.29 马赫 6 钝锥边界层利用根据壁面距离染色的湍流涡相干结构

Figure 6.29 Turbulent coherent vortical structures colored by distance from wall, about blunt cone flow at Ma 6

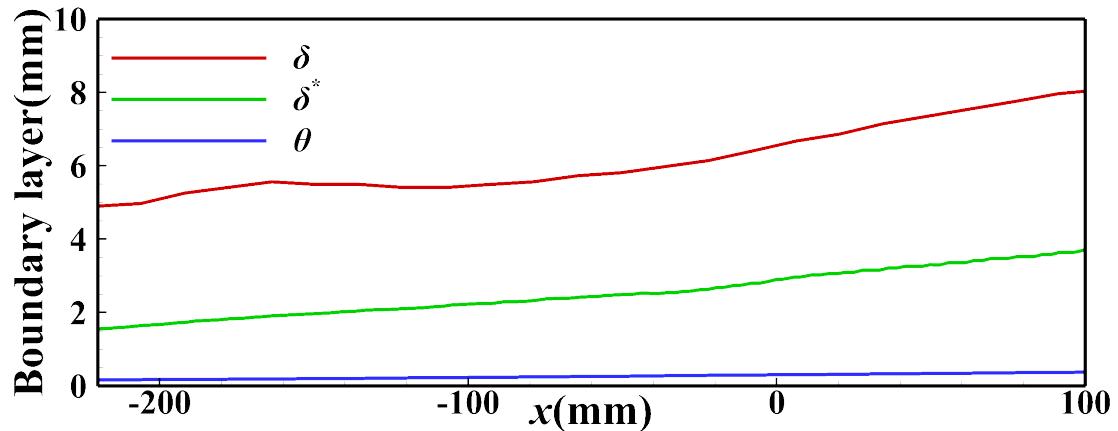


图 6.30 马赫 6 钝锥边界层厚度沿流向变化

Figure 6.30 Streamwise distribution of boundary layer thickness, about blunt cone flow at
Ma 6

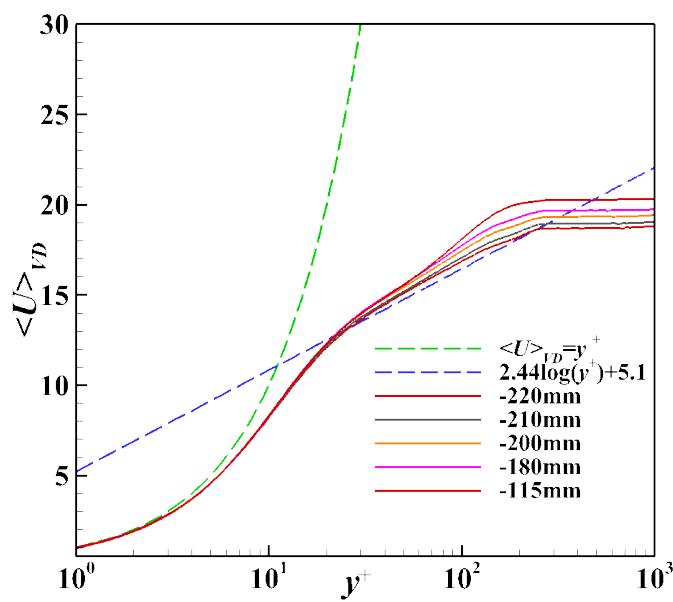


图 6.31 马赫 6 钝锥边界层不同流向位置经 Van Driest 变换后的平均速度剖面

Figure 6.31 Van Driest transformed mean velocity profiles at different positions, about blunt
cone flow at Ma 6

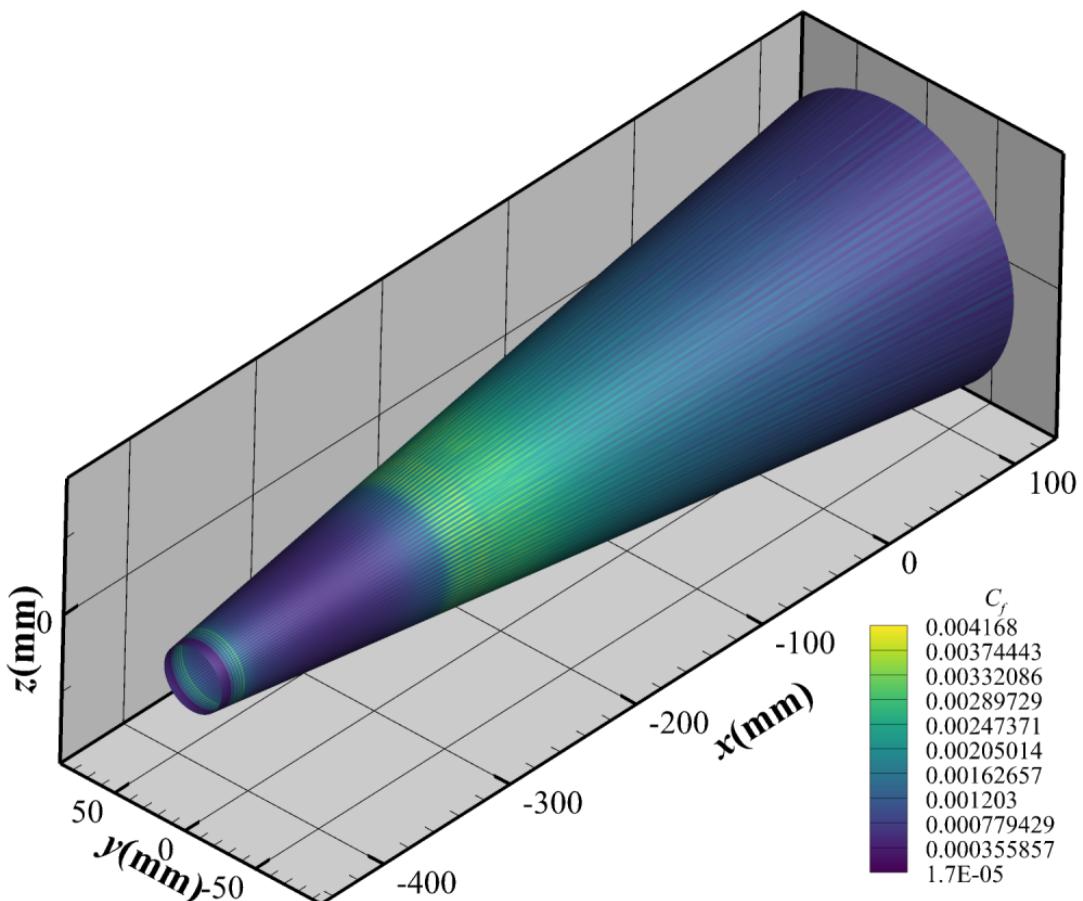


图 6.32 马赫 6 钝锥壁面摩擦阻力系数 C_f 分布

Figure 6.32 The skin friction coefficient C_f distribution of wall, about blunt cone flow at Ma 6

位置左右 C_f 达到峰值，峰值约为 0.0025。到达峰值之后 C_f 开始迅速减少。这种 C_f 迅速减少的现象类似于在 $\text{Ma}_\infty = 6$ 压缩折角算例中发现的流动冲击造成的 C_f 达峰后的减少，但机理不同，而且钝锥 C_f 减少的更快，这是由于随着钝锥扩展，钝锥表面的流体有逐渐层流化的趋势。

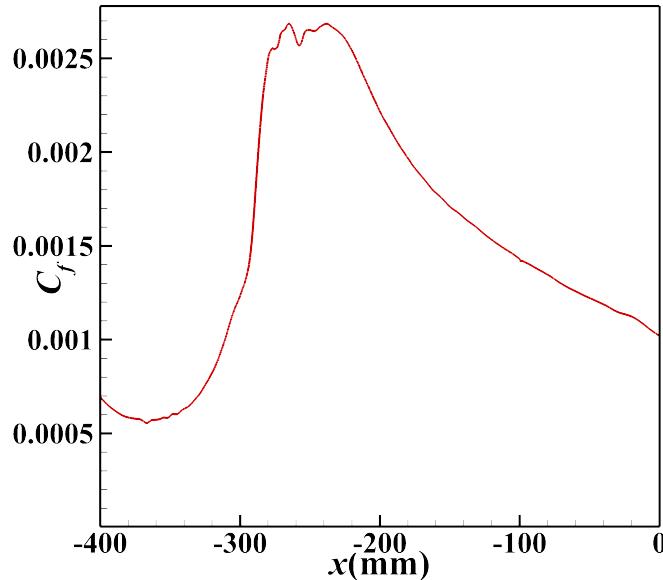


图 6.33 马赫 6 钝锥摩擦阻力系数 C_f 沿流向变化曲线

Figure 6.33 Streamwise distribution of skin friction coefficient C_f , about blunt cone flow at Ma 6

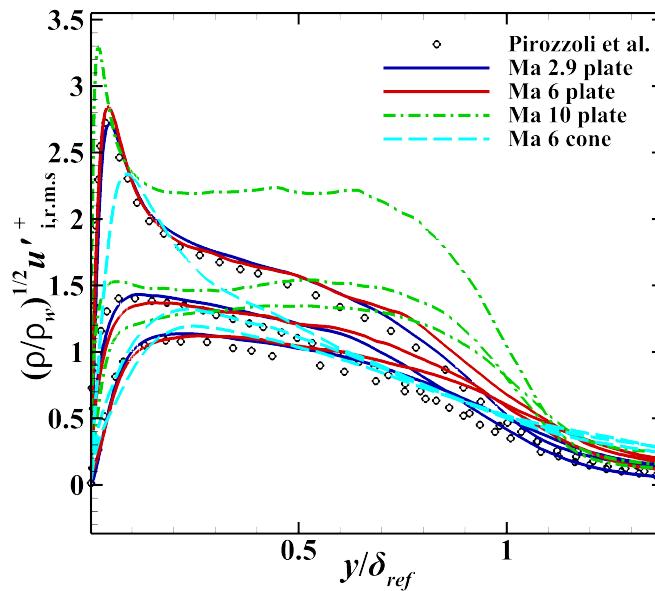


图 6.34 不同算例的脉动速度均方根

Figure 6.34 Root mean square velocity fluctuations at different cases

图 6.34 中，对比了前四个算例的脉动速度均方根，这四个算例分别是 $\text{Ma}_\infty = 2.9$ 压缩折角流动， $\text{Ma}_\infty = 6$ 压缩折角流动， $\text{Ma}_\infty = 10$ 平板边界层流动， $\text{Ma}_\infty = 6$

钝锥体边界层流动。图 6.34 中还添加了 Pirozzoli 等人^[71]超声速平板 DNS 的结果。结果表面, $Ma_{\infty} = 10$ 高雷诺数平板的湍流脉动强度明显高于其它情况。 $Ma_{\infty} = 6$ 压缩折角的平板区与 $Ma_{\infty} = 2.9$ 压缩折角平板区的湍流脉动情况类似, 也近似于 Pirozzoli 等人的结果, 而 $Ma_{\infty} = 6$ 钝锥体边界层与 $Ma_{\infty} = 6$ 压缩折角的平板区相比, 流向湍流脉动较弱, 而周向和法向湍流脉动较强, 这是由于钝锥存在更强的横流导致的。

6.6 马赫 6 升力体转捩的直接数值模拟

表 6.7 马赫 6 升力体计算参数与参考位置边界层相关参数

Table 6.7 The calculation parameters and the boundary layer parameters of lifting body flow at Ma 6

Ma_{∞}	Re_{mm}	T_{∞}	T_w	δ	δ^*	θ
6	10 000	79 K	293.88 K	5.41 mm	2.12 mm	0.216 mm

最后, 本文使用 OpenCFD-SCU 程序对更接近真实飞行器外形的高超声速升力体构型进行直接数值模拟。计算基于中国空气动力研究开发中心的高超声速转捩研究飞行器 (Hypersonic Transition Research Vehicle, HyTRV) 标准模型。在最近的研究中, Chen 等人^[127]对此外形开展了直接数值模拟, 在他的文章中提到, 由于分辨率的不足, 他们的空间离散不能捕捉到附着线不稳定性, 而附着线不稳定性在实验中已经被发现。与他人的计算相比^[127-129], 本文提供了更高分辨率的计算结果, 而且能够从翼侧捕捉到附着线位置的不稳定现象。本算例的具体参数设置与边界层信息参见表 6.7, 来流条件为: 来流 Mach 数为 6, 单位长度 Reynolds 数为 10000/mm, 来流静温 79K, 升力体壁面温度 300K。

图 6.35 展示了直接数值模拟的计算区域, 与钝锥类似, 升力体的计算分两步进行。先通过有限体积方法获得稳态流场数据; 再通过 OpenCFD-SCU 有限差分程序对小区域进行边界层转捩 DNS 计算, 入口条件和外边界条件由层流数据提供。为了节省计算资源, 计算采用半模, 对称面处采用对称边界条件。稳态计算的计算域较大, DNS 的计算域全部在头激波之内。无粘项计算先通过 Steger-Warming 分裂, 后采用混合格式加特征重构对无粘项进行离散, 粘性项离散采用八阶中心差分。转捩的触发形式与之前不同, 为了更加近似于自然转捩, 计算中在飞行器头部 50 mm-60 mm 区间壁面上添加了幅值为来流速度的 2% 的随机吹吸气扰动, 该扰动幅值较小近似于壁面粗糙度引起的扰动。

网格流向、周向、法向分别设置为 $8700 \times 4000 \times 320$, 总网格点数约 111 亿。下表面位置中心线的流向、周向和法向上的网格分辨率分别为 $\Delta x^+ = 5.1$ 、 $\Delta y^+ = 2.3$ 和 $\Delta z_w^+ = 0.4$ 。图 6.36 展示了不同流向位置的周向截面瞬时温度分布情况。可以看到, 在下表面的中心区和上表面的内凹区出现了一些明显的大尺度横流涡旋结构。下表面中心区的涡在上游时呈蘑菇状, 随着流动向下游发展涡变得不稳定, 高温区扩大, 横流涡逐渐破碎变为湍流。

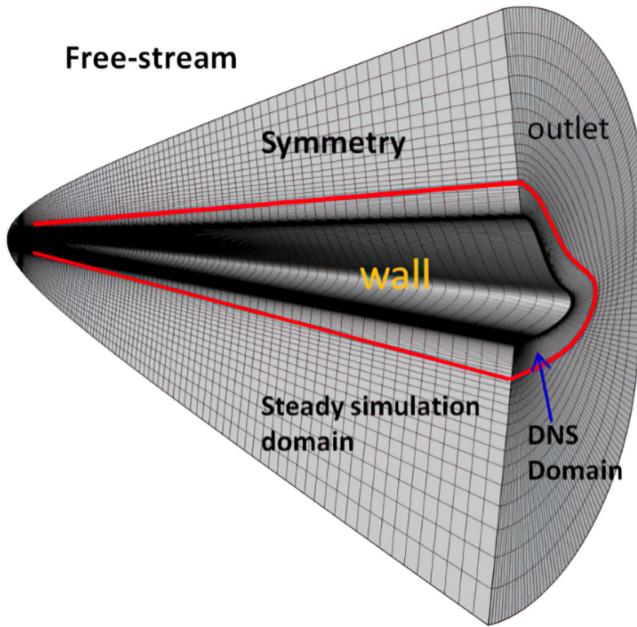


图 6.35 马赫 6 升力体计算域示意图

Figure 6.35 Schematic of computational domain, about lifting body flow at Ma 6

升力体表面摩擦阻力系数 C_f 与壁面温度 T_w 展示在图 6.37 和图 6.38 中，可以看出，高摩阻 C_f 所在的区域往往也是高温度 T_w 的区域， C_f 与 T_w 的分布具有很强的相似性，从中观察到的转捩位置也非常相似。在图 6.37(b) 中，展示了侧视角度的壁面摩阻 C_f ，可以看到在升力体翼侧，能够观察到附着线区域的不稳定性现象，由于缺乏分辨率，在他人的模拟中附着线不稳定性难以发现。图 6.38 中还显示出了升力体表面流线的分布情况，从中可以观察到，转捩发生区域多为流线汇聚的位置，其主要原因可能是扰动波沿流线进行传播，流线汇聚区会聚集更大的非线性扰动，从而更容易触发转捩。

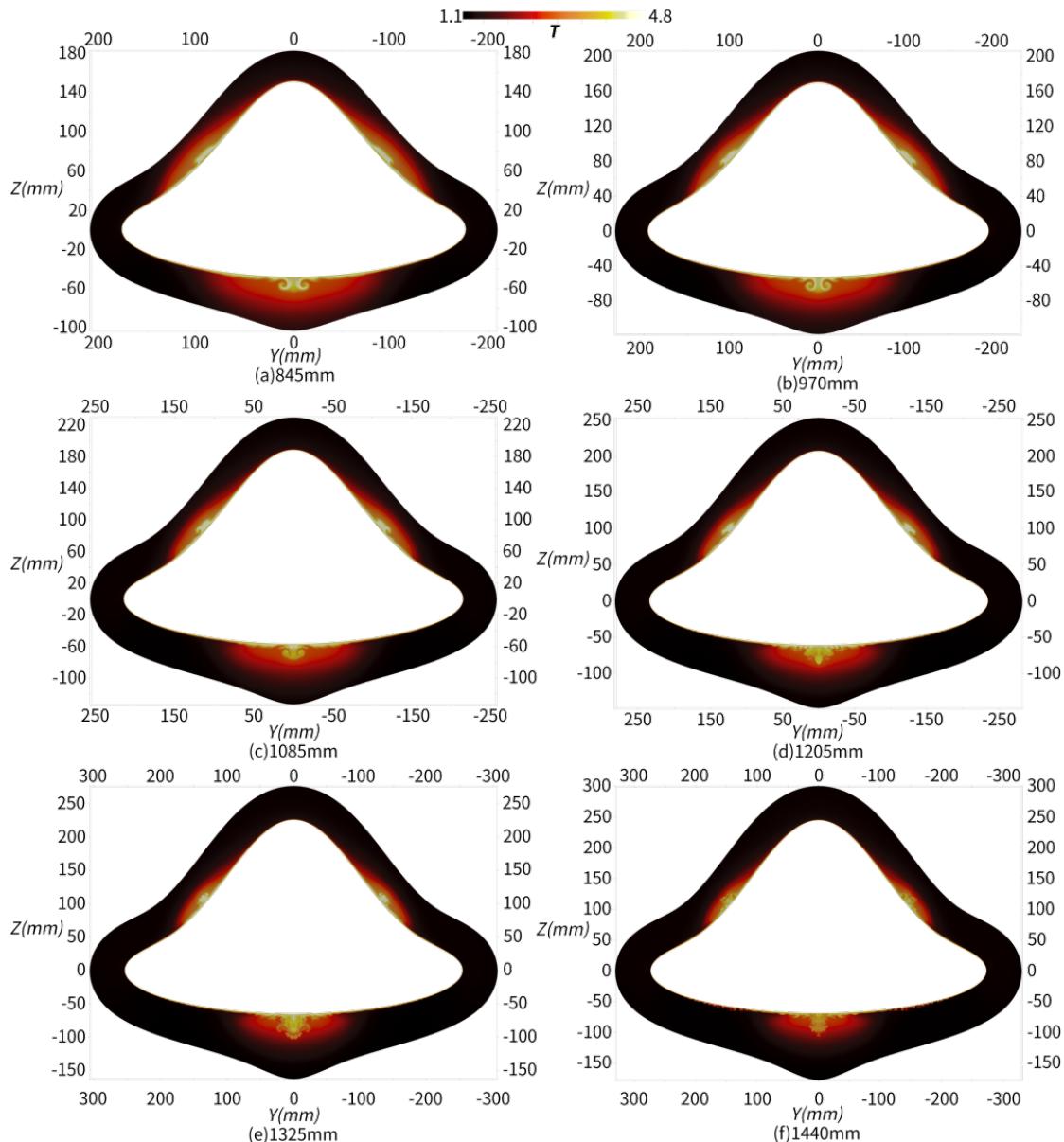


图 6.36 马赫 6 升力体不同流向位置的周向截面瞬时温度分布

Figure 6.36 Instantaneous temperature distribution on a spanwise section at different streamwise positions, about lifting body flow at Ma 6

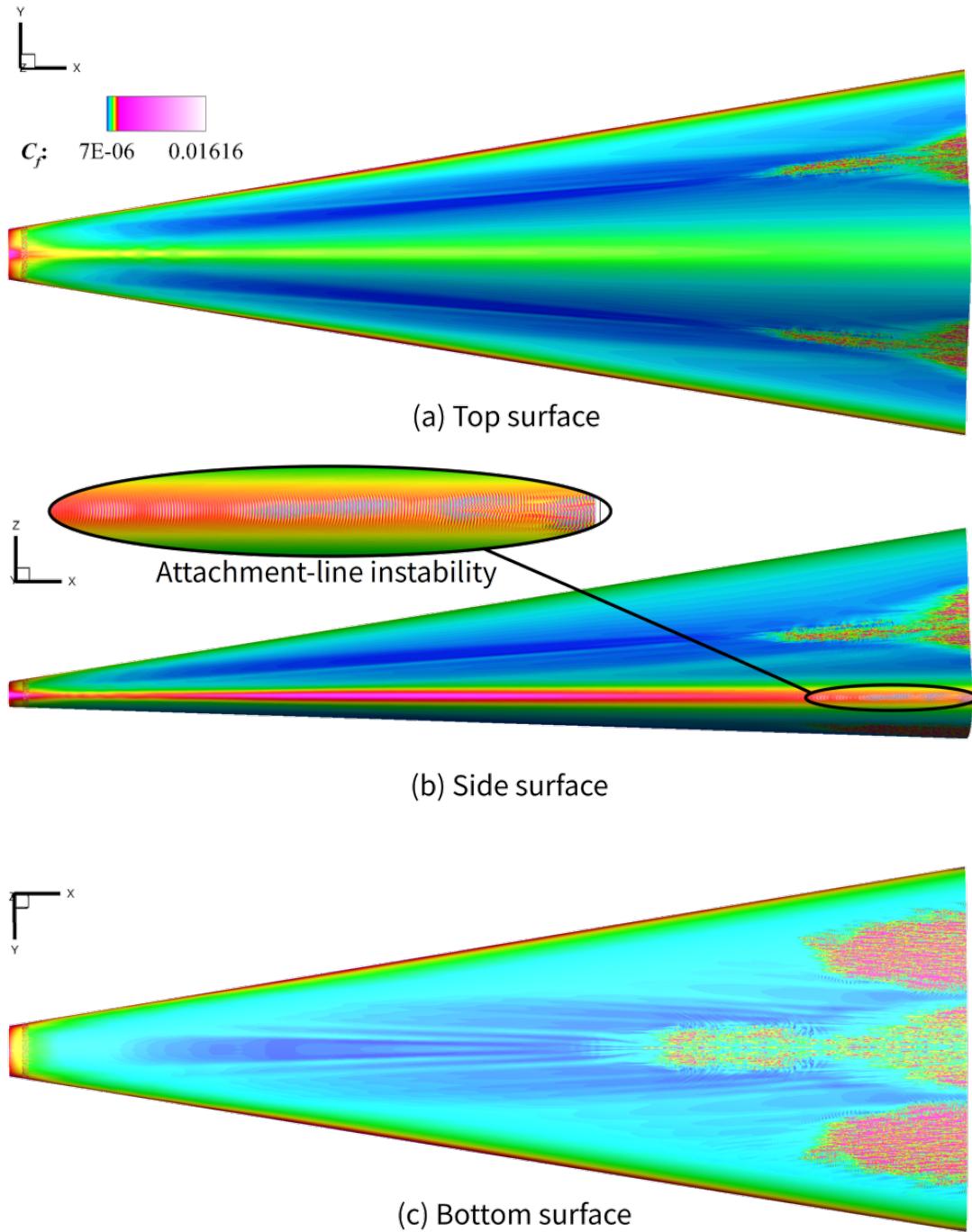


图 6.37 马赫 6 升力体壁面摩擦阻力系数 C_f 分布

Figure 6.37 Distribution of skin friction coefficient C_f on the wall, about lifting body flow at
Ma 6

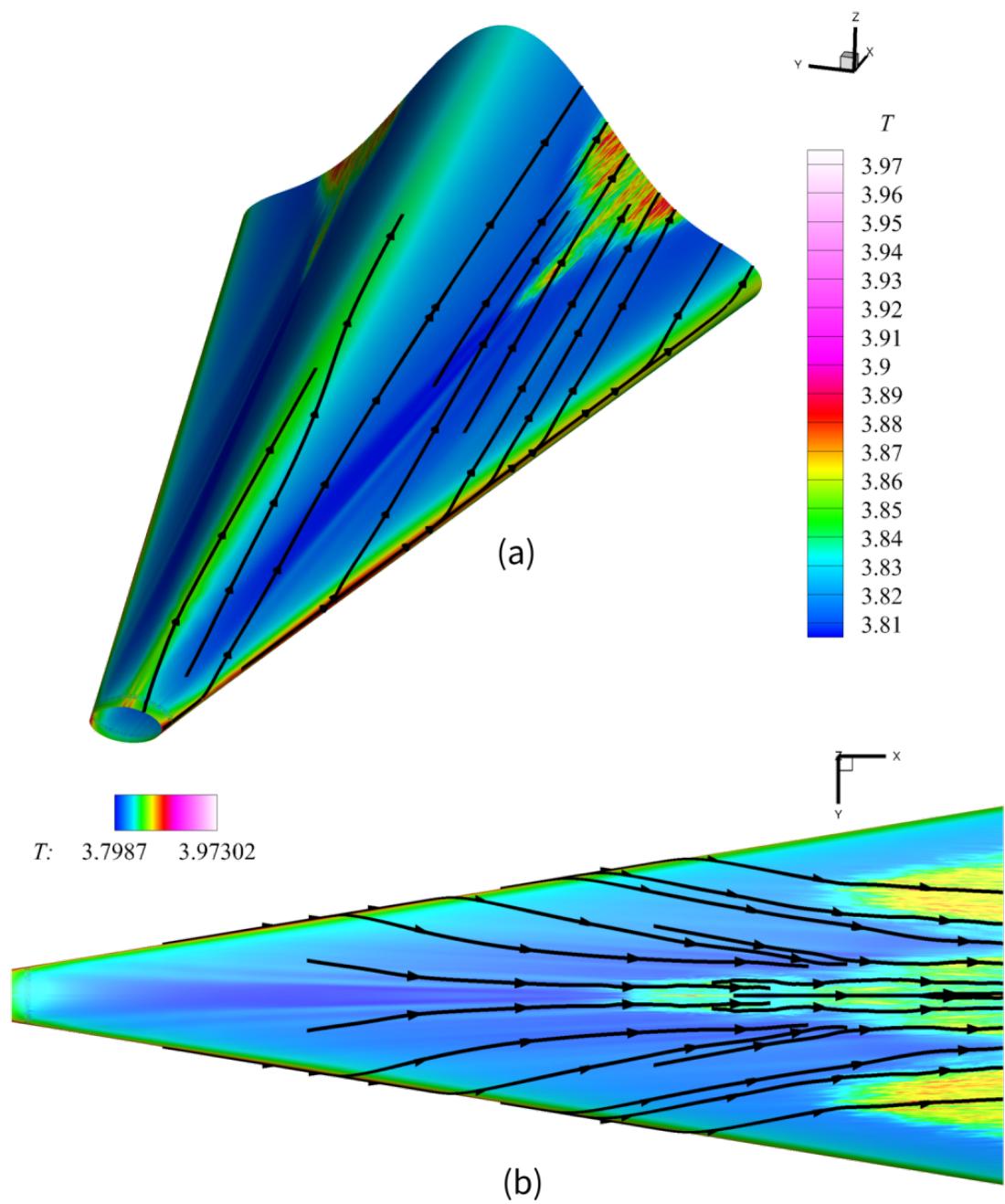


图 6.38 马赫 6 升力体壁温 T_w 以及壁面流线分布

Figure 6.38 Wall-temperature T_w distribution and streamtraces of lifting body flow at Ma 6,
about lifting body flow at Ma 6

6.7 小结

本部分介绍了使用开源有限差分软件 OpenCFD-SCU 开展的各种直接数值模拟实例，展示了该求解器在处理超声速和高超声速流动模拟问题的能力。

第一个算例是一个来流马赫数为 2.9 具有 24° 斜坡角的压缩折角流动。计算网格数达到了 76.8 亿个网格点，计算域具有 200 mm 的长距离坡道长度。

之后又展示了一个来流马赫数为 6 具有 34° 斜坡角的压缩折角流动，本文对比了超声速和高超声速情况下的不同，在 $Ma_\infty = 6$ 的情况下，表面摩擦阻力系数 C_f 和斯坦顿数 S_t 在坡道上达到最大值，然后迅速减小，而 $Ma_\infty = 2.9$ 时会保持较长距离的峰值，这是因为 $Ma_\infty = 6$ 时的分离激波比再附激波弱，气流将冲击再附区下游的壁面，产生过冲现象。马赫 6 斜坡角 34° 的压缩折角流动计算也证明了 OpenCFD-SCU 对较复杂与极端问题的模拟能力。

第三个算例是高雷诺数、高马赫数平板， $Ma_\infty = 10$, Re_τ 约为 1550, 在高雷诺数的情况下，湍流涡结构具有更多的小尺度结构，相比其它情况涡结构更为细碎，而且在冷壁效应的影响下，Van Driest 变换后的平均速度剖面截距更高。而且在 $Ma_\infty = 2.9$ 、 $Ma_\infty = 6$ 、 $Ma_\infty = 10$ 不同的情况下，平均速度剖面近壁区 $\langle U \rangle_{VD}$ 与 y^+ 的壁面率逐渐不再吻合。

之后计算了一个来流马赫数为 6 具有 7° 半锥角的钝锥体湍流边界层的直接数值模拟。相比于他人类似的计算，具有更大的计算域，网格规模达到 240 亿。在此工况下，钝锥表面流动通过强制的旁路转换为湍流之后会随着钝锥体扩展而呈现出逐渐层流化的趋势。

对比了以上几个算例的脉动速度均方根， $Ma_\infty = 10$ 高雷诺数平板的湍流脉动强度明显高于其它情况。 $Ma_\infty = 6$ 压缩折角的平板区与 $Ma_\infty = 2.9$ 压缩折角平板区的湍流脉动情况类似，也近似于 Pirozzoli 等人的结果，而 $Ma_\infty = 6$ 钝锥体边界层与 $Ma_\infty = 6$ 压缩折角的平板区相比，流向湍流脉动较弱，而周向和法向湍流脉动较强。

最后，模拟了一个更接近真实飞行器外形的高超声速升力体构型，在高分辨率的情况下，观察到了难以捕捉的翼侧位置附着线不稳定性。

以上算例所使用的开源程序 OpenCFD-SCU 的源代码可在<http://developer.hpccube.com/codes/danggl/opencfd-scu.git>上获取。数据库算例的获取方式与介绍也可在开源程序 OpenCFD-SCU 的源代码网站获取。

第 7 章 结论与展望

7.1 结论

本文介绍了利用 CPU-GPU 异构体系架构开展超大规模湍流直接数值模拟的情况。详细介绍了程序所使用的有限差分算法，程序框架与并行计算设计。重点介绍了程序在 CPU-GPU 异构系统下开发的方法、优化的技巧。并展示了大规模测试的结果。此外，本文介绍了一种混合差分格式，利用此格式对几个具有挑战性的问题开展了直接数值模拟，建立了壁湍流数据库。主要结论如下：

(1) 本文在 CPU 端的高精度有限差分法软件 OpenCFD-SC 的基础之上移植开发了一套 CPU-GPU 异构系统下的软件 OpenCFD-SCU，软件数值方法与 OpenCFD-SC 一致，在软件框架上进行了升级，通过调整计算顺序，改变函数计算方法，将无数据依赖的无粘项、粘性项计算并发执行。软件框架的升级使 OpenCFD-SCU 能够通过线程级、任务级、进程级多级并行的方式充分发挥硬件性能，并实现计算与通讯的重叠。并行 IO 的设计也让程序能够胜任大规模计算的需求。

(2) 有限差分方法在 GPU 上部署时优化的重点是访存。本文结合硬件特性使用多种优化方法来提升程序性能，优化方法包括内存访问合并、线程束内数据交换、冗余计算以减少全局内存访问，通过打包操作优化 CPU-GPU 间数据通讯。异构硬件结构带来的一个显著特点是，不同硬件上运行的指令具有天生的并发性，本文利用这一特点，通过多流水线技术完成了 GPU 上计算与进程间通讯的重叠，经过大规模测试，在相同 MPI 进程下 OpenCFD-SCU 比 OpenCFD-SC 加速 200 倍以上，弱扩展性在 24,576 个 GPU 上达到 98.7% 的并行效率，性能达到 17.1Pflops，占相应计算机节点理论峰值性能的 11.4%。本文在最大 312 亿网格的 DNS 中，测试了软件在真实情况下的表现。对于 300 亿或以下网格规模的 DNS 问题，利用最多 1280 块 GPU，任何一个算例计算耗时不超过两周。OpenCFD-SCU 使 DNS 模拟研究水平从十亿网格级提升到了百亿网格级。

(3) 本文在 Jameson 的人工粘性系数基础上，发展了修正的 Jameson 激波识别器，并以此构造了三格式混合的混合格式，将 7 阶迎风，7 阶 WENO，5 阶 WENO 混合，在大量算例的测试中，发现该混合格式能够很好地兼顾格式鲁棒性与分辨率，在高超声速激波湍流边界层干扰的模拟中，与特征变量重构相配合，也能保证在计算的全程不发散（前提是时间步长和网格长细比满足要求）。混合格式大大增强了计算鲁棒性，而且在所有算例的计算中，混合格式都能保持 90% 以上的点位采用 7 阶迎风格式，兼顾了高分辨率和低耗散。

(4) 本文利用 OpenCFD-SCU 与混合格式建立了一个开放的湍流数据库，其中包括 5 个算例，网格最小为 40 亿，最大为 240 亿，计算模型从最简单的平板模型，到接近真实飞行器的升力体外形，来流马赫数从最低的 2.9 到最高的马赫 10。摩擦雷诺数最高达到 1550。数据库中的算例或从分辨率、或从计算域或雷

诺数超越了目前类似算例的记录。本文对它们进行了分析和比较，(5) (6) (7)列举了分析与比较后得出的结论。

(5) 基于 Morkovin 假设的 Van Dierst 变换对数律在马赫 2.9 和马赫 6 的平板边界层中适用，但在马赫 10 冷壁的情况下，对数率的斜率依然适用，但截距高于 5.1 的理论值，为 5.9。Van Dierst 变换的壁面率随马赫数增加适用性逐渐降低。

(6) 马赫 2.9, 24° 压缩折角的摩擦阻力系数和壁压会在达到峰值后维持在最大值很长一段距离，而马赫 6, 34° 压缩折角的摩擦阻力系数和壁压会在达到峰值点后以近似线性的方式下降，马赫 6, 34° 压缩折角的热流达到峰值后会以两段不同的规律下降。而马赫 2.9, 24° 压缩折角的热流会在峰值点后以近似线性的方式下降。本文对这两种不同现象产生的原因进行了分析，通过比较流线和激波形状认为马赫 6, 34° 情况下，会出现流体对壁面的冲击作用，流动过冲是导致两者产生不同规律的原因。

(7) 本文开展了网格规模达到 240 亿的钝锥直接数值模拟，发现在钝锥表面添加吹吸扰动引发转捩后，随着流动扩展，流动出现层流化的趋势，Van Dierst 变换在钝锥边界层依然适用。钝锥的脉动速度均方根证明其与平板相比流向湍流脉动较弱，而周向和法向湍流脉动较强。

7.2 本文工作的创新点

本文工作主要创新点如下：

(1) 发展了 GPU 加速的异构并行有限差分法计算流体力学软件。取得了与 CPU 程序相比 200 倍以上的加速效果。此程序已开源。并根据开发经验，总结了一套结合硬件的 GPU 程序优化、开发的技巧，可为计算流体力学或其它学科的异构程序开发提供参考。

(2) 提出了一种基于 Jameson 人工粘性系数的混合格式，采用与大部分混合格式所不同三格式混合 (7 阶迎风 +7 阶 WENO+5 阶 WENO)。在数个高精度算例的测试中发现它可以很好地保证计算的鲁棒性与低耗散 (90% 以上采用低耗散的线性 7 阶迎风格式)。

(3) 对展向计算域达到 200mm 的来流马赫数 2.9 的压缩折角展开了模拟，网格达到 312 亿。迄今为止，未发现已发表的文献中，有网格数量如此多，计算域如此宽的压缩折角流直接数值模拟。由于较宽的展向计算域，发现压缩折角流动存在三维特征，不同于以往理解的压缩折角流为准二维流动。

(4) 对长坡面的来流马赫数 2.9 的压缩折角展开了模拟，坡面长度超过 200mm，网格 77 亿。迄今为止，未发现已发表的文献中，有如此长坡面计算域的压缩折角流直接数值模拟。由于较宽的坡面，发现马赫 2.9, 24° 压缩折角的摩擦阻力系数和壁压会在达到峰值后维持在最大值很长一段距离。

(5) 目前较少有高超声速压缩折角流直接数值模拟的报道。本文使用混合格式克服了计算发散的问题，并使直接数值模拟能达到较高计算精度。对比发现，

马赫 6, 34° 压缩折角流的热流、摩擦阻力、壁面压力规律与马赫 2.9, 24° 压缩折角流有较大不同。马赫 6, 34° 时再附激波更强，流动会冲击坡面产生局部峰值。

(6) 目前同时具备高马赫数和高雷诺数的平板湍流边界层直接数值模拟较少，在马赫 10 冷壁的情况下，本文扩展了雷诺数的范围，摩擦雷诺数达到 1550。研究发现基于 Morkovin 假设的 Van Dierst 变换对数率在此情况下斜率依然吻合。

(7) 本文进行了马赫 6 较大计算域钝锥的直接数值模拟，网格数到达 240 亿。计算域和网格数超出了钝锥 DNS 的文献记录。研究发现，与平板湍流边界层不同，钝锥表面流动转换后，由于扩展作用，会有层流化的趋势，但基于 Morkovin 假设的 Van Dierst 变换对于钝锥边界层依然适用。而且由于三维效应，钝锥的流向脉动速度均方根弱于平板湍流边界层，展向、法向的脉动速度均方根强于平板湍流边界层。

(8) 本文对高分辨率升力体转换开展了直接数值模拟，网格达到 110 亿，分辨率超过文献记录的升力体 DNS。首次在直接数值模拟中发现附着线不稳定现象，此现象在实验中已被观察到，在他人的 DNS 报道也指出，未观察到附着线不稳定是由于分辨率不足造成的。

(9) 本文建立了一个开放的湍流数据库，包含 5 个典型算例，算例质量达到国际一流水平，能为可压缩湍流机理研究和模型发展提供持续支持。

7.3 展望

数据库中的数据还有很多可挖掘的信息。软件和数据都是开源的，其它研究者可以下载数据进行进一步的分析，也可以参考本文的设置，自行设计算例。OpenCFD-SCU 源代码可在<http://developer.hpccube.com/codes/danggl/opencfd-scu.git> 上获得，数据库中数据获取方法也可从源代码网站上获得。这些数据可以通过网站下载，它们的存储位置就在我们开展模拟的超算上，由于算例规模比较大，一个流场动辄几百 G，用户可以联系超算中心直接在超算平台上开辟账号拷贝到本地进行分析，避免数据下载与个人电脑分析的困难。

目前以 GPU 为代表的专用芯片或 AI 芯片，性能仍然以每两年翻两番的速度发展，这被称之为黄氏定律。如果直接数值模拟能够不断追随高性能计算的发展，并且芯片性能增长速度不衰减的话。再过 20 年，计算机发展水平能够提升一千万倍，但那时候可能超级计算机就能够对大飞机整机开展 DNS 了，Splart 的估计会被大大提前。但这只是假设芯片发展速度不衰减的理想情况，考虑到目前最先进的晶体管的制程已经达到 3nm，晶体管制程到达 1nm 左右会有明显的量子隧穿现象，通过提升制程来提升芯片性能的道路可能要不了多久就要走到终点，因此，未来芯片发展可能会有另外两条道路，一是通过做大芯片尺寸容纳更大的晶体管，或通过设置更多低精度计算单元以提升芯片性能（高精度计算单元占用的晶体管数量更多），如果直接数值模拟程序想要追随高性能计算的发展，可能也要匹配这两条道路，前者是匹配更大规模和更细粒度的并行计算，后者则需要考虑用更多低精度计算单元完成 DNS 计算，如今的单芯片的半精度已

经能达到 Pflops 的计算能力，考虑单精度、半精度或混合精度又不影响结果准确性的 DNS 算法，或许将会是未来研究的热点。

目前越来越多的人们开始关注高超声速流动，在高超声速情况下，真实气体效应是不得不考虑的问题，下一步目标是发展具有多物质和化学反应计算功能的 GPU 异构程序。以便开展更高马赫数和壁温的直接数值模拟研究。

参考文献

- [1] Nvidia cuda toolkit documentation [EB/OL]. 2022. <https://docs.nvidia.com/cuda/archive/10.2/cuda-c-programming-guide/index.html#from-graphics-processing-to-general-purpose-parallel-computing-floating-point-operations-per-second-for-cpu-and-gpu>.
- [2] Tennekes H, Lumley J L, Lumley J L, et al. A first course in turbulence [M]. MIT press, 1972.
- [3] Westerweel J, Boersma B J, Nieuwstadt F T. Turbulence: Introduction to theory and applications of turbulent flows [M]. Springer International Publishing, 2016.
- [4] Antiga L, Steinman D A. Rethinking turbulence in blood [J]. *Biorheology*, 2009, 46(2): 77-81.
- [5] Mariotte E, de La Hire P. *Traité du mouvement des eaux et des autres corps fluides* [M]. Chez Claude-Jombert, 1718.
- [6] Darrigol O. *Worlds of flow: A history of hydrodynamics from the bernoullis to prandtl* [M]. Oxford University Press, 2005.
- [7] Eckert M. *Turbulence—an odysssey* [J]. *History of Physics*, 2022.
- [8] Boussinesq J. *Essai sur la théorie des eaux courantes* [M]. Impr. nationale, 1877.
- [9] Maffioli C S. *Out of galileo* [J]. *The Science of Waters (1628-1718)*, Rotterdam, 1994.
- [10] Reynolds O. Xxix. an experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels [J]. *Philosophical Transactions of the Royal society of London*, 1883(174): 935-982.
- [11] Reynolds O. On the dynamical theory of incompressible viscous fluids and the determination of the criterion (reprinted from papers on mechanical and physical subjects, vol 2, pg 535-577, 1901) [C]//Proceedings of the Royal Society-Mathematical and Physical Sciences: volume 451. ROYAL SOC 6-9 CARLTON HOUSE TERRACE, LONDON SW1Y 5AG, ENGLAND, 1995: 5-47.
- [12] Prandtl L. über flüssigkeitsbewegung bei sehr kleiner reibung, verh 3 int [J]. *Math-Kongr*, Heidelberg, 1904: 3.
- [13] Blasius H. *Grenzschichten in flüssigkeiten mit kleiner reibung* [M]. Druck von BG Teubner, 1907.
- [14] Von Karman T. *Turbulence and skin friction* [J]. *Journal of the Aeronautical Sciences*, 1934, 1(1): 1-20.
- [15] Karman T V. The fundamentals of the statistical theory of turbulence [J]. *Journal of the Aeronautical Sciences*, 1937, 4(4): 131-138.
- [16] Prandtl L, Tollmien W. Die windverteilung über dem erdboden, errechnet aus den gesetzen der rohrströmung [J]. *Z. Geophysik*, 1924, 1: 47-55.
- [17] Sagaut P, Terracol M, Deck S. Multiscale and multiresolution approaches in turbulence-les, des and hybrid rans/les methods: Applications and guidelines [M]. World Scientific, 2013.
- [18] Aoki K, Masumoto Y. Interpreting reynolds stress from perspective of eddy geometry [J]. *Dynamics of Atmospheres and Oceans*, 2021, 94: 101223.

- [19] Taylor G I. Diffusion by continuous movements [J]. Proceedings of the london mathematical society, 1922, 2(1): 196-212.
- [20] Taylor G I. Statistical theory of turbulence-ii [J]. Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences, 1935, 151(873): 444-454.
- [21] De Karman T, Howarth L. On the statistical theory of isotropic turbulence [J]. Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences, 1938, 164 (917): 192-215.
- [22] Simmons L, Salter C. Experimental investigation and analysis of the velocity variations in turbulent flow [J]. Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 1934, 145(854): 212-234.
- [23] Reichardt H. Die quadratischen mittelwerte der längsschwankungen in der turbulenten kanalströmung [J]. Zeitschrift für Angewandte Mathematik und Mechanik (ZAMM), 1933, 3: 177-180.
- [24] Dryden H L. Reduction of turbulence in wind tunnels [R]. 1931.
- [25] Dryden H L. Air flow in the boundary layer near a plate [R]. 1937.
- [26] Kolmogorov A N. The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers [J]. Cr Acad. Sci. URSS, 1941, 30: 301-305.
- [27] Yaglom A. An kolmogorov as a fluid mechanician and founder of a school in turbulence research [J]. Annual review of fluid mechanics, 1994, 26(1): 1-23.
- [28] Kline S J, Reynolds W C, Schraub F, et al. The structure of turbulent boundary layers [J]. Journal of Fluid Mechanics, 1967, 30(4): 741-773.
- [29] Corino E R, Brodkey R S. A visual investigation of the wall region in turbulent flow [J]. Journal of Fluid Mechanics, 1969, 37(1): 1-30.
- [30] Kim H, Kline S, Reynolds W. The production of turbulence near a smooth wall in a turbulent boundary layer [J]. Journal of Fluid Mechanics, 1971, 50(1): 133-160.
- [31] Cantwell B J. Organized motion in turbulent flow [J]. Annual review of fluid mechanics, 1981, 13(1): 457-515.
- [32] Robinson S K. Coherent motions in the turbulent boundary layer [J]. Annual review of fluid mechanics, 1991, 23(1): 601-639.
- [33] Wilcox D C, et al. Turbulence modeling for cfd: volume 2 [M]. DCW industries La Canada, CA, 1998.
- [34] Reed H, Saric W. Transition mechanisms for transport aircraft [C]//38th fluid dynamics conference and exhibit. 2008: 3743.
- [35] 傅德薰. 可压缩湍流直接数值模拟 [M]. 2010.
- [36] Morkovin M V. Effects of compressibility on turbulent flows [J]. Mécanique de la Turbulence, 1962, 367(380): 26.
- [37] Van Driest E R. Turbulent boundary layer in compressible fluids [J]. Journal of the Aeronautical Sciences, 1951, 18(3): 145-160.
- [38] Coleman G N, Kim J, Moser R D. A numerical study of turbulent supersonic isothermal-wall channel flow [J]. Journal of Fluid Mechanics, 1995, 305: 159-183.
- [39] Duan L, Beekman I, Martin M. Direct numerical simulation of hypersonic turbulent boundary layers. part 2. effect of wall temperature [J]. Journal of Fluid Mechanics, 2010, 655: 419-445.
- [40] Bradshaw P. Compressible turbulent shear layers [J]. Annual Review of Fluid Mechanics, 1977, 9(1): 33-52.

-
- [41] Duan L, Beekman I, Martin M. Direct numerical simulation of hypersonic turbulent boundary layers. part 3. effect of mach number [J]. *Journal of Fluid Mechanics*, 2011, 672: 245-267.
 - [42] Lagha M, Kim J, Eldredge J, et al. A numerical study of compressible turbulent boundary layers [J]. *Physics of fluids*, 2011, 23(1): 015106.
 - [43] Cebeci T. Analysis of turbulent boundary layers [M]. Elsevier, 2012.
 - [44] Huang P, Coleman G, Bradshaw P. Compressible turbulent channel flows: Dns results and modelling [J]. *Journal of Fluid Mechanics*, 1995, 305: 185-218.
 - [45] Gaviglio J. Reynolds analogies and experimental study of heat transfer in the supersonic boundary layer [J]. *International journal of heat and mass transfer*, 1987, 30(5): 911-926.
 - [46] Rubesin M W. Extra compressibility terms for favre-averaged two-equation models of inhomogeneous turbulent flows [R]. 1990.
 - [47] Zhang Y S, Bi W T, Hussain F, et al. A generalized reynolds analogy for compressible wall-bounded turbulent flows [J]. *Journal of Fluid Mechanics*, 2014, 739: 392-420.
 - [48] Hamilton J M, Kim J, Waleffe F. Regeneration mechanisms of near-wall turbulence structures [J]. *Journal of Fluid Mechanics*, 1995, 287: 317-348.
 - [49] Jiménez J, Pinelli A. The autonomous cycle of near-wall turbulence [J]. *Journal of Fluid Mechanics*, 1999, 389: 335-359.
 - [50] Hussain A F. Coherent structures and turbulence [J]. *Journal of Fluid Mechanics*, 1986, 173: 303-356.
 - [51] Panton R L. Overview of the self-sustaining mechanisms of wall turbulence [J]. *Progress in Aerospace Sciences*, 2001, 37(4): 341-383.
 - [52] Smits A, Spina E, Alving A, et al. A comparison of the turbulence structure of subsonic and supersonic boundary layers [J]. *Physics of Fluids A: Fluid Dynamics*, 1989, 1(11): 1865-1875.
 - [53] Smits A J, Dussauge J P. Turbulent shear layers in supersonic flow [M]. Springer Science & Business Media, 2006.
 - [54] Ganapathisubramani B, Clemens N, Dolling D. Large-scale motions in a supersonic turbulent boundary layer [J]. *Journal of fluid Mechanics*, 2006, 556: 271-282.
 - [55] Pirozzoli S, Bernardini M. Turbulence in supersonic boundary layers at moderate reynolds number [J]. *Journal of Fluid Mechanics*, 2011, 688: 120-168.
 - [56] Bross M, Scharnowski S, Kähler C J. Large-scale coherent structures in compressible turbulent boundary layers [J]. *Journal of Fluid Mechanics*, 2021, 911.
 - [57] Pope S B, Pope S B. Turbulent flows [M]. Cambridge university press, 2000.
 - [58] Moin P, Mahesh K. Direct numerical simulation: a tool in turbulence research [J]. *Annual review of fluid mechanics*, 1998, 30(1): 539-578.
 - [59] Orszag S A, Patterson Jr G. Numerical simulation of three-dimensional homogeneous isotropic turbulence [J]. *Physical review letters*, 1972, 28(2): 76.
 - [60] Rogallo R S. Numerical experiments in homogeneous turbulence: volume 81315 [M]. National Aeronautics and Space Administration, 1981.
 - [61] Moin P, Kim J. Numerical investigation of turbulent channel flow [J]. *Journal of fluid mechanics*, 1982, 118: 341-377.
 - [62] Kim J, Moin P, Moser R. Turbulence statistics in fully developed channel flow at low reynolds number [J]. *Journal of fluid mechanics*, 1987, 177: 133-166.
 - [63] Spalart P R. Direct simulation of a turbulent boundary layer up to $r\theta = 1410$ [J]. *Journal of fluid mechanics*, 1988, 187: 61-98.

- [64] Guo Y, Adams N A. Numerical investigation of supersonic turbulent boundary layers with high wall temperature [J]. Stanford Univ., Studying Turbulence Using Numerical Simulation Databases. 5: Proceedings of the 1994 Summer Program, 1994.
- [65] Rai M, Gatski T, Erlebacher G. Direct simulation of spatially evolving compressible turbulent boundary layers [C]//33rd Aerospace Sciences Meeting and Exhibit. 1995: 583.
- [66] Guarini S E, Moser R D, Shariff K, et al. Direct numerical simulation of a supersonic turbulent boundary layer at mach 2.5 [J]. Journal of Fluid Mechanics, 2000, 414: 1-33.
- [67] Duan L, Martin M. Direct numerical simulation of hypersonic turbulent boundary layers. part 4. effect of high enthalpy [J]. Journal of Fluid Mechanics, 2011, 684: 25-59.
- [68] Liang X, Li X. Dns of a spatially evolving hypersonic turbulent boundary layer at mach 8 [J]. Science China Physics, Mechanics and Astronomy, 2013, 56(7): 1408-1418.
- [69] Xin-Liang L, De-Xun F, Yan-Wen M. Direct numerical simulation of a spatially evolving supersonic turbulent boundary layer at $ma = 6$ [J]. Chinese Physics Letters, 2006, 23(6): 1519.
- [70] Xin-Liang L, De-Xun F, Yan-Wen M, et al. Acoustic calculation for supersonic turbulent boundary layer flow [J]. Chinese Physics Letters, 2009, 26(9): 094701.
- [71] Pirozzoli S, Grasso F, Gatski T. Direct numerical simulation and analysis of a spatially evolving supersonic turbulent boundary layer at $m = 2.25$ [J]. Physics of fluids, 2004, 16(3): 530-545.
- [72] Pirozzoli S, Bernardini M, Grasso F. Characterization of coherent vortical structures in a supersonic turbulent boundary layer [J]. Journal of Fluid Mechanics, 2008, 613: 205-231.
- [73] Pirozzoli S, Bernardini M, Grasso F. On the dynamical relevance of coherent vortical structures in turbulent boundary layers [J]. Journal of fluid mechanics, 2010, 648: 325-349.
- [74] Lesieur M, Métais O, Comte P, et al. Large-eddy simulations of turbulence [M]. Cambridge university press, 2005.
- [75] Spalart P R. Strategies for turbulence modelling and simulations [J]. International journal of heat and fluid flow, 2000, 21(3): 252-263.
- [76] Sood A, Forster R A, Archer B J, et al. Neutronics calculation advances at los alamos: Manhattan project to monte carlo [J]. Nuclear Technology, 2021, 207(sup1): S100-S133.
- [77] McCartney S. Eniac: The triumphs and tragedies of the world's first computer [J]. 1999.
- [78] Thornton J E. The cdc 6600 project [J]. Annals of the History of Computing, 1980, 2(4): 338-348.
- [79] Blankenbaker J V. Kenbak-1 digital computer [J]. The Journal of Data Education, 1973, 13 (4): 7-8.
- [80] Ashour M B A, Gee S J, Hammock B D. Use of a 96-well microplate reader for measuring routine enzyme activities [J]. Analytical biochemistry, 1987, 166(2): 353-360.
- [81] Warren M S, Salmon J K, Becker D J, et al. Pentium pro inside: I. a treecode at 430 gigaflops on asci red, ii. price/performance of 50\$/mflop on loki and hyglac [C]//SC'97: Proceedings of the 1997 ACM/IEEE Conference on Supercomputing. IEEE, 1997: 61-61.
- [82] Bader D A, Maccabe A B, Mastaler J R, et al. Design and analysis of the alliance/university of new mexico roadrunner linux smp supercluster [C]//ICWC 99. IEEE Computer Society International Workshop on Cluster Computing. IEEE, 1999: 9-18.
- [83] Schneider D. The exascale era is upon us: The frontier supercomputer may be the first to reach 1,000,000,000,000,000 operations per second [J]. IEEE Spectrum, 2022, 59(1): 34-35.

- [84] 中华人民共和国中央人民政府网站 [EB/OL]. 2018. http://www.gov.cn/xinwen/2018-10/22/content_5333505.htm.
- [85] Khokhar A A, Prasanna V K, Shaaban M E, et al. Heterogeneous computing: Challenges and opportunities [J]. Computer, 1993, 26(6): 18-27.
- [86] Mittal S, Vetter J S. A survey of cpu-gpu heterogeneous computing techniques [J]. ACM Computing Surveys (CSUR), 2015, 47(4): 1-35.
- [87] Brodtkorb A R, Dyken C, Hagen T R, et al. State-of-the-art in heterogeneous computing [J]. Scientific Programming, 2010, 18(1): 1-33.
- [88] Siegel H J, Antonio J K, Metzger R C, et al. Heterogeneous computing [J]. ECE Technical Reports, 1994: 206.
- [89] The green 500 [EB/OL]. <https://www.top500.org/lists/green500/2022/06/>.
- [90] The top 500 [EB/OL]. <https://www.top500.org/>.
- [91] Xin-Liang L I, Yan-Wen M A, De-Xun F U. Aliasing error analysis of the upwind compact difference method and comparison with the spectral method [J]. Chinese Journal of Computation Physics, 2002.
- [92] Steger J L, Warming R. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods [J]. Journal of computational physics, 1981, 40(2): 263-293.
- [93] Zhang H, Zhuang F. Nnd schemes and their applications to numerical simulation of two-and three-dimensional flows [J]. Advances in Applied Mechanics, 1991, 29: 193-256.
- [94] Li X l, Leng Y, He Z w. Optimized sixth-order monotonicity-preserving scheme by nonlinear spectral analysis [J]. International Journal for Numerical Methods in Fluids, 2013, 73(6): 560-577.
- [95] Jiang G S, Shu C W. Efficient implementation of weighted eno schemes [J]. Journal of computational physics, 1996, 126(1): 202-228.
- [96] Martín M P, Taylor E M, Wu M, et al. A bandwidth-optimized weno scheme for the effective direct numerical simulation of compressible turbulence [J]. Journal of Computational Physics, 2006, 220(1): 270-289.
- [97] 郑敏, 张涵信. 无波动, 无自由参数的耗散差分格式 (NND) 在喷流计算中的应用 [J]. 空气动力学学报, 1989, 7(3): 273-281.
- [98] Liu X D, Osher S, Chan T. Weighted essentially non-oscillatory schemes [J]. Journal of computational physics, 1994, 115(1): 200-212.
- [99] Levy D, Puppo G, Russo G. Central weno schemes for hyperbolic systems of conservation laws [J]. ESAIM: Mathematical Modelling and Numerical Analysis, 1999, 33(3): 547-571.
- [100] Borges R, Carmona M, Costa B, et al. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws [J]. Journal of Computational Physics, 2008, 227(6): 3191-3211.
- [101] Henrick A K, Aslam T D, Powers J M. Mapped weighted essentially non-oscillatory schemes: achieving optimal order near critical points [J]. Journal of Computational Physics, 2005, 207(2): 542-567.
- [102] Hu X, Wang Q, Adams N A. An adaptive central-upwind weighted essentially non-oscillatory scheme [J]. Journal of Computational Physics, 2010, 229(23): 8952-8965.
- [103] Li G, Qiu J. Hybrid weighted essentially non-oscillatory schemes with different indicators [J]. Journal of Computational Physics, 2010, 229(21): 8105-8129.

- [104] Ren Y X, Zhang H, et al. A characteristic-wise hybrid compact-weno scheme for solving hyperbolic conservation laws [J]. Journal of Computational Physics, 2003, 192(2): 365-386.
- [105] Sun Z, Ren Y X. A characteristic-wise hybrid compact-weno scheme for solving the navier-stokes equations on curvilinear coordinates [M]//Computational Fluid Dynamics 2008. Springer, 2009: 437-442.
- [106] Suresh A, Huynh H. Accurate monotonicity-preserving schemes with runge–kutta time stepping [J]. Journal of Computational Physics, 1997, 136(1): 83-99.
- [107] He Z, Li X, Fu D, et al. A 5th order monotonicity-preserving upwind compact difference scheme [J]. Science China Physics, Mechanics and Astronomy, 2011, 54(3): 511-522.
- [108] Kang J, Li X. On the ideal weights for weno/weno-like finite difference schemes for the first derivative, i [J]. International Journal of Modern Physics C, 2020, 31(07): 2050099.
- [109] Harten A, Zwas G. Self-adjusting hybrid schemes for shock computations [J]. Journal of Computational Physics, 1972, 9(3): 568-583.
- [110] Adams N A, Shariff K. A high-resolution hybrid compact-eno scheme for shock-turbulence interaction problems [J]. Journal of Computational Physics, 1996, 127(1): 27-51.
- [111] Pirozzoli S. Conservative hybrid compact-weno schemes for shock-turbulence interaction [J]. Journal of Computational Physics, 2002, 178(1): 81-117.
- [112] Sun Z S, Ren Y X, Larricq C, et al. A class of finite difference schemes with low dispersion and controllable dissipation for dns of compressible turbulence [J]. Journal of computational physics, 2011, 230(12): 4616-4635.
- [113] Zhang L, Wei L, Lixin H, et al. A class of hybrid dg/fv methods for conservation laws i: Basic formulation and one-dimensional systems [J]. Journal of Computational Physics, 2012, 231 (4): 1081-1103.
- [114] Sun Z s, Luo L, Ren Y x, et al. A sixth order hybrid finite difference scheme based on the minimized dispersion and controllable dissipation technique [J]. Journal of Computational Physics, 2014, 270: 238-254.
- [115] He Z, Li X, Liang X. Nonlinear spectral-like schemes for hybrid schemes [J]. Science China Physics, Mechanics and Astronomy, 2014, 57(4): 753-763.
- [116] Zhiwei H E, Xinliang L I, Liang X. Hybrid schemes for shock wave-turbulent boundary layer interaction problems [J]. Science China Physics Mechanics & Astronomy, 2013.
- [117] Zhao Z, Chen Y, Qiu J. A hybrid hermite weno scheme for hyperbolic conservation laws [J]. Journal of Computational Physics, 2020, 405: 109175.
- [118] Jameson A, Schmidt W, Turkel E. Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes [C]//14th fluid and plasma dynamics conference. 1981: 1259.
- [119] 李新亮. OpenCFD-SC 用户手册 [M]. 2016.
- [120] Shu C W, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes [J]. Journal of computational physics, 1988, 77(2): 439-471.
- [121] Amd [EB/OL]. 2022. https://en.wikipedia.org/wiki/List_of_AMD_graphics_processing_units#Radeon_Vega_series.
- [122] Hip [EB/OL]. 2022. https://docs.amd.com/bundle/HIP-Programming-Guide-v4.5/page/Transitioning_from_CUDA_to_HIP.html.
- [123] Li X, Fu D, Ma Y. Direct numerical simulation of passive scalar in decaying compressible turbulence [J]. Science in China Series G: Physics, Mechanics and Astronomy, 2004, 47(1): 52-63.

- [124] Li X, Fu D, Ma Y. Direct numerical simulation of hypersonic boundary layer transition over a blunt cone with a small angle of attack [J]. Physics of Fluids, 2010, 22(2): 025105.
- [125] Li X, Fu D, Ma Y, et al. Direct numerical simulation of shock/turbulent boundary layer interaction in a supersonic compression ramp [J]. Science China Physics, Mechanics and Astronomy, 2010, 53(9): 1651-1658.
- [126] Bookey P, Wyckham C, Smits A, et al. New experimental data of stbli at dns/les accessible reynolds numbers [C]//43rd AIAA Aerospace Sciences Meeting and Exhibit. 2005: 309.
- [127] Chen X, Dong S, Tu G, et al. Boundary layer transition and linear modal instabilities of hypersonic flow over a lifting body [J]. Journal of Fluid Mechanics, 2022, 938.
- [128] Liu S, Yuan X, Liu Z, et al. Design and transition characteristics of a standard model for hypersonic boundary layer transition research [J]. Acta Mechanica Sinica, 2021, 37(11): 1637-1647.
- [129] Qi H, Li X, Yu C, et al. Direct numerical simulation of hypersonic boundary layer transition over a lifting-body model hytrv [J]. Advances in Aerodynamics, 2021, 3(1): 1-21.
- [130] Adamson Jr T, Messiter A. Analysis of two-dimensional interactions between shock waves and boundary layers [J]. Annual review of fluid mechanics, 1980, 12(1): 103-138.
- [131] Andreopoulos Y, Agui J H, Briassulis G. Shock wave—turbulence interactions [J]. Annual Review of Fluid Mechanics, 2000, 32(1): 309-345.
- [132] Clemens N T, Narayanaswamy V. Low-frequency unsteadiness of shock wave/turbulent boundary layer interactions [J]. Annual Review of Fluid Mechanics, 2014, 46: 469-492.
- [133] Erengil M E, Dolling D S. Unsteady wave structure near separation in a mach 5 compression rampinteraction [J]. AIAA journal, 1991, 29(5): 728-735.
- [134] Ringuette M, Smits A. Wall-pressure measurements in a mach 3 shock-wave turbulent boundary layer interaction at a dns accessible reynolds number [C]//37th AIAA Fluid Dynamics Conference and Exhibit. 2007: 4113.
- [135] Narayanaswamy V, Raja L L, Clemens N T. Control of unsteadiness of a shock wave/turbulent boundary layer interaction by using a pulsed-plasma-jet actuator [J]. Physics of Fluids, 2012, 24(7): 076101.
- [136] Wu M, Martin M P. Analysis of shock motion in shockwave and turbulent boundary layer interaction using direct numerical simulation data [J]. Journal of Fluid Mechanics, 2008, 594: 71-83.
- [137] Porter K M, Poggie J. Selective upstream influence on the unsteadiness of a separated turbulent compression ramp flow [J]. Physics of Fluids, 2019, 31(1): 016104.
- [138] Deshpande A S, Poggie J. Unsteadiness of shock-wave/boundary-layer interaction with side-walls [C]//AIAA Scitech 2020 Forum. 2020: 0581.
- [139] Priebe S, Martín M P. Turbulence in a hypersonic compression ramp flow [J]. Physical Review Fluids, 2021, 6(3): 034601.
- [140] Spalart P R. Numerical study of sink-flow boundary layers [J]. Journal of Fluid Mechanics, 1986, 172: 307-328.
- [141] Zhang C, Duan L, Choudhari M M. Direct numerical simulation database for supersonic and hypersonic turbulent boundary layers [J]. AIAA journal, 2018, 56(11): 4297-4311.
- [142] Kara K, Balakumar P, Kandil O. Effects of nose bluntness on stability of hypersonic boundary layers over a blunt cone [C]//37th AIAA Fluid Dynamics Conference and Exhibit. 2007: 4492.

- [143] Li X, Fu D, Ma Y. Dns of compressible turbulent boundary layer around a sharp cone [J]. Science in China Series G: Physics, Mechanics and Astronomy, 2008, 51(6): 699-714.
- [144] Huang J, Duan L, Casper K M, et al. Direct numerical simulation of turbulent pressure fluctuations over a cone at mach 8 [C]//AIAA Scitech 2020 Forum. 2020: 1065.
- [145] Li X, Fu D, Ma Y. Direct numerical simulation of hypersonic boundary layer transition over a blunt cone [J]. AIAA journal, 2008, 46(11): 2899-2913.
- [146] Li X, Fu D, Ma Y. Direct numerical simulation of hypersonic boundary layer transition over a blunt cone with a small angle of attack [J]. Physics of Fluids, 2010, 22(2): 025105.
- [147] Liu Y, Schuabb M, Duan L, et al. Interaction of a tunnel-like acoustic disturbance field with a blunt cone boundary layer at mach 8 [C]//AIAA AVIATION 2022 Forum. 2022: 3250.

附录

.1 OpenCFD-SCU 控制文件与参数说明

.1.1 OpenCFD-SCU 参数定义模块

- OpenCFD-SCU 中所有可输入的参数都定义在变量类型为 configItem 的结构体数组 configList 中。
- 每个控制参数即为 configList 数组中的一项。

configItem 类型定义如下：

```
typedef struct configItem_
{
    char name[1000]; //变量名
    char value[1000]; //变量值
} configItem;
```

OpenCFD-SCU 目前主计算部分可输入的控制参数为 28 个：

```
configItem configList[27] = {
    {"GRID_3D", 0},           //x、y、z 三个方向的网格数
    {"PARALLEL_3D", 0},       //x、y、z 三个方向并行分割数目
    {"LAP", 0},               //重叠区宽度
    {"MSG_BLOCK_SIZE", 0},     //通讯方式
    {"STREAM", 0},             //计算通讯重叠
    {"TEST", 0},               //是否开启节点测试
    {"IPERIODIC", 0},          //周期性边界条件
    {"JAC_BOUND", 0},           //计算 Jacobian 时的边界格式
    {"DIF_BOUND", 0},           //差分计算时的边界格式
    {"NON_REFLETION", 0},        //无反射边界条件
    {"SCHEME_INVIS", 0},          //无粘项计算格式
    {"SCHEME_VIS", 0},            //粘性项计算格式
    {"RE", 0},                  //雷诺数
    {"AMA", 0},                  //马赫数
    {"GAMMA", 0},                //气体绝热指数
    {"PR", 0},                   //普朗特数
    {"T_REF", 0},                 //无量纲参考温度
    {"EPSL_SW", 0},                //S-W 分裂中的稳定系数
    {"DT", 0},                   //时间步长
};
```

```

{ "END_TIME", 0},           //计算结束时间
{ "KSTEP_SHOW", 0},         //显示步数
{ "KSTEP_SAVE", 0},         //流场保存步数
{ "INIT_STAT", 0},          //流场初始化
{ "IBC", 0},                //边界类型
{ "BC_NPARA", 0},           //边界参数（整型变量）
{ "BC_RPARA", 0},           //边界参数（浮点型变量）
{ "CHARTERIC", 0}           //特征重构
};


```

滤波模块部分可输入的控制参数为 2 个：

```

FILTER_NPARA&           //滤波参数（整型变量）
FILTER_RPARA&             //滤波参数（浮点型变量）

//此处 & 代指从 0 开始的自然数，若程序调用一个滤波，控制参数名只有 FILTER_NPARAO、
// FILTER_RPARAO，若程序同时调用多个滤波，则参数名应为：
// FILTER_NPARAO、FILTER_RPARAO、FILTER_NPARA1、FILTER_RPARA1, ...


```

流场分析与后处理部分可输入的控制参数为 3 个：

```

ANA_EVENT&                //分析事件
ANA_NPARA&                 //事件参数（整型变量）
ANA_RPARA&                  //事件参数（浮点型变量）

//此处 & 代指从 0 开始的自然数，若程序调用一个分析，控制参数名只有 ANA_EVENT0、
// ANA_NPARAO、ANA_RPARAO，若程序同时调用多个分析，则参数名应为 ANA_EVENT0、
// ANA_NPARAO、ANA_RPARAO、ANA_EVENT1、ANA_NPARA1、ANA_RPARA1, ...

//若分析的事件没有控制参数，则不填写对应的 ANA_NPARA 和 ANA_RPARA


```

混合格式相关参数为 5 个：

```

HY_STYLE                   //混合格式类型
HY_DP_INTV                 //阈值
HY_SMOOTH_DP               //光滑性处理
HY_PATCH_ZONE               //指定判别值的范围个数
HY_ZONE&                   //指定判别值的范围与指定的判别值


```

.1.2 OpenCFD-SCU 参数读取模块

- OpenCFD-SCU 通过遍历文本 opencfd-scu.in 获取以上定义的参数值。
- 在并行计算时，只有 my_id 为 0 的进程读取文本 opencfd-scu.in 获取参数值，后广播到其它进程。

参数读取模块代码如下：

```

1      FILE * file = fopen("opencfd-scu.in","r"); // 打开文
2     件opencfd-scu.in
3      int nk, nr;
4
5      if(file == NULL){
6          printf("\033[31mopencfd-scu.in is not find
7          !\033[0m\n");
8          exit(-1);
9      } // 如果没有发现文件opencfd-scu.in则退出
10
11     SearchItem(file, configList, configNum); // 遍历文件
12     opencfd-scu.in，找到控制文件中参数名对应的参数值
13
14     sscanf(configList[0].value,"%d%d%d",&NX_GLOBAL,&
15     NY_GLOBAL,&NZ_GLOBAL); // 将网格数参数GRID_3D的值写入
16     NX_GLOBAL,NY_GLOBAL,NZ_GLOBAL
17     sscanf(configList[1].value,"%d%d%d",&NPX0,&NPY0,&
18     NPZ0); // 将并行分割块数对应的参数PARALLEL_3D的值写入
19     NPX0,NPY0,NPZ0
20     sscanf(configList[2].value,"%d",&dummy_i); // 目前版
21     本程序LAP参数程序默认值为4，即使读取LAP值也不生效，
22     并将值写入到变量dummy_i中
23     sscanf(configList[3].value,"%d",&MSG_BLOCK_SIZE);
24     // 读取通讯方式，目前程序只有阻塞式MPI通讯一种，即使
25     读取MSG_BLOCK_SIZE值也不生效，并将值写入到无效变量
26     MSG_BLOCK_SIZE中
27     sscanf(configList[4].value,"%d",&Stream_MODE); // 读
28     取控制计算通讯重叠的参数STREAM，并将值写入到变量
29     Stream_MODE中
30     sscanf(configList[5].value,"%d",&TEST); // 读取控
31     制节点测试的参数TEST，并将值写入到变量TEST中
32
33     sscanf(configList[6].value,"%d%d%d",&Iperiodic
34     [0],&Iperiodic[1],&Iperiodic[2]); // 读取控制周期性边
35     条的参数IPERIODIC，并将值写入到数组变量Iperiodic[3]
36     中

```

```

    sscanf(configList[7].value, "%d%d%d", &Jacbound[0], &
Jacbound[1], &Jacbound[2]); // 读取控制 Jacob 计算边界格
式的参数 JAC_BOUND，并将值写入到数组变量 Jacbound[3]
中
21   sscanf(configList[8].value, "%d%d%d%d%d", &
D0_bound[0], &D0_bound[1], &D0_bound[2], &D0_bound
[3], &D0_bound[4], &D0_bound[5]); // 读取控制差分计算边
界格式的参数 DIF_BOUND，并将值写入到数组变量 D0_bound
[6] 中
    sscanf(configList[9].value, "%d%d%d%d%d", &Non_ref
[0], &Non_ref[1], &Non_ref[2], &Non_ref[3], &Non_ref
[4], &Non_ref[5]); // 读取控制无反射边界条件的参数
NON_REFLETION，并将值写入到数组变量 Non_ref[6] 中
23
    sscanf(configList[10].value, "%s", Scheme_invis); // 读
取控制无粘项计算格式的参数 SCHEME_INVIS，并将值写
入到变量 Scheme_invis 中
25   sscanf(configList[11].value, "%s", Scheme_vis); // 读
取控制粘性项计算格式的参数 SCHEME_VIS，并将值写入到
变量 Scheme_vis 中
    SCHEME_CHOOSE scheme = {Scheme_invis, Scheme_vis};
// 将格式信息传递到结构体变量 scheme 中
27   Schemes_Chose_ID(&scheme); // 判断选择的格式与对应
的编号

29   if(strcmp(Scheme_invis, "SCHEME_HYBRIDAUTO") == 0)
    IFLAG_HybridAuto = 1; // 判断是否是混合格式

31   HybridAuto.Num_Patch_zones = 0;
    HybridAuto.IF_Smooth_dp = 0;
33
    HybridAuto.P_intvs = (REAL *) malloc((HybridA_Stage
- 1)* sizeof(REAL));
35   HybridAuto.zones = (int *) malloc(6*Patch_max*
sizeof(int));
    HybridAuto.Pa_zones = (REAL *) malloc(Patch_max*
sizeof(REAL));
37
    if(IFLAG_HybridAuto == 1){ // 如何启用混合格式则读取

```

以下对应参数

```

39          int (*HybridAuto_zones)[6] = (int(*)[6])
        HybridAuto.zones;

41          configItem Hybridbuff = {"HY_DP_INTV", 0};
43          SearchItem(file, &Hybridbuff, 1);

45          tmp = PartItem(Hybridbuff.value, Part_buff);
        for(int i=0;i<(HybridA_Stage-1);i++) sscanf(
        Part_buff[i], "%f", &HybridAuto.P_intvs[i]); // 读取控
        制混合格式阈值的参数HY_STYLE，对应变量写入到
        HybridAuto.Style中，目前默认HybridA_Stage为3，即只
        有2个阈值

47          sprintf(Hybridbuff.name, "HY_STYLE");
49          SearchItem(file, &Hybridbuff, 1);
        sscanf(Hybridbuff.value, "%d", &HybridAuto.Style
        ); // 读取控制混合类型的参数HY_STYLE，对应值写入到
        HybridAuto.Style中

51          if(HybridAuto.Style != 1 && HybridAuto.Style
        != 2){
            printf("\033[31mHYBRID SCHEMES CHOOSE IS
        WRONG! ! ! \033[0m\n");
            exit(0);
        }

57          sprintf(Hybridbuff.name, "HY_SMOOTH_DP");
        SearchItem(file, &Hybridbuff, 1);
        sscanf(Hybridbuff.value, "%d", &HybridAuto.
        IF_Smooth_dp); // 读取是否做光滑性处理的参数
        HY_SMOOTH_DP，对应值写入到HybridAuto.IF_Smooth_dp中

61          sprintf(Hybridbuff.name, "HY_PATCH_ZONE");
        SearchItem(file, &Hybridbuff, 1);
        sscanf(Hybridbuff.value, "%d", &HybridAuto.
        Num_Patch_zones); // 读取patch区数量的变量

```

```

65         for (int i=0; i<HybridAuto.Num_Patch_zones; i
66             ++){
67             sprintf(Hybridbuff.name, "HY_ZONE%d", i);
68             SearchItem(file, &Hybridbuff, 1);
69
70             sscanf(Hybridbuff.value, "%d%d%d%d%d%lf"
71                 ,&HybridAuto_zones[i][0],&HybridAuto_zones[i][1],&
72                 HybridAuto_zones[i][2],
73                 &HybridAuto_zones[i][3],&HybridAuto_zones[
74                 i][4],&HybridAuto_zones[i][5],&HybridAuto.Pa_zones[
75                 i]); // 读取 patch 区对应的区域，将参数 HY_ZONE& 的值写入
76                 到数组 HybridAuto_zones 对应的行和列中，行表示 patch 区
77                 域起止的范围，列表示第几个 patch 区。
78         }
79     }
80
81     sscanf(configList[12].value, "%lf", &Re); // 读取控制
82     雷诺数的参数 RE，并将值写入到变量 Re 中
83     sscanf(configList[13].value, "%lf", &Ama); // 读取控制
84     马赫数的参数 AMA，并将值写入到变量 Ama 中
85     sscanf(configList[14].value, "%lf", &Gamma); // 读取控
86     制气体绝热指数的参数 GAMMA，并将值写入到变量 Gamma 中
87     sscanf(configList[15].value, "%lf", &Pr); // 读取控
88     制普朗特数的参数 PR，并将值写入到变量 Pr 中
89     sscanf(configList[16].value, "%lf", &Ref_T); // 读取控
90     制无量纲参考温度的参数 T_REF，并将值写入到变量 Ref_T
91     中
92     sscanf(configList[17].value, "%lf", &eps1_SW); // 读取控
93     制 S-W 分裂中的稳定系数的参数 EPSL_SW，并将值写入到
94     变量 epsl_SW 中
95
96     sscanf(configList[18].value, "%lf", &dt); // 读取控
97     制时间步长的参数 DT，并将值写入到变量 dt 中
98     sscanf(configList[19].value, "%lf", &end_time); // 读取
99     控制计算结束时间的参数 END_TIME，并将值写入到变量
100    end_time 中;
101    sscanf(configList[20].value, "%d", &Kstep_show); // 读
102    取控制显示步数的参数 KSTEP_SHOW，并将值写入到变量
103    Kstep_show 中

```

```

    sscanf(configList[21].value, "%d", &Kstep_save); // 读
    取控制流场保存步数的参数KSTEP_SHOW，并将值写入到变
    量Kstep_save中
85    sscanf(configList[22].value, "%d", &Init_stat); // 读
    取控制流场初始化的参数INIT_STAT，并将值写入到变量
    Init_stat中

87    sscanf(configList[23].value, "%d", &IBC_USER); // 读取
    控制边界类型的参数IBC，并将值写入到变量IBC_USER中
    BC_npara = (int*)malloc(sizeof(int)*100);
89    BC_rpara = (REAL*)malloc(sizeof(REAL)*100);

91    nk = PartItem(configList[24].value, Part_buff); // 统计边界参数（整形变量）BC_NPARA具有几个变量值，并
    写入到数组Part_buff中
    for(int i=0;i<nk;i++) sscanf(Part_buff[i], "%d",
    BC_npara+i); // 将数组Part_buff的值赋于数组BC_npara
93

94    nr = PartItem(configList[25].value, Part_buff); // 统计边界参数（浮点型变量）BC_RPARA具有几个变量值，并
    写入到数组Part_buff中
    for(int i=0;i<nr;i++) sscanf(Part_buff[i], "%lf",
    BC_rpara+i); // 将数组Part_buff的值赋于数组BC_rpara

97    sscanf(configList[26].value, "%d", &IF_CHARTERIC); // 读取控制特征重构的参数CHARTERIC，并将值写入到变量
    IF_CHARTERIC中

99    int NameNUM[1000];
    // 滤波功能的参数读取
101   NFiltering = ItemNUM(file, "FILTER_NPARA", &
    NameNUM[0]); // 判断调用滤波的次数

103   Filter_para = (int(*)[11])malloc(sizeof(int)*(NFiltering+1)*11);
    Filter_rpara = (REAL(*)[3])malloc(sizeof(REAL)*(NFiltering+1)*3);
105   for(int i=0;i<NFiltering;i++){

```

```

107     configItem Hybridbuff;
        // ntime , Filter_X , Filter_Y , Filter_Z , ib , ie ,
        jb , je , kb , ke , Filter_scheme
109     sprintf(Hybridbuff.name, "FILTER_NPARA%d",
NameNUM[ i ]);
        SearchItem(file , &Hybridbuff , 1);
111
111     tmp = PartItem(Hybridbuff.value , Part_buff);
113     for(int n=0;n<11;n++) sscanf(Part_buff[n],"%d"
,&Filter_para[ i ][n]); // 读取 FILTER_NPARA& (滤波整形
参数) 对应的值写入数组 Filter_para[&][11] 中
115
115     sprintf(Hybridbuff.name, "FILTER_RPARA%d",
NameNUM[ i ]);
        SearchItem(file , &Hybridbuff , 1);
117     tmp = PartItem(Hybridbuff.value , Part_buff);
        for(int n=0;n<3;n++) sscanf(Part_buff[n],"%lf"
,&Filter_rpara[ i ][n]); // 读取 FILTER_RPARA& (滤波浮点
型参数) 对应的值写入数组 Filter_rpara[&][3] 中
119 }

121 // 后处理和分析功能的参数读取
122 N_ana = ItemNUM(file , "ANA_EVENT" , &NameNUM[ 0 ]); // 判断开启了几个分析功能
123
123     ANA_npara = (int(*)[100]) malloc(sizeof(int)*100*
N_ana);
125     ANA_rpara = (REAL(*)[100]) malloc(sizeof(REAL)*100*
N_ana);
126     K_ana = (int*) malloc(sizeof(int)*N_ana);
127     Kstep_ana = (int*) malloc(sizeof(int)*N_ana);

129     for(int i=0;i<N_ana ; i++){
130         configItem Hybridbuff;
131         sprintf(Hybridbuff.name, "ANA_EVENT%d",
NameNUM[ i ]);
        SearchItem(file , &Hybridbuff , 1); // 按顺序查找
ANA_EVENT开头的变量，并获取值
133

```

```

    sscanf( Hybridbuff . value , "%d%d" , K_ana+i ,
Kstep_ana+i ); // 将分析类型与分析步数写入数组 K_ana 和
Kstep_ana

135
        sprintf( Hybridbuff . name , "ANA_NPARA%d" ,
NameNUM[ i ] );
137     SearchItem( file , &Hybridbuff , 1 ); // 查找分析所
需的整形参数ANA_NPARA&

139
        nk = PartItem( Hybridbuff . value , Part_buff ); // 
分析参数个数
        for ( int n=0;n<nk ;n++ ) sscanf( Part_buff [ n ] , "%d"
,&ANA_npara[ i ][ n ] ); // 将参数写入到数组ANA_npara 中
141
        sprintf( Hybridbuff . name , "ANA_RPARA%d" ,
NameNUM[ i ] );
143     SearchItem( file , &Hybridbuff , 1 ); // 查找分析所
需的浮点型参数ANA_RPARA

145
        nr = PartItem( Hybridbuff . value , Part_buff );
        for ( int n=0;n<nr ;n++ ) sscanf( Part_buff [ n ] , "%f"
,&ANA_rpara[ i ][ n ] ); // 将参数写入到数组ANA_rpara 中
147     }

149     fclose( file );

```

.1.3 控制文件参数填写注意事项与说明

- 为防止由于大小写容易混淆造成的错误，控制文件中所有参数均为**大写**，具体参数见参数定义模块说明。
- 参数名前后可加空格，但不可以加其它特殊字符（添加特殊字符会使此行参数无效）。
 - 参数填写没有顺序要求，只要在控制文件任意行中填写参数名与对应参数值，程序可自动查找识别。
 - 一行只能填一个参数名与对应值，其后加注释等其它文字不会造成错误。
 - 控制文件中参数赋值时，参数名与参数值以**等号**隔开，参数具有多个输入值时，输入值以**空格**隔开，空格个数无限制。
 - 对于滤波模块，程序支持计算时启用多个滤波。若程序只调用一个滤波，

滤波控制参数名应为 FILTER_NPARA0、FILTER_RPARA0，若程序同时调用多个滤波，则参数名末尾以自然数排列，如 FILTER_NPARA0、FILTER_RPARA0, FILTER_NPARA1、FILTER_RPARA1, FILTER_NPARA2、FILTER_RPARA2 …，程序可根据参数名自动判断调用了几个滤波。

- 对于流场分析与后处理模块，程序支持计算时启用多个分析与后处理。与滤波类似，参数名末尾以自然数排列，程序可自动识别调用了几个分析与后处理。如调用三个分析与后处理时，参数名应为 ANA_EVENT0、ANA_NPARA0、ANA_RPARA0, ANA_EVENT1、ANA_NPARA1、ANA_RPARA1, ANA_EVENT2、ANA_NPARA2、ANA_RPARA2。
- 混合格式相关参数与分析只有在混合格式启用时才生效。
- 控制文件目前可读入但无用的参数有（存入废变量中，以便后续扩展功能使用）：

MSG_BLOCK_SIZE

LAP

.1.4 OpenCFD-SCU 控制文件示例

```

GRID_3D = 8800 640 800
//此算例的网格规模为8800*640*800
PARALLEL_3D = 24 4 4
//并行分割方式为24*4*4，共用了384个进程
STREAM = 1
//启用了计算通讯重叠
CHARTERIC = 1
//启用了特征重构

IPERIODIC = 0 0 1
//z方向启用了周期性边界条件
JAC_BOUND = 1 1 1
//计算Jacobian系数时三个方向都使用了边界格式
DIF_BOUND = 1 1 1 1 0 0
//x, y的正负方向都使用了边界格式
NON_REFLETION = 0 1 0 1 0 0
//x+, y+方向使用了无反射边界条件

#SCHEME_INVIS = WENO7_SYMBO
SCHEME_VIS = CD8
//粘性项计算采用了8阶中心

```

```

SCHEME_INVIS = SCHEME_HYBRIDAUTO
//无粘性计算采用了混合格式
HY_STYLE = 2
//混合格式类型为2型
HY_DP_INTV = 0.02 0.1
//阈值为0.02和0.1
#HY_SMOOTH_DP = 1
#HY_PATCH_ZONE = 0
#HY_ZONE0 = 10 25 100 240 5 20 20.0

```

```

RE = 100000.0
//雷诺数为100000
AMA = 10.0
//马赫数为10.0
GAMMA = 1.40
//气体绝热指数为1.4
PR = 0.70
//普朗特数为0.7
T_REF = 79.0
//参考温度为79.0
EPSL_SW = 0.0
//sw分裂时没有添加增强稳定性的系数

```

```

DT = 0.002
//时间步长为0.002
END_TIME = 3000
//计算结束的无量纲时间为3000
KSTEP_SHOW = 1
//每一步进行一次流场显示，打印计算时间，平均温度、能量等信息
KSTEP_SAVE = 10000
//每10000步保存一次流场
INIT_STAT = 1
//读外部初始场和网格

```

```

IBC = 108
//边界条件选择为108
#mzmax, mtmax, Inlet_boundary, If_wall_not_normal
BC_NPARA = -10 1 1 0

```

//108边界条件对应整型参数

```
#Tw, epsl, x_dis_begin, x_dis_end, beta, x_wall_begin, x_up_bound_begin, SLZ  
BC_RPARA = 3.72 0.1 -320. -300 0.08 -1000. 1000. 24.
```

//108边界条件对应浮点型参数

```
#nstep_filter, IF_X, IF_Y, IF_Z, ib, ie, jb, je, kb, ke, Filter_scheme  
#FILTER_NPARA0 = 10 1 0 0 700 900 300 500 0 800 2  
#FILTER_RPARA0 = 1.0 1.e-6 310000
```

ANA_EVENT0 = 107 100

ANA_EVENT1 = 101 1

ANA_EVENT2 = 105 100

ANA_NPARA2 = 2 5 3000 5000

//启用了三个流场分析和后处理事件，第三个事件有需要输入的整型参数

表 1 OpenCFD-SCU 控制文件中基础参数说明

Table 1 Description of basic parameters in OpenCFD-SCU control file

	参数说明	参数值个数
GRID_3D	三个方向的网格数	3
PARALLEL_3D	三个方向的并行分割数目 (三维剖分时, 应保证每个进程的数据块三个维度网格数目接近)	3
	1.00.28 版本程序当单进程网格规模 设置为 280*280*280 时 , DCU 显存 (16G) 占比 70%	
LAP	重叠区宽度 (此参数目前版本不生效, 重叠区默认为 4)	1
MSG_BLOCK_SIZE	MPI 通讯类型 (此参数目前版本不生效, 只包含一种 MPI 通讯方式—阻塞式)	1
STREAM	是否开启计算通讯重叠 (建议开启, 开启后每个进程的每个方向网格数应大于 24, 否则会造成程序卡死)	1
	0 - 不开启计算通讯重叠	
	1 - 开启计算通讯重叠	
CHARTERIC	是否开启特征重构 (特征重构可使计算鲁棒性大幅提升, 但计算速度有 0.5 倍到 1 倍的减慢)	1
	0 - 不启用特征重构	
	1 - 启用特征重构	
TEST	是否进行节点测试 (每个 MPI 进程各自执行单独的任务, 打印各进程计算机名与计算时间, 可用来筛选慢节点)	1
	0 - 不进行测试	
	1 - 进行测试	
IPERIODIC	是否是周期边界条件	
	0 - 不是周期边界条件	3
	1 - 周期边界条件	
JAC_BOUND	计算 Jacobian 系数时是否调用边界格式 (只有当网格的两侧物理坐标相连接时, 如顿锥展向, 可以不调用边界格式计算 Jacobian 系数)	6
	0 - 不调用边界格式	
	1 - 调用边界格式	
DIF_BOUND	进行差分计算时是否调用边界格式 (只有周期性边界条件时可以不调用边界格式)	6
	0 - 不调用边界格式	
	1 - 调用边界格式	
NON_REFLETION	是否是无反射边界条件	
	0 - 不是无反射边界条件	6
	1 - 无反射边界条件	

表 2 OpenCFD-SCU 控制文件中差分选择参数说明**Table 2 Description of differential selection parameters in OpenCFD-SCU control file**

参数说明		参数值个数
无粘项差分格式		
	UP7	- 7 阶迎风
	WENO5	- 5 阶 WENO
	WENO7	- 7 阶 WENO
	WENO7_SYMBO	- WENO SYMBO 格式
SCHEME_INVIS	WENO7_SYMBO_LIM	- 加了限制器的 WENO SYMBO 格式
	NND2	- 2 阶 NND 格式
	OMP6_HR	- 6 阶高鲁棒优化保单调格式
	OMP6_LD	- 6 阶低耗散优化保单调格式
	OMP6_CD8	- 6 阶中心型优化保单调格式
	SCHEME_HYBRIDAUTO	- 混合格式
粘性项差分格式		
SCHEME_INVIS	CD6	- 6 阶中心格式
	CD8	- 8 阶中心格式
RE	雷诺数	1
AMA	马赫数	1
GAMMA	气体绝热指数	1
PR	普朗特数	1
T_REF	无量纲参考温度	1
EPSL_SW	S-W 分裂中的稳定系数	1
DT	时间步长	1
END_TIME	计算结束时间	1
KSTEP_SHOW	显示步数	1
KSTEP_SAVE	流场保存步数	1

表 3 OpenCFD-SCU 控制文件中边界条件相关参数**Table 3 Parameters related to boundary conditions in OpenCFD-SCU control file**

边界条件相关参数		
	参数说明	参数值个数
边界条件类型（目前程序中集成了 2 种边界条件）		
IBC	108 针对平板、压缩折角等算例的边界条件 124 针对圆锥、升力体等算例的边界条件	1
整型的边界条件参数		
108 边界条件 对应的整型参数	BC_NPARA[0] 空间上叠加的扰动波数量 MZMAX	
	BC_NPARA[1] 时间上叠加的扰动波数量 MTMAX	
BC_NPARA	BC_NPARA[2] 是否读取入口边界条件 INLET_BOUNDARY	4
	0 - 不读取	
	0 - 读取	
	BC_NPARA[3] 是否读壁面条件（当前版本程序此参数不生效） IFLAG_WALL_NOT_NORMAL	
展向是否是对称		
124 边界条件 对应的整型参数	BC_NPARA[0] 边界条件 IF_SYMMETRY	
	0 - 不采取对称边界条件 1 - 采取对称边界条件	
BC_NPARA	BC_NPARA[1] 是否包含头部（是否读入口边界条件文件） INLET_BOUNDARY	5
	0 - 包含头部 1 - 不包含头部（读文件）	
	BC_NPARA[2] 是否包含激波（是否读外边条文件） IFLAG_UPPERBOUNDARY	
	0 - 包含激波 1 - 不包含激波（读文件）	
	BC_NPARA[3] 空间上叠加的扰动波数量 MZMAX	
	BC_NPARA[4] 时间上叠加的扰动波数量 MTMAX	

表 4 OpenCFD-SCU 控制文件中滤波相关参数**Table 4 Filter related parameters in OpenCFD-SCU control file**

滤波相关参数		
	参数说明	参数值个数
整型滤波相关参数		
FILTER_NPARA&[0]	滤波步数间隔 Filter_step	
	x 方向是否滤波 filter_judge_X	
FILTER_NPARA&[1]	0 - 不开启滤波 1 - 开启滤波	
	y 方向是否滤波 filter_judge_Y	
FILTER_NPARA&[2]	0 - 不开启滤波 1 - 开启滤波	
	z 方向是否滤波 filter_judge_Z	
FILTER_NPARA&		11
FILTER_NPARA&[3]	0 - 不开启滤波 1 - 开启滤波	
FILTER_NPARA&[4]	x 方向滤波区域起始网格点 ib	
FILTER_NPARA&[5]	x 方向滤波区域结束网格点 ie	
FILTER_NPARA&[6]	y 方向滤波区域起始网格点 jb	
FILTER_NPARA&[7]	y 方向滤波区域结束网格点 je	
FILTER_NPARA&[8]	z 方向滤波区域起始网格点 kb	
FILTER_NPARA&[9]	z 方向滤波区域结束网格点 ke	
FILTER_NPARA&[10]	滤波类型（目前版本不生效，程序只集成了一种滤波，守恒形式）Filter_scheme	
浮点型滤波相关参数		
FILTER_RPARA&		3
FILTER_RPARA&[0]	滤波强度（滤波强度越大，对流场的改变程度越大， 默认值一般为 1.0） s0	
FILTER_RPARA&[1]	滤波的阀值（此值越小，接受滤波的范围越大，对流场改变越大， 默认值一般为 e-5） rth	
FILTER_RPARA&[2]	滤波结束的时刻，当流场时间超过此值后，滤波关闭 Filter_end	

表 5 OpenCFD-SCU 控制文件中后处理相关参数 (1)

Table 5 Post-processing related parameters in OpenCFD-SCU control file (1)

后处理相关参数		
	参数说明	参数值个数
ANA_EVENT&	100 对流场进行过滤, 分析出流场中的发散点, 即 nan 或出现负温度的点 ana_NAN_and_NT	
ANA_EVENT&[0]	分析或后处理功能模块选择 101 流场时间平均 ana_time_average	2
	102 当无粘项格式为混合格式时方可生效, 输出 x、y、z 各个中截面混合格式的分布情况 HybridAuto_scheme_IO	
	103 输出流场的 Q 判据文件 get_Q	
	104 保存 XY 块流场时间序列 ana_saveplaneXY	
	105 保存 YZ 块流场时间序列 ana_saveplaneYZ	
	106 保存 XZ 块流场时间序列 ana_saveplaneXZ	
	107 当无粘项格式为混合格式时方可生效, 屏幕打印混合格式的格式占比 HybridAuto_scheme_Proportion	
ANA_EVENT&[1]	- 不开启滤波	

表 6 OpenCFD-SCU 控制文件中后处理相关参数 (2)**Table 6 Post-processing related parameters in OpenCFD-SCU control file (2)**

后处理相关参数		
	参数说明	参数值个数
	ANA_NPARA[0] 需要保存的 XY 块流场 时间序列个数 point	
104	ANA_NPARA[1] 需要保存的 XY 块流场 宽度 andwidth	2+ANA_NPARA[0]
	ANA_NPARA[2] 需要保存的第一个 XY 块流场起始网格点	
	ANA_NPARA[2+...] 需要保存的第 n 个 XY 块流场起始网格点	
ANA_NPARA&	ANA_NPARA[2+point-1] 需要保存的第 point 个 XY 块流场起始网格点	
	ANA_NPARA[0] 需要保存的 YZ 块流场 时间序列个数 point	
105	ANA_NPARA[1] 需要保存的 YZ 块流场 宽度 andwidth	2+ANA_NPARA[0]
	ANA_NPARA[2] 需要保存的第一个 YZ 块流场起始网格点	
	ANA_NPARA[2+...] 需要保存的第 n 个 YZ 块流场起始网格点	
	ANA_NPARA[2+point-1] 需要保存的第 point 个 YZ 块流场起始网格点	
	ANA_NPARA[0] 需要保存的 XY 块流场 时间序列个数 point	
106	ANA_NPARA[1] 需要保存的 XZ 块流场 宽度 andwidth	2+ANA_NPARA[0]
	ANA_NPARA[2] 需要保存的第一个 XZ 块流场起始网格点	
	ANA_NPARA[2+...] 需要保存的第 n 个 XZ 块流场起始网格点	
	ANA_NPARA[2+point-1] 需要保存的第 point 个 XZ 块流场起始网格点	

表 7 OpenCFD-SCU 控制文件中混合格式相关参数
Table 7 Parameters related to hybrid scheme in OpenCFD-SCU control file

混合格式相关参数			
	参数说明	参数值个数	
混合格式类型选择			
HY_STYLE	1 2	利用压力梯度的混合格式（鲁棒性较强，耗散偏大，只适合存在激波的特定问题） 使用 Jameson 推荐的激波识别器进行格式选择的混合格式（推荐）	1
HY_DP_INTV		混合格式阈值，1型混合格式推荐 1 与 8，2型混合格式推荐 0.01 与 0.2	2
HY_SMOOTH_DP		是否启用光滑性处理（目前只有 1 型混合格式可用此功能）	1
HY_PATCH_ZONE		指定判别值的范围个数（目前只有 1 型混合格式可用此功能）	1
指定判别值的范围与指定的判别值			
HY_PATCH_ZONE	HY_ZONE&[0] HY_ZONE&[1] HY_ZONE&[2] HY_ZONE&[3] HY_ZONE&[4] HY_ZONE&[5] HY_ZONE&[6]	x 方向起始网格点 x 方向结束网格点 y 方向起始网格点 y 方向结束网格点 z 方向起始网格点 z 方向结束网格点 指定的判别值（处于哪个阈值区间内，此区域将全部使用阈值对应的格式）	7

.2 数据库说明

- 本数据库包含 5 个壁湍流直接数值模拟实例，由 OpenCFD-SCU 模拟获得。
- 对流项计算时使用的差分格式为 OpenCFD-SCU 中集成的混合格式 (UP7+WENO7+WENO5)。
- 粘性项计算时使用的差分格式为 CD8。
- 对流项通量重构时使用了 Steger-Warming 流通矢量分裂。
- 时间推进使用 3 步 3 阶 Runge-Kutta 方法。
- 部分实例计算时使用了特征变量重构（可以提升计算鲁棒性，但降低了计算效率，效率降低 30% 到 100%）。
- 数据获取地址：

<https://efile01.hpccube.com:65012/efile#Vshare/VHVYYmx1ZW5jZS1EYXRhc2V0>

密码：yuhs

• opencfd-scu.in 为程序控制文件；

OCFD3d-Mesh.dat 为网格文件；

OCFDxxxx.dat 为瞬时流场文件，序号 xxxx 代表计算的步数。

OCFDxxxx.average 为时间平均流场文件，序号 xxxx 代表时间平均的步数。

• 数据库算例说明参见章节 6 表 6.1。

表 8 数据库文件说明**Table 8 Database file description**

No.	算例	文件说明	文件大小
1	马赫 2.9 压缩折角	本数据集包含一个平均场，一个瞬时场，三个不同流向截面时间序列	文件总大小 2.1T
2	马赫 6 压缩折角	本数据集包含一个平均场，一个瞬时场，三个不同流向截面时间序列	文件总大小 1.9T
3	马赫 10 平板	本数据集包含两个平均场，两个瞬时场，两个不同流向截面时间序列	文件总大小 1.7T
4	马赫 6 钝锥	本数据集包含一个平均场，一个瞬时场，三个不同流向截面时间序列	文件总大小 5.0T
5	马赫 6 升力体	本数据集包含三个平均场，三个瞬时场	文件总大小 3.6T

致 谢

转眼间已经来到力学所五年有余了，直到时间到了马上要提交论文的时候，我才恍然惊醒。

在力学所这五年将是我一生中最重要的一段经历。我由衷地对这五年来带给我教导和帮助的父母、师长、亲朋、好友、同学表示感谢。

首先要感谢我的导师李新亮研究员，李老师博学多识，对流体力学的方方面面都有深刻的认识，每次与李老师交谈都能收获满满，让我深感钦佩。更重要的是，李老师为人端正，与人为善，他一直贯彻开源共享的精神，深深地影响了我。李老师给予学生充分的信任与支持，让我能够不受束缚地成长，受益匪浅。

我还要感谢我的父母，他们在我的背后给予我最大的精神支撑，无论在哪里，我都能感觉到内心的温暖。

感谢陈哲、李理、陈小平、彭峻师兄还有李多师姐，他们在我找工作的过程中给予了我帮助，让我有了很多就业选择。

感谢康健、周浩、李欣、闫政和符耀威师兄，尽管我现在已经是课题组年纪最大的学生了，但我还是更怀念刚进组的时候，与师兄们的交流中让我不断获得成长。

感谢 212 办公室现在的同学们，段俊亦，齐涵，何康，朱艳华，张吉，胡润宁，与你们讨论问题的片段让我印象深刻，也感谢你们陪我走完博士生涯的最后一段路程，让我最后的路程里充满欢声笑语。

感谢中科院数学与系统科学研究院的刘世伟师兄，在程序开发的前期，刘师兄做了很多工作，尽管我后来为了添加各种功能从底层重构了代码，但从刘师兄最初的代码里我学到了不少编程技巧。

感谢中科院计算机网络信息中心的张鉴研究员、胡晓东与刘夏真师兄，他们在程序开发过程中提出了不少意见与建议。

感谢郭同彪老师和谢祝轩老师，郭老师外语水平很高，他帮我修改了投稿论文摘要，引言和结论的部分，并在投稿过程中给予了我一定帮助。

感谢我曾经的室友孟锦、和曹高辉师兄。与你们相处的时光是我最难忘的时光。感谢一起玩的同学，段军，雷勇，你们让我生活不再无聊。

感谢中科曙光的张驭洲、卜景德在软件测试时提供的帮助与建议。

感谢课题组的于欣老师、刘洪伟老师、于长平老师以及研究生教育处的刘丽老师。

谨以此文献给师长、家人、伙伴。

党冠麟
2022年10月

作者简历及攻读学位期间发表的学术论文与研究成果

作者简历：

党冠麟，河南南阳市宛城区人。

2013.09 - 2017.07, 兰州大学, 理论与应用力学基地班, 学士;

2017.09 - 2022.12, 中国科学院力学研究所, 流体力学专业, 硕博连读研究生;

学术论文：

- (1) Guanlin Dang, Shiwei Liu, Tongbiao Guo, Junyi Dun, Xinliang Li, Direct numerical simulation of compressible turbulence accelerated by graphics processing unit: An open-source high accuracy accelerated computational fluid dynamic software. [J] Phys. Fluids (In Press, 2022), <https://doi.org/10.1063/5.0127684>
- (2) Guanlin Dang, Shiwei Liu, Tongbiao Guo, Junyi Duan, Xinliang Li, Direct numerical simulation of compressible turbulence accelerated by graphics processing unit: An open-access database of high-resolution direct numerical simulation. [J] AIP Advances (In Press, 2022), <https://doi.org/10.1063/5.0132489> (Editor's Pick)
- (3) 党冠麟, 刘世伟, 胡晓东, 等. 基于 CPU/GPU 异构系统架构的高超声速湍流直接数值模拟研究 [J]. 数据与计算发展前沿, 2020, 2(1):12.
- (4) Ji Zhang, Tongbiao Guo, Guanlin Dang, Xinliang Li, Direct numerical simulation of shock wave/turbulent boundary layer interaction in a swept compression ramp at Mach 6, [J] Phys. Fluids (In Press, 2022), <https://doi.org/10.1063/5.0118578> (Editor's Pick)

参加的研究项目及获奖情况：

中科院先导杯并行应用大奖赛开放应用组一等奖, 2020;

中科院先导杯并行应用大奖赛最短路径算法优胜奖, 2020;

中科院先导杯并行应用大奖赛 SgemmStridedBatched 优化算法优胜奖, 2020;

中国科学院大学三好学生 2020-2021;

中国科学院大学三好学生 2021-2022;

