Dong et al. EURASIP Journal on Wireless Communications and Networking (2019) 2019:62
https://doi.org/10.1186/s13638-019-1390-8

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Unstructured mesh generation based on Parallel Virtual Machine in cyber-physical system

Hao Dong[1], Haiqing Si[2*], Huiying Zong[3] and Xiaozhu Liu[4]

Abstract

A parallel unstructured mesh generation technique is proposed based on the built-in cyber-physical system (CPS) of Parallel Virtual Machine context. A static load-balancing strategy for computational domain decomposition is firstly presented in the paper. After dividing the whole computational domain into several sub-domains, unstructured grids are separately generated in each sub-domain using the advancing-front grid generation technique. Besides, for a dynamic load-balancing strategy, Lohner's advancing front domain-splitting algorithm is improved to make the sub-grids and their boundaries more favorable for the grid generation. Moreover, a new optimization strategy of sub-domain's boundary is simultaneously presented to smooth the boundaries and improve the quality of grids. Finally, conditions of receiving new points and elements are also developed during the grid generation in the sub-domain. Meanwhile, a new strategy of receiving new elements and refusing new points during the interface grid generation is proposed, which can save computational cost. A new parallel Laplacian smoother technique is implemented to generate high-quality mesh. Meshes for NACA0012 airfoil, cylinder, and multi-element airfoil are generated by the improved parallel algorithm using the static and dynamic load-balancing strategies. Some comparisons of parallel calculations are also made using the different number of processors in the tables.

Keywords: Cyber-physical system, Advancing-front method, Unstructured grid, Parallel algorithm, Domain-splitting, Parallel Virtual Machine

1 Introduction

The development of computer and network technology has brought great convenience to human life [1, 2]. However, with the advances of the hardware product performance and the rapid development of network communication technology, computer system and intelligence information for all kinds of engineering systems are highly expected to improve the data processing ability. Computing devices have not only confined to the system function expansion but also focused on the reasonable and effective system resources allocation efficiency and system performance optimization and personalized service and customer satisfaction. Under the guidance of these demands, a cyber-physical system (CPS) emerges as a new intelligent system and has attracted great attention from governments, academic researches, and business applications [3, 4]. Cyber-physical system (CPS) is an integration of computation, networking, and physical processes. Embedded computers and networks control and monitor the physical processes, with feedback loops where physical processes affect computations and vice versa. The economic and societal potential of such systems is vastly greater than what has been realized, and major investments are being made worldwide to develop the technology. The technology builds on the older (but still very young) discipline of embedded systems, computers, and software embedded in devices whose principal mission is not computation, such as cars, toys, medical devices, and scientific instruments. CPS integrates the dynamics of the physical processes with those of the software and networking, providing abstractions and modeling, design, and analysis techniques for the integrated whole.

Since CPS was first proposed by the US National Science Foundation in 2006, it has been viewed as the next

* Correspondence: haiqingshi18@163.com
[2]College of Civil Aviation and Flight, Nanjing University of Aeronautics and Astronautics, Nanjing, China
Full list of author information is available at the end of the article

generation intelligent system with the advantage of flexibility, high efficiency, and intelligence [5, 6]. Thus, it has the potential to serve as the interconnection and collaboration of virtual world and physical world. Besides, CPS can be described as a thematic subject rather than a disciplinary topic. Multidisciplinary areas such as mechatronics, robotics, and CPS often begin with themes and eventually evolve into subject areas. CPS has become an important direction for academic research and industrial application, such as in computer science, communication, control, and transportation [7, 8]. Therefore, the development of CPS has been supported by many governments. Specifically, applications of CPS include automotive systems, manufacturing, medical equipment, military systems, assisted living, traffic control and safety, process control, power generation and distribution, energy conservation, HVAC (heating, ventilation, and air conditioning), aircraft, instrumentation, water management systems, train, physical security (access control and monitoring), asset management, and distributed robotics (remote presentation, telemedicine) [9, 10]. Furthermore, CPS can be developed in the aspect of managing data and leveraging the interconnectivity of machines to achieve the goal of intelligent, resilient, and self-adaptable machines [11].

As for computational fluid dynamics (CFD), the calculation of flow fields is commonly divided into many correlative mesh parts. Usually, we handle both parts themselves and their connections independently to acquire the whole flow filed data, which is quite time-consuming. In this work, we introduce the idea of CPS into the mesh generation and parallelly process the different mesh parts to enhance the computing efficiency [12, 13].

Great progress in CFD using unstructured mesh method has been achieved over the recent years. However, conventional unstructured grid generation is quite time-consuming; for example, the generation of grids with more than 10^7 elements probably needs several hours. Therefore, a study on improving the efficiency for unstructured grids generation is of great significance for flow field calculations [14].

Despite that there are a large variety of grid generation schemes, the technique of parallel mesh generation is still not well addressed. Important attempts on parallel mesh generation have been made by Lohner [15] and Okusanya [16]. Lohner provided wave-front domain-splitting algorithm so that grids in the sub-domain and their boundaries are more favorable for grid generation. However, there exist several aspects to be improved in Lohner's methods, such as optimization of sub-domain boundary and grid smoothness; at the same time, there are too many communication overheads [17–19] in the method of Okusanya's Delaunay triangulation. Hence, it is necessary to further improve the parallel grid generation technique [20–22] to solve flow problem with complex geometries [23–25]. It is well known that CFD is becoming increasingly sophisticated: grids define highly complex geometries and flows are simulated involving very different length and time scales [26, 27]. The number of grid points, and thereby the number of degrees of freedom, is increasing as the memory of supercomputers is growing. CPS integrates computing, communication, and storage capabilities and can reliably, safely, stably, and efficiently run in real time [28]. Due to the demand for the mutual communication during the parallel grid generation, CPS can provide a good bridge for parallel calculations and can improve the performance of parallel mesh generation [29, 30].

This paper focuses on the efficient parallel mesh generation in CPS. Load balancing is a technique in which the workload is evenly assigned to every slaver processor as much as possible, which includes static and dynamic load balancing. To save computational cost, a static and dynamic load-balancing strategy for the domain decomposition technology is developed in the paper. For some applications, the workload on per processor can be estimated or even analytically determined in a preprocessing step before the simulation is started, so a static load-balancing method can be adopted. If the workload does not change during the course of the computation, static load balancing is sufficient. However, the majority of applications in computational science and engineering show a more complex runtime behavior, necessitating some kind of dynamic load balancing to provide the same workload on each processor at all times. The goal of a dynamic load balancing can be stated as follows: given a collection of tasks performing a computation and a set of processors on which these tasks are to be executed, find the mapping of tasks to processors that minimizes the run time.

After the whole computational domain is divided into sub-domains, advancing-front grid generation technique is used to generate unstructured grids in each sub-domain separately. Laplacian smoother is implemented on a parallel machine. Based on the 2D parallel algorithm presented by Lohner, a new simple algorithm which can decrease communication overheads is provided and implemented on the Parallel Virtual Machine (PVM) context. PVM is a software system that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resource. The individual computers may be shared- or local-memory multiprocessors, vector supercomputers, specialized graphics engines, or scalar workstations, which may be interconnected by a variety of networks, such as Ethernet. Parallel efficiency of generating grid and grids quality is desirable, which demonstrates high effectiveness of the proposed algorithms in the paper.

The remainder of this paper is organized as follows: Section 2 presents the proposed mesh generation scheme, Section 3 shows the numerical results and, finally, Section 4 concludes this paper.

2 The proposed parallel algorithm of mesh generation

In this section, an efficient mesh generation algorithm is proposed, which is easy to be coded, and it can decrease communication overheads and also enhance parallel efficiency. It mainly includes the following steps:

1. Inner and outer boundary points of the whole flow field domain should be defined firstly.
2. Background mesh is then generated by triangulating boundary points using Delaunay algorithm, and scales at boundary points are also calculated.
3. Computational domain of flow field is decomposed using the static or dynamic load balance strategy based on the number of boundary points.
4. Virtual boundary is determined by connecting the common boundary points from both neighboring sub-domains and then virtual points are inserted into the virtual boundary by background meshes.
5. Data of common boundary from sub-domains are sent to slaver processor by master processor, then mesh generation in the sub-domain is implemented using the wave-front domain-splitting algorithm on every slaver processor, separately.
6. Parallel Laplace smoother in the CPS. Mesh points inner the sub-domains are firstly smoothed except common virtual boundary points; secondly, virtual boundary points are smoothed independently due to the fact that the smoother for these points need mesh point information from neighboring sub-domains, where data information is mutually sent among slaver processors.
7. Information of the smoothed mesh points on every slaver processor is sent to master processor and stored in the CPS.

The present algorithm needs mutual communication only when mesh points are smoothed, which can decrease communication overheads. Using the present algorithm, not only high quality meshes are generated, but also computational time decreases.

Load balancing is to assign every slaver processor even workload as much as possible, including static and dynamic load balancing. For some applications, the workload on per processor can be estimated or even analytically determined in a preprocessing step before the simulation is started, so a static load-balancing method can be chosen. However, if the majority of applications in computational science and engineering show a more complex runtime behavior, a kind of dynamic load balancing should be adopted to provide the same workload on each processor at all times. Next, a static and dynamic load balancing strategy is introduced in the paper, separately.

2.1 Domain-splitting technique with a static load-balancing strategy

As a result of the domain decomposition procedure on a parallel platform of N processors, having N sub-domains means that there is one task only to carry out per worker; it is denoted as a static load balancing. Sometimes, there is no communication necessary between the workers, i.e., each sub-domain is generated independently, sub-dividing the domain into M sub-domains where $M > N$ makes it possible to carry out more than one task per processor (dynamic load balancing).

The standard of splitting flow field domain is to minimize communication overheads among slaver processors during the parallel calculations. A static load balancing technique is firstly adopted in the paper. Flow field domain is divided into several sub-domains according to the number of boundary points and processors in order that the number of boundary points assigned to every slaver processor is almost the same. Parallel mesh generation is implemented for the flow field domain of NACA0012 airfoil in order to validate the efficiency of the present static load balancing.

Common boundary points among neighboring sub-domains should be firstly determined, and these points are connected, then their scales δ can be calculated by the interpolation of background mesh. The point with the distance δ far away from the outer boundary point is denoted as the first virtual point. Scale δ_1 of the first one is interpolated by background mesh, then the second virtual point can be determined by the similar method where the distance between the first and second virtual point is δ_1. The search is stopped until the scale of the virtual point is larger than one of the inner boundary point. In order to better understand the insertion of the virtual point, the simple case of two points is shown next. Suppose that A and B are common boundary points from neighboring sub-domains, where A is outer boundary point and B is inner boundary point. The detailed steps are as following:

Step 1: Scales of A and B are calculated using background mesh, and they are denoted as SIZE(A) and SIZE(B), separately
Step 2: Triangle elements which C belongs to should be firstly found in the background mesh and its scale can be calculated using scales of three nodes of the present triangle

A ——— C ——— B

Step 3: The search is not stopped until SIZE(C) is larger than SIZE(B); otherwise, Step 2 is executed again.

Statistics of mesh elements of four sub-domains can be found in Table 1, and it is clearly seen that elements in every sub-domain are almost the same. Figure 1 shows the background mesh of airfoil using Delaunay algorithm. Figure 2 reveals the smoothed meshes using a wave-front domain-splitting algorithm in the serial context.

2.2 Wave-front domain-splitting algorithm with a dynamic load-balancing strategy

Wave-front domain-splitting algorithm was presented by Lohner, which is fit for the parallel unstructured mesh generation as shown in Fig. 2. However, there exist some flaws in the algorithm. Its disadvantages are as follows: (1) neighboring domains are too many; (2) common boundaries among neighbors are also many; and (3) additionally, it is very difficult to search for the common boundaries. They make it inconvenient for generating sub-grids and coding programs in the parallel context. The amount of interfaces should be consistent with that of neighboring sub-domains. According to their existing weakness, the algorithm should be improved in order to save computational cost using a dynamic load balancing technique.

Given the background grid of the field domain, the global domain can be divided into sub-domains with smooth boundaries using the improved algorithm. Grid up each sub-domain separately by the advancing front technique, then slots that are interfaces between neighbors are made, and finally, we grid up the inter-sub-domain regions by mutual communication of processors. However, the advancing front technique in the parallel context differs from that in the serial context.

In contrast to Lohner's methods, advantages of the parallel generation method in the paper are as follows:

Table 1 Elements and nodes for four sub-domains

Sub-domain	Number of elements	Number of nodes
1st one	2213	1169
2nd one	2212	1167
3rd one	2210	1164
4th one	2215	1171

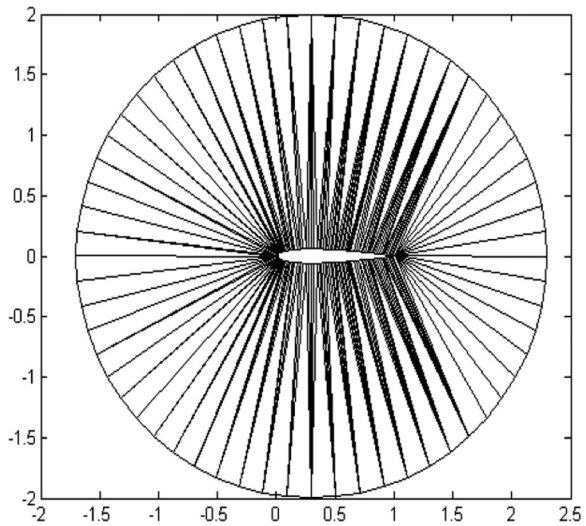

Fig. 1 Background meshes

(1) we improve the conditions of receiving new points and elements in generating sub-grids; (2) concepts of the transient inactive face and the perpetual inactive face are presented in the paper, which helps to understand the grid generation algorithm; and (3) steps of the grid generation in the inter-subdomain regions are included here. A new strategy of just receiving new element and refusing new point is presented in generating the interface grids, which can spare much central processing unit (CPU) time.

Given the sub-domains, there are two possible parallel grid generation strategies: (1) in-out and (2)

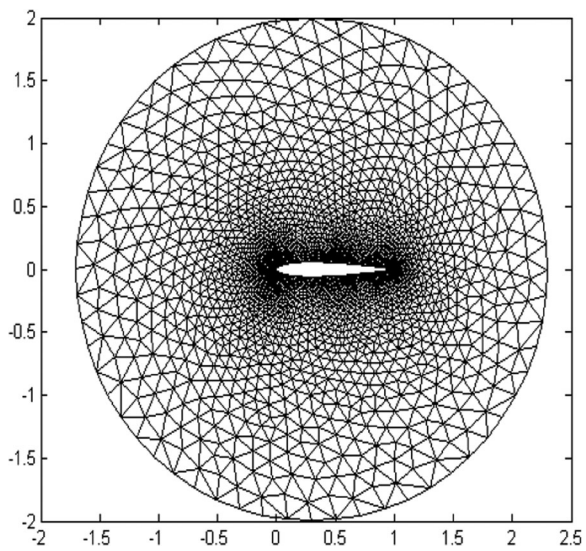

Fig. 2 Smoothed meshes

Table 2 Comparison between wave-front algorithm and its improvement

Grids of NACA0012	Wave-front algorithm	The improved algorithm
NCB	950	924
ANND	4.1	1.75
MAND	5	2

out-in. The first approach is implemented in the paper due to its convenience. According to the improved algorithm, there are at most two neighbors for every sub-domain, which can avoid the generation of corners; therefore, the third step in the strategy of in-out can be omitted.

2.2.1 Generation of sub-domain grids

In each sub-domain, the advancing front technique is used to generate an unstructured grid. Compared to the steps, the required modification is the fourth step. In order to understand the generation algorithm better, the classification of edges and the concept of transient and perpetual inactive faces are defined below.

Active faces on the front can be used to form faces of elements. The transient inactive face is not the active face in generating sub-domain grids, but the active one during the course of generating interface grids. The perpetual inactive face is not the active face in generating sub-domain grids or interface grids. In a word, faces include active and inactive ones, while inactive faces constitute transient and perpetual ones.

The step F.4 in reference [15] is on how to determine new points and elements. The modified steps in the paper are as follows:

Step 1: The line segment joining the new point with the midpoint of the selected front edge cannot intersect any other fronts.
Step 2: New point and sub-domain boundary point cannot coincide.
Step 3: New element cannot include any boundary point of the front.
Step 4: New element cannot intersect boundaries of the sub-domain.
Step 5: New element cannot cross any transient inactive faces (including coincidence).
Step 6: New element cannot incorporate points of the transient inactive faces.

New point or element can be received if all the above six items must be satisfied; otherwise, the selected front face is denoted as a transient inactive face.

2.2.2 Generation of the interface grids

In generating the interface grids, a new strategy of only receiving new element and rejecting new point is presented here, according to the geometry features of the slots between sub-domains. The following steps should be added in the generation of interface grids.

1. Receive the information of the transient inactive faces from the neighbors in the present processor.
2. Find the coincident points of the transient inactive faces from the receiving processor and the present processor, then start with the coincident points and merge all the transient inactive faces upon each other into the closed sequential fronts.
3. Call the advancing front technique program (serial program) and generate meshes.
4. Assemble the results and store them.

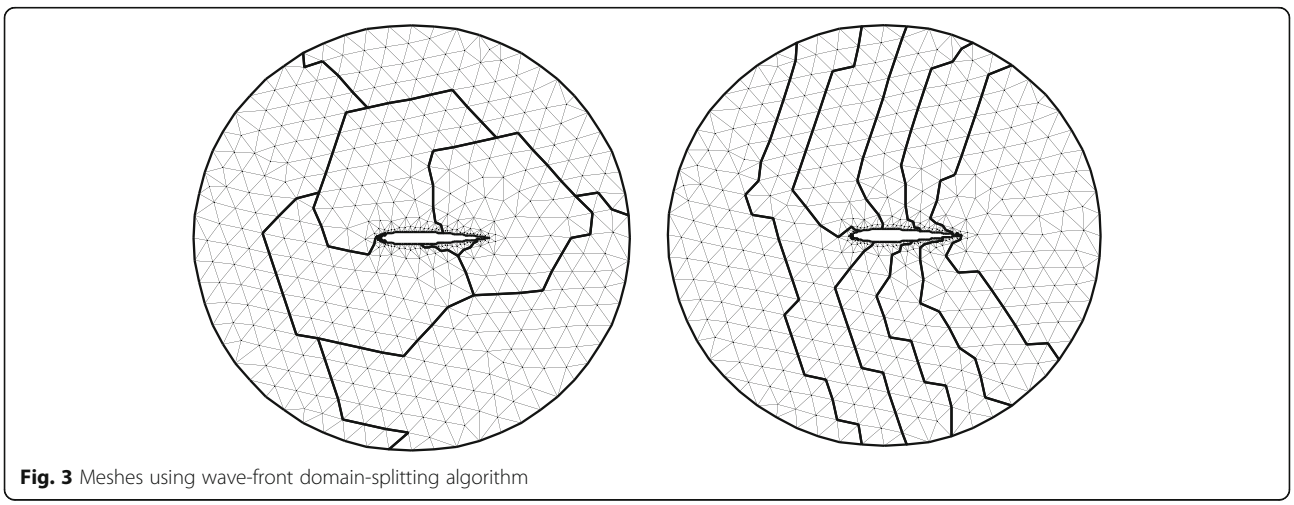

Fig. 3 Meshes using wave-front domain-splitting algorithm

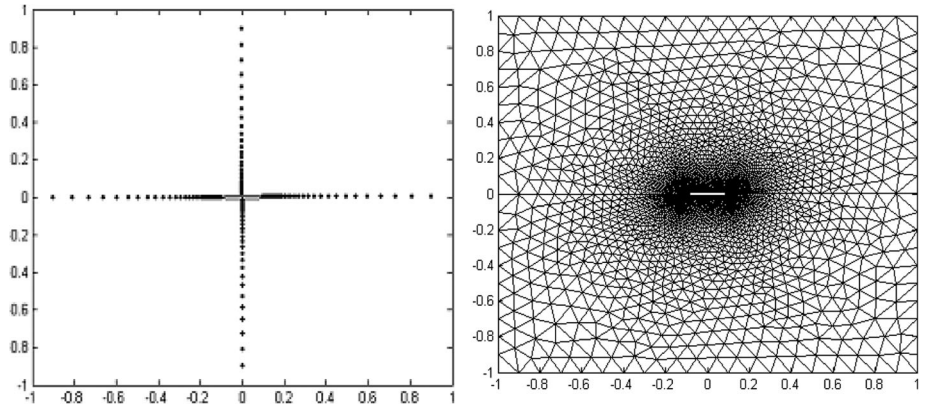

Fig. 4 Inserted virtual points and parallel meshes in sub-domains

Advantages of the algorithm improved are as follows: (1) the number of neighboring domains decreases very much, which is generally at most 2; (2) the amount of common boundaries also declines accordingly; (3) this cuts down the complexity of compiling parallel codes; (4) furthermore, it is very easy to search for common boundaries; (5) complex interfaces are avoided after each sub-domain is gridded up separately; 6) the third step can be omitted, which attempts to grid up the corners in parallel grid generation strategies.

2.3 Parallel smoother of mesh

In order to enhance the quality of the whole mesh, grid needs to be smoothed, and simple Laplace iteration is adopted in the proposed scheme:

$$x_i^{n+1} = x_i^n + \frac{\omega}{m}\sum_{k=1}^{m}\left(x_k^n - x_i^n\right) \tag{1}$$

$$y_i^{n+1} = y_i^n + \frac{\omega}{m}\sum_{k=1}^{m}\left(y_k^n - y_i^n\right) \tag{2}$$

where m is denoted as the number of points which share the common edge with the point i, n is defined as iteration time and it is chosen to be the range from 50 to 100, and ω is the relaxation factor and it is 0.2 normally.

Due to the fact that virtual boundary points still belong to inner ones in the whole flow field domain, they should be smoothed independently; otherwise, it will affect the quality of whole grids. Therefore, mutual communication from neighboring sub-domains is needed when virtual boundary points are smoothed. The detailed steps are as following:

1. Mesh is firstly smoothed independently in the sub-domain excluding virtual boundary points.
2. Triangle elements to which virtual point belongs are gathered by slaver processors.
3. Send the information of triangle elements to neighboring slaver processors.
4. Virtual points are then smoothed when information is received in the slaver processor.
5. Send the update coordinates of mesh to master processor.

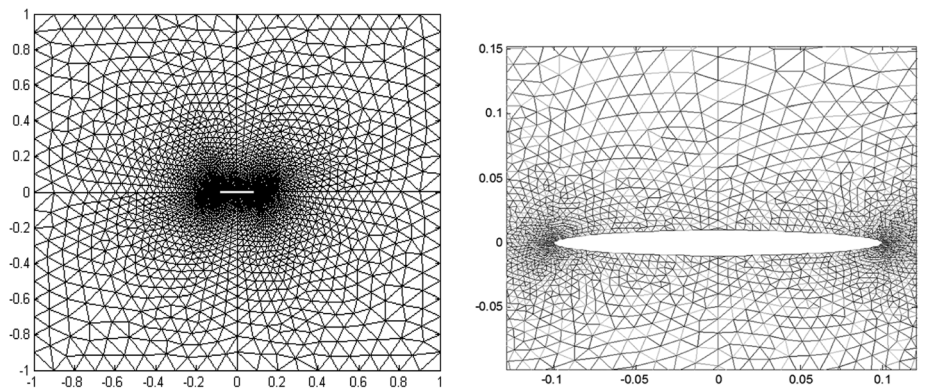

Fig. 5 Meshes for four sub-domains

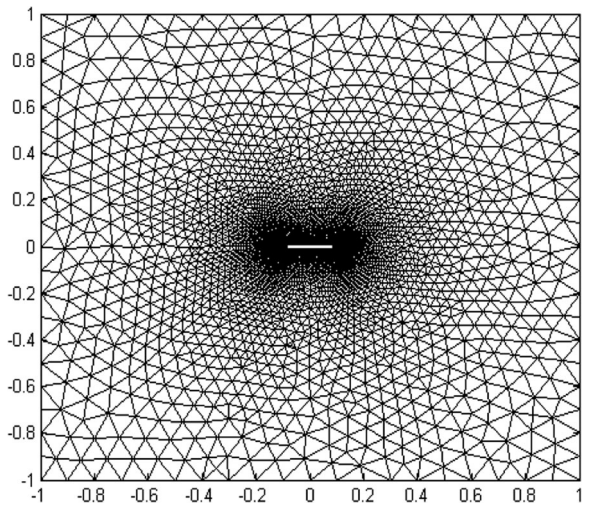

Fig. 6 Smoothed grids for airfoil

What outlined above is successfully implemented on a workstation with many processors. Parallel unstructured grid generator is ported to a multiple instruction multiple data (MIMD) PVM context [18] in the CPS. PVM is a software system that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resource. The individual computers may be shared- or local-memory multiprocessors, vector supercomputers, specialized graphics engines, or scalar workstations, which may be interconnected by a variety of networks in the CPS. PVM software executes on each machine in a user-configurable pool and presents a unified, general, and powerful computational environment of concurrent applications. User programs written in C or Fortran are provided access to PVM through the use of calls to PVM library routines for functions such as process initiation, message transmission and reception, and synchronization via barriers or rendezvous. Users may optionally control the execution location of specific application components. The PVM system transparently handles message routing, data conversion for incompatible architectures, and other tasks that are necessary for operation in a heterogeneous, network environment in the CPS.

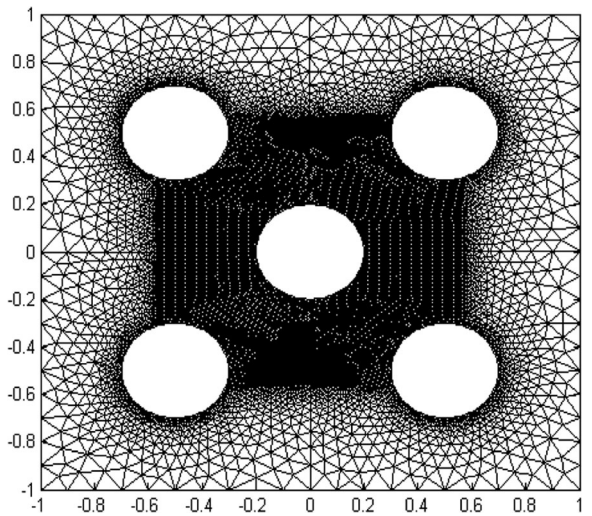

Fig. 7 Mesh using four sub-domains for cylinder

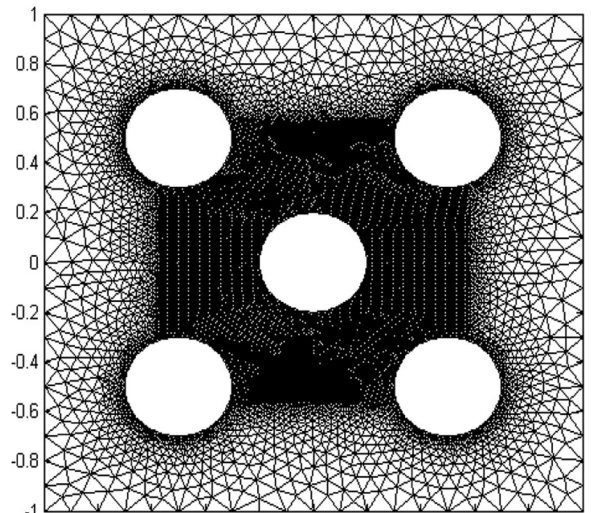

Fig. 8 Smoothed grids for cylinder

Table 2 lists the statistics of the number of common boundaries (NCB), the average number of neighboring domains (ANND), and the maximum amount of neighbor domains (MAND) in order to compare wave-front algorithm with the improved one. They show that the value of three parameters decrease after the algorithm is done in the paper, especially for the third one, which contributes to the grid generation. Figure 3 shows the meshes using the wave-front algorithm and the improved one, respectively.

3 Experimental results and parallel efficiency

3.1 Numerical validation for a static load-balancing strategy

The NACA airfoils are airfoil shapes for aircraft wings developed by the National Advisory Committee for Aeronautics (NACA). In order to validate the effectiveness of the proposed parallel algorithm, meshes for

Table 3 Computational time, speedup, and parallel efficiency for airfoil meshes

	1 CPU	2 CPUs	4 CPUs
Computational time(s)	168.3	89.67	48.0
Speedup	1.0	1.82	3.4
Parallel efficiency	100%	91%	85%

Table 4 Computational time, speedup, and parallel efficiency for airfoil meshes

	1 CPU	2 CPUs	4 CPUs
Computational time(s)	49.128	26.41	13.8
Speedup	1.0	1.86	3.56
Parallel efficiency	100%	93%	89%

NACA0012 airfoil and five cylinders are generated in the paper, where NACA 0012 airfoil is symmetrical, the 00 indicating that it has no camber, and the 12 indicates that the airfoil has a 12% thickness to chord length ratio: it is 12% as thick as its length. Figures 4, 5, and 6 reveal meshes of NACA0012 airfoil.

Figure 4 shows that the NACA0012 airfoil flow field domain is decomposed into two sub-domains, where 88 virtual boundary points are inserted into the present computational domain. As depicted in Fig. 5, computational domain is split into four sub-domains, where 166 virtual boundary points are inserted in the common boundary of sub-domains. The high resolution of the local grid around the NACA0012 airfoil has to be also displayed clearly in Fig. 5. Figure 6 shows the smoothed meshes of the airfoil, and it is clearly seen that mesh quality is better than the initial meshes. Mesh for five cylinders can be found in Figs. 7 and 8, where the computation domain is divided into four sub-domains and 132 virtual points are inserted into the common boundary of sub-domains.

The quality of a parallel implementation is often measured by the speed up or parallel efficiency.

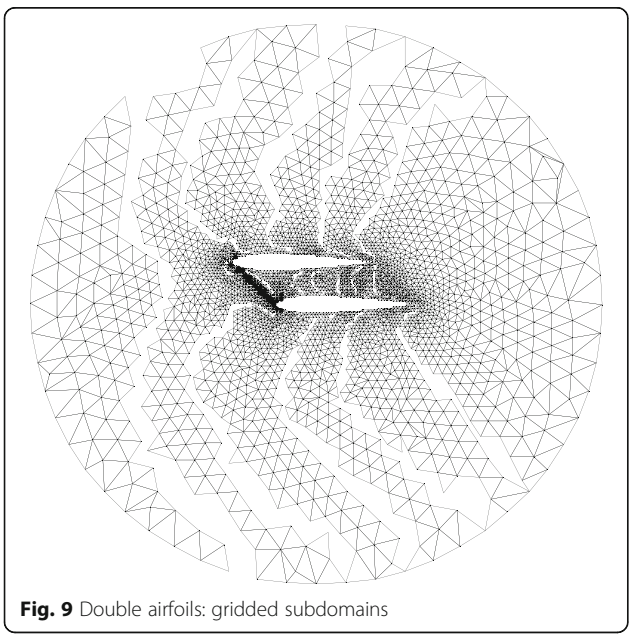

Fig. 9 Double airfoils: gridded subdomains

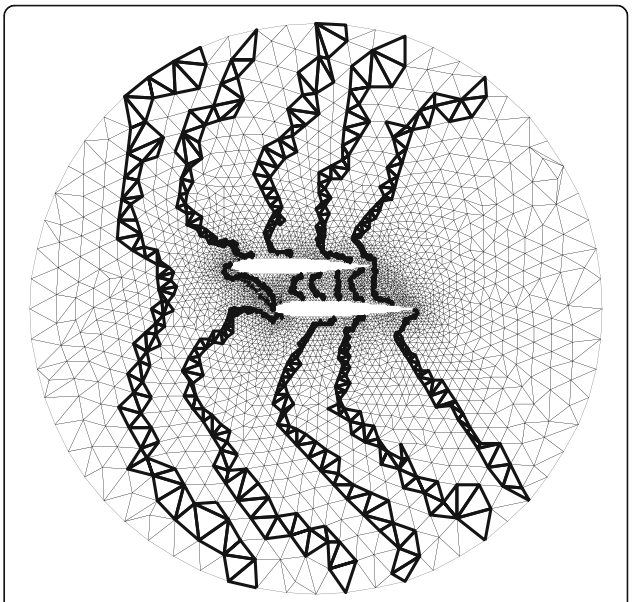

Fig. 10 Double airfoils: interface grids

$$S_p = \frac{t_1}{t_p} \tag{3}$$

$$E_p = \frac{S_p}{p} \tag{4}$$

where S_p and E_p denote speedup and parallel efficiency separately; t_1 and t_p denote the execution time of the algorithm on one and P processors, respectively. Tables 3 and 4 show the speedup ratio and parallel efficiency of mesh generation for airfoil and cylinder.

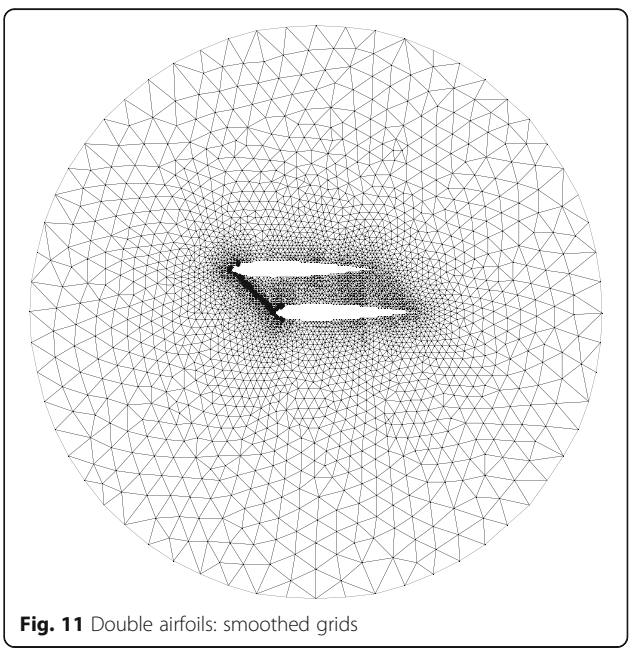

Fig. 11 Double airfoils: smoothed grids

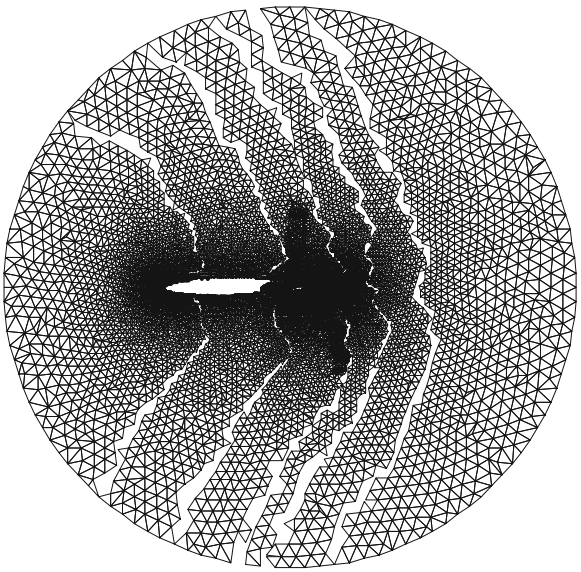

Fig. 12 Multi-element airfoil: gridded subdomains

From the tables, it clearly reveals the comparisons of computational time, speedup, and parallel efficiency using the different number of slaver processors. The following conclusions can be drawn from Tables 3 and 4:

1. Computing time decreases very much, and the aim of quickly generating unstructured grids is attained.
2. Parallel efficiency of two processors is much higher than that of four processors. The reason why parallel efficiency is on the decline is that mutual communication overheads between neighbor processors accordingly increase with the number of sub-domains during the course of parallel smoothing.

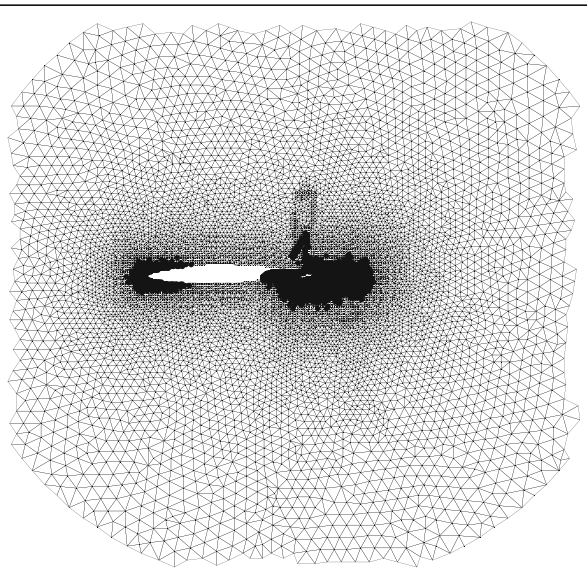

Fig. 13 Multi-element airfoil: interface grid

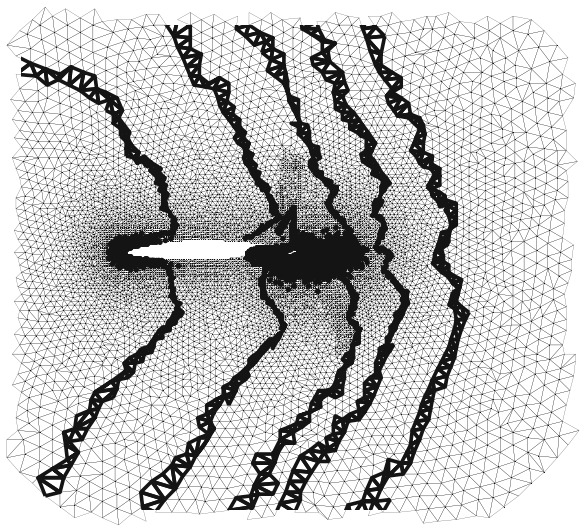

Fig. 14 Multi-element airfoil: smoothed grids

3.2 Numerical validations for a dynamic load-balancing strategy

If the majority of applications in computational science and engineering show a more complex runtime behavior, a kind of dynamic load balancing to provide the same workload on each processor at all times should be adopted. Next, a dynamic load balancing is validated in the paper.

What outlined above is successfully implemented on a workstation with many processors. Parallel unstructured grid generator is ported to a MIMD PVM context. Numerical examples show the success of the improved

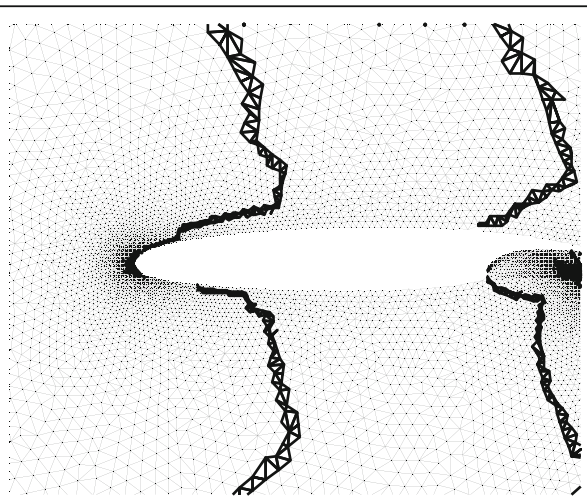

Fig. 15 Local meshes for multi-element airfoil

Table 5 Computational time, speedup, and parallel efficiency for airfoil meshes

	1 CPU	2 CPUs	4 CPUs
Computational time(s)	168.75	92.805	56.626
Speedup	1.0	1.818	2.98
Parallel efficiency	100%	90.9%	74.5%

parallel unstructured grid generation algorithm. Figures 9, 10, and 11 depict sub-grids, interface grids, and smoothed whole grids for double airfoils, separately.

Figures 12, 13, and 14 show sub-grids, interface grids, and smoothed grids for complex multi-element airfoil, separately. They demonstrate that the quality of parallel grids is satisfactory for flow field calculation and the improved parallel grid generation technique is feasible. Figure 15 reveals the local grid around multi-element airfoil.

Table 5 displayed the comparison of time, speed up, and parallel efficiency using the different number of processors. As stated from the above Tables 3 and 4, the similar conclusions can be also drawn from Table 5. Due to the limited computer resources, four PCs are optimal and used in the course of the calculations. Speedup and parallel efficiency demonstrate that the effectiveness of a dynamic load balancing developed by us is enough for fast calculation of flow field, especially for the complex bodies.

4 Conclusions

Based on Lohner's algorithm, a parallel unstructured grid generation technique is studied and developed in CPS to improve the performance. For some applications, the workload per processor can be estimated or even analytically determined in a preprocessing step before the simulation is started, so a static load-balancing method can be recommended. However, if the majority of applications in computational science and engineering show a more complex runtime behavior, a kind of dynamic load balancing to provide the same workload on each processor at all times should be adopted. A static and dynamic load-balancing strategy for the domain decomposition has been proposed, where communication overhead can decrease greatly. A new parallel smoother technique is also provided in order to enhance mesh quality in CPS. Based on the PVM context, mesh for the NACA0012 airfoil and cylinder is generated on PC clusters. By comparison, parallel efficiency can be enhanced as the number of processors used in the calculations.

Abbreviations

ANND: Average number of neighboring domains; CFD: Computational fluid dynamics; CPS: Cyber-physical system; CPU: Central processing unit; HVAC: Heating, ventilation and air conditioning; MAND: Maximum amount of neighbor domains; MIMD: Multiple instruction multiple data; NCB: Number of common boundaries; PVM: Parallel Virtual Machine

Funding

This work was supported by Special Foundation of China Postdoctoral Science (Grant No. 201104565) and National Natural Science Foundation of China (Grant No. 11272151, Grant No. 61103019).

Availability of data and materials

The data will be shared if request received.

Authors' contributions

HD and HS contributed to the conception and algorithm design of the study. HZ contributed to the acquisition of simulation. XL and HD contributed to the analysis of simulation data and approved the final manuscript. All authors read and approved the final manuscript.

Authors' information

Hao Dong received the B. S. in Aircraft Design from Northwestern Polytechnical University in 2005, and his Ph. D. degree in Fluid Mechanics from Nanjing University of Aeronautics and Astronautics in 2010. Dr. Dong is an associate professor at college of Aerospace Engineering, Nanjing University of Aeronautics and Astronautics, and is the deputy director of the department of aerodynamics. He has published more than 20 journal articles in the areas of computational fluid dynamics, and mesh generation in CPS.

Haiqing Si received the B.S., M.S. and Ph.D. degrees in College of Aerospace Engineering from Nanjing University of Aeronautics and Astronautics, China, in 1999, 2004 and 2007, respectively. His research interests include parallel computing, numerical method in computational fluid dynamics. He has published over 40 papers in international journals and conferences in the areas of parallel computing, parallel generation technology of calculation meshes.

Huiying Zong is pursuing her M.S. degree in the science college from Nanjing University of Aeronautics and Astronautics, China. Her research interests include CPS, parallel computation in numerical simulation.

Xiaozhu Liu is an associate professor in the school of automation, Wuhan University of Technology, China. She received the Ph. D. in Computer Software and Theory, Wuhan University, in 2011. Her research interests include CPS, mobile computing, and numerical simulation.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]College of Aerospace Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China. [2]College of Civil Aviation and Flight, Nanjing University of Aeronautics and Astronautics, Nanjing, China. [3]College of Science, Nanjing University of Aeronautics and Astronautics, Nanjing, China. [4]School of Automation, Wuhan University of Technology, Wuhan, China.

Received: 2 November 2018 Accepted: 28 February 2019
Published online: 13 March 2019

References

1. R. Rajkumar, I. Lee, L. Sha, J. Stankovic, *Cyber-physical systems: The next computing revolution. 47th ACM/IEEE Design Automation Conference (DAC)* (2010), pp. 731–736.
2. X. Liu, R. Zhu, B. Jalaian, Y. Sun, Dynamic spectrum access algorithm based on game theory in cognitive radio networks. Mob. Netw. Appl. **20**(6), 817–827 (2015).
3. R. Zhu, X. Zhang, X. Liu, W. Shu, T. Mao, B. Jalaeian, ERDT: Energy-efficient reliable decision transmission for cooperative spectrum sensing in industrial IoT. IEEE Access **3**, 2366–2378 (2015).
4. B. Jalaeian, R. Zhu, H. Samani, M. Motani, An optimal cross-layer framework for cognitive radio network under interference temperature model. IEEE Syst. J. **10**(1), 293–301 (2016).

5. E.A. Lee, *Cyber physical systems: Design challenges*, 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC) (2008), pp. 363–369.
6. J. Du, X. Liu, L. Rao, Proactive Doppler shift compensation in vehicular cyber-physical systems. IEEE/ACM Trans. Networking **26**(2), 807–818 (April 2018).
7. K. Gai, M. Qiu, H. Zhao, X. Sun, Resource management in sustainable cyber-physical systems using heterogeneous cloud computing. IEEE Trans. Sustain. Comp. **3**(2), 60–72 (April-June 2018).
8. S. Cai, V.K.N. Lau, Zero MAC latency sensor networking for cyber-physical systems. IEEE Trans. Signal Process. **66**(14), 3814–3823 (July 15 2018).
9. X. Lu, B. Chen, C. Chen, J. Wang, Coupled cyber and physical systems: Embracing smart cities with multistream data flow. IEEE Electrification Mag. **6**(2), 73–83 (June 2018).
10. C. Lu, IoT-enabled adaptive context-aware and playful cyber-physical system for everyday energy savings. IEEE Trans. Hum. Mach. Syst. **48**(4), 380–391 (August 2018).
11. B.H. Krogh, *Cyber physical systems: the need for new models and design paradigms* (Carnegie Mellon University), pp. 1–31.
12. H.Q. Si, T.G. Wang, J. Cheng, Domain decompositions and parallel algorithms to solve Euler equations on the unstructured grid. Acta Aerodynamica Sinica **24**(1), 102–108 (2006).
13. Y.P. Chien, F. Carpenter, A. Ecer, H.U. Akay, Load-balancing for parallel computation of fluid dynamics problems. Comput. Methods Appl. Mech. Eng. **120**, 119–130 (1995).
14. P.A. Cavallo, N. Sinha, G.M. Feldman, *Parallel unstructured mesh adaptation for transient moving body and aero-propulsive applications, AIAA 2004-1057* (2004).
15. R. Lohner, Parallel unstructured grid generation. Comput. Methods Appl. Mech. Eng. **95**, 343–357 (1992).
16. T. Okusanya, J. Peraire, *Parallel unstructured mesh generation, 5th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields* (1996), pp. 719–729.
17. Y.P. Chien, A. Ecer, H.U. Akay, S. Secer, R. Blech, Communication cost estimation for parallel CFD using variable time-stepping algorithms. Comput. Methods Appl. Mech. Eng. **190**, 1379–1389 (2000).
18. R.U. Payli, E. Yilmaz, A. Ecer, H.U. Akay, S. Chien, *DLB-A dynamic load balancing tool for grid computing, Parallel CFD conference, May 24-27* (2004).
19. H.Q. Si, T.G. Wang, Load balancing strategy for parallel calculation and time cost estimation. Acta Aeronautica et Astronautica Sinica **28**(Sup), S57–S61 (2007).
20. R. Löhner, Recent advances in parallel advancing front grid generation. Arch. Comput. Meth. Eng. **21**(2), 127–140 (2014).
21. A. Lintermann, S. Schlimpert, J.H. Grimmen, et al., Massively parallel grid generation on HPC systems. Comput. Meth. Appl. Mech. Eng. **277**(2), 131–153 (2014).
22. A. Bihlo, R.D. Haynes, Parallel stochastic methods for PDE based grid generation. Comput. Math. Appl. **68**(8), 804–820 (2014).
23. R. Haynes, F. Kwok, Discrete analysis of domain decomposition approaches for mesh generation via the equidistribution principle. Math. Comput. **86**, 303, pp. 233–273 (2017).
24. G.J. Page, Rapid, parallel CFD grid generation using octrees. Aeronaut. J. **117**(1188), 133–146 (2013).
25. E.A. Hereth, K. Sreenivas, L.K. Taylor, D.S. Nichols. An automatic parallel octree grid generation software with an extensible solver framework and a focus on urban simulation, 55th AIAA Aerospace Sciences Meeting, AIAA SciTech Forum, (2017).
26. O. Meister, K. Rahnema, M. Bader, Parallel memory-efficient adaptive mesh refinement on structured triangular meshes with billions of grid cells. ACM Trans. Math. Softw. **43**(3), 19 (2016).
27. O. Meister, M. Bader, 2D adaptivity for 3D problems: Parallel SPE10 reservoir simulation on dynamically adaptive prism grids. J. Comput. Sci. **9**, 101–106 (2015).
28. C.F. JanâŸEn, N. Koliha, T. Rung, A fast and rigorously parallel surface voxelization technique for GPU-accelerated CFD simulations. Commun. Comput. Phys. **17**(5), 1246–1270 (2015).
29. M. O'Connell, S.L. Karman, Advances in parallelization for large scale Oct-tree 681 mesh generation. AIAA Science and Technology forum and exposition, (2015).
30. I. Fejtek, T. Barfoot, G. Lo, Turboprop nacelle optimization using automated surface and grid generation and coarse-grain parallelization. J. Aircr. **33**(6), 1166–1173 (2015).