计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(5):238−243 [doi: 10.15888/j.cnki.csa.006371]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于非结构网格隐式算法的 GPU 加速研究①

陈　龙, 徐添豪, 田书玲

(南京航空航天大学 航空宇航学院, 南京 210016)

摘　要: 针对非结构网格隐式算法在 GPU 上的加速效果不佳的问题, 通过分析 GPU 的架构及并行模式, 研究并实现了基于非结构网格格点格式的隐式 LU-SGS 算法的 GPU 并行加速. 通过采用 RCM 和 Metis 网格重排序 (重组) 方法, 优化非结构网格的数据局部性, 改善非结构网格的隐式算法在 GPU 上的并行加速效果. 通过三维机翼算例验证了本文实现的正确性及效率. 结果表明两种网格重排序 (重组) 方法分别得到了 63% 和 69% 的加速效果提高. 优化后的 LU-SGS 隐式 GPU 并行算法获得了相较于 CPU 串行算法 27 倍的加速比, 充分说明了本文方法的高效性.

关键词: GPU 加速; 并行计算; 网格排序; 计算流体力学; 隐式格式

引用格式: 陈龙,徐添豪,田书玲.基于非结构网格隐式算法的 GPU 加速研究.计算机系统应用,2018,27(5):238–243. http://www.c-s-a.org.cn/1003-3254/6371.html

Research on GPU Acceleration of Implicit Schemes Based on Unstructured Grids

CHEN Long, XU Tian-Hao, TIAN Shu-Ling

(College of Aerospace Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: With regard to the poor acceleration performance on GPU using the unstructured grids implicit method, this study realizes the GPU acceleration of LU-SGS implicit method based on unstructured grids with the cell-vertex scheme. With introduce the architecture of a GPU and its parallelization method, two grid reordering methods are set forth based on RCM and METIS, to improve data locality of unstructured grids and to improve acceleration performance on GPU using the unstructured grids implicit method. The ONERA M6 Wing test case is carried out to verify and validate this implementation. With two grid reordering methods, the GPU implementations achieve 63% and 69% improvements respectively. The GPU implementation obtains a speedup of 27 times compared to the CPU version running on a single core. It indicates that the proposed GPU implementation has a solid performance.

Key words: GPU acceleration; parallel computing; grid reordering; computational fluid dynamics; implicit schemes

近年来, 图形处理器 (GPU) 在科学计算领域中发挥了重要的作用. 由于其出色的浮点运算性能以及具有优势的显存带宽, GPU 往往能给科学计算带来可观的加速. 计算流体力学 (CFD) 领域的 GPU 加速已得到广泛的研究, Hagen 等人[1,2]先后在 GPU 上实现了基于欧拉方程以及基于 NS 方程的求解器, 得到了良好的加速效果. 随着 GPU 硬件自身的发展, 更复杂的算法得到了实现, 基于 RANS 方程、LES 方法以及 DNS 方法的国内外先进研究成果不断被发表[3−7]. 为了进一步地提高运算能力, 基于多 GPU 的实现也得到了越来越多的关注[8]. 刘枫等人[9]基于 MPI+CUDA 的异构并行可压缩流求解器, 用于高超声速流动的数值模拟计算效率较 CPU 同构并行计算提高 10 倍以上. 采用 GPU 加速的 CFD 算法在航空航天工程中具有广阔应用前景.

① 基金项目: 江苏高校优势学科建设工程资助项目
收稿时间: 2017-09-19; 修改时间: 2017-10-10; 采用时间: 2017-10-26; csa 在线出版时间: 2018-04-23

CFD 算法 GPU 加速目前的研究前沿主要集中在结构网格和显式时间推进. 非结构网格相比于结构网格更适用于复杂的几何外形, 但由于其存储结构的特点难其内存存取的连续性较差, 在 GPU 架构上难以获得很高的效率. 同时, 以 LU-SGS 算法为代表的隐式方法具有较显式方法高数倍的计算效率, 但其每次迭代过程中多次需要相邻单元数据, 需要大量的内存访问, 在 GPU 架构上需要较好的数据局部性才能得到较高的效率, 因此通常非结构网格的隐式算法在 GPU 上的加速效果不佳.

针对这一问题, 本文发展了 LU-SGS 算法在非结构网格上的高效率的 GPU 实现方法. 首先, 介绍了基本控制方程和 LU-SGS 算法. 然后, 给出了算法的 GPU 实现方法. 并采用 RCM 以及基于 Metis 网格分区的网格重排序方法优化算法的加速效果, 详细讨论各网格排序算法在 GPU 实现中的适用性. 最后, 通过算例验证了算法的正确性以及实现方案的高效性.

1 数值方法

惯性系下, 忽略源项的 RANS 方程形式如下:

$$\frac{\partial}{\partial t}\int_{\Omega} W d\Omega + \int_{d\Omega} (F-G)dS = 0 \tag{1}$$

其中 W, F, G 分别代表守恒变量、对流通量及粘性通量. 本文采用非结构网格下格点格式离散有限体积控制体, 使用 Roe 格式离散对流通量. 通过线性重构实现空间二阶精度. 使用收敛性良好的 Venkatakrishnan 限制器. 采用隐式 LU-SGS 方法进行时间推进, 其形式如下[10]

$$(\mathbf{D}+\mathbf{L})\mathbf{D}^{-1}(\mathbf{D}+\mathbf{L})\Delta\mathbf{W} = \mathbf{R} + (\mathbf{L}\mathbf{D}^{-1}\mathbf{U})\Delta\mathbf{W} \tag{2}$$

其中, $\mathbf{U}$, $\mathbf{L}$ 和 $\mathbf{D}$ 分别为严格上三角矩阵, 严格下三角矩阵和对角矩阵.

采用向前扫掠和向后扫掠求解该方程, 过程可以写成:

向前扫掠:

$$(\mathbf{D}+\mathbf{L})\Delta\mathbf{W}^{*} = \mathbf{R} \tag{3}$$

向后扫掠:

$$(\mathbf{D}+\mathbf{U})\Delta\mathbf{W} = \mathbf{D}\Delta\mathbf{W}^{*} \tag{4}$$

该方法最早由 Jameson 和 Yoon[11]应用于结构网格, 在扫掠时确定超平面, 如图 1. 同一超平面内的控制体可以同步更新而不存在相互的依赖, 这也为计算的数据并行提供了理论基础. 后 Nakahashi 等人[12]将其应用到非结构网格上, 扫掠时也需要确定类似形式的超平面, 如图 2.

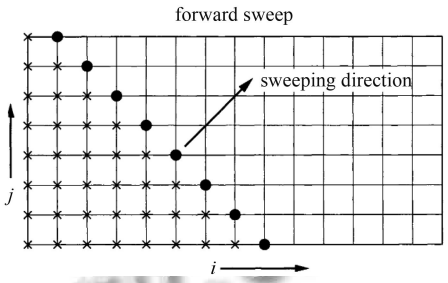


图 1 结构网格上的超平面形式

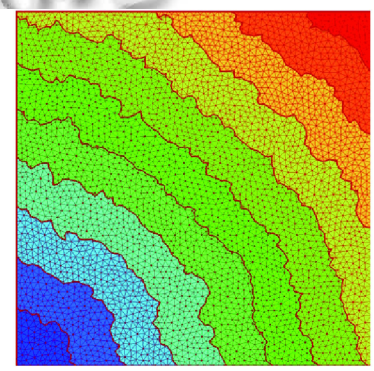


图 2 非结构网格上的超平面形式

湍流模型选用 Spalart-Allmaras 一方程模型[13]. 该模型广泛应用于飞机机翼、翼身组合体等绕流的数值模拟中.

2 算法 GPU 并行加速

2.1 GPU 并行方法

单个 GPU, 从硬件上来说, 由多个流多处理器 SM 组成. 每个 SM可以同时处理上千个线程. 在英伟达公司提出的 CUDA 框架下, 线程以 32 为一束进行调度, 即同一线程束 (Warp) 内的线程在同一时间执行相同的指令, 这也对应着并行的最细粒度. 而若干个 Warp 又组成线程块 (Block) 被分发到不同 SM 上执行. 每个计算任务会对应一个线程网格 (Grid) 进行线程组织, 而一个 Grid 是由多个 Block 组成的. 从 Grid 到 Block 到 Thread 都可以在编程时进行一维、二维乃至三维的索引, 如图 3 所示.

运行于 GPU 上的并行程序是基于数据并行的, 每个线程执行相同的任务, 程序员也无需过多地关心线程的调度方式, 它由 GPU 自行处理. 因此, 在 CFD 领

域内, 涉及大量数据并行的 CPU 程序很容易移植到 GPU 上. 在传统的 CPU 求解器中, 非结构网格结合格点格式会涉及两种形式的循环, 即基于点的和基于边的. 对于这样两种循环, 我们在 GPU 上最为直接地将单个点或者单条边映射到单个线程上, 而每个线程则执行原来循环体内的指令.

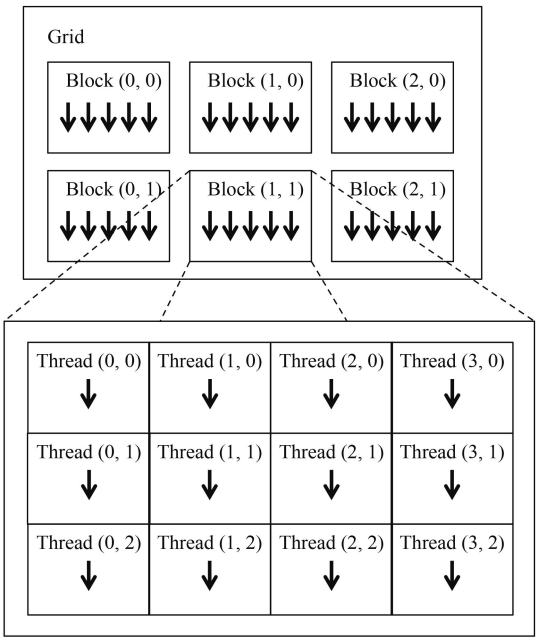


图 3 线程、线程块、线程块网络的组织形式

```
For iNode=0 to nNode do
  // code
End for
```

可以写成对应的由 CPU 调用、GPU 执行的函数 (Kernel 函数) 的形式

```
Kernel<<<Grid(nNode threads in total)>>>() {
  // code (almost the same)
}
```

代码区域则几乎相同, 这为快速的代码移植提供了相当的便捷性. 可见 GPU 的细粒度并行方式通常与 CPU 上的多线程、多进程实现的任务并行不同 (将单个任务拆分成多个子任务). 当然在单个或者多个 GPU 上进行类似于 CPU 的任务并行也是实现大规模 GPU 运算的必要手段.

2.2 算法的 GPU 实现

涉及算法在 GPU 上的流程图如图 4.

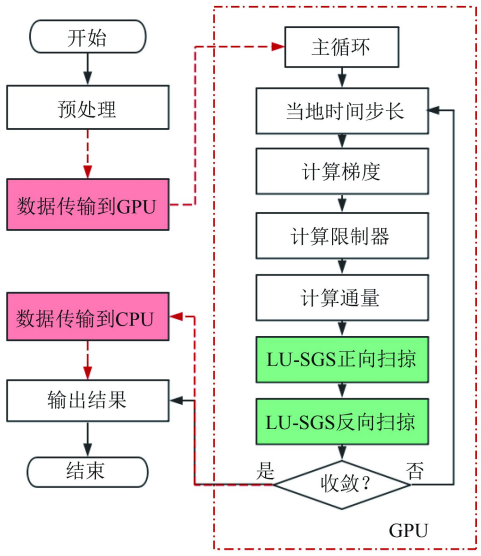


图 4 CPU 及 GPU 算法流程图

GPU 流程与 CPU 上的实现几乎完全相同. 在 GPU 上进行计算的前后, 需要将数据拷贝到 GPU 上或者从 GPU 拷贝回 CPU, 但是所有的求解过程是在 GPU 上进行的, 如图中红色虚线框. 可以认为 GPU 上实现的每一个步骤对应一个 Kernel 函数. 不同的是, LU-SGS 向前、向后扫掠步骤中涉及了数百甚至上千个 Kernel 函数 (与超平面个数有关). LU-SGS 算法本身就适合于进行超平面内的数据并行. 而进行扫掠的时候, 下一个超平面内的变量依赖于更新过后的当前面内的变量. 因此, 可以采用每一层对应一个 Kernel 函数, 而这些 Kernel 函数串行执行来实现对应语义.

2.3 性能优化

对应于上述流程图的实现并不一定立即获得显著的加速比. GPU 上程序的性能优化需要更多地关注硬件底层. 通常认为会有基于硬件利用率、内存以及指令三个层面的优化方法. 在进行了基本的包括上述三个层面的常见优化[14]后, 本文着重关注非结构网格下全局内存访问效率的提升, 属于内存层面的深度优化.

全局内存访问的优化关键在于合并访问. 合并访问指的是同一个 Warp 内的 Thread 访问全局内存时, 内存访问指令可以合并成若干个 32、64 或者 128 字节的内存读取指令. 当相邻 Thread 内存访问可以合并时, 说明其访问的内存在地址上是连续的. 一般在结构网格下, 合理安排数据形式就能实现合并的内存访问, 采用 SoA(Structure of Array) 而不是 CPU 实现常用的 AoS(Array of Structure) 形式, 如守恒变量安排如图 5 所示.

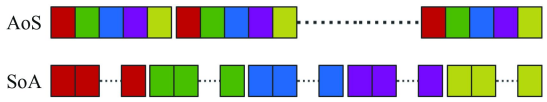
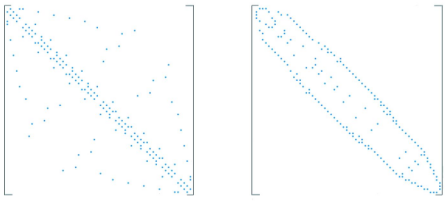


图 5 SoA 和 AoS 内存组织方式对比

在非结构网格中, 根据本文使用的 Thread 映射方式, 连续的 Thread 对应连续的边, 但是每条边需要访问它相连的左右节点, 而左右节点由于非结构网格的不规则性, 在内存上却是不连续的. 得益于 GK110 核心对应的 L2 Cache 以及 Read-only Data Cache, 对于内存连续性的要求并没有之前的硬件那么高, 不再需要保证完全按照顺序在地址上连续, 仅需保证有良好的数据局部性 (Data Locality). 因此, 本文采用对网格重排序的方法来改善基于非结构网格的内存合并访问性能.

考虑由边描述的节点连接关系的邻接关系矩阵 (nNodes×nNodes), 可以通过 RCM(Reverse Cuthill-McKee algorithm) 排序[14]来减小该稀疏矩阵带宽, 也意味着空间中邻近的点在内存上也是邻近的, 效果如图 6 所示.



(a) 排序前 (b) 排序后

图 6 RCM 排序用于减小稀疏矩阵带宽

对网格点进行 RCM 排序, 然后根据排序后的网格点顺序, 按照每个网格点遍历连接边的贪婪算法进行边的重排序, 完成排序后加速效果显著.

RCM 排序可以在线性的时间复杂度下完成, 这是该方法的优势. 本文再提出另一种基于 Metis 网格分区方法[15]的排序方法用于改善数据局部性, 旨在通过时间代价换取更高的加速比. Metis 方法常用于并行计算过程的中的网格分区及负载均衡. 针对 GPU 结构优化时, 基于 Metis 的排序方法较 RCM 排序更为直观, 语义上直接对应: 靠的近的点分进同一组. 本文通过将网格使用 Metis 分成 N 个小区域, 每个区域包含 100 到 1000 个点 (不同点的个数在这个区域内对效果影响不明显). 如简化了的只包含四种颜色的示意图 7.

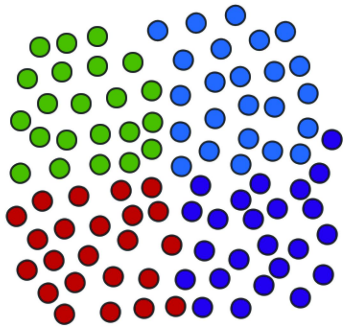


图 7 基于 Metis 的网格排序方法示意图

基于 Metis 的排序方法同样也需要根据新的点顺序使用贪婪算法对边进行重排序. 这种方法在绝大多数情况下性能优于 RCM 方法, 但是问题在于需要占用大量的内存并且需要较长的时间. 对于 CPU 内存满足条件的设备, 使用这种排序算法能够实现更高的加速比.

3 算例分析

本节通过 ONERA M6 机翼算例验证求解器求解三维绕流问题的正确性. 该算例是 CFD 中的经典算例, 几何模型简单却又包含了流动的复杂性, 涉及跨音速流动、局部超音速流动、激波以及湍流边界层分离等现象. 并于实验结果进行了对比. 计算条件设置为: 来流马赫数 0.8395, 攻角 3.06°, 无侧滑角, 雷诺数 11.72×106. 网格由四面体网格和附面层区域的棱柱网格组成, 包含 210 万格点, 467 万单元, 877 万条边, 241 万棱柱单元, 第一层网格高度$y^+ \leqslant 1$. 计算得到的采用音速无量纲化后的压力云图与物面网格如图 8 所示.

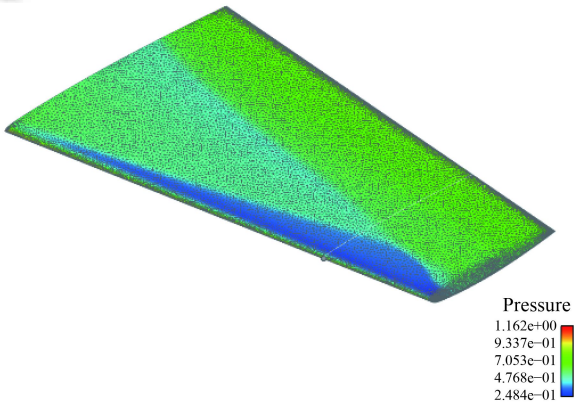
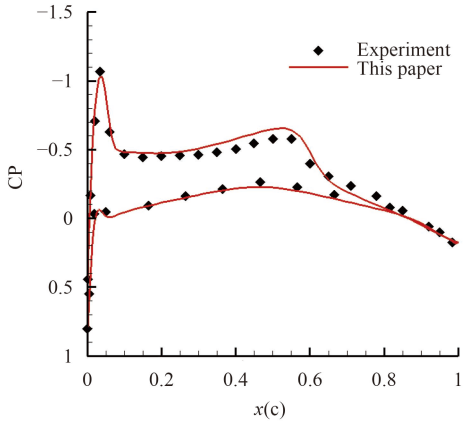


图 8 ONERA M6 机翼表面网格及压力分布

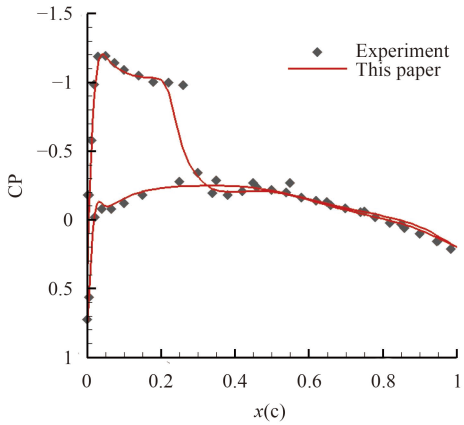
3.1 数值结果分析

实验结果给出的数据为展向七个截面处压力系数

分布, 以翼展 b 为基准, 本节选出 20% 和 90% 翼展处截面的压力系数分布作为对比, 结果与实验数据基本吻合, 如图 9 所示.



(a) 翼展20%截面压力系数分布



(b) 翼展90%截面压力系数分布

图 9 机翼展向不同截面处压力系数分布

3.2 加速性能分析

本节所用 CPU 设备为 Intel(R) Xeon(R) CPU E3-1230 v3, GPU 设备为 NVIDIA Tesla K40c. 操作系统环境为 Ubuntu14.04. C++编译器选用 gcc-4.8, CUDA 版本为 7.5. GPU 端以及 CPU 端代码均采用-O2 等级优化. 加速比通过执行并对比算法相同的 CPU 串行程序和 GPU 并行程序得到. 计算采用双精度变量.

采用本章算例网格, 分别在 CPU 和 GPU 上进行 1000 步迭代步计算, 并在 GPU 实现中结合本文提出的基于 RCM 及基于 Metis 的网格排序方法同样进行 1000 步迭代, 所需计算时间及相应加速比如表 1 所示.

结合基于 Metis 的网格排序的隐式算法求解过程在 GPU 上相较于 CPU 得到了 27 倍的加速比. 表 1 也反应出了本文提出的网格排序算法的高效性, 分别得到了 63% 和 69% 的性能提升. 如果原始网格过于规整, 该提升也将会因为对合并访问改善幅度的下降而有所下降. 此外, 通过不断改善网格质量, 或者使用结构网格, 本文实现的 GPU 并行程序可以实现 50 倍的加速比. 可见, 对内存合并访问的改善仍然有很大空间. 另外, 测试所用网格应避免过小以防 GPU 未被充分利用, 这一点不再展开.

表 1 计算时间及相应加速比

平台及排序算法	计算时间(s)	加速比
CPU	5944	1
GPU	377	16
GPU+RCM	230	26
GPU+Metis	219	27

4 结论

针对非结构网格格点格式在 GPU 上计算加速效果不佳的问题, 提出了采用 RCM 以及基于 Metis 网格分区的网格重排序方法, 大大地改善了基于非结构网格的隐式算法在 GPU 上的并行加速效果. 针对文中算例, 两种方法分别得到了 63% 和 69% 的加速效果. 基于 Metis 的排序方法在效果上往往优于 RCM 方法, 但对于规模较大的网格, 其耗时较长. 采用了基于 Metis 的网格重排序方法后, 文中算例结果得到了相较于 CPU 串行程序 27 倍的加速比, 充分证明了本文提出的方法的可行性及优越性. 此外, 本文的结果也意味着基于非结构网格格点格式的隐式算法在 GPU 上也能得到良好的加速性能, 相关应用拥有着广阔的前景.

致谢: 英伟达公司为本研究工作的展开赞助了 Tesla K40 计算卡, 我们为此表示由衷的感谢.

参考文献

1 Hagen TR, Lie KA, Natvig JR. Solving the Euler equations on graphics processing units. Proceedings of the 6th International Conference Computational Science-ICCS 2006. Berlin, Heidelberg, Germany. 2006. 220–227.

2 Cohen J, Molemaker MJ. A fast double precision CFD code using CUDA. Proceedings of Parallel Computational Fluid Dynamics: Recent Advances and Future Directions. Lancaster, UK. 2009. 414–429.

3 Zimmerman BJ, Wie B. Graphics-processing-unit-accelerated multiphase computational tool for asteroid fragmentation/pulverization simulation. AIAA Journal, 2017,

55(2): 599–609.

4 Croaker P, Kessissoglou N. Aeroacoustic scattering using a particle accelerated computational fluid dynamics/boundary element technique. AIAA Journal, 2016, 54(7): 2116–2133.

5 Mostafazadeh B, Marti F, Pourghassemi B, *et al*. Unsteady navier-stokes computations on GPU architectures. Proceedings of the 23rd AIAA Computational Fluid Dynamics Conference. Denver, CD, USA. 2017. 4508.

6 Jude D, Baeder J. Extending a three-dimensional GPU RANS solver for unsteady grid motion and free-wake coupling. Proceedings of the 54th AIAA Aerospace Sciences Meeting. San Diego, CA, USA. 2016. 1811.

7 董廷星, 李新亮, 李森, 等. GPU 上计算流体力学的加速. 计算机系统应用, 2011, 20(1): 104–109.

8 Watkins J, Romero J, Jameson A. Multi-GPU, implicit time stepping for high-order methods on unstructured grids. Proceedings of the 46th AIAA Fluid Dynamics Conference. Washington, DC, USA. 2016. 3956.

9 刘枫, 李桦, 田正雨, 等. 基于 MPI + CUDA 的异构并行可压缩流求解器. 国防科技大学学报, 2014, 36(1): 6–10.

10 陈龙. 基于 CFD/CSD 耦合的旋翼气动弹性数值模拟[博士学位论文]. 南京: 南京航空航天大学, 2011.

11 Jameson A, Yoon S. Lower-upper implicit schemes with multiple grids for the Euler equations. AIAA Journal, 1987, 25(7): 929–935.

12 Sharov D, Nakahashi K. Reordering of 3-D hybrid unstructured grids for vectorized LU-SGS Navier-Stokes computations. Proceedings of the 13th Computational Fluid Dynamics Conference. Snowmass Village, CO, USA. 1997.

13 Spalart PR, Allmaras SR. A one-equation turbulence model for aerodynamic flows. Proceedings of the 30th Aerospace Sciences Meeting and Exhibit. Reno, NV, USA. 1992.

14 George A, Liu JW. Computer solution of large sparse positive definite. Biometrical Journal, 1981, 26(2): 218–225.

15 Karypis G, Kumar V. Metis-unstructured graph partitioning and sparse matrix ordering system, version 5.1.0. University of Minnesota, Department of Computer Science, 2013.