Journal of Computational Physics 480 (2023) 112009

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A cache-efficient reordering method for unstructured meshes with applications to wall-resolved large-eddy simulations

Yi Liu [a,b], Hongping Wang [a,b], Shizhao Wang [a,b,*], Guowei He [a,b]

[a] Institute of Mechanics, Chinese Academy of Sciences, Beijing, 100190, China
[b] School of Engineering Science, University of Chinese Academy of Sciences, Beijing, 100049, China

ARTICLE INFO

Article history:
Received 31 July 2022
Received in revised form 7 February 2023
Accepted 8 February 2023
Available online 23 February 2023

Keywords:
Large-eddy simulations
Unstructured mesh
High performance computing
Mesh reordering
Cache utilization

ABSTRACT

Unstructured meshes provide a distinct advantage for handing complex geometries. However, the low cache utilization due to mesh-related data access patterns raises particular challenges in achieving a high computing efficiency for large-eddy simulations. We propose a geometrical-based mesh reordering method to improve cache utilization. The proposed method utilizes a Hilbert space-filling curve that passes through each cell once, guiding the reordering of the cells of the unstructured mesh. The reordering enables neighboring cells to be stored in contiguous areas of memory as much as possible. The performance of the proposed method is validated by two- and three-dimensional unstructured meshes. According to the memory and spatial distances, the proposed reordering method significantly improves data localities. Consequently, the cache hit rate is increased and the computing efficiency is improved. The proposed reordering method is then applied to large-eddy simulations of flows around an underwater vehicle model with $Re_L = 1.2 \times 10^6$. To fully resolve the near-wall flows, an unstructured mesh consisting of 1.476 billion cells is used, which is partitioned into 12800 subdomains. The computed velocity profiles, pressure and friction coefficients are in good agreement with the experimental measurements. The computational costs are reduced by using the proposed reordering method.

1. Introduction

Large-eddy simulation (LES) is a promising approach for predicting and analyzing flows around underwater vehicles [1,2]. In this approach, the energy-containing eddies are resolved on computational grids [3]. The number of grid points required to fully resolve the energy-containing eddies scales as $\mathrm{Re}_{L_x}^{13/7}$ [4], where $\mathrm{Re}_{L_x} = UL_x/\nu$, U is the freestream velocity, L_x is the reference length in the streamwise direction, and ν is the kinematic viscosity. The need to fully resolve these energy-containing eddies places high demands on the spatial resolution. For example, hundreds of millions of mesh cells are usually required even in simulating flows around an underwater vehicle at a relatively low Reynolds number of $Re_L = UL/\nu = 1.1 \times 10^6$ [5,6], where L is the length of the underwater vehicle. Parallel computing is essential for large-scale simulations [7].

* Corresponding author.
E-mail address: wangsz@lnm.imech.ac.cn (S. Wang).

https://doi.org/10.1016/j.jcp.2023.112009
0021-9991/

One of the challenges in parallel computing of flows around an underwater vehicle is efficiently handling complicated boundaries. The complicated boundaries of underwater vehicles are usually handled by either body conformal meshes or nonbody conformal meshes. The recent work of Posa and Balaras [8] successfully conducted simulations on a nonbody conformal mesh with 8.1 billion grid points on 3000 CPU cores, as the nonbody conformal grid has advantages in high-efficiency parallel computing. A detailed discussion of the advantages and disadvantages of nonbody conformal grids can be found in the review of Mittal and Iaccarino [9]. However, highly efficient parallel computing on a body conformal mesh, especially unstructured mesh, is very difficult. In a flow solver on an unstructured mesh, the mesh connectivity and irregular element orders [10] are required to be explicitly stored in memory. Indirect and irregular memory access patterns appear when iterating over the mesh cells to evaluate fluxes, gradients, and limiters, increasing the memory access latency and dramatically decreasing the computational speed [11–13]. Memory access latency has been reported to be one of the top five challenges for exascale computing systems [14,15]. Memory latency becomes worse as the mesh size increases; however, many of these problems can be alleviated by the programmer through the use of memory caches [16–18]. Mesh reordering is considered to be an effective method for reducing cache-related latency and achieving high computing efficiency [19–21].

Due to the increasing size of the mesh used in computational fluid dynamics (CFD), mesh reordering methods have been studied to exploit the performance gains of CFD solvers [22–24]. Mudigere et al. [23] conducted a detailed exploration on optimizing the memory of a solver using an unstructured mesh (PETSc-FUN3D), demonstrating that optimizing the data structure and cache utilization significantly improved solver performance. Economon et al. [24] studied the impact of edge reordering and data layout transformations on improving the computing performance of modern multicore architectures. Their results show that a speedup ranging from 1.2X to 1.7X was obtained by reordering the mesh. Hadade et al. [13] reviewed some useful optimizations for CFD solvers using unstructured meshes on multicore and manycore architectures. In a flow solver on an unstructured mesh, most of the computations are performed at the local elements (edges, faces, or cells), including time integration, gradient calculations, and gather and scatter operations. However, unstructured mesh generators, such as the Delaunay generator [25], usually create numbers for the vertices and cells as they produce them, which effectively results in irregular ordering. Consequently, computations based on an unstructured mesh suffer from low cache utilization due to these irregular and nonlocal memory access patterns. Improving the data access pattern is helpful in exploiting cache utilization, increasing computational efficiency [13,24]. Previous works have optimized data access patterns by reordering the mesh cells. Löhner [26] presented reordering techniques based on shared-memory, cache-based parallel machines, focusing on eliminating cache-line overwriting. Aubry et al. [12] presented reordering methods for vertex-centered discretization that guaranteed that nodes belonging to one thread were not accessed by other threads. Cheng et al. [27] proposed a grid reordering method for hybrid unstructured meshes, in which the negative effect of the interface between threads was weakened, thereby improving the convergence of the solution. This type of reordering approach makes better use of the cache and improves the performance of the solver for shared-memory OpenMP implementations.

One widely used reordering strategy for message passing interface (MPI) platforms is the reverse Cuthill-McKee (RCM) method [28,29], which has been adopted by SU2 [22], FUN3D [23], and some in-house codes [30]. Duff and Meurant [31] showed that the RCM method effectively reduced the matrix bandwidth for lower-upper (LU) factorization and incomplete LU (ILU) preconditioning. Shi et al. [32] used the RCM method to compress the bandwidth of sparse matrices, considerably reducing the complexity of LU factorization and improving the computational efficiency. To increase the speed of solving linear equations, Diosady et al. [33] and Mathews et al. [34] performed the RCM method on the Jacobian matrix. Amir et al. [35] used the RCM method as a preconditioner in a high-order unstructured Newton-GMRES solver, resulting in an acceleration in the convergence of the solver. In the RCM method, a starting cell is selected and relabeled as 1. The cells adjacent to the starting cell are relabeled as $(1+i)$ in the order of their increasing number (i) of cells. The cells at a distance of 1 from the starting cell are said to be in the first level. The cells adjacent to the cells in the first level that have not yet been labeled become the next level. This procedure is repeated until all cells are numbered. In each level, the first cell in the preceding level can be connected to any cell in the current level, even the last cell in the current level in the worst-case scenario. Mesh connectivity can be broken arbitrarily, thereby deteriorating the data locality [36,37]. Data locality indicates that the data access is linear and therefore predictable and that the data are accessed soon after nearby data in the same domain have been accessed [38]. In the conditional source-term estimation (CSE) method [39], which has been applied in CFD simulations of turbulent reacting flows, the data locality guarantees correct results due to the requirement that a sufficient number of localized points must be used to compute the conditionally averaged scalars. In computer science, better locality of the data reference results in faster execution times. Data that will be used in the near future are prefetched and stored in the cache, eliminating the latency penalty from the main memory [18]. Modern computer architectures use various cache levels, and these caches access data quickly but are small in size. Poor locality in the data access pattern may cause cache misses, leading to dramatic decreases in the computational speed. The study presented by Günther et al. [11] shows that cache misses lead to a dramatic slowdown of the data access when solving the Poisson equation. More recently, Akkurt et al. [18] presented cache optimization strategies in the context of a high-order Euler solver, obtaining a significant speedup of approximately 2 for hexahedral elements. Cache misses are a serious bottleneck in high-performance computing [17,18].

The objective of this paper is to develop a Hilbert space-filling curve (SFC)-based mesh reordering method to improve cache utilization when performing large-scale numerical simulations with unstructured meshes. This idea is inspired by graph partitioning algorithms, which use an SFC to identify locally optimal partitions that minimize edge cuts [40]. The SFC

is a function that maps a multidimensional space to a one-dimensional space with good locality properties [41]. In scientific computing, SFCs are also powerful tools for multidimensional data mining and computer graphics or can be used to build data structures that allow fast data access and exploit cache hierarchies [41,42]. Self-similar Hilbert SFCs are characterized by the properties that the curves are very tightly packed or coiled within a local area. Thus, these SFCs can localize spatially neighboring cells in a one-dimensional array, preserving the locality [43]. The properties of preserving locality make for efficient gathering and scattering of data from and to pairs of cells that share a face. Motivated by the locality property of Hilbert SFCs, we propose a Hilbert SFC-based mesh reordering method to improve the performance of flow solvers for unstructured meshes in parallel computations. The proposed mesh reordering method was implemented in an in-house CFD solver [44–48] that can run on a range of computing platforms from desktop workstations to HPC platforms.

The remainder of this paper is organized as follows. The Hilbert SFC-based mesh reordering approach and basic numerical methods are briefly described in Section 2. The computational results and detailed analyses of the statistical data are discussed in Section 3.1 and Section 3.2. A wall-resolved LES of the flow over SUBOFF without appendages is presented in Section 3.3. Finally, the conclusions are drawn in Section 4.

2. Numerical method

The governing equations for a large-eddy simulation of a compressible flow are obtained by filtering the time-dependent compressible Navier–Stokes equations. The filtered Navier–Stokes equations are given by Ref. [49]

$$\frac{\partial \bar{\rho}}{\partial t} + \frac{\partial}{\partial x_i}(\bar{\rho}\tilde{u}_i) = 0, \tag{1a}$$

$$\frac{\partial (\bar{\rho}\tilde{u}_i)}{\partial t} + \frac{\partial}{\partial x_j}(\bar{\rho}\tilde{u}_i\tilde{u}_j) + \frac{\partial \tau_{ij}^{SGS}}{\partial x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial \tilde{\tau}_{ij}}{\partial x_j}, \tag{1b}$$

$$\frac{\partial (\bar{\rho}\tilde{E})}{\partial t} + \frac{\partial}{\partial x_j}(\bar{\rho}\tilde{H}\tilde{u}_j) + \frac{\partial}{\partial x_j}(\tau_{ij}^{SGS}\tilde{u}_i) = \frac{\partial (\tilde{\tau}_{ij}\tilde{u}_i)}{\partial x_k} - \frac{\partial \tilde{q}_j}{\partial x_j} - \frac{\partial Q_j^{SGS}}{\partial x_j}, \tag{1c}$$

where $\bar{\varphi}$ denotes the filtered quantity and $\tilde{\varphi}=\overline{\rho\varphi}/\bar{\rho}$ represents the Favre-filtered quantity. $\bar{\rho}$, $\bar{p}$ and $\tilde{u}_i$ represent the density, pressure and velocity, respectively. The total energy and total enthalpy are given by $\tilde{E} = \tilde{e} + 0.5\tilde{u}_i\tilde{u}_i + 0.5\tau_{ii}^{SGS}/\bar{\rho}$ and $\tilde{H} = \tilde{E} + \bar{p}/\bar{\rho}$, respectively. Here, $\tilde{e}$ represents the internal energy. By convention, the molecular viscous stress tensor $\tilde{\tau}_{ij}$ is approximated as

$$\tilde{\tau}_{ij} = \bar{\mu}\left(\tilde{T}\right)\left(\frac{\partial \tilde{u}_i}{\partial x_j} + \frac{\partial \tilde{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial \tilde{u}_k}{\partial x_k}\delta_{ij}\right), \tag{2}$$

where $\bar{\mu}\left(\tilde{T}\right)$ is the molecular viscosity based on the Favre-filtered static temperature $\tilde{T}$, which is evaluated by Sutherland's law. The subgrid stress tensor τ_{ij}^{SGS} is defined as,

$$\tau_{ij}^{SGS} = -\bar{\rho}\left(\widetilde{u_i u_j} - \tilde{u}_i\tilde{u}_j\right). \tag{3}$$

The molecular heat flux is defined as

$$\bar{q}_i = -\lambda\frac{\partial \tilde{T}}{\partial x_i}, \tag{4}$$

where λ represents the thermal conductivity coefficient. The corresponding components of the subgrid-scale heat flux are

$$Q_j^{SGS} = \frac{1}{\gamma - 1}\bar{\rho}\left(\overline{u_j T} - \tilde{u}_j\tilde{T}\right). \tag{5}$$

Further details on these terms can be found in Refs. [49–51].

The governing equations, namely, Eqs. (1a)-(1c), were discretized with the cell-centered finite volume method on unstructured hybrid meshes composed of hexahedrons, prisms, tetrahedrons, and pyramids. An in-house CFD solver [44–48] was used to solve Eq. (1). The convective flux terms were computed with second-order Roe discretization schemes, and the viscous flux terms were obtained with a reconstructed central scheme. For the Roe scheme, second-order accuracy was achieved by reconstructing the solution following Frink's interpolation method [52]. The lower-upper symmetric Gauss–Seidel (LU-SGS) relaxation-based implicit backward-Euler scheme was implemented for simulations of steady flows [53,54], and a corresponding second-order fully implicit dual-time scheme [55,56] was adopted for unsteady flows. An adaptive local time stepping method was developed to eliminate the adverse influence of poor-quality grids on the stability and convergence of the solution. In addition, locally adaptive flux blending [57] was implemented in the Roe scheme to ensure the dominance of low dissipation. To allow large-scale computations, this solver was parallelized by using a domain decomposition strategy with an MPI protocol. Nonblocking communications were used to overlap computations with communications to exploit possible performance gains. The details of the numerical methods employed in this work are summarized in Table 1.

Table 1
Details of the numerical methods.

Governing equations	3D, compressible, filtered Navier-Stokes equation
Spatial discretization	Cell-centered FVM
Temporal evolution	LU-SGS
Time marching	2nd-order fully implicit dual-time scheme
Reconstruction	2nd-order Roe scheme for inviscid term 2nd-order central scheme for viscous term

2.1. Subgrid-scale model

The eddy viscosity model proposed by Vreman [58] was employed to close the system of momentum and energy equations. The Vreman subgrid-scale (SGS) model formulates the SGS eddy viscosity μ_t as follows:

$$\mu_t = C\sqrt{\frac{B_\beta}{\alpha_{ij}\alpha_{ij}}}, \tag{6}$$

with

$$B_\beta = \beta_{11}\beta_{22} - \beta_{12}^2 + \beta_{11}\beta_{33} - \beta_{13}^2 + \beta_{22}\beta_{33} - \beta_{23}^2, \tag{7}$$

$$\beta_{ij} = \Delta_m^2 \alpha_{mi}\alpha_{mj}, \tag{8}$$

where, C is a model constant that is specified as $C = 0.06$; $\alpha_{ij} = \partial\tilde{u}_j/\partial x_i$ represents the derivatives of the filtered velocity; and Δ_m is the filter width in the spatial domain. In this paper, the filter width Δ_m was taken as the cubic root of the cell volume.

2.2. Implementation of mesh reordering

The performance of flow solvers for unstructured meshes can be improved by changing the order in which data are accessed. We achieved this improvement by using a Hilbert SFC-based mesh reordering method. The Hilbert SFC-based reordering method generates a Hilbert SFC that passes through each point (the centroid of cells) once, establishing a one-to-one mapping between the index of the curve and the discretized cell in the computational domain. The SFC connects only neighboring mesh cells to cluster the nonzero elements of the system matrix, thereby preserving data locality as much as possible.

Algorithm 1 Implementation of Hilbert SFC-based mesh reordering algorithm.

Step 1	Define an axis-aligned bounding box. A bounding box for the mesh is defined based on the maximum and minimum coordinates of the nodes.
Step 2	Assign each cell coordinate to a Cartesian grid. The bounding box is divided into N bins. These "overflow" bins are subdivided into N equal bins. A specific refinement pattern is followed for each of these basic bins to obtain the next level, and this process is continued until the desired level of refinement is achieved.
Step 3	Generate the Hilbert SFC. A coherent Hilbert SFC can be constructed by joining all local SFCs obtained by scaling, rotating, or reflecting the fundamental elements.
Step 4	Reorder the vertices according to the Hilbert SFC order. The cells are reordered according to the order provided by the Hilbert SFC.

The algorithm for the Hilbert SFC-based mesh reordering method is briefly summarized in **Algorithm 1**. Here, we use a simple mesh [shown in Fig. 1(b)] to illustrate the implementation of this strategy. First, the smallest axis-aligned box that contains all of the point ensembles [e.g., the centroid of the mesh cells in Fig. 1(c)] is defined according to the spatial coordinates of these points, as shown in Fig. 1(d). This bounding box is used to scale the coordinates to a $[0, 1]^d$ unit volume (where $d = 2$ represents the number of dimensions in Fig. 1). Next, all centroids of the mesh cells are assigned to a Cartesian grid in the $[0, 1]^d$ unit volume. To accomplish this, the bounding box is divided into N bins, where $N = 2^d$ and d is the number of spatial dimensions[$N = 4$ for 2D, as shown in Fig. 1(e)]. The left and right endpoints of these bins specify a half-open interval and form a nonoverlapping cover over the bounding box. These bins are ordered such that the curve traverses them with an "⊐" shape, which is referred to as the fundamental pattern of the Hilbert SFC. The "overflow" bins, which are marked in blue in Fig. 1(e), are each subdivided into N equal sub-bins. The four sub-bins in the next level are arranged by rotating or reflecting the fundamental pattern given in Fig. 1(e), allowing their start and end points to be connected. Thus, they respectively take the shapes "⊓", "⊐", "⊏", and "⊔", which are indicated by the bold polygonal lines in Fig. 1(f). Following the spatial refinement for each of these basic bins, we obtain the next level [see Fig. 1(g)], and this process is continued until the desired level of refinement is achieved. In the programming design and implementation, the

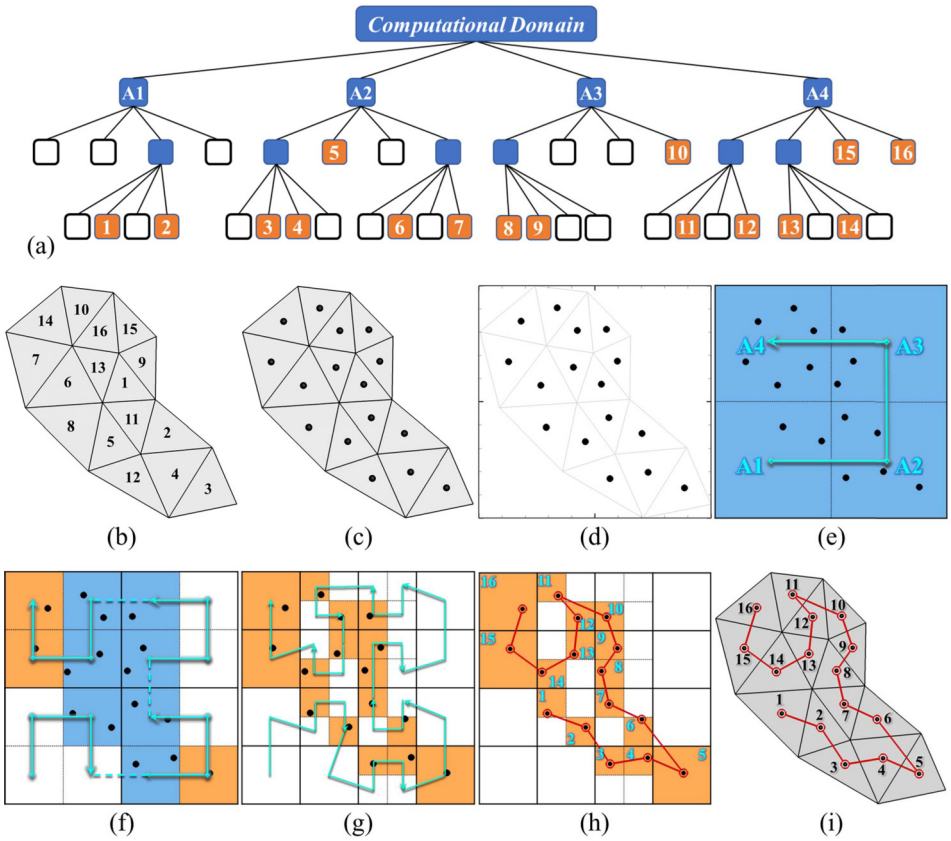


Fig. 1. Recursive generation of the Hilbert SFC (in a 2D domain). (a) Quadtree representation of a given object and the sequential order of the quadtree cells. (b) The objective mesh with the original ordering. (c) The centroid of the mesh cells. (d) Axis-aligned bounding box. (e) Initial coarse grid definition. (f) The second construction steps. The "overflow" bins marked in blue are subdivided into smaller cells, unless the finest resolution marked in orange has already been achieved. (g) In the third construction step, the resolution exactly satisfies the requirements throughout the domain. (h) Recursive generation of the SFC and the sequential orders. (i) The reordering mesh. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

data structure is described by quadtrees, as shown in Fig. 1(a), and each mesh cell corresponds to a node of the tree. The root of the trees represents the computational domain, and the sub-bins of each bin are the children of the respective node. The same idea is followed in the 3D case, where the basic shapes are defined on $2 \times 2 \times 2$ sub-bins. This loop terminates when the resolution satisfies the requirements throughout the domain.

In the third step, we traverse the sub-bins represented by the leaf cells [see Fig. 1(a)]. This traversal is achieved by recursively descending in the tree structure following Hilbert iterations until a leaf cell is reached. After this leaf cell is processed, the traversal ascends up the tree until the first node with a child node that was not visited throughout the traversal is reached. After all the leaf cells are processed, a Hilbert SFC that traverses all sub-bins is obtained as shown in Fig. 1(g). We then remove those "empty" bins and replace the remaining bins with the centroids of the mesh cells. Fig. 1(h) depicts the SFC that traverses all centroids of mesh cells. Further details about the generation of the Hilbert SFC can be found in Ref. [59]. The vertices on the curve are plainly ordered, and this ordering provides guidance for reordering the cells in the mesh. In a mathematical sense, the mapping between the indices of the vertices and cells is bijective [see Fig. 1(i)]. Finally, the cells are reordered according to the mapping, and the reordered mesh is shown in Fig. 1(i). Compared with the original ordering in Fig. 1(b), jumps between cells and their successors in sequential order are effectively eliminated by the Hilbert SFC-based reordering method.

A simple test case is presented to illustrate how the cells in a simple unstructured mesh are reordered. The unstructured mesh consists of 100 triangular cells, and the order of the mesh cells is indicated by the numbers shown in Fig. 2(a). Fig. 2(b) shows a curve that traverses all the cells in the same order as shown in Fig. 2(a). The curves are chaotically displayed in the figure. Fig. 2(c) presents the ordering obtained by the RCM method. Starting from a given cell (the starting point is marked by •), new cells are added in layers by searching the smallest connectivity, and these cells are sorted according to the order of entry. Then, the "front" of the reordered cells advances through the mesh until all cells have been covered. It is clear that the selection of the starting cell has an impact on bandwidth reduction, which has also been reported by Burgess et al. [37]. Fig. 2(d) presents the ordering of cells after Hilbert SFC reordering. The SFC ordering starts in the lower left corner

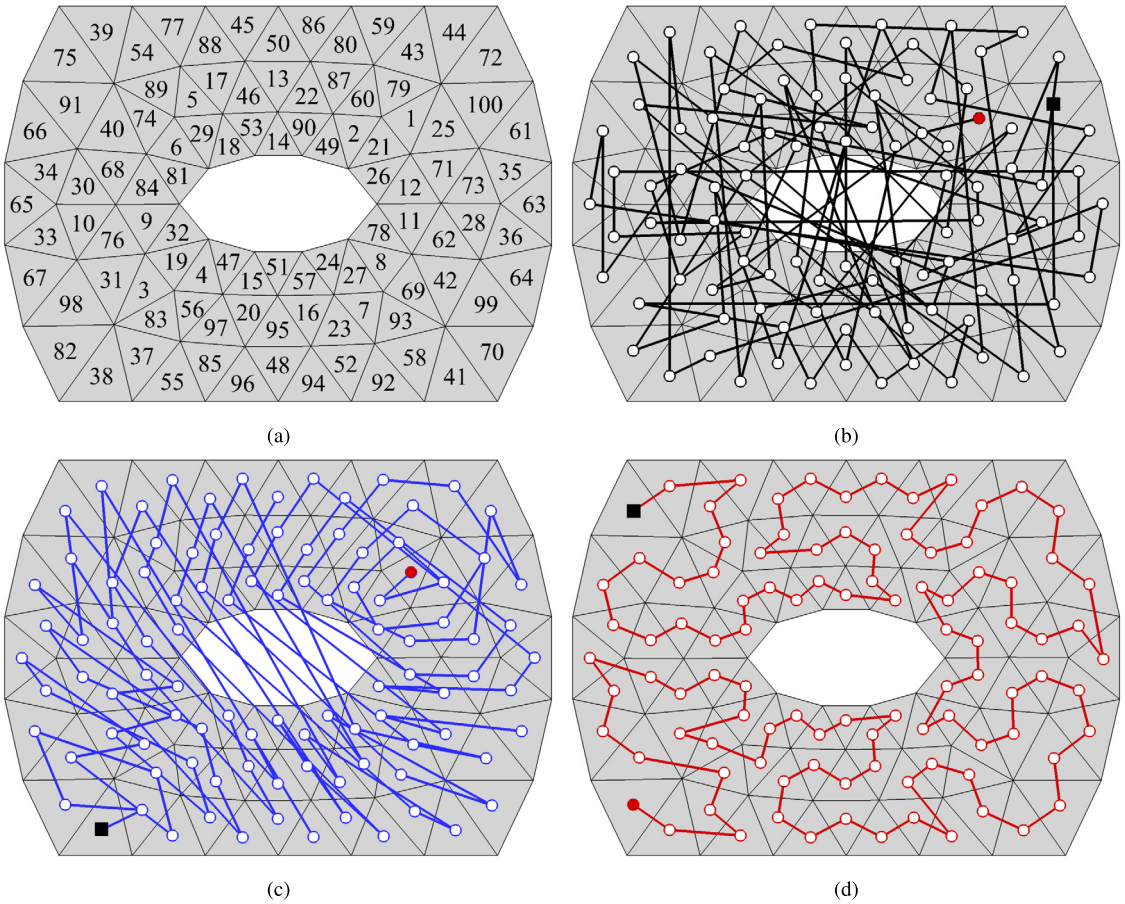


Fig. 2. Schematic of a simple test case. (a) The unstructured mesh, with the cells numbered. (b) Original ordering of the cell coordinates. (c) Ordering of the cell coordinates after RCM method reordering; (d) Ordering of the cell coordinates after Hilbert SFC reordering. ● represents the starting point; ■ represents the ending point; and ○ represents the cell location.

and follows the Hilbert curves to the upper left corner, and the neighboring cells remain as neighboring cells in memory reference. This clustering of data is important for exploiting cache memory during computations.

3. Results and discussion

Two test cases are selected to illustrate the performance of the Hilbert SFC-based reordering method: a 2D unstructured mesh case and a 3D ONERA M6 wing case. We examine the improvement in the data locality and the reduction in the system matrix bandwidth. After the 2D and 3D test cases were validated and their performance was evaluated, the Hilbert SFC-based reordering method was applied to LESs of flows over an axisymmetric body of revolution.

3.1. 2D unstructured mesh test case

A typical unstructured mesh that consists of 26,545 cells and 18,724 nodes is shown in Fig. 3(a). This mesh was generated by the Delaunay method [25] and is locally refined near the "CFD" region, which represents the wall in most simulations. The rest of the computational domain is filled with triangles. A line that traverses all cells according to the ordering of the original mesh given by the mesh generator is presented in Fig. 3(b). The line exhibits disorganization in space, which confirms the irregular ordering of the original mesh.

The application of the RCM method on this mesh is presented in Fig. 3(c). The ordering shows a certain improvement and follows some sort of pattern; however some irregularities remain. A number of spanning "jumps" in the traversal order can be observed between adjacent layers. Since the searching layer is formed by the cells adjacent to the reordered cells, the first cell in the previous level can be connected to any cell in the current level. Thus, the data locality worsens. Fig. 3(d) presents the ordering of the cell coordinates after Hilbert SFC reordering, and the curve connecting the centers of each cell emphasizes their ordering. The ordering starts in the lower left corner and follows the Hilbert curve to the upper left corner. The neighboring cells are mapped to adjacent cells in the index space, ensuring that the data locality is preserved as much

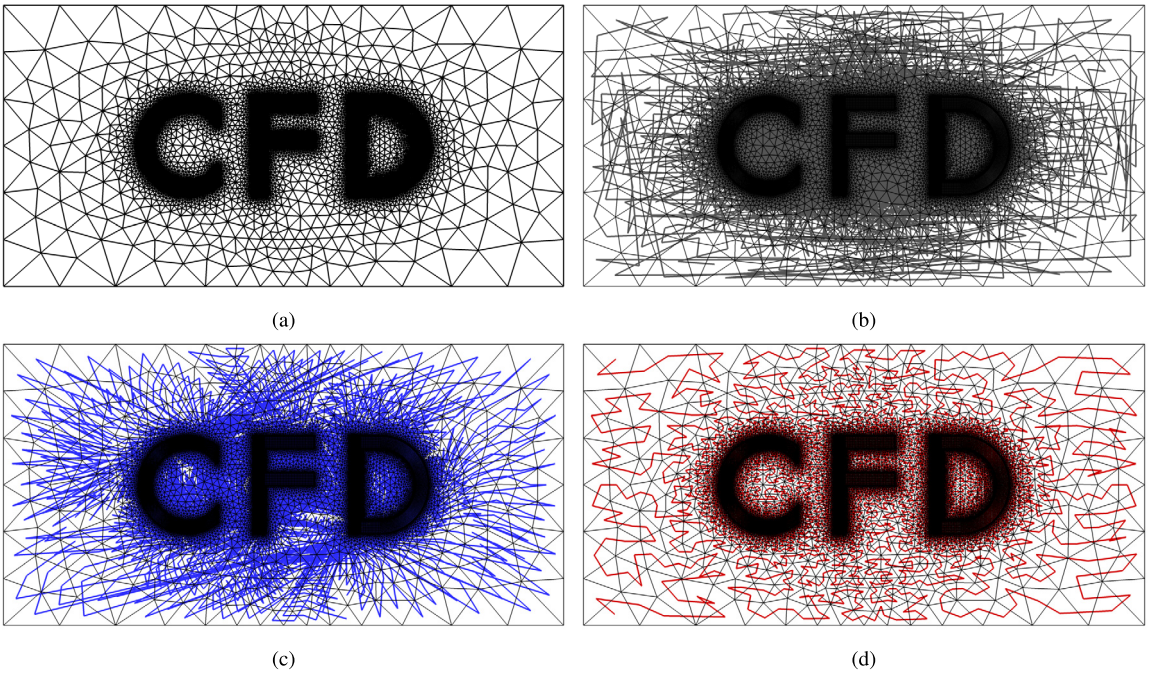


Fig. 3. Schematic of an unstructured mesh and the lines that traverse all cells in the ordering given by the mesh generator, RCM method, and Hilbert SFC-based reordering method. (a) The unstructured mesh that is locally refined in the nearby the "CFD" region. (b) Original ordering of the cells given by the mesh generator. (c) Ordering of the cells after RCM method reordering. (d) Ordering of the cells after Hilbert SFC reordering.

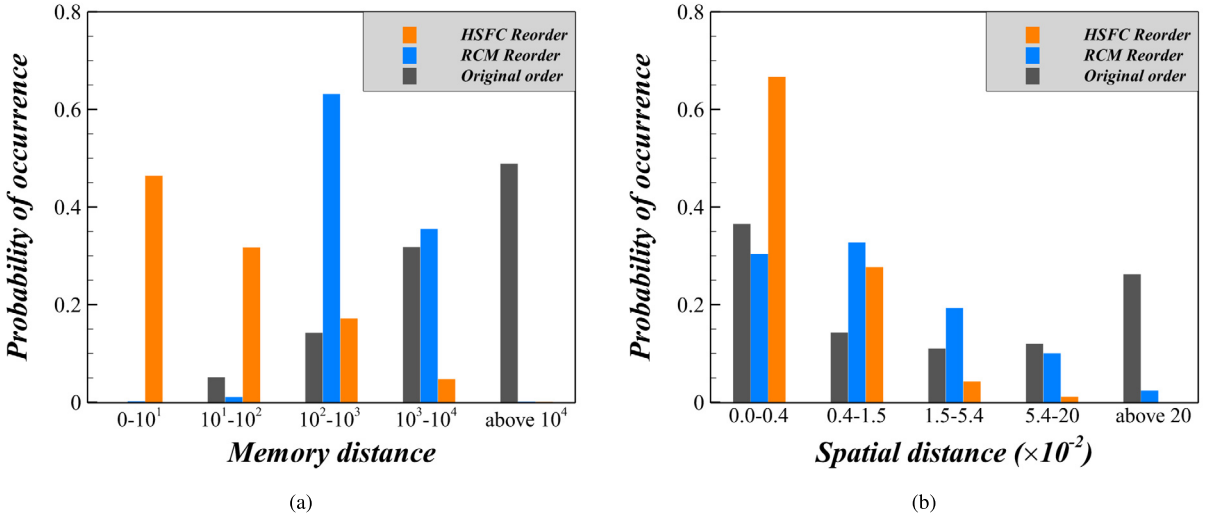


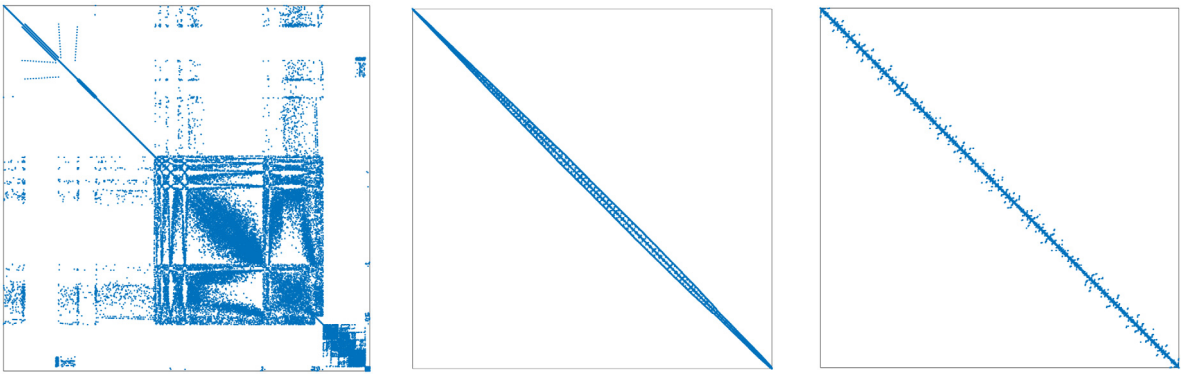
Fig. 4. Data locality measured according to the memory distance and spatial distance. (a) Probability of the memory distance between a cell and its neighboring cells. (b) Probability of the spatial distance between the i-*th* and $(i+1)$-*th* cells in the physical domain.

as possible. Fig. 4(a) and Fig. 4(b) show the data locality measured according to the memory distance and spatial distance. Here, the memory distance (D_m) is defined as the difference between the array indices of neighboring cells ($N(i)$), and the spatial distance (D_s) is defined as the distance between the i-*th* cell and $(i+1)$-*th* cell in the physical domain.

$$D_m(i) = \sum_{j \in N(i)} |i - j|, \tag{9a}$$

$$D_s(i) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}, \tag{9b}$$

The vertical axis in Fig. 4 represents the probability of occurrence, which is defined as the ratio of the number of possible outcomes of an event to the total number of cells. Fig. 4(a) presents the statistical result of the memory distance, demon-



(a) (b) (c)

Fig. 5. Sparse matrix bandwidth (β) reduction with the reverse Cuthill-McKee method and the Hilbert SFC-based reordering method. (a) Original mesh given by the mesh generator, $\beta = 24709$, $\chi = 1706.5$. (b) Reordering by the reverse Cuthill McKee method, $\beta = 521$, $\chi = 235.2$. (c) Reordering by the Hilbert SFC-based method, $\beta = 546$, $\chi = 112.9$.

Table 2
The bandwidth (β) and average bandwidth (χ) of the matrix for each of the mesh orders.

Ordering	Original ordering	RCM ordering	Hilbert SFC ordering
Bandwidth	24709	521	546
Average bandwidth	1706.5	235.2	112.9

strating that the memory distance decreases when the mesh is reordered by the Hilbert SFC-based reordering method. Fig. 4(b) presents the probability distribution of the spatial distance. Since the Hilbert SFC connects neighboring cells, the spatial distance is also reduced by the Hilbert SFC-based reordering method.

In terms of bandwidth reduction, the Hilbert SFC-based reordering method is expected to be as good as the RCM method. The bandwidth β of a matrix is defined by

$$\beta = \max_{a_{ij} \neq 0} |i - j| . \tag{10}$$

Fig. 5 shows the sparsity pattern of the adjacency matrix. The quasi-random distribution of nonzero elements in the original mesh is undesirable, and the original mesh has a bandwidth of 24709 [Fig. 5(a)]. The RCM algorithm produces good numbering and reduces the matrix bandwidth to 521 [Fig. 5(b)]. The test result [Fig. 5(c)] indicates that the Hilbert SFC-based reordering method results in a bandwidth of 546, which is comparable to that of the RCM method. Since the matrix bandwidth does not provide too much information about the grouping pattern of nonzero elements along and around the main diagonal, an indicator of the nonzero element compactness known as the average bandwidth is presented. The average bandwidth is defined as

$$\chi = \frac{1}{m} \sum_{\substack{i,j=1,n \\ a_{ij} \neq 0}} |i - j|, \tag{11}$$

where, m denotes the number of nonzero elements, and i and j represent the row and column indices of the nonzero elements. Table 2 lists the average bandwidth (χ) of the matrix for each of the mesh orders. The adjacency matrix of the original mesh has an average bandwidth of 1706.5, and the RCM algorithm compresses the average bandwidth to 235.2. The Herbert SFC-based reordering method has an average bandwidth of 112.9, which is substantially better than that of the RCM method. In a matrix with the minimum average bandwidth, most nonzero elements are very close to the main diagonal and very few nonzero elements are away from the main diagonal, which is advantageous for large-scale computations when the cache size is limited.

3.2. ONERA M6 wing

As a 3D test case, we examine the flow over an ONERA M6 wing. The most widely computed case corresponds to the conditions of $Ma = 0.84$, $\alpha = 3.06^o$ and $\mathrm{Re}_{MAC} = 11.72 \times 10^6$, where the mean aerodynamic chord (MAC) is 0.801673 m. Computations with the Menter shear stress transport (SST) turbulence model [60] were carried out on a hybrid unstructured mesh, as shown in Fig. 6. The mesh employs a nonreflecting boundary condition at a distance of approximately 500 chord

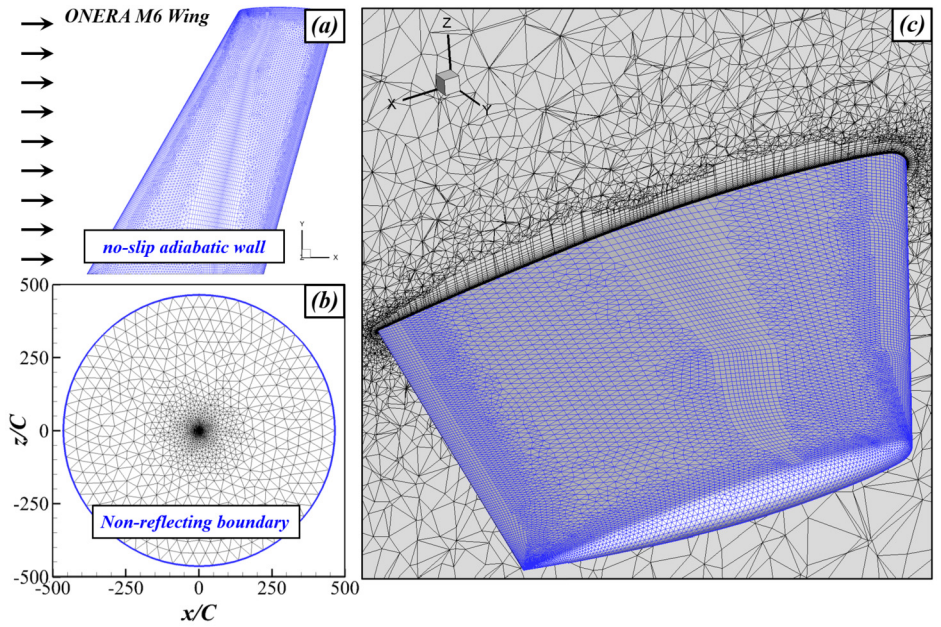


Fig. 6. Computational mesh used for the ONERA M6 wing simulation. (a) Surface grid. (b) Symmetry plane and far-field boundary condition. (c) Zoomed-in view of the volume mesh.

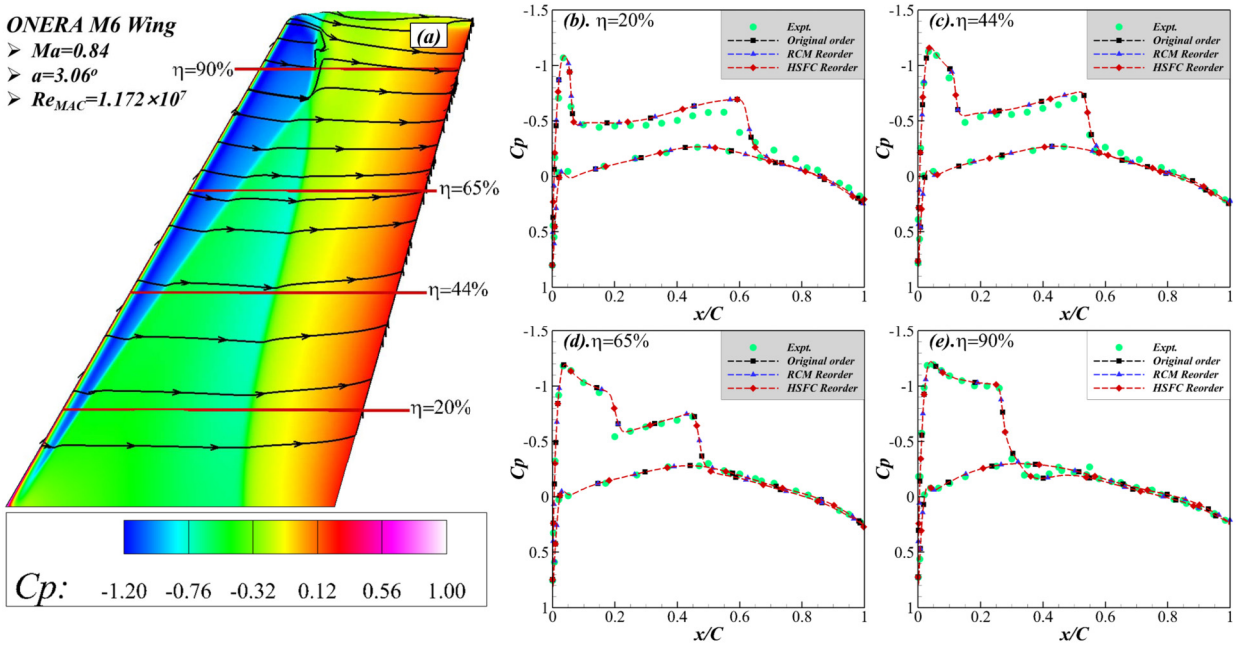


Fig. 7. Pressure coefficient contour on the wing surface for the ONERA M6 wing at $Ma = 0.84$, $\alpha = 3.06^o$ and $\mathrm{Re}_{MAC} = 1.172 \times 10^7$, and a comparison of the computed surface C_p distribution in four outboard sections (a $\eta = 20\%$; b $\eta = 44\%$; c $\eta = 65\%$; d $\eta = 90\%$) with the experimental data.

lengths, a symmetry boundary condition on the inboard wing side, and an adiabatic no-slip wall on the wing surface. The mesh contains a total of 3,627,926 cells and 1,602,721 points and is divided into 20 blocks by the Metis software package [61].

The mesh reordering is performed immediately after the solver is initialized across all processes. Each process is in charge of reordering the local cells. Since the mesh is reordered only once at initialization and without MPI communication, this reordering process has a negligible effect on the overall execution. After the reordering process is completed, simulations are conducted. The results are shown in Fig. 7, and the computed C_p distributions at the $20\% - 90\%$ span in the four sections

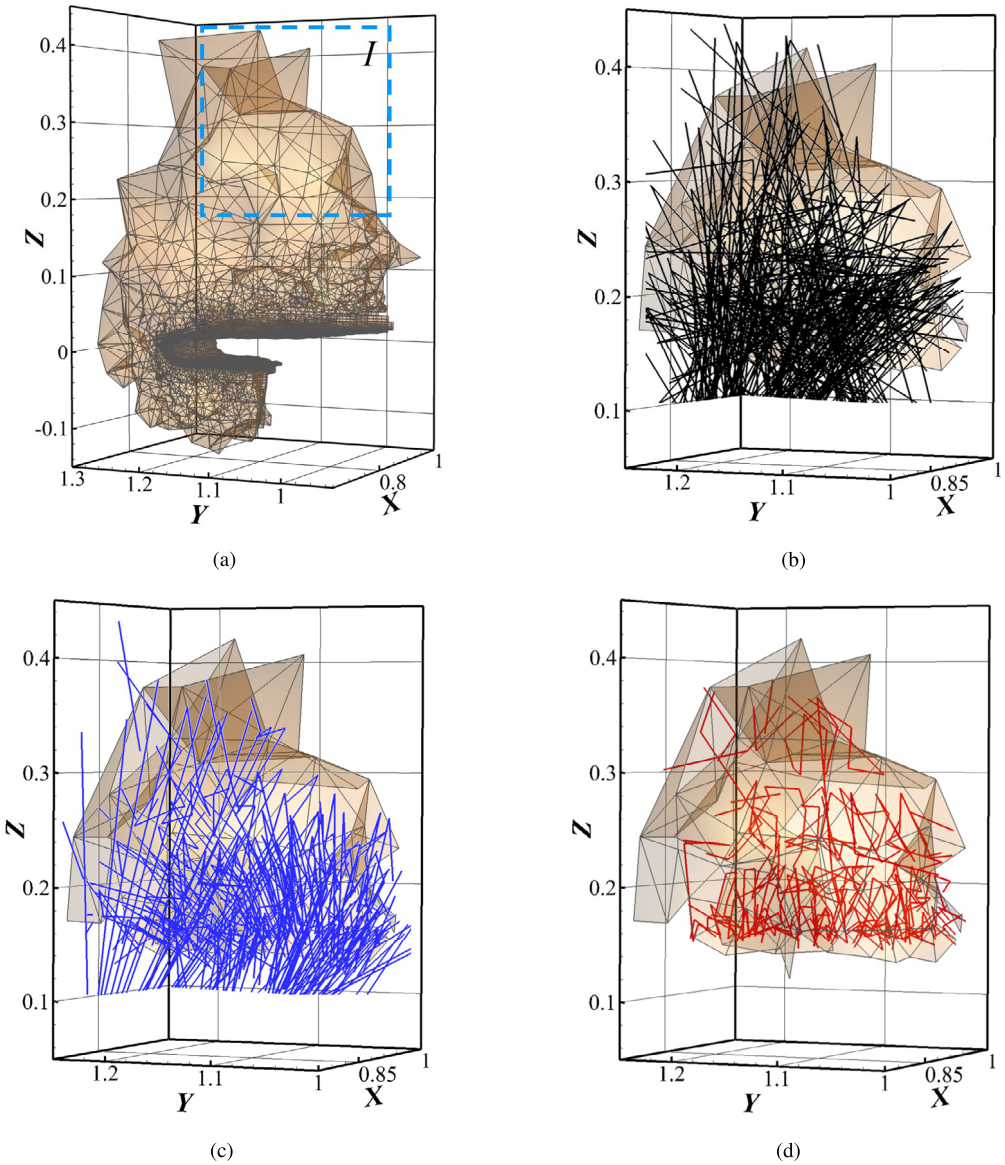


Fig. 8. Tenth part of the unstructured mesh used for the ONERA M6 wing simulation. For easy of reading, the details of mesh in region *I* and its ordering of the cells are shown in the following panels. (a) Hybrid unstructured mesh. (b) Original ordering of the cell coordinates. (c) Ordering of the cell coordinates after RCM method reordering. (d) Ordering of the cell coordinates after Hilbert SFC reordering.

are plotted against the experimental data [62]. For both reordering methods, good agreement is observed throughout the sections, and a sharp shock wave is accurately predicted on the upper surface.

To provide details of the mesh reordering for the 3D cases, we use the 10th part of [Fig. 8 (a)] the 20 blocks as an example for illustration. The orders computed using the RCM and Hilbert SFC-based methods are shown in Fig. 8 (c) and Fig. 8 (d), respectively, with the ghost cells on the interface also included. For easy of reading, the details of mesh in region *I* and its ordering of the cells are shown in the following panels. In Fig. 8 (b) and (c), many curve segments cross the region *I* and connected to the cells that outside the region *I*. A computation loop based on such an ordering will slow down the convergence of iterative inversion methods such as the Gauss–Seidel method [63]. The Hilbert SFC ordering starts at the bottom of the domain in a finely discretized area and ends at the top of the domain. The data locality indicated by the average memory distance and average spatial distance among the 20 blocks is presented in Fig. 9. Compared with the ordering given by the RCM method and the original ordering, the ordering given by the Hilbert SFC-based method significantly reduces these distances. This clustering of data is important for exploiting cache memory during computations because the cells referenced according to the ordering are quasi-contiguous in memory, allowing the computed data to be reused as much as possible. Thus, mesh reordering enables efficient cache traversal during cell looping and leads to performance gains.

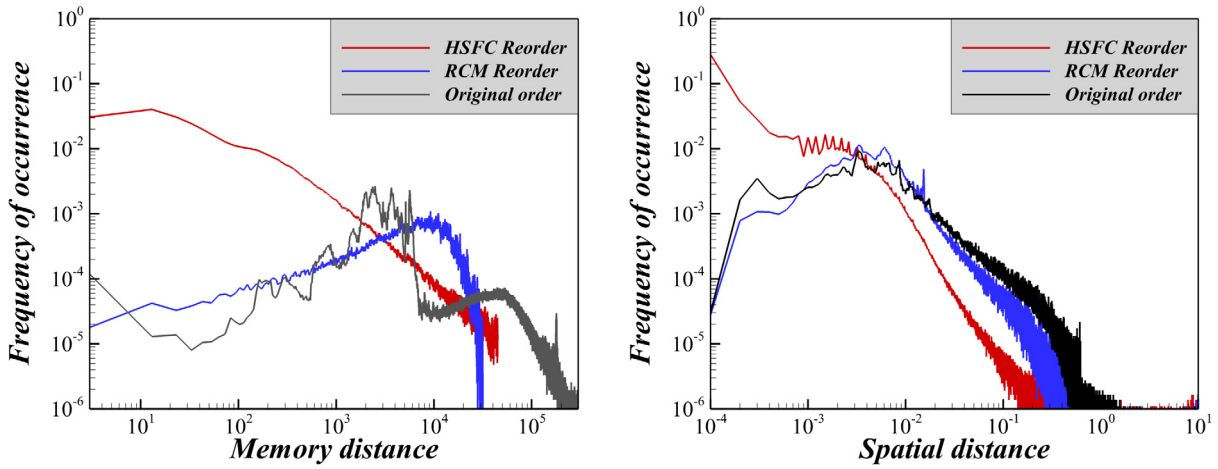


Fig. 9. Data locality indicated by the memory distance and spatial distance. (a) Memory distance between a cell and its neighboring cells. (b) Spatial distance between the $i-th$ and $(i+1)-th$ cells in the physical domain.

Table 3
The total number of last-level cache misses for each mesh order during 100 iterations.

Ordering	LU-SGS[1]	Green-Gauss[2]	Convective flux[3]
Original order	181,964,020	35,362,804	43,795,641
RCM Reorder	103,341,268	23,814,375	30,039,360
Hilbert SFC Reorder	83,671,823	16,287,130	30,133,705

[1] The computation of LU-SGS.
[2] The computation of gradients with the Green-Gauss method.
[3] The computation of the convective flux with the Roe scheme.

Then, we use the memory access analysis tool of the Intel VTune Profiler [64] to identify cache missing. A GNU/Linux system equipped with a dual Intel Xeon Gold 6226R CPU and 32 cores was used to conduct these simulations. All cores were run at 2.9 GHz, and the total cache size was 22 MB. Because the benefits of Hilbert SFC-based reordering may be affected by the computation work per element, three functions with different computational complexities are examined. We isolate the objective function and call it 100 times. Table 3 shows the total number of last-level cache (LLC) misses for each of the mesh orders. The Hilbert SFC-based mesh reordering method significantly reduces the cache misses by 54% for LU-SGS operations, 53.9% for Green-Gauss operations, and 31.2% for convective flux computations. When the two ordering methods are compared, we find that the Hilbert SFC-based reordering method is more effective, especially for LU-SGS operations. This difference indicates that the effectiveness of sorting for improving cache efficiency is related to the computational complexity of the operation. The LU-SGS operation is conducted over all faces of the current cell and neighboring cells with a common face, whereas the convective flux computation is performed only on a face and the two cells that share this face. The higher the computational complexity, the more types and number of elements involved, and the better the improvement in cache utilization with the Hilbert SFC-based reordering method. We also evaluated the improvement in the run time and average cache missing rate due to the reordering on a real system. The run time was provided by the VTune Profiler, and the average cache missing rate is defined as:

$$\text{Cache miss rate} = \frac{1}{N_p}\sum_{i=1}^{N_p}\left[\frac{\text{Total cache misses}}{\text{Total cache accesses}}\right]_i, \tag{12}$$

where N_p represents the total number of parts. The results are shown in Fig. 10. Since the total number of loads and stores from memory depends on the algorithm, the total number of times that the cache is accessed is almost the same for a specified operation. The cache miss rates shown in Fig. 10(a) follow the same trend as the data presented in Table 3. Fig. 10(b) compares the run times of the objective functions. The Hilbert SFC-based mesh reordering method yielded the lowest computing time. When the gradient was calculated by using the Green-Gauss method, we see favorable results, with the execution time of the Green-Gauss functions reduced by 24.2%.

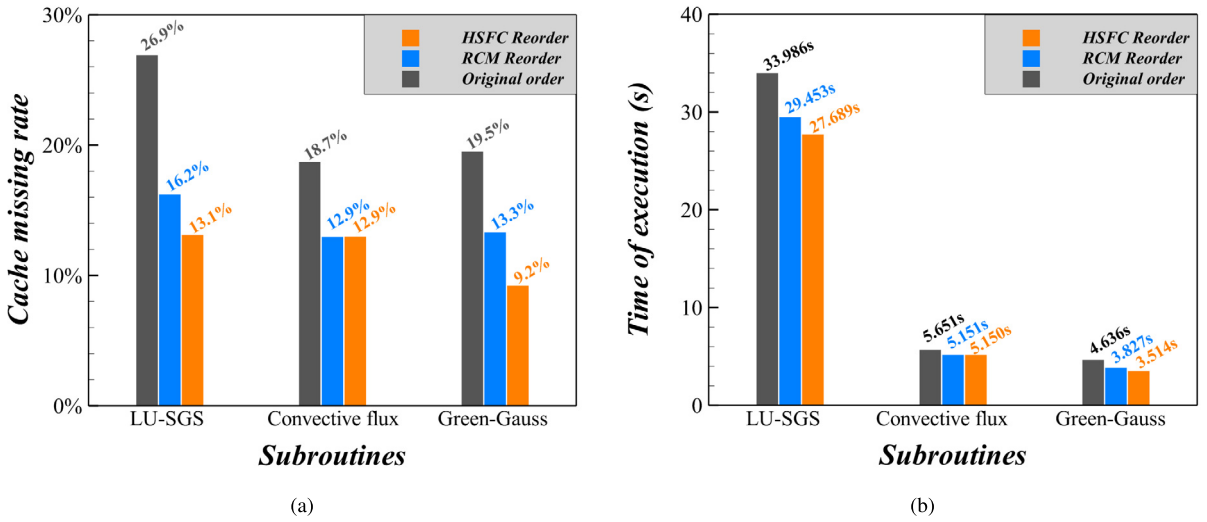


Fig. 10. The averaged cache misses rate (a) and run time (b) of the three functions during 100 computations.

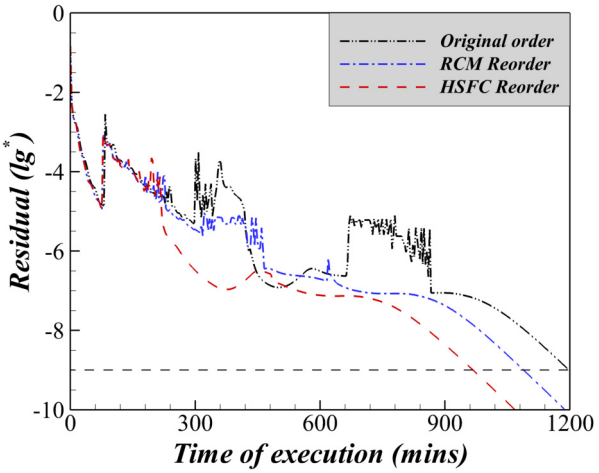


Fig. 11. Convergence histories for the ONERA M6 wing at $Ma = 0.84$, $\alpha = 3.06^o$ and $\mathrm{Re}_{MAC} = 11.72 \times 10^6$ monitored by tracking the l_2-norm residual, which is defined in Eq. (1c) according to the energy equation.

Fig. 11 compares the convergence histories of the three mesh orders for the M6 wing test cases. The convergence of the solution was monitored by tracking the l_2-norm residual which is defined in Eq. (1c) according to the energy equation. All test cases were conducted using a Courant-Friedrichs-Lewy (CFL) number of 10. The residuals are plotted on logarithmic scales, and the execution time is calculated in minutes. Compared to the baseline performance with an unordered mesh, the Hilbert SFC-based method reduces the execution time required for the residual to fall below 10^{-9} by $11.3 - 29.6\%$. The average reduction in execution time is 18.2% over 10 tested cases, and Fig. 11 shows an example in which the execution time is reduced by 18.36%.

3.3. AFF1 SUBOFF

In this section, a wall-resolved LES of the flow over SUBOFF is conducted with the Hilbert SFC-based reordering method. The geometry of SUBOFF without appendages (AFF1), which is composed of a forebody, a parallel middle body, and an afterbody stern, is shown in Fig. 12. The hull length is $L = 14.292$ ft, and the maximum diameter is $D = 1.667$ ft. The freestream conditions are selected according to the numerical studies presented by Posa [1] and Shi [65]; the angle of attack is set as 0^O, and the Reynolds number based on the hull length L is equal to $Re_L = 1.2 \times 10^6$. The computational domain is composed of a hemisphere with a radius of $90D$ in the front and a cylinder that extends downstream by $120D$ from the trailing edge. An unstructured mesh consisting of approximately 1.476 billion cells is generated and partitioned into 12800 subdomains to perform the parallel simulations.

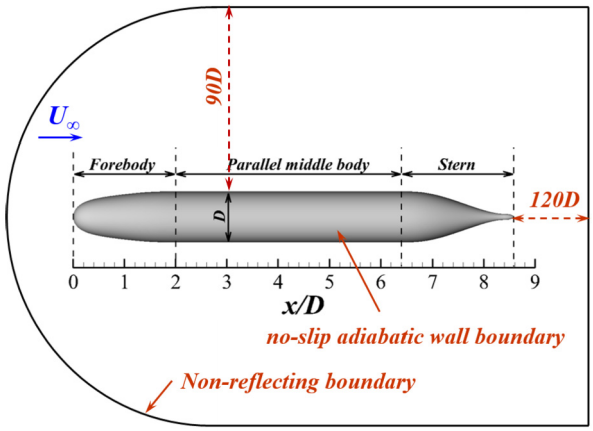


Fig. 12. Schematic of the SUBOFF model without appendages and the computational domain used for the wall-resolved LES.

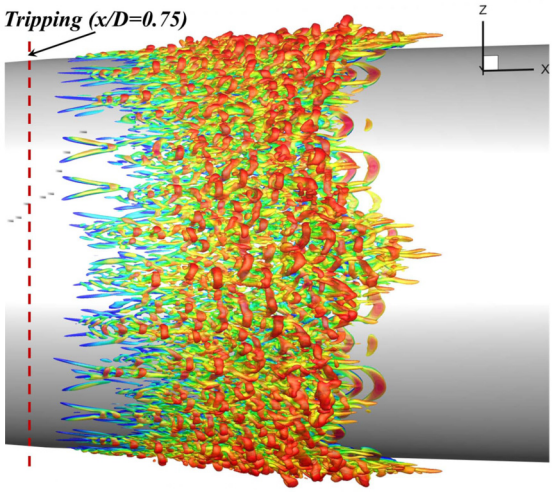


Fig. 13. Near-wall instantaneous flow structures in the initial stage visualized using the Q-criterion isosurface. The boundary layer is tripped at the same location ($x/D = 0.75$) as in the experiments of Jimenez et al. by applying a steady wall-normal velocity of $0.08U_\infty$.

To the best of our knowledge, the size of the unstructured mesh used in present simulation goes beyond what has been reported for LESs of flows over SUBOFF model. The first layer spacing of all grids was chosen such that the average $y^+ \approx 0.6$ and the growth rate in the boundary layer was 1.008. A no-slip adiabatic wall condition was employed on the hull surface, and a nonreflecting boundary condition was implemented at the pressure far-field (PFF) boundary. The nondimensional time step used in the simulations is $\Delta t^* = \Delta t \cdot U_\alpha / L = 3.35 \times 10^{-4}$, where U_α denotes the freestream velocity and Δt is the dimensional time step size. Simulations were conducted on a new-generation domestic TH-3F supercomputer platform with 12,800 cores and the overall computation took approximately 1.15 million CPU hours. Considering that the LES testing is too expensive, we simplify the numerical test in which the unsteady simulations are performed over 100 physical time steps with both reordered and unordered meshes. The results indicate that execution time is reduced by 18% with the Hilbert SFC-based reordering method.

In the simulations, a numerical trip wire was used to force the transition of the boundary layer to occur at the same location as in the experiments of Jimenez et al. [66]. The trip wire in the computations was represented by a wall-normal velocity perturbation and located at a distance $x/D = 0.75$ from the nose. Several perturbations with different amplitudes were tested, and a value of $0.08U_\infty$ was found to be the minimum value that satisfied the requirements. A small steady wall-normal velocity over a few cells around the bow promotes a quick transition of the boundary flow, as shown in Fig. 13. A sequence of vortical structures induced by tripping is convected downstream and becomes complex 3D vortices in the wake. Fig. 14 shows the instantaneous flow structures near the wall visualized by a Q-criterion isosurface [67] after the flow has fully developed. Here, Q is computed as $Q = 0.5(\Omega_{ij}\Omega_{ij} - S_{ij}S_{ij})$, where Ω_{ij} and S_{ij} denote the antisymmetric and symmetric components of ∇u, respectively. A global view of the axial velocity, pressure coefficient and vorticity magnitude field in the XZ plane is shown in Fig. 15, where the contour of the axial velocity [Fig. 15(a)] shows that the axisymmetric turbulent boundary layer gradually thickens after the trip wire is implemented. The axial velocity undergoes a significant

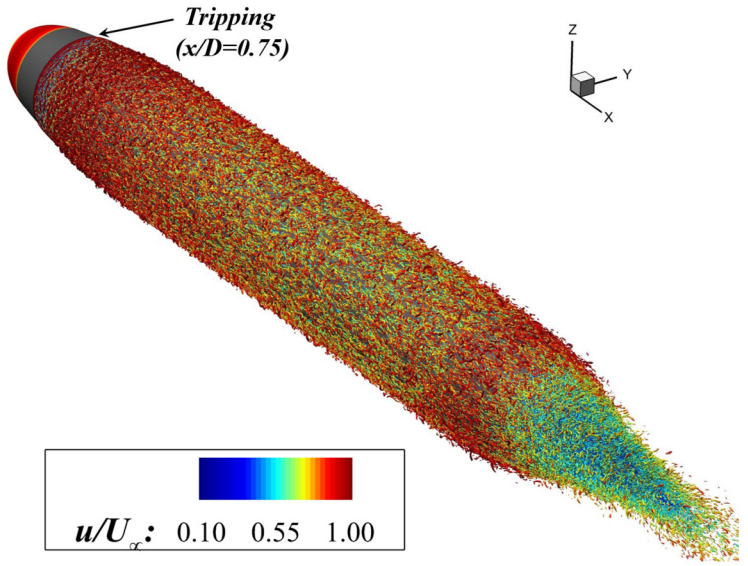


Fig. 14. Instantaneous flow structures near the wall visualized using the Q-criterion isosurface. The boundary layer is tripped at the same location ($x/D = 0.75$) as in the experiments of Jimenez et al. by applying a steady wall-normal velocity of $0.08U_\infty$.

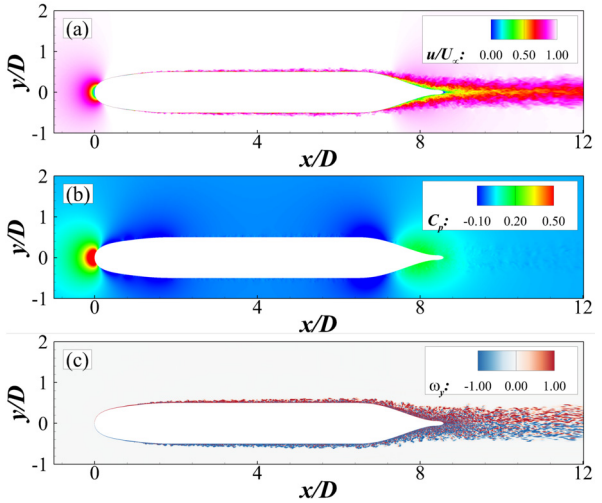


Fig. 15. Instantaneous flow field in the XZ plane. (a) Axial velocity; (b) pressure coefficient; (c) vorticity magnitude.

deceleration due to the reverse pressure gradient on the stern [Fig. 15(b)]. The contour of the vorticity magnitude shown in Fig. 15(c) shows regions with intense turbulent activity near the hull boundary layer and the wake.

After the initial transient stage ended, statistics were gathered over 1.5 flow-through times. A database with 4500 snapshots was collected to compute the statistics, which include the time-averaged pressure coefficient C_p, the skin friction coefficient C_f, and the velocity profiles. The time-averaged pressure (C_p) and skin friction (C_f) coefficients are computed as follows:

$$C_p = \frac{p - p_\infty}{0.5\rho_\infty {u_0}^2} \quad , \quad \text{and} \quad C_f = \frac{\tau_w}{0.5\rho_\infty {u_0}^2} \quad , \tag{13}$$

where p_∞ is the freestream pressure, and ρ_∞ represents the density of the fluid in the far field. τ_w and p represent the time-averaged wall shear stress and pressure, respectively. The predicted hull pressure and skin friction coefficients are compared with the experimental data and the available LES results in Fig. 16. The experimental data reported by Huang et al. [68] and the LES results presented by Posa et al. [1] and Kumar et al. [5] are used in our comparison. Posa and Balaras conducted wall-resolved LESs of SUBOFF with appendages at a Reynolds number of $Re_L = 1.2 \times 10^6$. Kumar and Mahesh also conducted wall-resolved LESs of SUBOFF without appendages at a slightly lower Reynolds number of $Re_L = 1.1 \times 10^6$. The predicted C_p profiles [shown in Fig. 16(a)] are in good agreement with both the measurements by Huang et al. [68]

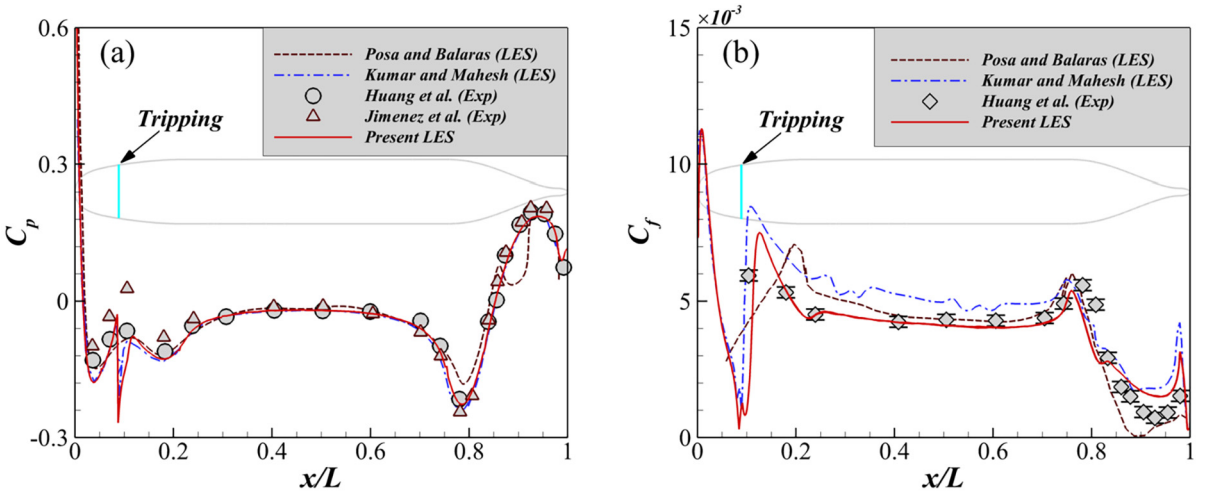


Fig. 16. Distribution of the computed mean pressure C_p (a) and skin friction C_f (b) coefficients on the hull. The symbols represent measurements taken from the experiment by Huang et al. (1992) at $Re_L = 1.2 \times 10^7$, and the C_f measurements are scaled to the Re of the current simulations by using a scaling law ($C_f \sim \mathrm{Re}^{-0.2}$). The LES results presented by Posa et al. (2016) and Kumar et al. (2018) are also shown for comparison.

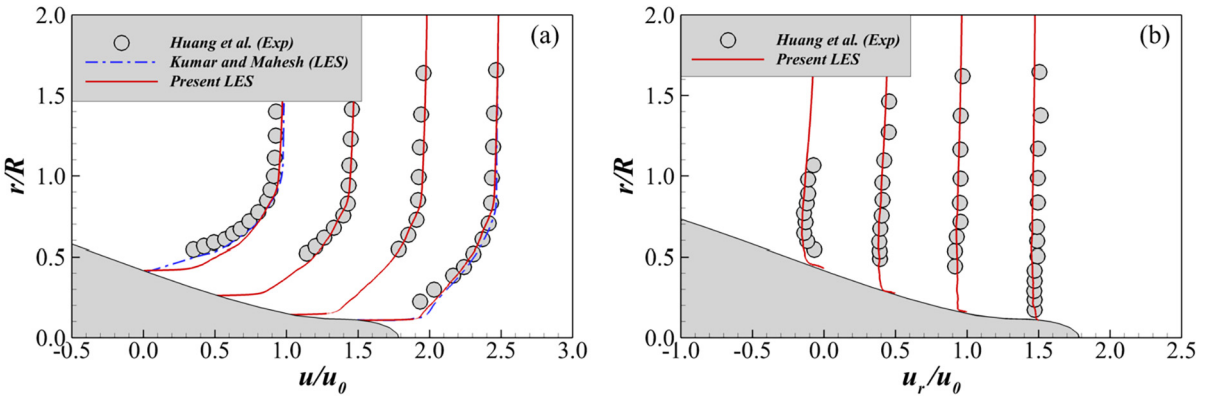


Fig. 17. Time-averaged axial (a) and radial (b) velocity profiles at x/L=0.904, 0.927, 0.956 and 0.978. The circles indicate the measurements obtained by Huang et al. (1992) at $Re_L = 1.2 \times 10^7$. The LES results presented by Mahesh et al. (2018) are also shown for comparison.

and the wall-resolved LES results, including the C_p levels in the forebody and middle sections, as well as the C_p details on the stern. There is a spike in the predicted C_p curve at x/D=0.75 that was induced by the numerical trip wire. Fig. 16(b) presents a comparison of the experimental data, the LES results and the computed C_f along the meridian line of the hull. According to the work of Posa and Balaras [1] and Kumar and Mahesh [5], the skin friction coefficient C_f of the experiments is scaled to the Reynolds number of the simulation using a scaling law ($C_f \sim \mathrm{Re}^{-0.2}$) based on the measurements of Huang et al. [68]. Huang et al. also reported a measurement uncertainty of ± 0.0002 for C_f, which is indicated by the error bars in Fig. 16(b). Qualitatively, the C_f predicted in the current study is consistent with the experimental results and the results of Posa et al. [1] in the middle section. While the numerical results converge in the tapered stern region, they are slightly smaller than the experimental results. It should be noted that the amplitude of the spike in the curve presented by Kumar and Mahesh [5] is higher than that presented in the current study, which may be due to the weaker effect of the numerical trip wire we used, resulting in the difference between the red solid line and the blue dashed-dotted line.

The time-averaged axial and radial velocity profiles at four streamwise locations on the stern, namely, x/L = 0.904, 0.927, 0.956, and 0.978, are compared with the measurements of Huang et al. [68] and the LES results presented by Kumar and Mahesh [5] in Fig. 17(a) and Fig. 17(b), respectively. The development of the boundary layer over the stern is properly reproduced, and the adverse pressure gradient causes the hull boundary layer on the stern to thicken. The velocity deficit increases downstream, as shown in Fig. 17(a). The overall agreement among the prediction results, the available experimental data and the LES results is satisfactory, with the exception of a small underprediction in the velocity deficit at x/L=0.904.

Fig. 18 compares the profiles of the root mean square (rms) velocity with the measurements and LES results at x/L=0.904, 0.927, 0.956 and 0.978. The root mean square velocity in the axial direction is shown in Fig. 18(a), and the root mean square

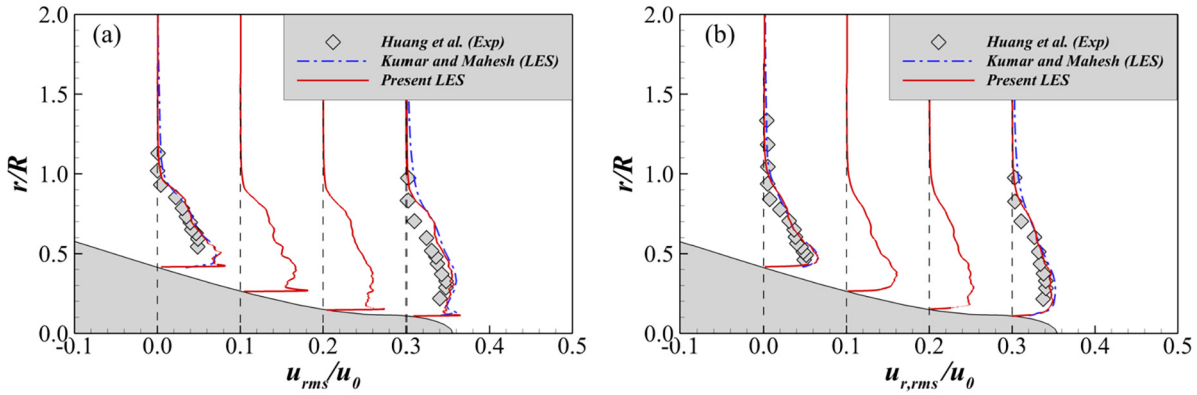


Fig. 18. Profiles of the axial rms velocity (a) and radial rms velocity (b) at x/L=0.904, 0.927, 0.956 and 0.978. The diamonds indicate the measurements obtained by Huang et al. (1992) $Re_L = 1.2 \times 10^7$. The LES results presented by Mahesh et al. (2018) are also shown for comparison.

velocity in the radial direction is shown in Fig. 18(b). The measurements denoted by diamonds were obtained by Huang et al. [68] at $Re_L = 1.2 \times 10^7$, and only two of the four cross-sectional datasets are available. The LES results represented by the dashed line were obtained from the wall-resolved LES data reported by Kumar and Mahesh [5]. Notably, the present results show reasonably good agreement with the measurements and the LES results of Kumar and Mahesh [5] at x/L=0.904 and 0.978. Moreover, the agreement is satisfactory even for fluctuations in the normal velocity, which are challenging to capture.

4. Summary and conclusion

To improve the performance of a flow solver on an unstructured mesh, we propose a cache-efficient mesh reordering method. In the proposed mesh reordering method, a Hilbert SFC that traverses all mesh cells and provides a new ordering is generated. The cells are rearranged according to this ordering, improving data locality and reducing memory latency. This mesh reordering method is effectively implemented and validated using an in-house CFD solver. Then, we use the proposed method to conduct a wall-resolved LES of the flow around SUBOFF at $Re_l = 1.2 \times 10^6$ with zero yaw angle. All essential flow features, including the flow transition induced by numerical tripping and the development of a turbulent boundary layer along the hull, are well resolved by the high-resolution mesh. The results are compared with the results of experiments and the available LES results. The main observations and conclusions are summarized as follows:

(1) Compared with the baseline performance with an unordered mesh, the computational costs of CFD computations with the reordered mesh are significantly reduced. The reordered unstructured mesh cells allow the data to be accessed as continuously as possible when looping over the cells, reducing the number of cache misses and thereby improving the performance of the solver.
(2) We compare the performance of the RCM and Hilbert SFC-based reordering methods in terms of preserving the data locality. The Hilbert SFC-based reordering method is found to generate a better order according to the memory and spatial distances. When the Hilbert SFC-based reordering method is coupled with a CFD solver, the execution time required for convergence in the M6 wing test cases is reduced by $11.3 - 29.6\%$.
(3) Wall-resolved LES of turbulent flows around SUBOFF is conducted on an unstructured mesh with approximately 1.476 billion cells over 12800 CPU cores. To the best of our knowledge, this is the largest number of cells used in LES of turbulent flows over SUBOFF on an unstructured mesh. The development of the boundary layer around the SUBOFF hull is well reproduced by the present wall-resolved LES, which shows a significant deceleration along the stern under an adverse pressure gradient. The quantitative results, including the time-averaged C_p, C_f, mean velocity profiles, and root mean square velocity profiles, are in good agreement with the experimental and LES results. These comparisons demonstrate the capability of the proposed method to conduct large-scale simulations and to accurately predict asymmetric turbulence boundary layers under the influence of a pressure gradient. The overall computational costs in the wall-resolved LESs are reduced by using the proposed Hilbert SFC-based reordering method.

CRediT authorship contribution statement

Yi Liu: Conceptualization, Formal analysis, Methodology, Visualization, Writing – original draft. **Hongping Wang:** Validation, Visualization. **Shizhao Wang:** Conceptualization, Formal analysis, Writing – review & editing. **Guowei He:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

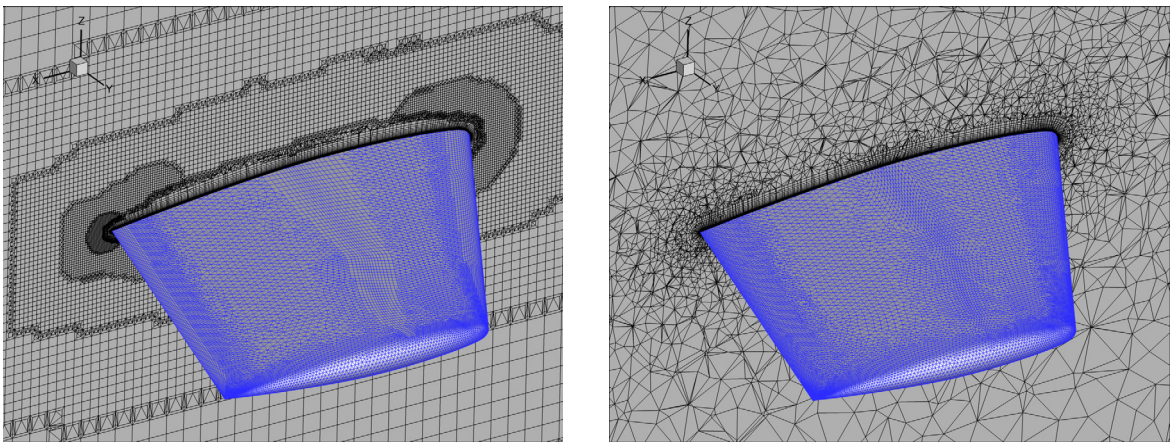
Acknowledgements

This work was supported by the NSFC Basic Science Center Program for "Multiscale Problems in Nonlinear Mechanics" (No. 11988102), the National Natural Science Foundation of China (Nos. 12102439, 11922214, 92252203 and 91952301), and the China Postdoctoral Science Foundation (No. 2021M703290). The computations were conducted on Tianhe-3F at the National Supercomputer Center in Tianjin.

Appendix A. Unstructured meshes generated by different methods

To assess the performance of the proposed Hilbert SFC-based reordering method, we generated two additional meshes for the M6 wing as shown in Fig. A.1. The two hybrid unstructured meshes are generated with the same surface mesh on the wing by the voxel-blocking method and advancing-front method [46], respectively. In Fig. A.1(a), the volume blocks are mainly composed of hierarchical hexahedral cells clustered about and sized by the surface mesh. Transitions between levels in the hexahedral cells are made using tetrahedral cells and pyramids for maintaining the point-to-point connections. The total cell counts of unstructured voxel-mesh were approximately 9.53 million. In Fig. A.1(b), the hybrid unstructured mesh is generated using a bottom-up meshing approach using the advancing-front method. Since the advancing-front generator is only applicable to the surface mesh that uses triangular elements [46], some quadrangular elements at the front, rear, and middle of the wing are diagonalized. The volume mesh will then be generated from this surface mesh, and finally 3.65 million cells are generated.

The proposed Hilbert SFC-based reordering method can improve the data locality of mesh for unstructured meshes generated by both the voxel-type and advancing-front. We divide the two unstructured meshes into 20 blocks to conduct the numerical test. The process, the numerical settings, and the testing platform are the same as in section 3.2. Fig. A.2 plots the data locality of the unstructured mesh generated using the voxel-blocking method. The Hilbert SFC-based reordering method significantly reduces the memory and spatial distance, where the maximum value of the distance is compressed to one-tenth of the original mesh. Fig. A.3 plots the data locality of the unstructured mesh generated using the advancing-front method, which shows the frequency of relatively short memory distances substantially increases after Hilbert SFC reordering. Table A.1 lists the frequency of occurrence that the memory distance falls within the range of 0 to 10^4. For the mesh generated by advancing-front, the Hilbert SFC-based reordering method increases the frequency of memory distance in $[0, 10^4]$ from 9.95% to 86.83%, which is better than that of voxel-type mesh from 24.43% to 80.72%. The improvement might be due to the difference in mesh generation strategies. The voxel-blocking method generates a mesh using a top-down meshing approach. First, a background mesh of hexahedral cells that fills the entire region is created. Then, cell



(a) (b)

Fig. A.1. Computational mesh used for the ONERA M6 wing simulation. (a) Generated by using voxel-blocking method. (b) Generated by using advancing-front method.

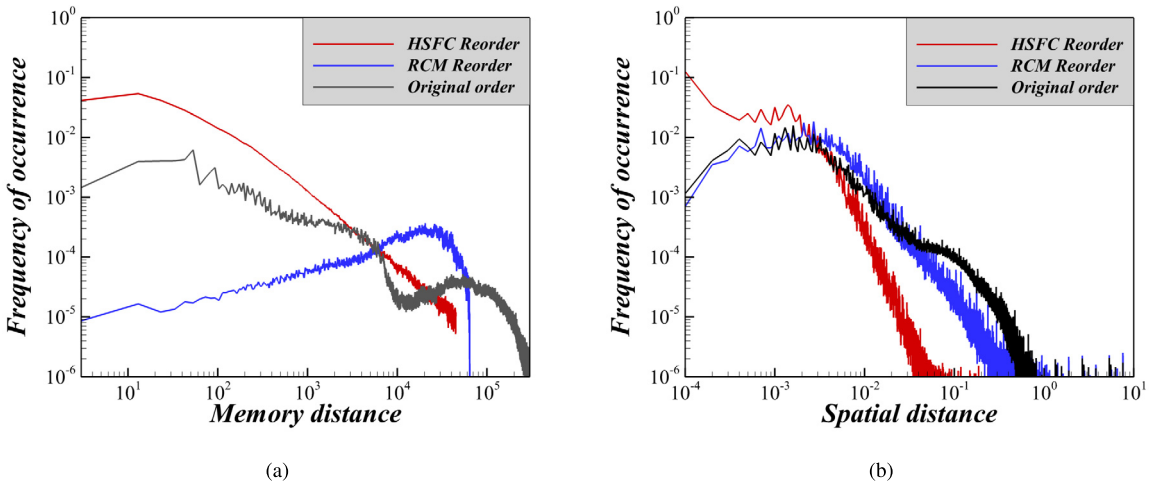


Fig. A.2. Data locality of unstructured voxel-type mesh, which is indicated by the memory distance and spatial distance. (a) Memory distance between a cell and its neighboring cells. (b) Spatial distance between the $i-th$ and $(i+1)-th$ cells in the physical domain.

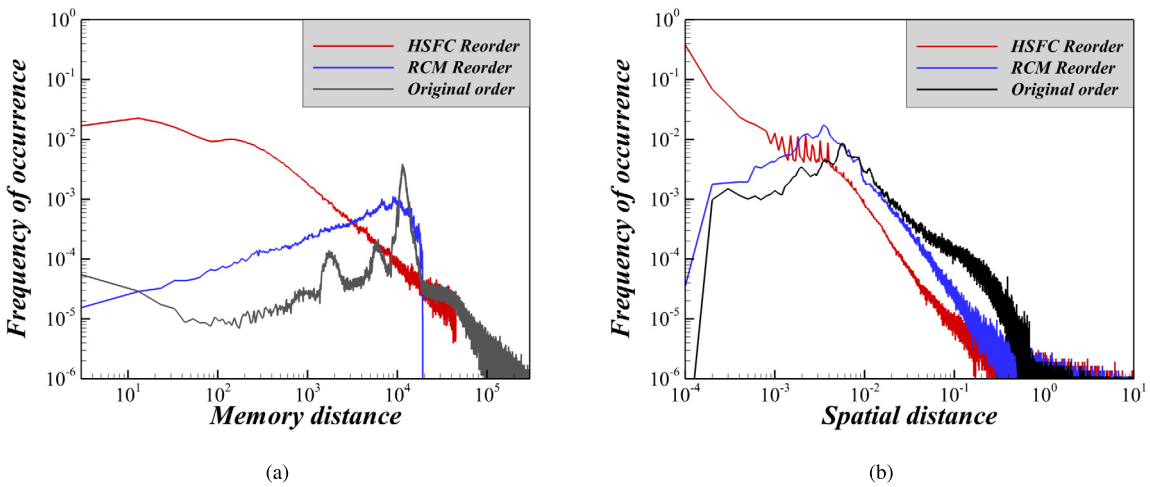


Fig. A.3. Data locality of advancing-front mesh, which is indicated by the memory distance and spatial distance. (a) Memory distance between a cell and its neighboring cells. (b) Spatial distance between the $i-th$ and $(i+1)-th$ cells in the physical domain.

Table A.1
Frequency of occurrence that the memory distance falls in the range of 0 to 10^4.

Mesh type	Original order	RCM reorder	Hilbert SFC reorder
Voxel-type mesh	24.43%	12.35%	80.72%
Advancing-front mesh	9.95%	49.02%	86.83%

splitting is repeatedly performed according to the surface mesh or parameters. In contrast, the advancing-front method adopts a bottom-up meshing approach. The volume mesh is generated from the surface mesh and successively pushes the front layers far away from the surface until it fills the entire computational domain.

The proposed Hilbert SFC-based reordering method can reduce the cache misses for the unstructured meshes generated by both the voxel-blocking method and the advancing-front method. The memory access analyses of two type meshes are summarized in Table A.2, which presents the total number of last-level cache misses and run time for LU-SGS functions during 100 iterations. We find that the proposed Hilbert SFC-based reordering method is still valid for both tested meshes and performs better than the RCM method. In terms of LU-SGS operations, the Hilbert SFC-based reordering method reduces the cache misses by 57.4% in the voxel-type mesh and 53.4% in the advancing front. Furthermore, the execution time of LU-SGS functions decreases with the reduction of cache misses. The Hilbert SFC-based reordering method reduces the execution time by 8.7% in the voxel-type mesh and 15.2% in the advancing-front mesh.

Table A.2
The total number of last-level cache misses and run time for LU-SGS functions during 100 iterations.

Item	Voxel-type mesh		Advancing-front mesh	
	cache misses	run time(s)	cache misses	run time(s)
Original order	557,414,040	82.637	143,510,045	31.861
RCM reorder	379,270,948	79.183	116,383,146	28.367
Hilbert SFC reorder	237,512,120	75.416	66,878,981	26.996

Appendix B. Unstructured mesh with irregularly-shaped cells

Although the Hilbert SFC-based reordering method works well with the three commonly used meshes, it also suffers degraded performance when dealing with irregularly-shaped cells. Similar performance degradation problems were reported in Ref. [39] and Ref. [69]. Ref. [39] presented a heuristic that improves partition locality in arbitrary geometries by using a Morton order curve, and Ref. [69] used adaptive Hilbert curve algorithm to enhance the efficiency of searching several adjacent grids. To illustrate the irregular cells related problem, we test a mesh with elongated triangle cells [in Fig. B.1(a)] and compare it with a mesh consisting of multiple identical triangle cells [in Fig. B.1(b)]. Fig. B.1(c) presents the probability of the memory distance over the specified interval, indicating that the irregularly-shaped cells increase the likelihood of longer memory distance.

Further, we estimate the upper bound of the memory distance for a mesh with irregularly-shape cells using the resolution requirement of the Hilbert SFC. Considering a computational domain Ω_i that contains the i–th mesh cell and all its neighbors $N(i)$, we define the maximum and minimum spatial distance of i–th mesh cell as $D_{s,\max}$ and $D_{s,\min}$ according to Eq. (9b).

$$D_{s,\max}(i) = \max\left[D_s(j)\right], \quad j \in N(i), \tag{B.1}$$

$$D_{s,\min}(i) = \min\left[D_s(j)\right], \quad j \in N(i). \tag{B.2}$$

Then, we define the maximum ($L_{\max}$) and minimum ($L_{\min}$) characteristic lengths of those cells in Ω_i

$$L_{\max} = \max\left[D_{s,\max}(i)\right], \quad i \in \Omega_i, \tag{B.3}$$

$$L_{\min} = \min\left[D_{s,\max}(i)\right], \quad i \in \Omega_i. \tag{B.4}$$

For this n–th refinement level, the length of the nested intervals of the Hilbert curve construction is 2^{-n}. The smallest bins that cover all the center of cells belonging to the Ω_i should satisfy

$$2^{-(n+1)} < L_{\max} < 2^{-n}. \tag{B.5}$$

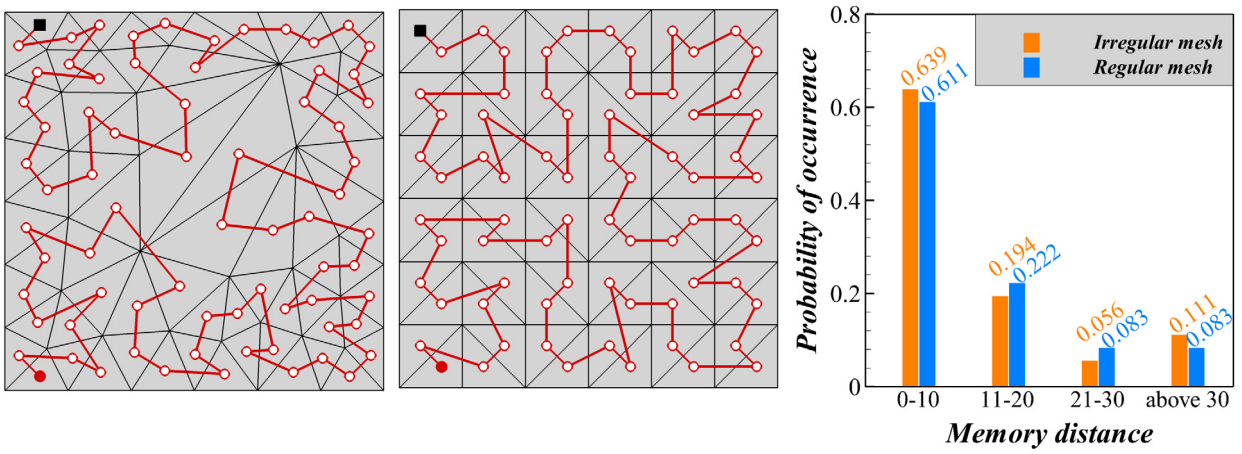


Fig. B.1. Performance of the proposed Hilbert SFC-based reordering method on a mesh with irregularly-shaped cells. (a) An unstructured mesh with elongated triangle cells, and the ordering of the cell after Hilbert SFC reordering. (b) An unstructured mesh consisting of multiple identical triangle cells, and the ordering of the cell after Hilbert SFC reordering. (c) Probability of the memory distance between a cell and its neighboring cells. ● represents the starting point; ■ represents the ending point; and ○ represents the cell location.

The highest resolution should be able to distinguish the minimum spatial distance

$$2^{-(m+1)} < L_{\min} < 2^{-m}. \tag{B.6}$$

Therefore, we obtain an upper bound for the memory distance between the i–th cell and its neighbors: $D_m < 3 \cdot 4^{(m-n)}$ (for the 2D case). Combining Eq. (B.5) and Eq. (B.6), we find that an increase in the ratio of $L_{\max}/L_{\min}$ would result in a dramatic increase in memory distance. Since the $L_{\max}/L_{\min}$ is an estimate of the skewness, considerable value of $L_{\max}/L_{\min}$ is often occurring in the region where the irregularly-shaped cells appear. The above analysis shows that the proposed Hilbert SFC-based reordering method will achieve the best performance in uniformly distributed meshes and degrade in meshes with irregularly-shaped cells.

References

[1] A. Posa, E. Balaras, A numerical investigation of the wake of an axisymmetric body with appendages, J. Fluid Mech. 792 (2016) 470–498, https://doi.org/10.1017/jfm.2016.47.

[2] D. Zhou, K. Wang, M. Wang, Large-eddy simulation of an axisymmetric boundary layer on a body of revolution, in: AIAA Aviation 2020 Forum, 2020, p. 2989.

[3] S.T. Bose, G.I. Park, Wall-modeled large-eddy simulation for complex turbulent flows, Annu. Rev. Fluid Mech. 50 (2018) 535–561, https://doi.org/10.1146/annurev-fluid-122316-045241.

[4] H. Choi, P. Moin, Grid-point requirements for large eddy simulation: Chapman's estimates revisited, Phys. Fluids 24 (2012) 011702, https://doi.org/10.1063/1.3676783.

[5] P. Kumar, K. Mahesh, Large-eddy simulation of flow over an axisymmetric body of revolution, J. Fluid Mech. 853 (2018) 537–563, https://doi.org/10.1017/jfm.2018.585.

[6] N. Morse, K. Mahesh, Large-eddy simulation and streamline coordinate analysis of flow over an axisymmetric hull, J. Fluid Mech. 926 (2021), https://doi.org/10.1017/jfm.2021.714.

[7] Y. Du, J.A. Ekaterinaris, Time-marching schemes for spatially high order accurate discretizations of the Euler and Navier–Stokes equations, Prog. Aerosp. Sci. 130 (2022) 100795, https://doi.org/10.1016/j.paerosci.2021.100795.

[8] A. Posa, E. Balaras, A numerical investigation about the effects of Reynolds number on the flow around an appended axisymmetric body of revolution, J. Fluid Mech. 884 (2020), https://doi.org/10.1017/jfm.2019.961.

[9] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261, https://doi.org/10.1146/annurev.fluid.37.061903.175743.

[10] D. Mavriplis, Unstructured grid techniques, Annu. Rev. Fluid Mech. 29 (1997) 473–514, https://doi.org/10.1146/annurev.fluid.29.1.473.

[11] F. Günther, M. Mehl, M. Pögl, C. Zenger, A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves, SIAM J. Sci. Comput. 28 (2006) 1634–1650, https://doi.org/10.1137/040604078.

[12] R. Aubry, G. Houzeaux, M. Vazquez, J. Cela, Some useful strategies for unstructured edge-based solvers on shared memory machines, Int. J. Numer. Methods Eng. 85 (2011) 537–561, https://doi.org/10.1002/nme.2973.

[13] I. Hadade, F. Wang, M. Carnevale, L. di Mare, Some useful optimisations for unstructured computational fluid dynamics codes on multicore and manycore architectures, Comput. Phys. Commun. 235 (2019) 305–323, https://doi.org/10.1016/j.cpc.2018.07.001.

[14] G. Steinmacher-Burow, Some Challenges on Road from Petascale to Exascale [EB/DB], 2010, p. 4.

[15] J.P. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, D.J. Mavriplis, CFD vision 2030 study: a path to revolutionary computational aerosciences, Technical Report, Langley Research Center, Hampton, Virginia, USA, 2014.

[16] Y. Liu, C. Hu, C. Zhao, Efficient parallel implementation of Ewald summation in molecular dynamics simulations on multi-core platforms, Comput. Phys. Commun. 182 (2011) 1111–1119, https://doi.org/10.1016/j.cpc.2011.01.007.

[17] A. Srivatsa, N. Fasfous, N.A.V. Doan, S. Nagel, T. Wild, A. Herkersdorf, Exploring a hybrid voting-based eviction policy for caches and sparse directories on manycore architectures, Microprocess. Microsyst. (2021) 104384, https://doi.org/10.1016/j.micpro.2021.104384.

[18] S. Akkurt, F. Witherden, P. Vincent, Cache blocking strategies applied to flux reconstruction, Comput. Phys. Commun. (2021) 108193, https://doi.org/10.1016/j.cpc.2021.108193.

[19] R.J. Thacker, H.M. Couchman, A parallel adaptive P3M code with hierarchical particle reordering, Comput. Phys. Commun. 174 (2006) 540–554, https://doi.org/10.1016/j.cpc.2005.12.001.

[20] D.I. Lyakh, An efficient tensor transpose algorithm for multicore CPU, Intel Xeon Phi, and NVidia Tesla GPU, Comput. Phys. Commun. 189 (2015) 84–91, https://doi.org/10.1016/j.cpc.2014.12.013.

[21] A. Bocharov, N.M. Evstigneev, V. Petrovskiy, O.I. Ryabkov, I. Teplyakov, Implicit method for the solution of supersonic and hypersonic 3D flow problems with Lower-Upper Symmetric-Gauss-Seidel preconditioner on multiple graphics processing units, J. Comput. Phys. 406 (2020) 109189, https://doi.org/10.1016/j.jcp.2019.109189.

[22] G. Mudalige, M. Giles, I. Reguly, C. Bertolli, P. Kelly, Op2: an active library framework for solving unstructured mesh-based applications on multi-core and many-core architectures, in: 2012 Innovative Parallel Computing (InPar), IEEE, 2012, pp. 1–12.

[23] D. Mudigere, S. Sridharan, A. Deshpande, J. Park, A. Heinecke, M. Smelyanskiy, B. Kaul, P. Dubey, D. Kaushik, D. Keyes, Exploring shared-memory optimizations for an unstructured mesh CFD application on modern parallel systems, in: 2015 IEEE International Parallel and Distributed Processing Symposium, IEEE, 2015, pp. 723–732.

[24] T.D. Economon, D. Mudigere, G. Bansal, A. Heinecke, F. Palacios, J. Park, M. Smelyanskiy, J.J. Alonso, P. Dubey, Performance optimizations for scalable implicit RANS calculations with SU2, Comput. Fluids 129 (2016) 146–158, https://doi.org/10.1016/j.compfluid.2016.02.003.

[25] N.P. Weatherill, O. Hassan, Efficient three-dimensional grid generation using the Delaunay triangulation, Comput. Fluid Dynam. 92 (1992) 961–968, https://doi.org/10.1002/nme.1620371203.

[26] R. Löhner, Renumbering strategies for unstructured-grid solvers operating on shared-memory, cache-based parallel machines, Comput. Methods Appl. Mech. Eng. 163 (1998) 95–109, https://doi.org/10.1016/S0045-7825(98)00005-X.

[27] M. Cheng, G. Wang, H.H. Mian, Reordering of hybrid unstructured grids for an implicit Navier-Stokes solver based on openMP parallelization, Comput. Fluids 110 (2015) 245–253, https://doi.org/10.1016/j.compfluid.2014.05.003.

[28] N.E. Gibbs, W.G. Poole Jr, P.K. Stockmeyer, An algorithm for reducing the bandwidth and profile of a sparse matrix, SIAM J. Numer. Anal. 13 (1976) 236–250, https://doi.org/10.1137/0713023.

[29] J. Blazek, Computational Fluid Dynamics: Principles and Applications, third ed., Joe Hayton, Sankt Augustin, Germany, 2015.

[30] S. Akkurt, M. Sahin, An efficient edge based data structure for the compressible Reynolds-averaged Navier-Stokes equations on hybrid unstructured meshes, Int. J. Numer. Methods Fluids (2021), https://doi.org/10.1002/fld.5045.

[31] I.S. Duff, G.A. Meurant, The effect of ordering on preconditioned conjugate gradients, BIT Numer. Math. 29 (1989) 635–657, https://doi.org/10.1007/BF01932738.
[32] S.-B. Shi, W. Shao, J. Ma, C. Jin, X.-H. Wang, Newmark-Beta-FDTD method for super-resolution analysis of time reversal waves, J. Comput. Phys. 345 (2017) 475–483, https://doi.org/10.1016/j.jcp.2017.05.036.
[33] L.T. Diosady, D.L. Darmofal, Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations, J. Comput. Phys. 228 (2009) 3917–3935, https://doi.org/10.1016/j.jcp.2009.02.035.
[34] N.H. Mathews, N. Flyer, S.E. Gibson, Solving 3D magnetohydrostatics with RBF-FD: applications to the solar corona, J. Comput. Phys. 462 (2022) 111214, https://doi.org/10.1016/j.jcp.2022.111214.
[35] A. Nejat, C. Ollivier-Gooch, Effect of discretization order on preconditioning and convergence of a high-order unstructured Newton-GMRES solver for the Euler equations, J. Comput. Phys. 227 (2008) 2366–2386, https://doi.org/10.1016/j.jcp.2007.10.024.
[36] R. Löhner, Some useful renumbering strategies for unstructured grids, Int. J. Numer. Methods Eng. 36 (1993) 3259–3270, https://doi.org/10.1002/nme.1620361904.
[37] D. Burgess, M.B. Giles, Renumbering unstructured grids to improve the performance of codes on hierarchical memory machines, Adv. Eng. Softw. 28 (1997) 189–201, https://doi.org/10.1016/S0965-9978(96)00039-7.
[38] P.J. Denning, The locality principle, in: Communication Networks and Computer Systems: A Tribute to Professor Erol Gelenbe, World Scientific, 2006, pp. 43–67.
[39] G.V. Nivarti, M.M. Salehi, W.K. Bushe, A mesh partitioning algorithm for preserving spatial locality in arbitrary geometries, J. Comput. Phys. 281 (2015) 352–364, https://doi.org/10.1016/j.jcp.2014.10.022.
[40] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1998) 359–392, https://doi.org/10.1137/S1064827595287997.
[41] M. Bader, Space-Filling Curves: An Introduction with Applications in Scientific Computing, vol. 9, Springer Science & Business Media, 2012.
[42] A.-J.N. Yzelman, D. Roose, High-level strategies for parallel shared-memory sparse matrix-vector multiplication, IEEE Trans. Parallel Distrib. Syst. 25 (2013) 116–125, https://doi.org/10.1109/TPDS.2013.31.
[43] R. Borrell, J.C. Cajas, D. Mira, A. Taha, S. Koric, M. Vázquez, G. Houzeaux, Parallel mesh partitioning based on space filling curves, Comput. Fluids 173 (2018) 264–272, https://doi.org/10.1016/j.compfluid.2018.01.040.
[44] Y. Liu, G. Wang, H. Zhu, Z. Ye, Numerical analysis of transonic buffet flow around a hammerhead payload fairing, Aerosp. Sci. Technol. 84 (2019) 604–619, https://doi.org/10.1016/j.ast.2018.11.002.
[45] Y. Liu, Turbulence modeling and its uncertainty quantification for complex aerodynamic flows, Ph.D. thesis, Northwestern Polytechnical University, 2020.
[46] G. Wang, Z.-Y. Ye, Mixed element type unstructured grid generation and its application to viscous flow simulation, in: 24th International Congress of Aeronautical Sciences, Yokohama, Japan, 2004, pp. 1–8.
[47] Y. Liu, G. Wang, Z. Ye, Dynamic mode extrapolation to improve the efficiency of dual time stepping method, J. Comput. Phys. 352 (2018) 190–212, https://doi.org/10.1016/j.jcp.2017.09.043.
[48] Y. Liu, Z. Zhou, L. Zhu, S. Wang, Numerical investigation of flows around an axisymmetric body of revolution by using Reynolds-stress model based hybrid Reynolds-averaged Navier-Stokes/large eddy simulation, Phys. Fluids 33 (2021) 085115, https://doi.org/10.1063/5.0058016.
[49] E. Garnier, N. Adams, P. Sagaut, Large Eddy Simulation for Compressible Flows, Springer Science & Business Media, 2009.
[50] U. Piomelli, Large-eddy simulation: achievements and challenges, Prog. Aerosp. Sci. 35 (1999) 335–362, https://doi.org/10.1016/S0376-0421(98)00014-1.
[51] P. Moin, K. Squires, W. Cabot, S. Lee, A dynamic subgrid-scale model for compressible turbulence and scalar transport, Phys. Fluids A, Fluid Dyn. 3 (1991) 2746–2757, https://doi.org/10.1063/1.858164.
[52] N.T. Frink, Upwind scheme for solving the Euler equations on unstructured tetrahedral meshes, AIAA J. 30 (1992) 70–77, https://doi.org/10.2514/3.10884.
[53] S. Yoon, A. Jameson, Lower-upper symmetric-Gauss-Seidel method for the Euler and Navier-Stokes equations, AIAA J. 26 (1988) 1025–1026, https://doi.org/10.2514/3.10007.
[54] G. Wang, H.H. Mian, Y. Liu, Z. Ye, A hybrid implicit scheme for solving Navier-Stokes equations, Int. J. Numer. Methods Fluids 78 (2015) 319–334, https://doi.org/10.1002/fld.4019.
[55] A. Jameson, Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings, in: 10th Computational Fluid Dynamics Conference, 1991, p. 1596.
[56] V. Venkatakrishnan, D. Mavriplis, Implicit method for the computation of unsteady flows on unstructured grids, J. Comput. Phys. 127 (1996) 380–397, https://doi.org/10.1006/jcph.1996.0182.
[57] F. Ducros, V. Ferrand, F. Nicoud, C. Weber, D. Darracq, C. Gacherieu, T. Poinsot, Large-eddy simulation of the shock/turbulence interaction, J. Comput. Phys. 152 (1999) 517–549, https://doi.org/10.1006/jcph.1999.6238.
[58] A. Vreman, An eddy-viscosity subgrid-scale model for turbulent shear flow: algebraic theory and applications, Phys. Fluids 16 (2004) 3670–3681, https://doi.org/10.1063/1.1785131.
[59] M. Bader, Space-Filling Curves: an Introduction with Applications in Scientific Computing, vol. 9, Springer Science & Business Media, Berlin, Heidelberg, 2012.
[60] F.R. Menter, Two-equation eddy-viscosity turbulence models for engineering applications, AIAA J. 32 (1994) 1598–1605, https://doi.org/10.2514/3.12149.
[61] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1998) 359–392, https://doi.org/10.1137/S1064827595287997.
[62] V. Schmitt, Pressure distributions on the ONERA M6-wing at transonic Mach numbers, experimental data base for computer program assessment, AGARD AR-138, 1979.
[63] J.-L. Zhang, Z.-H. Ma, H.-Q. Chen, C. Cao, A GPU-accelerated implicit meshless method for compressible flows, J. Comput. Phys. 360 (2018) 39–56, https://doi.org/10.1016/j.jcp.2018.01.037.
[64] Intel, Intel VTune Profiler user guide, https://www.intel.com/content/www/us/en/develop/documentation/vtune-help/top/analyze-performance/microarchitecture-analysis-group/memory-access-analysis.html, 2022, 2, 25.
[65] B. Shi, X. Yang, G. Jin, G. He, S. Wang, Wall-modeling for large-eddy simulation of flows around an axisymmetric body using the diffuse-interface immersed boundary method, Appl. Math. Mech. 40 (2019) 305–320, https://doi.org/10.1007/s10483-019-2425-6.
[66] J.M. Jiménez, M. Hultmark, A.J. Smits, The intermediate wake of a body of revolution at high Reynolds numbers, J. Fluid Mech. 659 (2010) 516–539, https://doi.org/10.1017/S0022112010002715.
[67] Y. Dubief, F. Delcayre, On coherent-vortex identification in turbulence, J. Turbul. 1 (2000) 011, https://doi.org/10.1088/1468-5248/1/1/011.
[68] T. Huang, H.L. Liu, N. Grooves, T. Forlini, J. Blanton, S. Gowing, Measurements of flows over an axisymmetric body with various appendages in a wind tunnel: the DARPA SUBOFF experimental program, in: Proceedings of the 19th Symposium on Naval Hydrodynamics, National Academy Press, Seoul, Korea, 1992.
[69] T. Su, W. Wang, Z. Lv, W. Wu, X. Li, Rapid Delaunay triangulation for randomly distributed point cloud data using adaptive Hilbert curve, Comput. Graph. 54 (2016) 65–74, https://doi.org/10.1016/j.cag.2015.07.019.