Intelligent Automation And Soft Computing, 2019
Copyright © 2019, TSI® Press
Vol. 25, no. 3, 527–546
https://doi.org/10.31209/2019.100000109

Numerical optimization algorithm for unsteady flows of rotor based on web service

Jilin Zhang[1,4,5], Xuechao Liu[1,5], Jian Wan[2,1,5], Yongjian Ren[1,5], Binglin Xu[1,5], Jianfan He[1,5], Yuchen Fan[1,5], Li Zhou[1,5] , Zhenguo Wei[6], Juncong Zhang[6] and Jue Wang[3]

[1]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
[2]Zhejiang University of Science and Technology, Hangzhou 310023, China
[3]Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
[4]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[5]Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou 310018, China
[6]Zhejiang Dawning Information Technology Co., Ltd, Hangzhou 310051, China

The two authors Jilin Zhang and Xuechao Liu contribute equally to this paper, and they are co-first authors.

ABSTRACT
A numerical optimization algorithm for unsteady flows of rotor based on web service is proposed. Space discretization uses the finite volume method, time discretization uses the implicit dual-time steps method, and turbulence model uses the Spalart–Allmaras (S–A) model. In order to efficiently use the computing resources of the cluster, a service-oriented service computing architecture is used in a parallel computing service program. In order to realize the load balance of hybrid grid partition, the grid is partitioned by Metis Library. Meanwhile, data communication based on Message Passing Interface (MPI) technology guarantees the consistency of convergence between parallel algorithm and serial algorithm and establishes virtual boundaries for each partition to ensure data exchange among partitions. By simulating the forward flight state of the Robin fuselage/rotor model, the experimental results coincide with the serial program, and the parallel efficiency is high.

KEY WORDS: Service-oriented; LU-SGS; MPI; METIS; Load Balancing

1 INTRODUCTION

COMPUTATIONAL fluid dynamics (CFD) is an important tool for fluid analysis by means of computer and numerical simulation to solve fluid dynamics control equations and analyse the motion law of a model. At present, numerical simulation systems are developing towards being large scale, high-precision and real-time, which has enhanced the requirements of computational complexity, computational scale and accuracy. These problems need to be solved by parallel numerical simulation. In order to make parallel computing better use of computer resources and achieve resources dynamic management, this paper integrate computing cluster resources through service computing to improve the utilization of computing resources. The advantage of using service computing lies in the small changes in the original program, the designer only needs to encapsulate its original computing service program into web service, and then deploy web service on the cluster node. In this way, a service node assigns computing resources to the computing node according to the client's request for resources and the availability of the current resource. Designers can focus on the parallel design of numerical simulation and shorten the development cycle.

The common methods of solving through numerical simulation are divided into two categories: explicit and implicit methods.

The explicit method generally used at present is multi-step Runge–Kutta(Jameson, 1981) method. The

CONTACT Jian Wan wanjian@hdu.edu.cn

advantage of the explicit method is that the requirement computation and storage is little with each time-step iteration, and parallel computation is convenient to implement(Liu, 2013). The disadvantage of the explicit method is that the time step is constrained by the stability condition, which results in excessive calculation steps and low iterative efficiency. At present, the LU-SGS(Jameson, 1987) method is the most popular implicit calculation method. The advantage of implicit method is the adequate stability, preferable value of calculating step size and enhanced iterative efficiency that it provides. Because the LU-SGS method is highly dependent on the data in the iterative process, the method is not convenient for parallelization. By improving the LU-SGS method, Wright(Candler, 1994) and others proposed the Data-Parallel Lower–Upper Relaxation (DP-LUR) implicit method. The DP-LUR method shifts the non-diagonal item to the right end of the equation in order to replace the symmetric Gauss–Seidel iteration with the Jacobi iteration, which avoids the upper and lower triangular matrices that are used in the iterative process by the LU-SGS method(Candler, 1994). This iterative method removes all dependencies of data, reduces the number of communications and conveniently achieves parallel computing. However, the DP-LUR method is similar to the Jacobi iteration, which requires a higher number of iterations to achieve a similar convergence as the LU-SGS method; this increases the computational requirement of the DP-LUR method.

In the field of large-scale CFD numerical simulation system design, technology available globally has been very mature; a typical representative is Fluent. In recent years, our country has also made new progress in this aspect. Wang Song(Wang, 2017) et al. designed large-scale computational fluid dynamics visualization analysis system (FVAS) for large-scale three-dimensional steady and unsteady flow analysis of a system. In the field of magnetic fluid dynamics(Zhao, 2009), Pan Yong and Wu Yizhao(Pan, 2007) of Nanjing University of Aeronautics and Astronautics had conducted a numerical simulation experiment on the field interference effect of supersonic flow field and magnetic field. Using MPI technology to realize parallel computing, the second-order precision interpolation is carried out using the Monotonic Upwind Scheme for Conservation Laws (MUSCL)(Van, 1979) method, and the Time-Marching method adopted the five-step Runge–Kutta method.

In the field of impeller machinery, domestic research has also been conducted. Xu Jianzhong(Li, 2008) of the Chinese Academy of Sciences had adopted the DP-LUR method to realize Navier–Stokes parallel solution based on unsteady computation. A consistent amount of research has been devoted on high performance computing field in China. Deng Shouchun(Zheng, 2011) of the Chinese Academy of Sciences developed the parallel algorithm based on MPI communication technology; here, the space discretization adopted the third order upwind Harten–Lax-Van Leer–Einfeldt–Wada (HLLEW)(Obayashi, 1994) or Advection Upstream Splitting Method (AUSM)(Liou, 1993) format based on Roe(Roe, 1986), the turbulence model adopted the two-equation model and the time advance adopted the convenient parallel DP-LUR method. In recent years, China has made new progress in the field of high-performance computing of large aircraft; Chen Gang(Chen, 2011) of the Chinese Academy of Sciences had developed a large-scale aircraft aerodynamic numerical simulation software—China computational fluid dynamics (CCFD) —based on ten–thousand–core parallel computing. The space discrete scheme is constructed by finite volume method. The second-order Roe-upwind flux difference scheme and the central difference scheme were used for discrete processing of the inviscid flux and viscous flux. The time advance adopted the implicit method, which exhibits adequate reliability in the complex flow field and high-resolution numerical simulation; moreover, it can solve the engineering problems in the field of large aircraft design.

In the area of helicopter design, the flow field of helicopter rotor is highly complicated compared with the fixed wing flow field; moreover, the complexity of the rotor flow field is embodied in three aspects: Firstly, the flow velocity of the rotor is changed along the span direction. Secondly, the rotor flow is a typical unsteady flow, which results from the rotation of the rotor; the flow velocity and angle of attack of the rotor are changed with the azimuth. Thirdly, the rotor rotation process produces a tip vortex and tail vortex, which are located around the rotor, causing complex vortex/propeller interference and vortex/fuselage interference; the aerodynamic performance of the aircraft is thus adversely affected(Xiao, 2007). Therefore, accurate and efficient numerical simulation of rotor flow field is a challenging task. In this regard, domestic scholars have conducted a large amount of important research. Ji Changrui(Ji, 2014) developed the strong coupling Reynolds-averaged Navier–Stokes (RANS) method based on multi block overlapped grids, which was used in the numerical simulation of helicopter rotor flow field in hover; this method adopted LU-SGS to solve the time advance and don't support parallel computing. Li Peng(Li, 2014) had developed a CFD computation method for the analysis of the tilt rotor/wing interference flow field in hover state by using the Runge–Kutta method and the parallel acceleration technique of the Single Program/Multiple Data (SPMD) model. This method is applicable to the structural mesh. Jiang Yuening(Jiang, 2017) et al. proposed an efficient and accurate numerical simulation of Unmanned Aerial

Vehicle CFD multi-core parallel computing method, a method using Fluent software for simulation.

However, our country rarely uses implicit algorithm in the field of helicopter to conduct large-scale parallel numerical simulation. Therefore, this paper presents a numerical optimization algorithm for unsteady flows of rotor based on web service, which is used to solve the problem of low efficiency in numerical simulation of unsteady rotor flow field with three-dimensional hybrid grids. This paper uses Metis to realize the load balance of partition and realizes data communication among different partitions by MPI. This algorithm can be extended to large-scale parallel numerical simulation. Through verification of the Robin fuselage/rotor model and comparison with the calculation results of the original LU-SGS method, the parallel computation method proposed in this paper substantially improves computational efficiency and ensures that the parallel computation results coincide with the serial results.

This paper is divided into five chapters: the first chapter introduces CFD high-performance computing, which has achieved remarkable results and commonly used solutions in numerous areas; however, in the helicopter field, domestic development is relatively primitive. Furthermore, the chapter introduces the main work of this study. In the second chapter, this study analyses the serial process of numerical simulation of helicopter rotor flow field and the derivation process of the control equations involving numerical calculation; further, it sums up the serial program, which can be conducted in parallel. The third chapter introduces the optimization strategy of the numerical simulation of rotor flow field, including load balancing strategy, communication optimization and LU-SGS + Jacobi parallel algorithm. The fourth chapter verifies the computational efficiency of parallel programs and the correctness of the program through the Robin fuselage/rotor model. The fifth chapter summarizes the optimization method proposed in this paper.

2 PROCESS ANALYSIS FOR NUMERICAL SIMULATION OF ROTOR FLOW FIELD

2.1 Serial Process Analysis

IN practical engineering problems, the numerical simulation of unsteady flow field with shifting boundary is more and more concerned, such as bombing by fighter planes, the swing of helicopter rotor, etc. It is of high engineering significance to develop a numerical simulation method that is highly precise, efficient and stabile, in order to solve this type of practical problems.

This paper is based on a serial three-dimensional dynamic overlapping grid Navier–Stokes equation solver; its numerical simulation of the overall flow is depicted in Figure 1:

(1) The establishment of grid system and flow field system, including the establishment of nested grid relationship and the initialization of flow field.

(2) Enter dual-time steps iteration.

(3) Flux computation. It includes the calculation of viscous flux and inviscid flux.

(4) If the problem is unsteady, the calculation of the unsteady time item is carried out; otherwise, go to step (5).

(5) LU-SGS iteration.

(6) If it is a viscous flow field, the S–A turbulence model is calculated; otherwise, go to step (7).

(7) Calculation of residual value and interpolation among nested grids.

(8) Go to step (9) if the residual value satisfies the requirement or the number of pseudo-time step iteration is attained. Otherwise, the next pseudo-time step iteration is performed.

(9) If the output condition is attained, the result of the calculation is the output. Otherwise, go to step (10).

(10) Dynamic grid motion, geometric data update.

(11) Rebuild nested grid relationships. Re-establish the interpolation relationship among nested grids, update the minimum wall distance, etc.

(12) Physical time step interpolation among nesting grids. Interpolation among the nested grids is performed according to the interpolation relationship established in step (11).

(13) If the number of iterations of the physical time step is attained, the program is terminated; otherwise, the next physical time step iteration is entered.

The pseudo-time step iteration process is the key step of the numerical simulation process, which is also the solution process of the Navier–Stokes equation. This process has the largest share of the total number of iterations in and the time consumed by the serial program. Therefore, this study can improve the computational efficiency of the numerical simulation by optimizing the pseudo-time step iteration. This is mainly concerned with the parallelization of the LU-SGS implicit iterative algorithm, which is one of the challenges in CFD research. In this paper, a numerical optimization algorithm for unsteady flows of rotor based on web service is proposed; i.e., LU-SGS iteration is performed on the inner cells of each computational process, and Jacobi iteration is performed on the boundary cells to improve the computational efficiency of the numerical simulation.

2.2 Fluid Control equations

In this study, the control equation of the numerical simulation is the Reynolds-averaged Navier–Stokes equations(Zhu, 1998) based on three-dimensional unsteady compressible fluid:

$$\iiint_V \frac{\partial W}{\partial t}\,dV + \oiint_S (F(W) - F_v)\,dS\#(1)$$

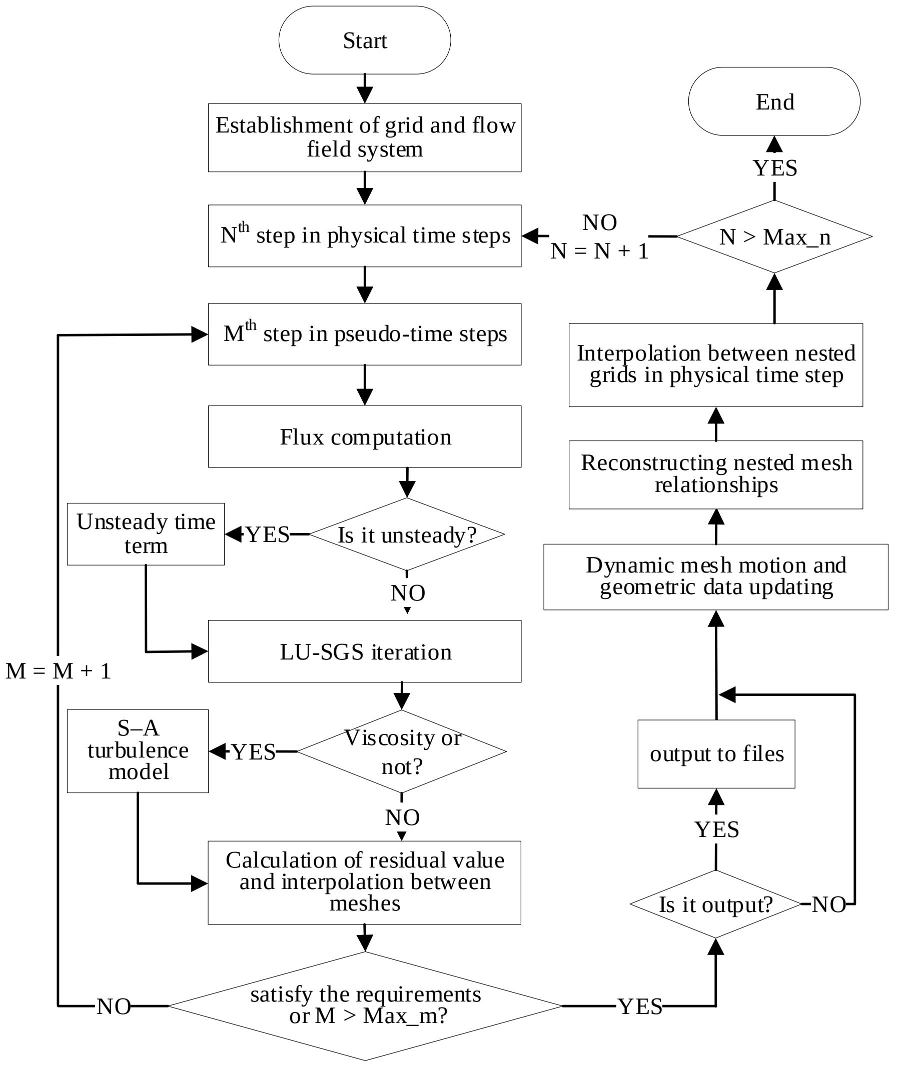

Start
Establishment of grid and flow field system
Nth step in physical time steps
Mth step in pseudo-time steps
Flux computation
Is it unsteady?
YES
Unsteady time term
NO
LU-SGS iteration
Viscosity or not?
YES
S–A turbulence model
NO
Calculation of residual value and interpolation between meshes
satisfy the requirements or M > Max_m?
NO
M = M + 1
YES
Is it output?
YES
output to files
NO
Dynamic mesh motion and geometric data updating
Reconstructing nested mesh relationships
Interpolation between nested grids in physical time step
N > Max_n
NO
N = N + 1
YES
End

Figure 1. **Serial flowchart**

where t denotes time, V represents the control volume, S is the closed surface enclosing the control volume, W is the conserved quantity, $F(W)$ is the inviscid flux and F_v is the viscous flux.

Based on the Reynolds transport equation:

$$\frac{\partial}{\partial t}\iiint_V W dV = \iiint_V \frac{\partial W}{\partial t} dV + \oiint_S (x \cdot \vec{n}) W dS \#(2)$$

where x and $\vec{n}$ represent the motion velocity and the normal vector, respectively, of the control surface and $v_{gn} = x \cdot \vec{n}$ is defined. Equation (1) can be denoted in the following form:

$$\frac{\partial}{\partial t}\iiint_V W dV + \oiint_S (F(W) - v_{gn} W) dS = \oiint_S F_v dS \#(3)$$

When $v_{gn} = \vec{V} \cdot \vec{n}$ ($\vec{V}$is the velocity vector of the fluid), equation (3) is the Lagrange equation. When $v_{gn} = 0$, equation (3) is the Euler equation.

2.3 Space discretization and Time discretization

In order to improve the time-calculation precision of an unsteady flow problem and to ensure high computational efficiency, this study adopts dual-time step propulsion method(Jameson, 1991). In equation (3), the pre-processed pseudo-time derivative is introduced, i.e.,

$$\Gamma \frac{\partial}{\partial \tau}\iiint_V Q dV + \frac{\partial}{\partial t}\iiint_V W dV +$$
$$\oiint_S (F(Q) - v_{gn} W) dS = \oiint_S F_v dS \#(4)$$

where τ and t denote the pseudo-time step and physical time step, respectively, Γ is the pre-processing matrix proposed by Weiss and Smith(Weiss, 1995) and $Q = (p, u, v, w, T)^T$ is the original variable.

In the arbitrary control volume V_i , the finite volume method is discretized for equation (4).

$$\Gamma_i \frac{\partial (QV)_i}{\partial \tau} + \frac{\partial (WV)_i}{\partial t} + \sum_{j=1}^{nfaces} \tilde{F}(Q)_{ij} S_{ij} = \sum_{j=1}^{nfaces} F_{vij} S_{ij} \#(5)$$

wherein $\tilde{F}(Q) = F(Q) - v_{gn}W$, j is the neighbour control volume of V_i , ij represents the interface between V_i and V_j and *nfaces* is the amount of boundary faces of V_i. If the left and right flow field value of S_{ij} is considered as the central value of the control volume, the result is of the first order precision. In order to obtain high accuracy, the linear reconstruction technique of Gauss integral method(Mavriplis, 2003) is adopted. In order to prevent the emergence of the new extremum, the limiter proposed by Venkatakrishnan(Venkatakrishnan, 1993) is adopted.

In time discretization, pseudo-time step uses first-order back difference and physical time uses implicit k-order posterior difference. Pseudo-time step and physical time step are represented by m + 1 and n + 1, respectively. Then, equation (5) becomes:

$$\Gamma_i \frac{Q_i^{m+1} V_i^{n+1} - Q_i^m V_i^{n+1}}{\Delta \tau} + \frac{\varphi_{n+1} (WV)^{n+1}}{\Delta t} + \frac{1}{\Delta t} \sum_{\square=0}^{k-1} \varphi_{n-\square} (WV)^{n-\square} + RES_i(Q^{m+1}) = 0 \#(6)$$

m and n are the pseudo-time steps and physical time steps, respectively. RES_i is the residual value defined as:

$$RES_i(Q) = \sum_{j=1}^{nfaces} \left(\tilde{F}(Q) - F_v\right)_{ij} S_{ij} \#(7)$$

The sequence of $\{\varphi_n\}$ represents the coefficient of a backward difference format, which is used to control the time precision of discrete equations.

In each physical time step, the flow field is solved by pseudo-time $\Delta\tau$ propulsion. When $m \to \infty$, W^{m+1} approaches W^{n+1}; it is considered to be the solution of the n + 1th time step. Define $\Delta Q = Q^{m+1} - Q^m$ and $W^{n+1} = W^m + M\Delta Q$, where M represents the Jacobi matrix. Equation (6) can be expressed in the following form:

$$\left(\frac{V_i^{n+1}}{\Delta \tau} \Gamma_i + \frac{\varphi_{n+1} V_i^{n+1}}{\Delta t} M_i \right) \Delta Q_i + RES_i^*(Q^{m+1}) = 0 \#(7)$$

The unsteady residual $RES_i^*(Q)$ is:

$$RES_i^*(Q) = \frac{\varphi_{n+1} W^m V^{n+1}}{\Delta t} + \frac{1}{\Delta t} \sum_{\square=0}^{k-1} \varphi_{n-\square} (WV)^{n-\square} + RES_i(Q) \#(8)$$

In equation (7) in this paper, the serial program adopts the implicit LU-SGS iterative format with no matrix, which exhibits higher convergence speed and higher efficiency and is more apparent in the calculation of viscous flow field.

2.4 Implicit LU-SGS method

Jameson and Yoon(Jameson, 1987) proposed an implicit LU-SGS method in 1987; however, this method is applicable only to the structure grid. In 1998, Luo(Luo, 1998) and Nakahashi(Sharov, 1998) extended this method to unstructured grids. The fundamental concept of the LU-SGS method is to decompose the Jacobi matrix into the upper triangular matrix U, lower triangular matrix L and diagonal matrix D and to eschew the complex matrix operation by linear approximation of the Jacobi matrix of the flux, thus reducing the implicit method's demand for memory hardware(Han, 2013).

For structural grids, discretization of equation (3) results in:

$$\frac{\partial W}{\partial t} V + RES(W) = 0 \#(9)$$

where $RES(W) = \sum_{j=1}^{nfaces} \left(\tilde{F}(W) - F_v\right)_{ij} S_{ij}$. The first-order backward difference is applied to equation (9):

$$\frac{\Delta W}{\Delta t} V = -RES(W^{n+1}) \#(10)$$

where $\Delta W = W^{n+1} - W^n$, $RES(W^{n+1})$ is the residual value of the n + 1th time step. The Taylor expansion of the right end term of the equation (10) can be obtained as the following equation:

$$\frac{\Delta W}{\Delta t} V = -\left(RES(W^n) + \frac{\partial RES(W^n)}{\partial W} \Delta W \right) \#(11)$$

Equation (11) can be expressed as:

$$A\Delta W = -RES(W) \#(12)$$

where

$$A = \frac{V}{\Delta t} I + \frac{\partial RES(W^n)}{\partial W} \#(13)$$

Define A = D + L + U; D, L and U are the diagonal matrix, lower triangular matrix and upper triangular matrix, respectively, of matrix A. The approximate decomposition of A is:

$$\begin{aligned} (D + L + U) &= D(I + D^{-1}L + D^{-1}U) \\ &\approx D(I + D^{-1}L)(I + D^{-1}U) \\ &= (D + L)D^{-1}(D + U) \#(14) \end{aligned}$$

Equation (12) can be converted into:

$$(D + L)D^{-1}(D + U)\Delta W = -RES(W) \#(15)$$

This equation is solved by the two processes of forward sweeping and backward sweeping. Set $\Delta W^* = D^{-1}(D + U)\Delta W$.

Forward sweep:

$$\Delta W^* = D^{-1}(-RES(W) - L\Delta W^*) \#(16)$$

Backward sweep:

$$\Delta W = \Delta W^* - D^{-1} U \Delta W \#(17)$$

Then, update the conservation variables of the flow field: $W^{n+1} = W^n + \Delta W$.

For two-dimensional cases, Figure 2 illustrates the two processes—forward sweeping and backward

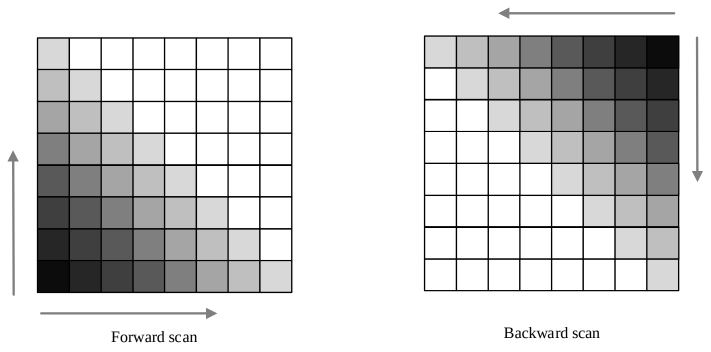


Figure 2. LU-SGS - diagrams of the two sweeps

sweeping, corresponding to equations (16) and (17), respectively.

It is observed from equations (16) and (17) that we need to only reverse the matrix *D* in the iterative process, which makes the LU-SGS method require highly marginal computational capacity relative to other implicit methods.

For unstructured grids, the residual value items of the *m* + 1 pseudo-time steps in equation (6) are linearized(Xiao, 2007), and equation (7) can be expressed as:

$$\left\{V_i^{n+1}\left(\frac{\Gamma_i}{\Delta\tau}+\frac{\varphi_{n+1}M_i}{\Delta t}+\frac{\partial RES_i(Q)}{\partial Q_i}\right)\right\}\Delta Q_i + \frac{\partial RES_i(Q)}{\partial Q_j}\Delta Q_j = -RES_i^*(Q^m)\#(18)$$

To simplify the calculation of implicit iterations, the flux calculation of the left side of the equation (18) takes the first order of precision, i.e.

$$RES_i(Q)=\frac{1}{2}\left(\tilde{F}_i+\tilde{F}_j-\Gamma_{ij}\tilde{\lambda}_{ij}\left(Q_j-Q_i\right)\right)\ \#(19)$$

where $\tilde{\lambda}_{ij}$ is the spectral radius of the boundary surface matrix $\Gamma^{-1}\tilde{A}_T$ of the control volume and $\tilde{A}_T=\partial\tilde{F}/\partial Q$. For closed control volume V_i, $\sum_{j=1}^{nfaces}\tilde{A}_{Ti}S_{ij}=0$; similarly, $\tilde{A}_T\Delta Q=\Delta\tilde{F}$. Equation (18) can be reduced to:

$$\left(\left(\frac{V_i^{n+1}}{\Delta\tau}+\frac{1}{2}\sum_{j=1}^{nfaces}\tilde{\lambda}_{ij}S_{ij}\right)I+\frac{\varphi_{n+1}V_i^{n+1}}{\Delta t}M_i\Gamma_i^{-1}\right)\Gamma_i\Delta Q_i = -RES_i^*(Q^m)-\frac{1}{2}\sum_{j=1}^{nfaces}\left(\Delta\tilde{F}_j-\Gamma_j\tilde{\lambda}_{ij}\Delta Q_j\right)S_{ij}\ \#(20)$$

In order to realize no-matrix computation, this study defines:

$$D=aI+bdiag(M_i\Gamma_i^{-1})\#(21)$$

where

$$a=\frac{V_i^{n+1}}{\Delta\tau}\frac{1}{2}\sum_{j=1}^{nfaces}\tilde{\lambda}_{ij}S_{ij}\,,b=\frac{\varphi_{n+1}V_i^{n+1}}{\Delta t}$$

After the reordering of unstructured grids(Sharov, 1998), equation (20) is solved by LU-SGS iteration

Forward sweep:

$$\Gamma_i\Delta Q^*=-D^{-1}RES_i^*(Q^m)-\frac{1}{2}D^{-1}\sum_{j\in L(i)}\left(\Delta\tilde{F}_j^*-\Gamma_j\tilde{\lambda}_{ij}\Delta Q_j^*\right)S_{ij}\ \#(22)$$

Backward sweep:

$$\Gamma_i\Delta Q_i=\Gamma_i\Delta Q^* - D^{-1}\left(\frac{1}{2}\sum_{j\in U(i)}\left(\Delta\tilde{F}_j-\Gamma_j\tilde{\lambda}_{ij}\Delta Q_j\right)S_{ij}\right)\#(23)$$

As observed from equations (22) and (23), the LU-SGS method uses $U$and L of V_i. For the structure grid, it is necessary to use the hyperplane to determine the upper and lower diagonals in the calculation to balance the upper and lower triangular matrices(Xu, 2015). However, for unstructured grids, because the order of grid nodes and cells does not satisfy the requirement of balance between the upper triangular matrix and lower triangular matrix, it is necessary to realize the hyperplane of similar structure grids by the reordering of meshes(Sharov, 1998), in order to reduce the time cost of calling the neighbour control in the process of calculation and to improve computational efficiency.

2.5 Law of Conservation of Geometry

The law of conservation of geometry is the fundamental condition to be satisfied in the process of dynamic grid flow calculation(Lesoinne, 1996). In order to prevent the non-physical solution caused by the movement of grids, the law of geometrical conservation must be satisfied in the calculation.

$$\frac{\partial}{\partial t}\iiint_V dV-\oiint_S v_{gn}dS=0\#(24)$$

The equation reveals the relationship between the volume change of the grid cell and the speed of the grid surface. That is, the volume change of each grid cell is equal to the sum of the volume swept by each grid surface during the motion. Discretization of equation (24) can be expressed as:

$$\frac{V_i^{n+1}-V_i^n}{\Delta t}=\sum_{j=1}^{nfaces}v_{gn}S_{ij}\ \#(25)$$

The volume $v_{gn}S$ of the mesh surface sweep can be calculated according to the coordinates of the two-time grid points. According to the volume V^{n+1} of the grid cell calculated n + 1 times by equation (25), the geometrical conservation law can be automatically satisfied rather than the cell volume calculated from the real grid coordinates by V^{n+1}.

3 OPTIMIZATION STRATEGY FOR NUMERICAL SIMULATION OF ROTOR FLOW FIELD

3.1 Parallel Design Scheme

THE parallel mode of this study adopts the master–slave mode, which is divided into the management node and the compute nodes. The management node calculates the serial part; the compute nodes carry on the parallel part computation. According to the serial flow of Figure 1, this study provides the parallel whole flowchart. As shown in Figure 3, the dashed line in the graph shows the serial and parallel parts of the process. The grey part indicates the need for communication; the communication method of this study utilizes MPI. Iterative algorithm using LU-SGS + Jacobi parallel algorithm.

3.2 Load Balancing Strategy

3.2.1 Load-balanced partitioning method

In the field of CFD parallel computing, data parallelism is one of the most common parallel methods. The fundamental concept is to divide the whole grid into N regions; then, the N regions are allocated to N calculation processes; each process carries out the initialization of the flow field information and the dual-time steps iteration. The exchange of data across virtual boundaries is carried out in the pseudo-time step iterative process. During the grid partition, it is necessary to realize the load balance of each compute node in order to improve the parallel efficiency. The method of this study is to guarantee that the number of grid cells of each compute node is fundamentally similar, because the dynamic grid is rigid motion(Yang, 2014), just once grid partition after the first assembly of the grid.

In this study, the partition of the grid is realized by calling Metis Library, which can guarantee the load balance of compute node. Metis is a partitioned algorithm library based on graph; the graph elements can be grid cell or grid point, and the topological relation of the graph element is required in partition; this is suitable for unstructured grid as well as structure mesh. In this study, in the case of the Metis library function, the graph element is the grid element, and a multi-level K-way partition method(Karypis, 1999) is used for partitioning the mixed grids. As shown in Figure 4, the method includes coarsening phase, initial partitioning phase and uncoarsening phase. Here, the coarsening phase is the condensation process of the graph, reduces the complexity of the graph and constructs the multi-level level of the graph. The initial partitioning phase is the first division of the graph after the coarsening; the uncoarsening phase is the splitting process of the graph, which restores the diagram to its original form and optimizes it in the recovery process layer by layer.

In this study, we call the Metis library function by C++ and integrate the partition function into the solution program of the Navier–Stokes equation. As shown in Table 1, the flow field grid of Robin model has 2 146 697 grid cells, which are divided into different regions using the multi-level K-way partitioning method; the number of grid cells in different regions is fundamentally similar, which realizes the load balance of each process calculation and communication.

3.2.2 Partition boundary processing method

For parallel CFD solver, in a time step iterative process, the internal flow field data of different partitions are computed independently, and the information exchange among the different partitions is realized by extending the virtual boundary. The mesh of a virtual boundary part does not participate in the calculation and is used only to receive data from the boundary cells of other zones. In the calculation of the flow field, gradient, limiter, original variable and flux of the virtual boundary are required. As shown in Figure 5, the concept is to identify neighbour cells that share grid points with the current cell, by traversing the grid cells within the partition. Then, program determine whether the neighbour cells and the current cell are in the same zone. If they are not in the same zone, the current cell is a boundary cell and the neighbour cells are the virtual boundary cells of the zone to which the current cell belongs. In this manner, after each region has traversed all the grid cells, the partition boundary and virtual boundary are established.

In order to store the virtual boundary information, this study designs a new data structure vector_map template class, which can be used to store the internal data of the region, partition and virtual boundaries and mapping relation of local ID and global ID. The data structure of the vector_map template class is as follows:

```
template <class T> class vector_map {
TDMap map_;
std::vector<T> items_;
std::array<std::vector<std::vector<T>>, layer>
item_recv_buf_;
std::array<std::vector<std::vector<T>>, layer>
item_send_buf_;
}
```

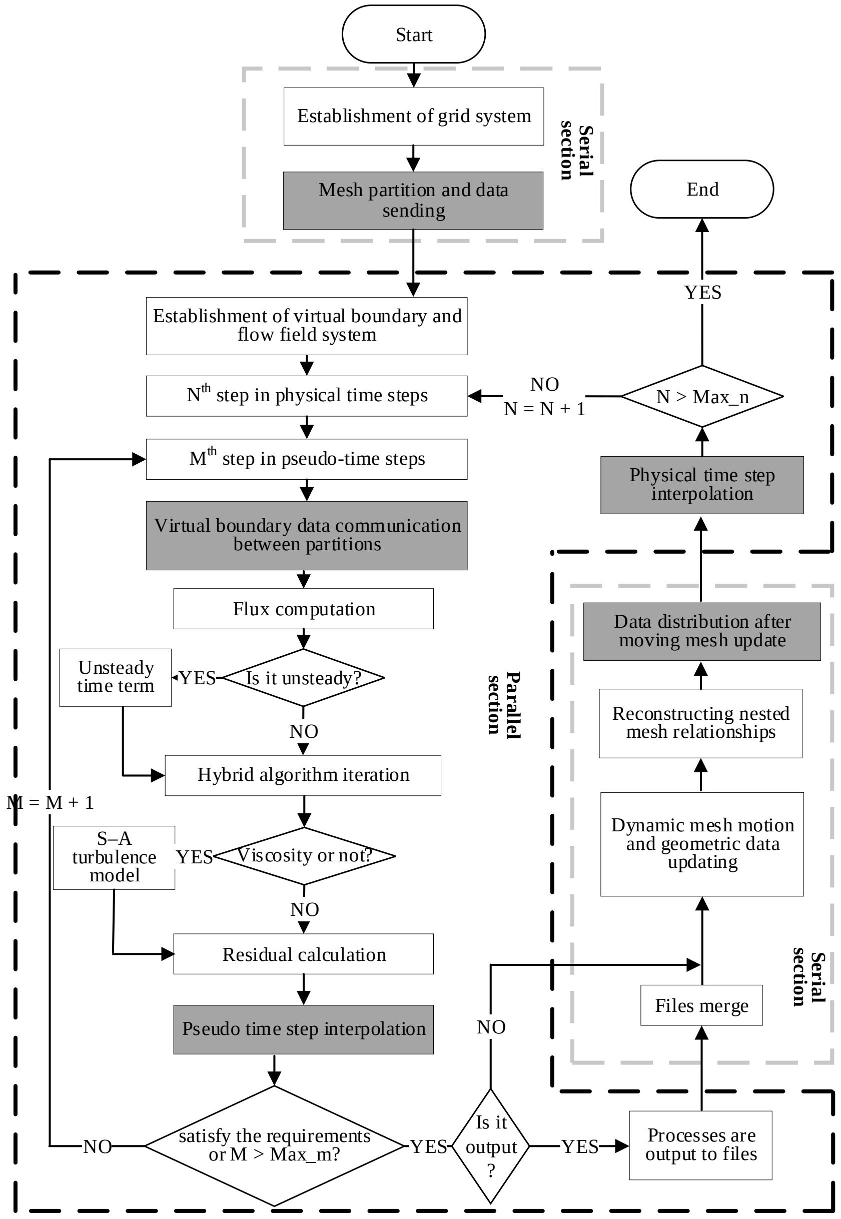

Start
Establishment of grid system
Mesh partition and data sending
Serial section
End
Establishment of virtual boundary and flow field system
N th step in physical time steps
NO
N = N + 1
N > Max_n
YES
M th step in pseudo-time steps
Physical time step interpolation
Virtual boundary data communication between partitions
Flux computation
Data distribution after moving mesh update
Unsteady time term
YES
Is it unsteady?
NO
Parallel section
Reconstructing nested mesh relationships
Hybrid algorithm iteration
M = M + 1
S–A turbulence model
YES
Viscosity or not?
NO
Dynamic mesh motion and geometric data updating
Residual calculation
Serial section
Files merge
Pseudo time step interpolation
NO
NO
satisfy the requirements or M > Max_m?
YES
Is it output ?
YES
Processes are output to files

Figure 3. Parallel whole flow chart

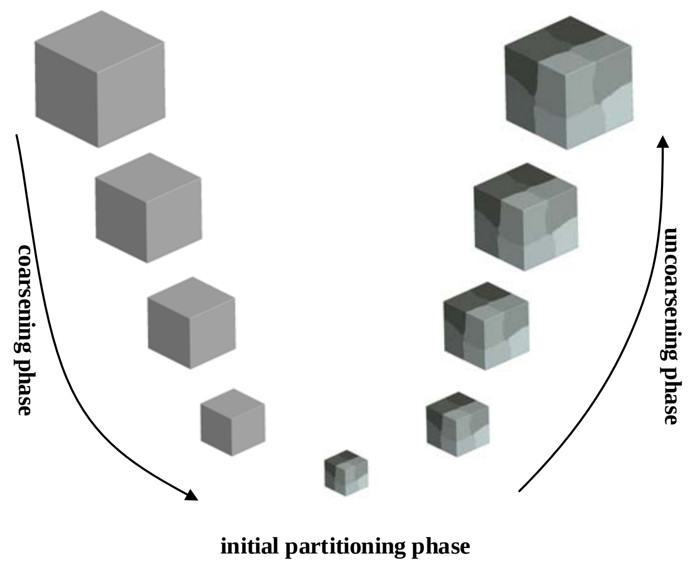
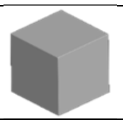


Figure 4. Schematic diagram of multi-level K-way partition method

Table 1. Results of different partitions in a grid

Grid partition diagram			
Number of partitions	1	4	8
Average number of cells	2 146 697	536 674	268 337
Maximum number of cells	2 146 697	536 695	268 357
Minimum number of cells	2 146 697	536 662	268 323

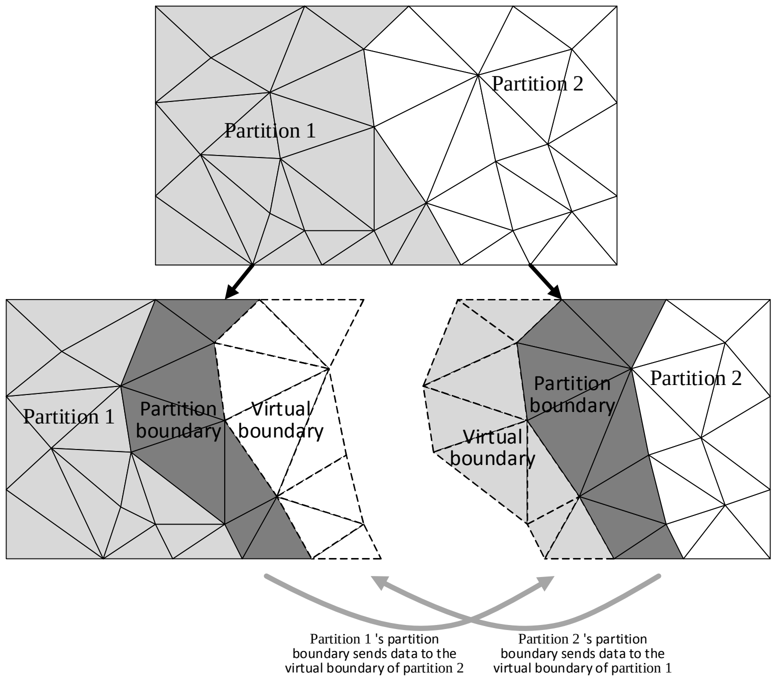

Partition 1
Partition 2
Partition 1
Partition boundary
Virtual boundary
Partition boundary
Partition 2
Virtual boundary
Partition 1 's partition boundary sends data to the virtual boundary of partition 2
Partition 2 's partition boundary sends data to the virtual boundary of partition 1

Figure 5. Parallel partition boundary processing diagram for unstructured grids

where "TDMap" is a multiple-index container, "map_" is used to store the mapping relationship between global ID and local ID and multiple index containers can be searched by global or local ID as indexes; "items_" is used to store internal data for partitions; "layer" represents the number of layers of virtual boundary; "item_send_buf_" is used to store partition boundaries' data; and "item_recv_buf_" is used to store virtual boundaries' data.

3.3 LU-SGS + Jacobi parallel algorithm

Wright and others studied the parallelization of implicit algorithm and put forward the DP-LUR method(Candler, 1994)(Wright, 1996). The DP-LUR method shifts the non-diagonal item to the right end of the equation to replace the symmetric Gauss–Seidel iteration with the Jacobi iteration; this removes the data dependency of the LU-SGS in the iterative process. The process of the DP-LUR method is as follows:

(1) Firstly:

$$\Delta Q_i^{(0)} = D^{-1}R_i \#(1)$$

(2) Then, a series of relaxation iterations are carried out; $m = 1{\sim}m_{max}$.

$$\Delta Q_i^{(m)} = D^{-1}\left(R - (U + L)\Delta Q_i^{(m-1)}\right)\#(2)$$

(3) Last:

$$\Delta Q_i = \Delta Q_i^{(m_{max})}\#(3)$$

In the DP-LUR method, m_{max} represents the number of sweeps for this algorithm in each partition, m_{max} generally assumes a value from three–six(Wright, 1996)(Wissink, 1996). The DP-LUR algorithm is preferable for parallel processing because the Jacobi iteration uses data from the previous step, the partition boundary of each partition also stores the data from the previous step and the parallel results can be consistent with the serial guarantee. Data communication occurs only at the partition boundary, the number of communication is related to the number of sweeps.

While DP-LUR is more convenient for parallelization than LU-SGS, it requires more computation. Because the convergence rate of the Jacobi iteration is lower than that of the Gauss–Seidel iteration, Jacobi requires a higher number of iterations to achieve convergence. Each time step that uses the DP-LUR iteration requires three–six sweeps to guarantee the convergence of the computation result. However, LU-SGS requires only two sweeps per time step, i.e., forward and backward sweep.

The DP-LUR algorithm is proposed for more effective parallelization; however, its essence is Jacobi iterative. The diagonal method of DP-LUR converges very gradually, this low rate of convergence is caused in part by the diagonal approximation(Wright, 1996). Therefore, this paper presents a LU-SGS + Jacobi parallel algorithm, that is, the LU-SGS iteration is performed within each partition, and the Jacobi iteration similar to the DP-LUR is used at each partition boundary to improve the parallel computational efficiency of numerical simulation. The process of the LU-SGS + Jacobi parallel algorithm is as follows:

(1) Firstly, initialization flux and conservation variable:

$$RES_i^*\left(Q^{(0)}\right) = 0\#(4)$$

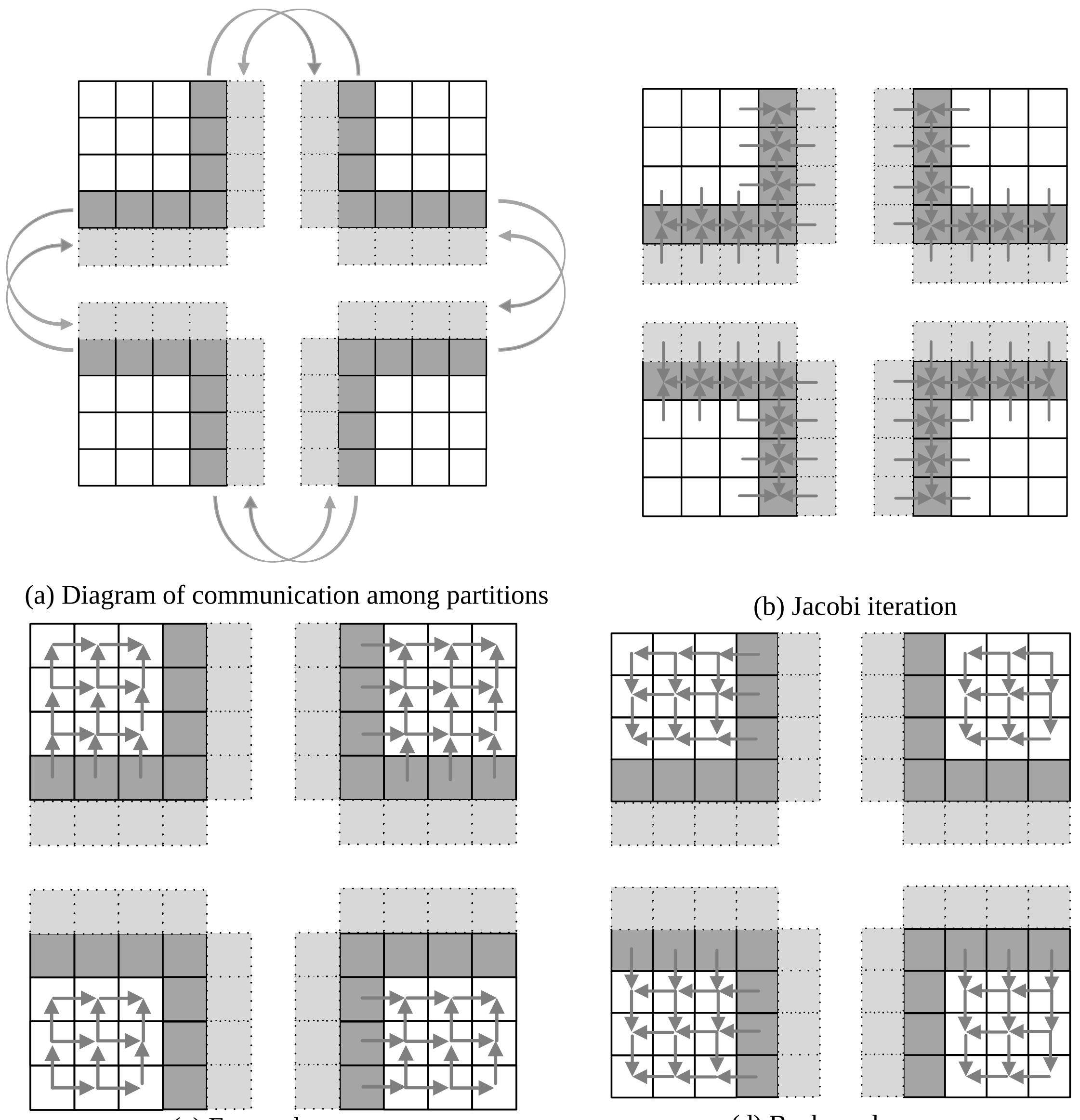
$$\Delta Q_i^{(0)} = D^{-1}R_i\#(5)$$

(2) Then, perform the pseudo-time step iteration:

```
for $m$ = 1 to $m_{steps}$ do
    Exchanging virtual boundary data: $\Delta Q^m = \Delta Q^{m-1}$.
    The formula (19) is used for flux calculation:
        $RES_i\left(Q_i^{(m)}\right) = \frac{1}{2}\left(\tilde{F}_i + \tilde{F}_j - \Gamma_{ij}\tilde{\lambda}_{ij}\left(Q_j^{(m)} - Q_i^{(m)}\right)\right)$
    Partition boundary adoption (2) for Jacobi iteration:
        $\Delta Q_i^{(m)} = D^{-1}\left(R - (U + L)\Delta Q_i^{(m-1)}\right)$
    LU-SGS iterations are performed with equations (22) and (22) within the partition:
    Forward sweep:
        $\Gamma_i\Delta Q^* = -D^{-1}RES_i^*(Q^m) - \frac{1}{2}D^{-1}\sum_{j\in L(i)}\left(\Delta\tilde{F}_j^* - \Gamma_j\tilde{\lambda}_{ij}\Delta Q_j^*\right)S_{ij}$
    Backward sweep:
        $\Gamma_i\Delta Q_i = \Gamma_i\Delta Q^* - D^{-1}\left(\frac{1}{2}\sum_{j\in U(i)}\left(\Delta\tilde{F}_j - \Gamma_j\tilde{\lambda}_{ij}\Delta Q_j\right)S_{ij}\right)$
    Turbulence model and interpolation calculation.
    Exchanging interpolate data in partitions.
end for
```

(3) Go into the next physical time step.

This study uses dual-time steps propulsion method; m_{steps} represents the number of iterations of the pseudo-time step. When only one partition is used, the parallel algorithm degrades into the original LU-SGS algorithm. When there are multiple partitions, the virtual boundary data of the adjacent partitions communicates with each other at each pseudo-time step; then, it takes the form of the Jacobi and LU-SGS iterations. The parallel algorithm in this study does not result in slow convergence as the parallel algorithm uses the LU-SGS algorithm in the partition interior. The computational flow of a pseudo-time step iteration of the parallel algorithm is shown in the following figure; the grey solid line portion of the Figure 6(a) represents the partition boundary, the grey dotted line represents the virtual boundary and the arrows represent data communication among the partitions; In order to simplify, only one layer of the virtual boundary is expanded. The following description uses V_i to represent the current control volume. V_{ij} represents neighbour control of V_i, and W is used to represent the conserved quantity of the control volume. Figure 6(a) indicates data communication among the partitions, and Figure 6(b) represents the Jacobi iteration. In each iteration, V_i uses V_{ij}'s W_{ij}^{n-1} to update W_i^n. Figure 6(c) represents


(a) Diagram of communication among partitions

(b) Jacobi iteration

(c) Forward sweep

(d) Backward sweep

Figure 6. Parallel implicit algorithm execution flow diagram

the forward sweep of LU-SGS, and Figure 6(d) represents the backward sweep of LU-SGS. During the dual-times sweep, V_i uses V_{ij}'s W_{ij}^n to update W_i^n. The arrows in Figures 6(b), (c) and (d) represent data dependencies. The virtual boundary is not involved in the calculation and is used only to store the data of the previous time step. The partition boundary zone only conducts Jacobi iterations. The partition interior zone (white cells in the diagram) conducts LU-SGS iteration. The partition's internal control volume can use the data of the partition boundary control volume during the iteration.

In the process of solving the serial Navier–Stokes equation, V_i requires V_{ij}'s W_{ij}^n to update W_i^n without considering whether V_{ij} is located in the various partitions. However, in a parallel program, if the V_{ij}'s W_{ij}^n that V_i requires is in another partition, it is necessary to obtain W_{ij}^n through communication. If V_i obtains W_{ij}^n in real time from a different partition during the calculation, the overhead of the communication will be large. In order to solve this problem, two-layer virtual boundary is established in each partition boundary to store W^{n-1}. After one pseudo-time step iteration, the virtual boundary of the various partitions is communicated to update W^{n-1} of virtual boundary. When V_i updates W_i^n on the partition boundary, it requires a partial V_{ij}'s W_{ij}^{n-1}in the virtual boundary because the Jacobi algorithm uses W^{n-1} to update W^n. The Jacobi iterative algorithm is used at the partition boundary. The internal V_i of each partition uses the efficient implicit LU-SGS iteration. With the use of the LU-SGS + Jacobi parallel algorithm to simulate the Robin model, the experimental results coincide with the serial program results, and the parallel efficiency is high.

4 EXPERIMENT AND RESULT ANALYSIS

IN order to test the parallel computing results and parallel efficiency of the hybrid algorithm, a set of three-dimensional mixed grids is tested. The mesh scale is 13 million mesh points. The test platform is the Inspur high-performance server cluster with five operational nodes. The CPU is 2.2 the GHz Intel Xeon E5-2630v 20 core processor, and the memory for each node is 256 Gb. The Robin fuselage/rotor model is shown in Figure 7.

4.1 Example verification

The parameters of the Robin model are as follows: Ma = 0.08, alpha = 0, beta = 0, cfl = 50, number of blades = 4 and number of cycles per period = 180. Each physical time step iteration contains 15 pseudo-time step iterations. The airfoil is NACA0012, and the turbulence model is the S–A model. In this paper, the forward flight state of the model flow field is simulated. When the calculation is stable, the pressure coefficient distribution of the fuselage and rotor in this study is compared with the serial results, as shown in Figures 8, 9, 10 and 11. Figure 11 shows the parallel pressure distribution and serial pressure distribution comparison of rotor blade in the span-wise directions at the four positions r/R = 20%, r/R = 40%, r/R = 60% and r/R = 80% depicted in Figure 10. The calculated results are consistent with the serial data.

Vorticity is used to describe the rotational motion of fluid clusters; it is an important physical quantity to describe the vortex motion of an object. Figures 12 and 13 show the comparison between the parallel tip vortex trajectory and the serial calculation results. Figure 14 shows the comparison of lift coefficient results between serial programs and parallel programs, where the dashed lines represent serial lift coefficient curves, and the solid lines represent parallel lift coefficient curves. The serial and parallel errors of lift coefficient are approximately 1%. The example demonstrates that the results of the hybrid algorithm coincide with the serial results. Through the comparison of the above data, the correctness of the hybrid algorithm is explained.

4.2 Speedup and parallel efficiency analysis

In order to evaluate the efficiency of parallel computing, the definition formulas of the speedup ratio *S* and parallel efficiency *E* are introduced:

$$S = \frac{T_1}{T_n}; E = \frac{S}{n} \times 100\%$$

Here, T_1 represents the average running time of a pseudo-time iteration of the serial program, n represents the number of processes and T_n represents the average running time of a pseudo-time iteration for each process. Table 2 presents the results of the hybrid algorithm on different processes with grid size of 13 million, including the average run time T_n, the acceleration ratio *S* and the parallel efficiency *E* of a pseudo time step.

Figure 15 shows the average run time of a pseudo-time step iteration of the hybrid algorithm in the different processes. It is observed from the graph that when the number of processes reaches over 32, the running time begins to stabilize the number of processes in less cases, the area of grid division is less, each region of the calculation time is significantly larger than the communication time between the processes and the impact of the communication time on the computing time is marginal. Moreover, the average running time decreases rapidly as the number of processes increases. When the number of processes reaches a certain number, the average running time tends to be stable because with the increase in the number of processes, the area of grid division keeps reducing, each process requires fewer and fewer calculation data and the communication time keeps increasing; then, the communication time exerts a larger impact on the average running time.

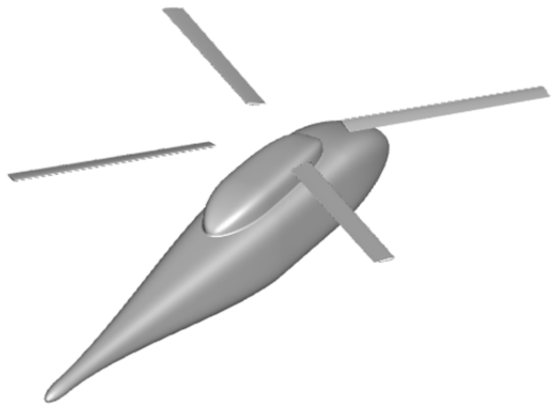


Figure 7. Robin fuselage/rotor model

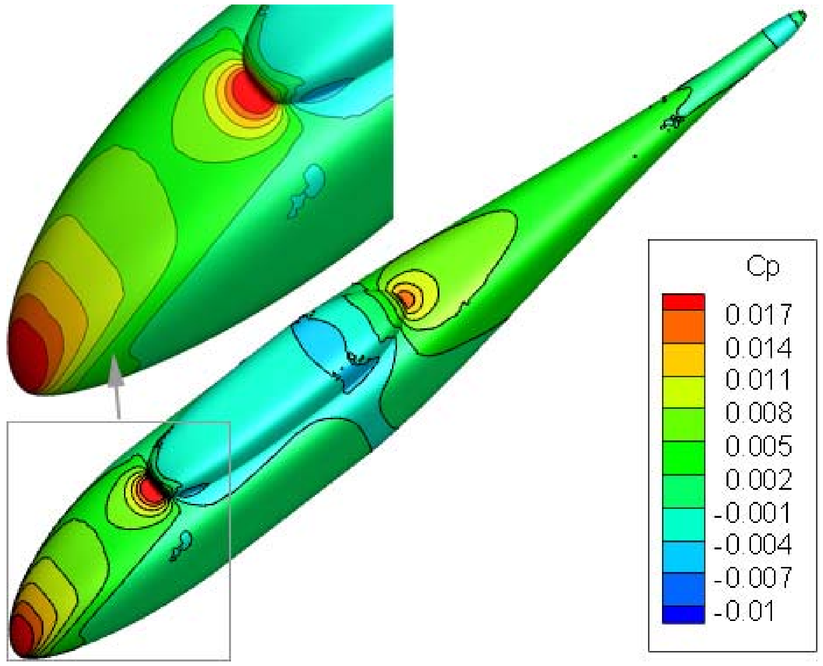


Figure 8. Serial fuselage pressure coefficient distribution cloud chart

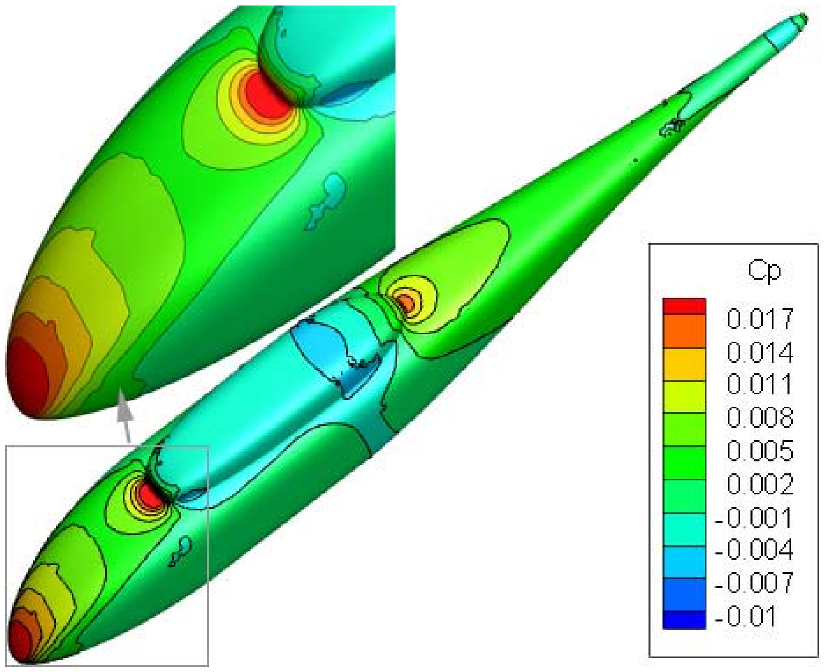


Figure 9. Parallel fuselage pressure coefficient distribution cloud chart

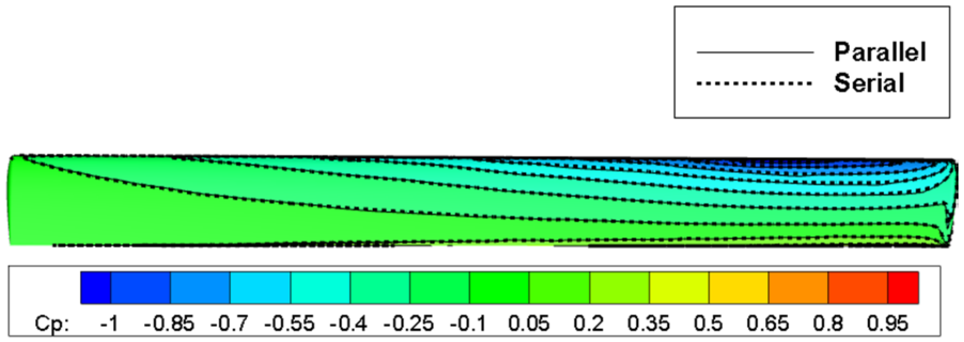
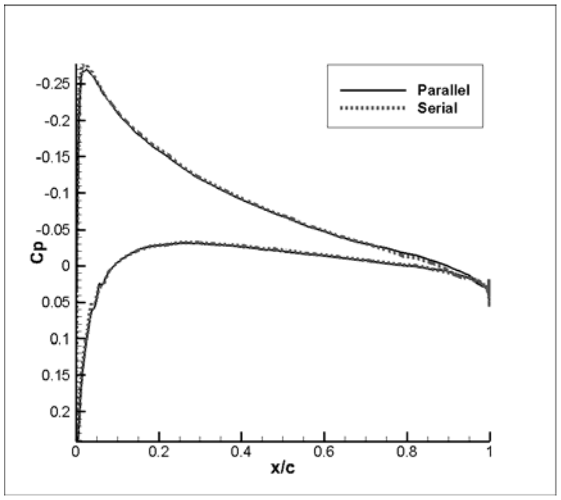
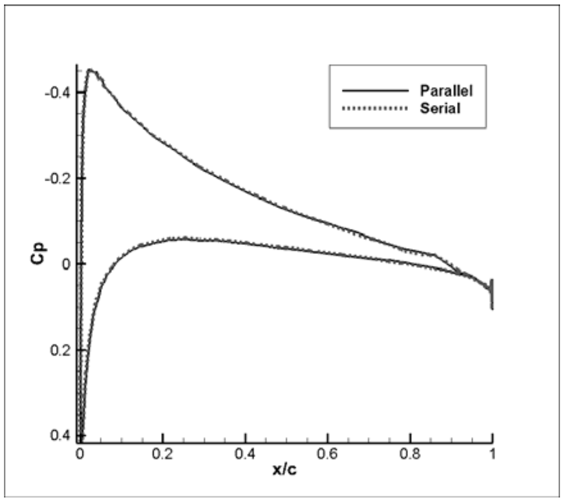


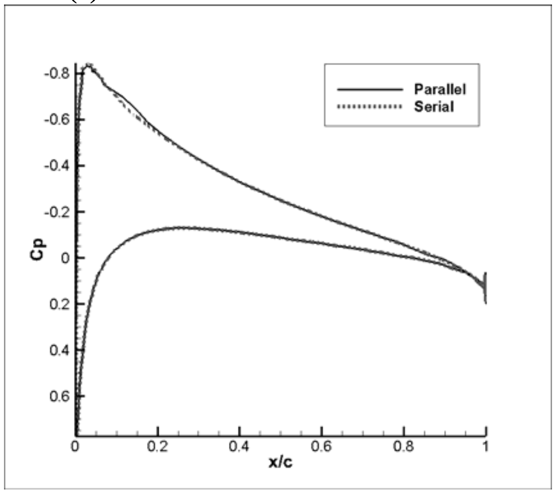
Figure 10. **NACA0012 rotor pressure coefficient distribution cloud chart**



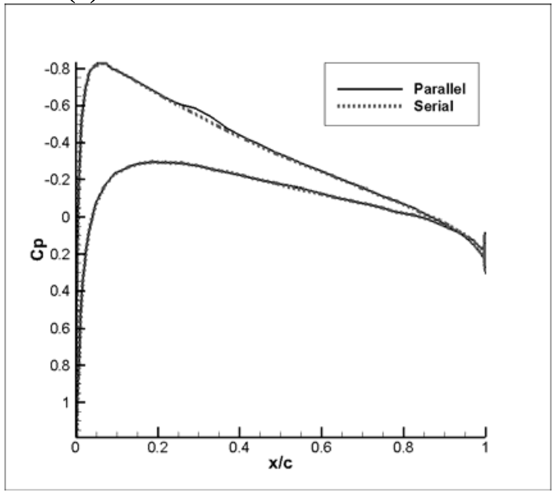
(a) Pressure distribution at r/R = 20%



(b) Pressure distribution at r/R = 40%



(c) Pressure distribution at r/R = 60%



(d) Pressure distribution at r/R = 80%

Figure 11. **NACA0012 rotor pressure coefficient distribution**

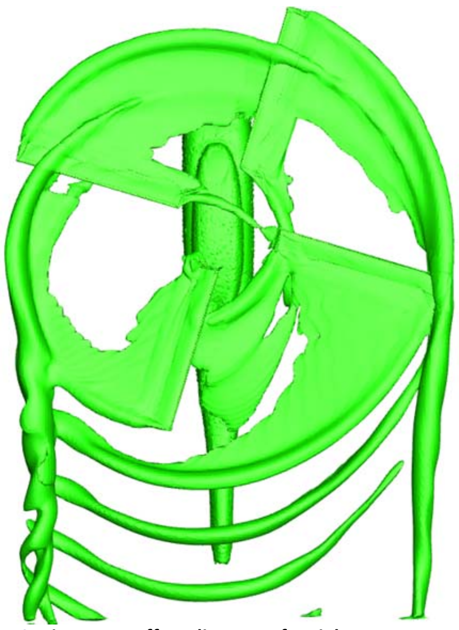


Figure 12. Tip vortex effect diagram of serial programs

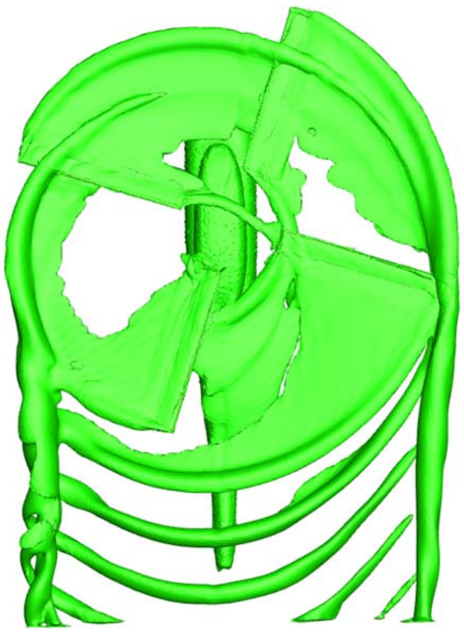


Figure 13. Tip vortex effect diagram of parallel programs

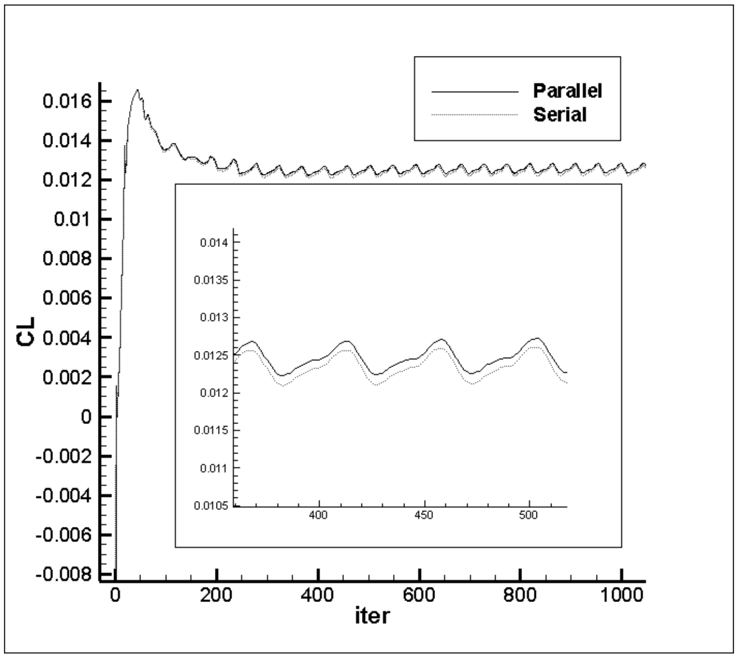


Figure 14. Comparison of lift coefficient results between serial programs and parallel programs

Table 2. Parallel speedup and parallel efficiency of 1300W grid scale

n	T_n (s)	S	E (%)
1	1051	1	100
2	600	1.75	87.5
4	313	3.36	84
8	168	6.26	78.3
16	105	10.01	62.6
32	66	15.91	49.7
64	54	19.46	30.4

Figure 16 shows the parallel speedup of the hybrid algorithm in the Robin model. The number of processes is less than 16, and the speedup ratio grows almost linearly. When the number of processes reaches a certain number, the parallel speedup ratio tends to be stable, because the average running time of the hybrid algorithm tends to stabilize as the number of processes increases; From the acceleration formula defined above *S* will also be stable.

Figure 17 shows the parallel efficiency of the hybrid algorithm in the Robin model. With the increase of the number of processes, the parallel efficiency declines continuously, which is due to the non-linear increase of the parallel acceleration ratio; the phenomenon results in the gradual decrease of the parallel efficiency. Therefore, it is necessary to select an appropriate number of processes to ensure the higher speedup ratio and parallel efficiency. As observed in Figs 15 and 16, the larger is the grid size, the higher is the parallel speedup ratio and the parallel efficiency.

5 CONCLUSION

A numerical optimization algorithm for unsteady flows of rotor based on web service is proposed, which is suitable for numerical simulation of steady and unsteady rotor flow field in three-dimensional hybrid grids. The efficient use of computer resources is realized by service computing, which greatly improves the speed of parallel computing. Using Metis to partition the grid, the load balance of the partition is realized. The problem of communication efficiency among the partitions is solved by establishing virtual boundary for partition boundary. The method is numerically simulated by a Robin model, and the experimental results coincide with those of the serial program; moreover, the parallel efficiency is high.

6 ACKNOWLEDGMENT

THIS work is partly supported by the National Key Technology Research and Development Program under Grant No. 2018YFB0204001; National Natural Science Foundation of China under Grants No. 61672200, No. 61572163; Key Technology Research and Development Program of the Zhejiang Province under Grants No. 2019C01059, No. 2019C03135 and No. 2019C03134; The Zhejiang Natural Science Funds under Grants No. LY17F020029 and No. LY16F020018; State Key Laboratory of Computer Architecture Project No. CARCH201712; Hangzhou Dianzi University Postgraduate Research Innovation Fund Program under Grants No. CXJJ2018052.

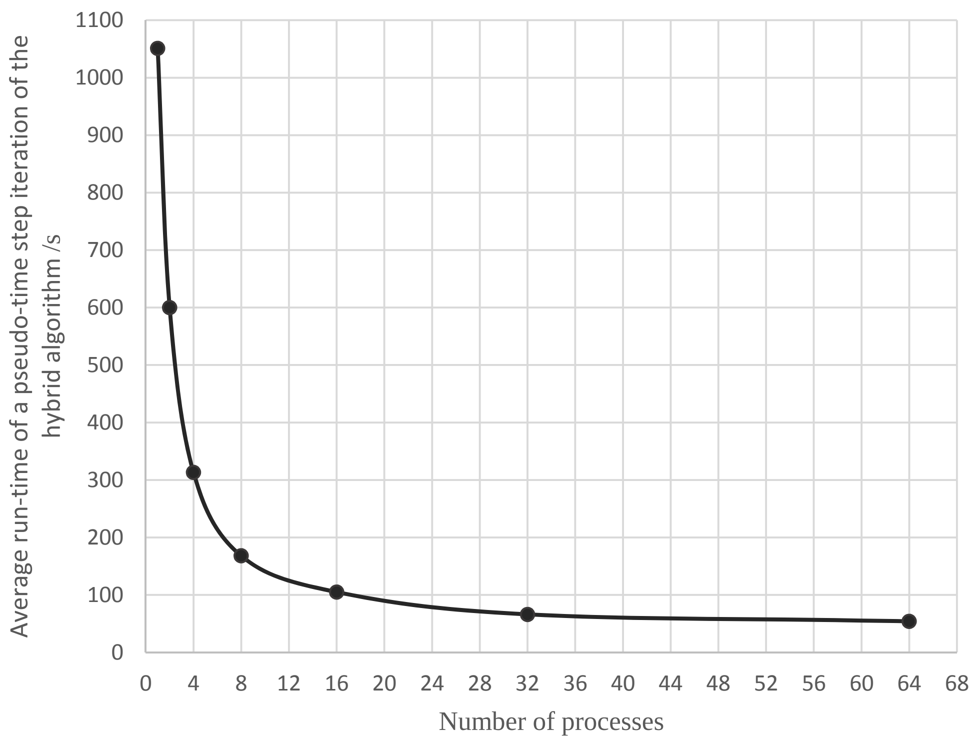


Figure 15. **Average runn-time of the hybrid algorithm in an inner iteration**

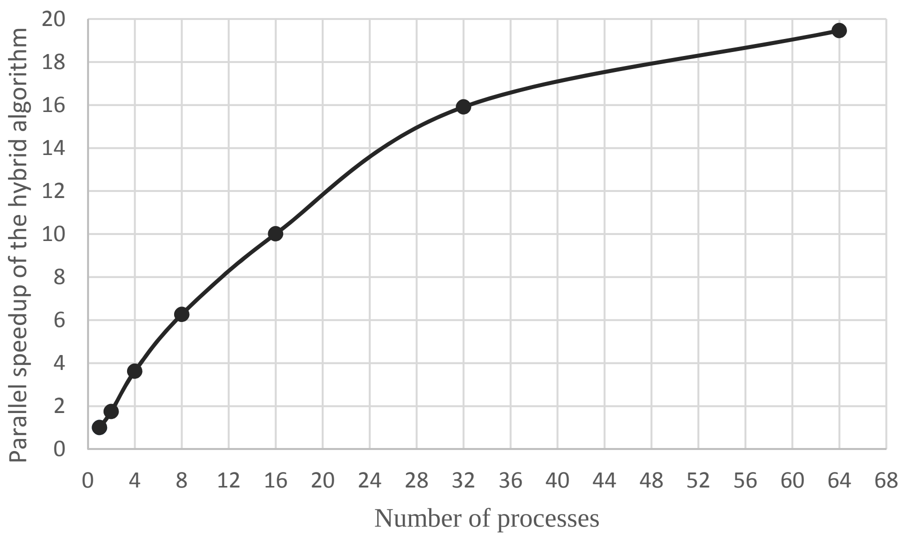


Figure 16. **Parallel speedup of the hybrid algorithm in the Robin model**

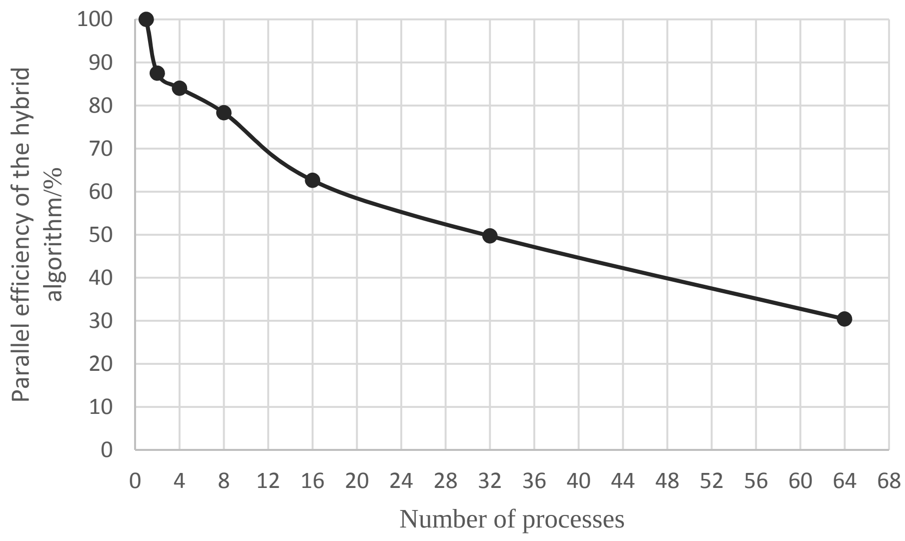


Figure 17. **Parallel efficiency of the hybrid algorithm in the Robin model**

7 REFERENCES

Jameson, A., Schmidt, W., & Turkel, E. (1981, June). Numerical solution of the Euler equations by finite volume methods using Runge Kutta time stepping schemes. In 14th fluid and plasma dynamics conference (p. 1259).

Wei, L., Lilun, Z., & Yongxian, W. (2013). Foundations of Computational Aerodynamics Parallel Programing. National Defense Industry Press, Beijing.

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

Candler, G. V., Wright, M. J., & McDonald, J. D. (1994). Data-parallel lower-upper relaxation method for reacting flows. AIAA journal, 32(12), 2380-2386.

Wang Song, Wang Hai-yang, Wu Ya-dong, Wu Bin & Wu Ying-chuan. (2017). Application of large-scale CFD flowfield visualization analysis system. Journal of Aerospace Power, 32(5):1138-1147.

Zhao Xiuyan. (2009). MHD analysis of metal fluid (Doctoral dissertation, Jinan: Shandong Agricultural University).

Pan Yong. (2007). Numerical Method for Hypersonic Flowfield with Magnetic Interface (Doctoral dissertation, Nanjing: Nanjing University of Aeronautics and Astronautics).

Van Leer, B. (1979). Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. Journal of computational Physics, 32(1), 101-136.

Li Xuesong, & Xu Jianzhong. (2008). A parallel solution design for unsteady Navier-Stokes equation. Journal of Engineering Thermophysics, 29(1):52-54.

Zheng, G., Deng, S., Han, T., & Yang, G. (2011). An implicit parallel computing method based on the Navier-Stokes equations with hybrid grids. Yingyong Lixue Xuebao(Chinese Journal of Applied Mechanics), 28(3), 211-218.

Obayashi, S., & Guruswamy, G. P. (1995). Convergence acceleration of a Navier-Stokes solver for efficient static aeroelastic computations. AIAA journal, 33(6), 1134-1141.

Liou, M. S., & Steffen Jr, C. J. (1993). A new flux splitting scheme. Journal of Computational physics, 107(1), 23-39.

Roe, P. L. (1986). Characteristic-based schemes for the Euler equations. Annual review of fluid mechanics, 18(1), 337-365.

Chen, G., Wang, L., Lu, Z., & Liang, X. (2011). Development of ten-thousand-core parallel software CCFD for aircraft aerodynamics simulation. Huazhong Keji Daxue Xuebao(Ziran Kexue Ban)/ Journal of Huazhong University of Science and Technology(Nature Science Edition), 39.

Zhongyun, X. (2005). Investigation of Computational Modeling Techniques for Rotor Flow Fields (Doctoral dissertation, Ph. D. Dissertation, Fluid Mechanics, Mianyang, China Aerodynamics Research and Development Center, 2007 (in Chinese)).

Ji Changrui, Yang Xiaoquan, Yang Aiming, Si Jiangtao, & Liu Peiqing. (2014). Strong coupled RANS algorithm for simulating hovering rotor flow. Journal of Aerospace Power, 29(8), 1894-1903.

Li, P., & Zhao, Q. J. (2014). CFD calculations on the interaction flowfield and aerodynamic force of tiltrotor/wing in hover. Acta Aeronautica et Astronautica Sinica, 35(2), 361-371.

Jiang Yuening, Jia Hongguang, & Li Ming. (2018). CFD numerical simulation of unmanned aerial vehicle based on multi-core parallel computation. Computer Engineering and Applications, 54(7), 221-225.

Zhu Ziqiang, Wu Ziniu, & Li Jin. (1998). Applied computational fluid dynamics.

Jameson, A. (1991, June). Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings. In 10th Computational Fluid Dynamics Conference (p. 1596).

Weiss J M, Smith W A. Preconditioning applied to variable and constant density flows[J]. AIAA journal, 1995, 33(11): 2050-2057.

Mavriplis, D. (2003, June). Revisiting the least-squares procedure for gradient reconstruction on unstructured meshes. In 16th AIAA Computational Fluid Dynamics Conference (p. 3986).

Venkatakrishnan, V. (1993, January). On the accuracy of limiters and convergence to steady state solutions. In 31st Aerospace Sciences Meeting (p. 880).

Luo, H., Baum, J. D., & Löhner, R. (1998). A fast, matrix-free implicit method for compressible flows on unstructured grids. Journal of Computational Physics, 146(2), 664-690.

Sharov, D., & Nakahashi, K. (1998). Low speed preconditioning and LU-SGS scheme for 3-D viscous flow computations on unstructured grids. In 36th AIAA Aerospace Sciences Meeting and Exhibit (p. 614).

Han Zhirong. (2013). Grid Adaption and Parallel Computing in the Application of Airloads Computations. (Doctoral dissertation, Nanjing University of Aeronautics and Astronautics).

Xiao, T., Ang, H., & Yu, S. (2007). A preconditioned dual time-stepping procedure coupled with matrix-free LU-SGS scheme for unsteady low speed viscous flows with moving objects. International Journal of Computational Fluid Dynamics, 21(3-4), 165-173.

Xu, T., & Chen, L. (2016). Gpu Implementation of a Viscous Flow Solver on Unstructured Grids. In International Journal of Modern Physics: Conference Series (Vol. 42, p. 1660167). World Scientific Publishing Company.

Lesoinne, M., & Farhat, C. (1996). Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. Computer methods in applied mechanics and engineering, 134(1-2), 71-90.

Yang Xiaochuan, Wang Yuntao, Zhang Yulun, & Meng Dehong. (2014). Numerical simulation of wind turbine based on rigid moving mesh method. National Conference on Fluid Mechanics.

Karypis, G., & Kumar, V. (1999). Parallel multilevel series k-way partitioning scheme for irregular graphs. Siam Review, 41(2), 278-300.

Wright, M. , Candler, G. , & Prampolini, M. . (2013). A data-parallel lu relaxation method for the navier-stokes equations. Aiaa Journal, 34(7), 1371-1377.

Wissink, A. M. , Lyrintzis, A. S. , & Strawn, R. C. . (1996). Parallelization of a three-dimensional flow solver for euler rotorcraft aerodynamics predictions. AIAA Journal, 34(11), 2276-2283.

8 DISCLOSURE STATEMENT

NO potential conflict of interest was reported by the authors.

9 NOTES ON CONTRIBUTORS

Jilin Zhang received the PhD degree in Computer Application Technology from University of Science Technology Beijing, Beijing, China, in 2009. He serves as an associate professor in School of Computer Science and Technology, Hangzhou Dianzi University. His research interests include High Performance Computing and Cloud Computing.
Email: jilin.zhang@hdu.edu.cn

Xuechao Liu is now M.S. in School of Computer Science and Technology in Hangzhou Dianzi University, China. His research interests include Parallel Computing, Computational Fluid Dynamics and High Performance Computing.
Email: xuechao_liu@yeah.net

Jian Wan received the Ph.D. degree in Computer Application Technology from Zhejiang University, Zhejiang, China, in 1989. He is currently a Professor in software engineering in Zhejiang University of Science and Technology, China. His research interests include Grid Computing, Service Computing and Cloud Computing.
Email: wanjian@hdu.edu.cn

Yongjian Ren received the PhD degree in Engineering from Zhejiang University, Hangzhou, China, in 1989. He is currently an distinguished professor at Hangzhou Dianzi University. His research interests include mass storage and cloud computing.
Email: yongjian.ren@hdu.edu.cn

Binglin Xu is now M.S. in School of Computer Science and Technology in Hangzhou Dianzi University, China. His research interests include Parallel Computing, Computational Fluid Dynamics and High Performance Computing.

Jianfan He is now M.S. in School of Computer Science and Technology in Hangzhou Dianzi University, China. His research interests include Parallel Computing, Computational Fluid Dynamics and High Performance Computing.

YuChen Fan is now M.S. in School of Computer Science and Technology in HangZhou Dianzi University, China. His research interests include Parallel Computing, Machine Learning, Computational Fluid Dynamics and High Performance Computing.

Li Zhou received her Master Degree from Hangzhou Dianzi University, Hangzhou, China, in 2003. She is currently an associate professor in School of Computer Science and Technology, Hangzhou Dianzi University. Her current research interests include virtual storage system, cloud storage, cloud computing and high performance computing.

Zhenguo Wei graduated from Business Administration Major of Zhejiang University of Technology. In 2000, he worked in Dawning Information Technology Co., Ltd. In 2018, he worked in Zhejiang Dawning Information Technology Co., Ltd. As the general manager. He was fully responsible for the business of the enterprise and responsible for the research and development and management of software products related to high-performance computing, big data and artificial intelligence.

Juncong Zhang graduated from Computer Science and Technology Major of Xiangfan University. In 2018, he worked in Zhejiang Dawning Information Technology Co., Ltd. As a manager of the technology center. Responsible for the research and development of software products and technical support related to High-performance computing, big data and artificial intelligence.

Jue Wang is currently working as a associate professor in the supercomputing center of Chinese Academy of Science. The motivation behind his work is to improve soft systems by increasing the productivity of programmers and by increasing software performance on modern architectures including many cores clusters and GPU.