ISSN 1349-1113
JAXA-RR-07-024E

JAXA Research and Development Report

Comprehensive Study on Interpolation Methods for Helicopter Simulation : Interpolation between Cartesian and Curvilinear Grids in Parallel Computation

Choongmo YANG, Takashi AOYAMA

February 2008

Japan Aerospace Exploration Agency

Comprehensive Study on Interpolation Methods for Helicopter Simulation: Interpolation between Cartesian and Curvilinear Grids in Parallel Computation*

Choongmo YANG*1 and Takashi AOYAMA*1

ヘリコプタのシミュレーションにおける補間法の研究：並列計算における直交格子と曲線格子の補間*

梁　忠模*1、青山　剛史*1

ABSTRACT

The importance of accurate and fast interpolation algorithm is growing up for multi-body configuration with arbitrary overlap. One of the critical cases can be a helicopter simulation because of the complex relative movement of rotor-rotation and body motion. In this paper, new searching algorithms are implemented for the interpolation between two different grids, Cartesian grid and curvilinear grid, of which the overlapped grid system consists for the massive computation of the full helicopter configuration. These searching algorithms are proposed to make full use of (1) the characteristics of Cartesian grid, (2) special geometric configuration of helicopter, and (3) load balancing in parallel computation. In the first stage, Alternating Index Searching (AIS) algorithm, which alternates a searching direction by jumping the grid index to the searching point, is constructed to compare the iteration speed with a conventional Linear Searching (LS) algorithm. Simple two dimensional problems and three dimensional helicopter simulations are conducted to compare the efficiency of these searching algorithms. The result shows a considerable enhancement in computing time for whole computation domain. In the second stage, Reverse Index Searching (RIS) algorithm, which is developed to consider the load balance among each processing element (PE) during parallel computation, is proposed. By applying these searching algorithms, efficient massive computation can be achieved for the helicopter configuration.

Key Words: *Helicopter, CFD, Interpolation, Searching algorithm*

概　　要

ロータの回転や６自由度の機体運動等の複雑な動きのため、メイン／テールロータ及び胴体からなるヘリコプタの数値シミュレーションにおいて、正確で速い補間アルゴリズムの重要性が高まってきている。本報告では、直交格子と曲線格子で構成される移動重合格子を利用した CFD コードに対して、より正確で早い補間法を提案した。新しい補間法では、（1）直交格子の特性、（2）ヘリコプタ・ブレードの特殊な幾何学的配置、及び（3）並列計算時の計算負荷バランスなどを十分有効に利用できるアルゴリズムが考案されている。第一章では、Alternating Index Searching (AIS) アルゴリズムを提案し、従来の Linear Searching アルゴリズムに対し、2 次元の簡単なケースと実際のヘリコプタを模擬した 3 次元計算のケースで補間計算の速度を比べた。第二章では、並列計算における各計算ノードの負荷バランスを考慮した Reverse Index Searching (RIS) アルゴリズムを提案した。この補間法を利用することによって、ヘリコプタのより効果的な大型計算を実現することができた。

* 平成 19 年 11 月 29 日　受付（Received 29 November, 2007）

*1 総合技術研究本部　計算科学研究グループ(Computational Science Research Group, Institute of Aerospace Technology)

1. INTRODUCTION

When using overlapped grid or other kinds of multiple meshes, solutions on the first grid must be accurately interpolated and transferred to the second grid for the calculation to proceed. In the case of structured/unstructured grids or the more general multi-physics case of two independent grids with arbitrary overlap, this grid transfer operation (i.e. operation to transfer information from one grid to the other grid) is more complicated. It involves a search for which element of the first mesh contains each nodal point of the second mesh, and a subsequent interpolation. [1] If the two meshes move relatively to each other, then the search operation must be repeatedly invoked. It needs considerably high computational cost especially for the calculation of complex motion such as helicopter in maneuver.

JAXA has been developing its own full helicopter simulation code by combining accurate CFD solver and acoustic solver[2]. The flow solver uses moving overlapped grid method, which is one of the most advanced techniques for tip-vortex capturing at present. The moving overlapped grid system is composed of three different types of grids (blade grid, inner and outer background grids), and simple bi-linear interpolation method is used to exchange the information between each grid during calculation. The acoustic code, which is based on Ffowcs Williams and Hawkings (FW-H) formulation, uses the pressure distribution on blade surface obtained by the CFD code as input data.

Our previous researches[2,3] have shown its ability to capture the distinct peak of BVI noise for several problems. Also the code is expanding its ability to solve the flow-field including tail-rotor and fuselage configuration for interaction noise analysis. The noise generated by a maneuvering rotorcraft is the next step to full helicopter simulation. To get accurate noise signals for the noise problems of helicopter, it is very important to solve the fully unsteady flow field with high-accuracy for several rotor revolutions. Because of complex movement of helicopter including rotor-rotation and flight motion of all parts of helicopter, the importance of accurate and fast interpolation algorithm between overlapped grids is growing up. Specially, for the computation of a maneuvering helicopter, the computing efficiency becomes one of the bottle-necks. In these previous works, several interpolation algorithms were implemented and compared for elapsed computing time.

Recent helicopter simulations in world-leading research groups including JAXA in Japan, ONERA[4, 5] in France, and NASA[6, 7] in US are conducted by CFD code using overlapped or overset grid, which is consisted of both Cartesian and curvilinear grid. And in the middle of the development process of the CFD code, the proper interpolation or searching algorithm is one of bottle neck to achieve good computation performance. Even the interpolation methods of these CFD codes are very important, not so much information is open to public. Concerning the searching algorithm, Alternating Digital Tree (ADT) [8] algorithm is also known to have excellent searching speed to find the intersection line or plane between two meshes. But if we restrict the interpolation for a structured grid, which can be characterized by linear indexing along computational coordinate axis, we need to choose or develop other kinds of searching algorithm to make full use of this grid indexing characteristic. As a new searching algorithm, we proposed Alternating Index Searching (AIS) algorithm. It searches the nearest point by checking neighbor points to alternate its index direction for a given point. Compared to linear index searching algorithm, AIS algorithm showed better performance to accelerate calculation with overlapped grid system[9]. Moreover, for massive computation such as the present helicopter simulation, parallel computing is inevitably necessary. In the respect of interpolation method or searching algorithm, parallel computing requires new algorithm to achieve the optimal load balance between each processing elements (PEs).

In this paper, an innovative searching algorithm is tested considering the characteristics of present CFD code using overlapped grid, which is consisted of both Cartesian and curvilinear grid. At first, spatial searching algorithm is checked when using a single process element. Then the adaptability for parallel computation is considered to achieve the better performance in considering the geometric relation during rotor rotation. By applying this algorithm, efficient massive computation can be achieved for the full helicopter configuration. The results can provide the valuable information to the CFD code developers for helicopter simulations which use the same configuration of grid system with either a single process element or parallel computation.

2. OVERLAPPED GRID SYSTEM

2.1 Grid System and Numerical Methods

A moving overlapped grid system with three different types of grids (rotor grid, inner and outer background grids) is used to simulate BVI of helicopter. Figure 1 shows the computational domain for three types of grid and the geometric dimensions for whole domain. The blade grid rotates in the Cartesian background grid. The Cartesian background grid is divided into the two parts as shown in Fig. 2, which shows a perspective view of grid system for the whole computational domain. One is the inner background grid and the other is the outer background grid. The inner background grid is placed around the rotor disk, and the outer background grid covers whole computation region with sparse grid density. The calculated flow data are exchanged between inner and outer background grids during calculation steps. Huge numbers of grid points are distributed to the inner background grid to achieve higher resolution because the density of grid directly affects the strength of numerical viscosity. The body-fitted blade grid in O-H topology moves along with the blade motion including rotation, flapping, feathering, and lagging, which is shown in Fig. 3. The number of grid points in span-wise direction is considerably increased to match the grid density of the blade grid with that of the inner background grid. The size of the blade grid in normal direction is nearly equal to the chord length as shown in Fig. 4.

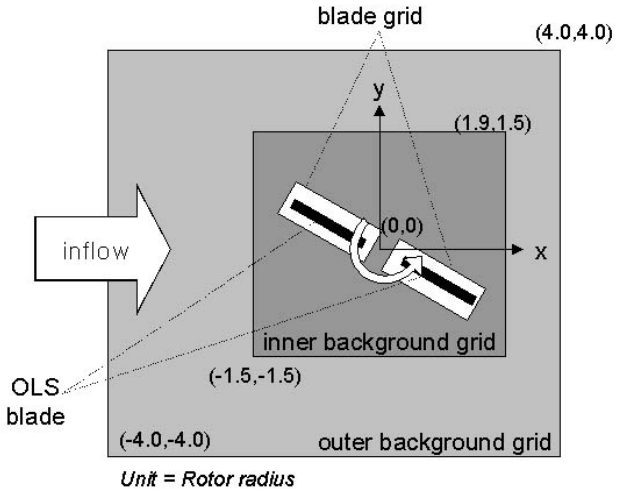


Figure 1. Geometric dimensions of computational domain of inner and outer background grids

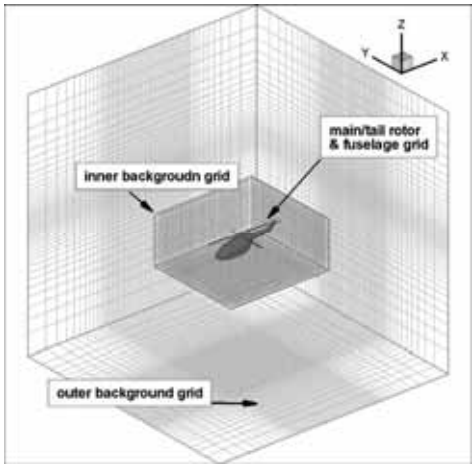


Figure 2. Perspective view of overlapped grid system

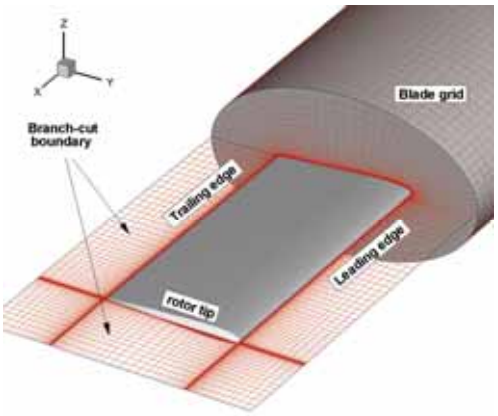
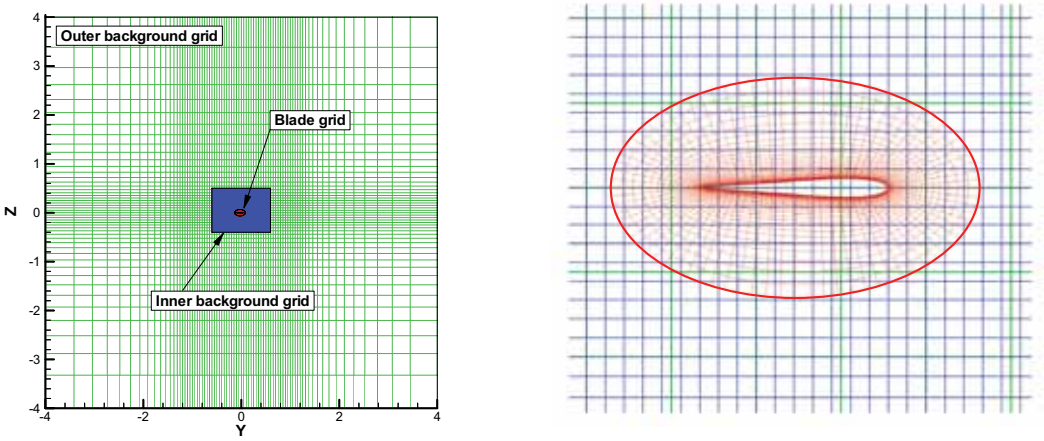


Figure 3. Cross section of blade grid and boundary condition

Table 1 shows the specification of overlapped grid system. Most of the grid is concentrated in inner-background grid, which captures the trajectory of tip vortex during several rotations around rotor disk. Two different background grids, coarse and fine grids, are used to check the effect of grid number during interpolation process. The number of grid points in span-wise direction is considerably increased to match the grid density of the blade grid with that of the inner background grid. The grid spacing of the two inner background grids corresponds to 0.1c and 0.05c, respectively, where c is the blade chord length.



(a) whole grid system (b) blade and inner background grid

Figure 4. Cross section of blade grid and inner background grid

Table 1: Specification of grid system

	Coarse grid	Fine grid
inner background grid	(X×Y×Z) 290×230×50 = 3,335,000	(X×Y×Z) 450×400×80 = 14,400,000
outer background grid	(X×Y×Z) 83×79×49 = 321,293	
blade grid	(chord×normal×span) × blade (83×25×131) × 1 = 1,087,300	
total	~4,740,000 points	~15,800,000 points
inner background spacing	0.1c (=0.012R)	0.05c (=0.006R)

2.2 Numerical Schemes

A three-dimensional numerical flow solver for the compressible Euler equation is used to analyze the detailed behavior of tip vortex. For the calculation of blade grid, inviscid flux vectors are separated using Roe's flux difference splitting (FDS) [10] algorithm, with second-order accuracy using a TVD scheme. For the time integration, Euler backward scheme is used in the conventional delta form, then, Newton iterative method is applied to achieve higher accuracy. The unsteady calculation is impulsively started from 0˚azimuth angle. A diagonalized ADI method with an upwind flux-split technique is used in the linearized implicit part for the discretionary governing equations. A detailed derivation of the governing equation and numerical schemes is described in a previous work by Aoyama et al.[11] The typical dividing number along the azimuthal direction is about 4800 per revolution, which corresponds to the azimuth angle of about 0.075˚. For the calculations over background grid, the flux difference across cell interface is divided using a compact TVD scheme[12] to get third order accuracy. MUSCL cell interface value is modified to achieve 4th-order high accuracy in the background Cartesian grid. Simple High-resolution Upwind Scheme (SHUS)[13] is employed to obtain numerical flux. SHUS is one of the Advection Upstream Splitting Method (AUSM) type approximate Riemann solvers and has small numerical diffusion. The four stage Runge-Kutta method is used for the present calculation. The free stream condition is applied for the outer boundary of the outer background grid.

2.3 Calculation Procedure

The dynamic blade motions such as flapping, feathering, and lagging are defined by the input data. These input data can include azimuth-wise data or 1st harmonic function data obtained by measurements or other codes, such as CAMRADII (Comprehensive Analytical Code for Rotorcraft Aerodynamics and Dynamics) [16]. In the present calculation, the collective pitch, cyclic pitch, flapping, and lagging angles measured by the wind tunnel experiment by Advanced Technology Institute of

Commuter-helicopter, Inc. (ATIC) are used as the input data. The calculation procedure is shown in the diagram of Fig. 5. The search and interpolation to exchange flow data between the grids are executed in each time step because the blade grid rotates with the rotor blade in the background grids. The computation time spent for search and interpolation is one of the disadvantages of the moving overlapped grid approach. Figure 6 shows the procedure of search and interpolation algorithm. In the first step, the grid indexed (i,j,k) of the background grid point that might be inside of the each blade cell are listed. In the second step, the listed indexes are checked whether they are located inside or outside of the grid cell. The position of the point is expressed by three scalar parameters, s, t, and u for the use of tri-linear interpolation. In this step, values of s, t, and u for each index are calculated. When all s, t, and u are between zero and one, the point is judged to be located inside of the cell. Then, the grid points outside of the cell are removed from the list and the flow data are interpolated to be saved in temporal array. The each processing element (PE) of the supercomputer performs these procedures in parallel. Finally, the interpolated values are exchanged between the processing elements. [17] In this step, values of s, t, and u for each index are calculated. Figure 7 shows diagrams of 2D bi-linear interpolation and 3D tri-linear interpolation. Compared to the simple diagram of 2D interpolation, 3D interpolation is difficult to show the general expression for base grid cell volume and Cartesian background grid together.

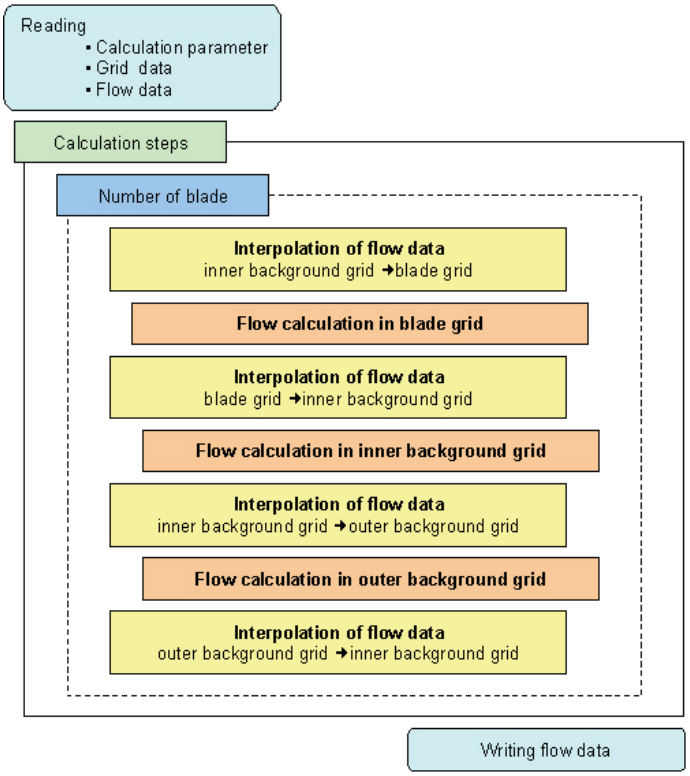


Figure 5. Diagram of procedure for flow calculation

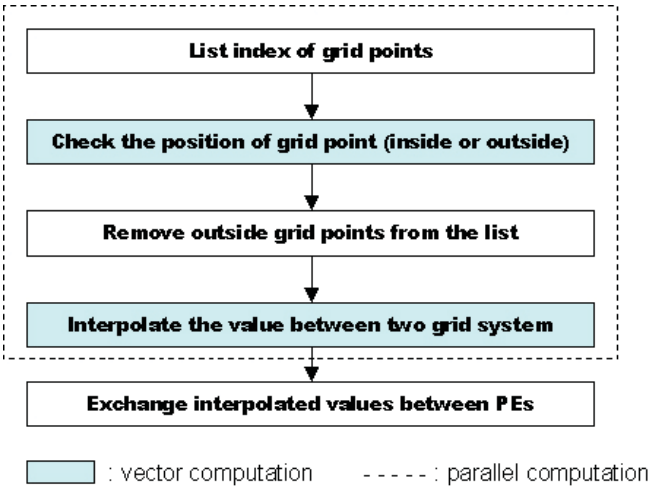
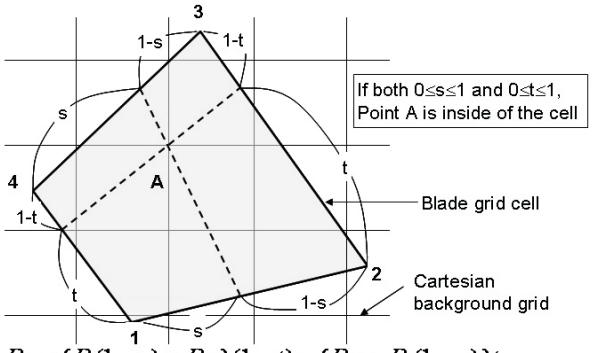
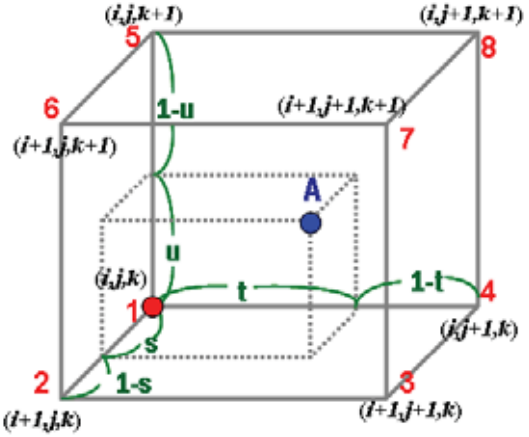


Figure 6. Procedure of new search and interpolation algorithm



$$P_A = \{P_1(1-s)+P_2 s\}(1-t)+\{P_3 s+P_4(1-s)\}t$$
$$= \{P_1 - P_2 + P_3 - P_4\}st + (-P_1+P_2)s + (-P_1+P_4)t + P_1$$

(a) 2D bi-linear interpolation



$$P_A = (1-s)(1-t)(1-u)P_1 + s(1-t)(1-u)P_2 + (1-s)t(1-u)P_4 + st(1-u)P_3$$
$$+ (1-s)(1-t)uP_5 + s(1-t)uP_6 + (1-s)tuP_8 + stuP_7$$

$$s = \frac{x_p - x_1}{x_2 - x_1}, \quad t = \frac{y_p - y_1}{y_4 - y_1}, \quad u = \frac{z_p - z_1}{z_5 - z_1}$$

(b) 3D tri-linear interpolation

Figure 7. Diagram of interpolation for overlapped grid system

2.4 Computational Platform

Calculations are performed using Central Numerical Simulation System (CeNSS), the main engine of the third-generation numerical simulator of JAXA. It is composed of high performance UNIX servers, FUJITSU PRIMEPOWER, which are connected by a crossbar network. CeNSS has 9TFLOPS peak performance, 3TB shared memory, 50TB disk storage, and 600TB tape archive.

3. SPATIAL SEARCHING ALGORITHMS

Among several kinds of interpolation between different grids, only the procedure of the data exchange from the curvilinear blade grid to Cartesian background grid is described in detail here. As shown in Fig. 8, the other procedures of the data exchange from the Cartesian background grid to the Cartesian or curvilinear blade grid are much easier than that from the blade grid to the background grid.

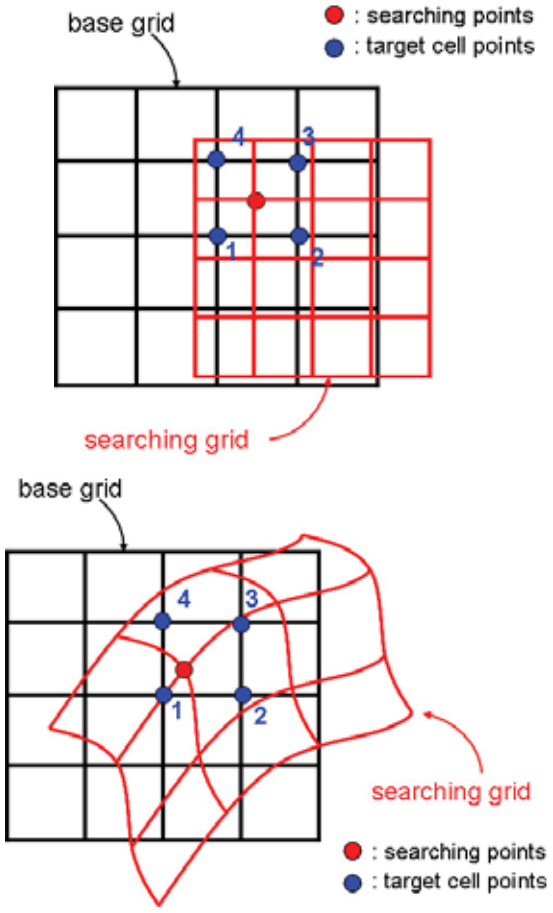


Figure 8. Diagrams of relatively simple interpolation in Cartesian grid

For simplicity, 2-dimensional searching is discussed in this section. As shown in Fig.9, we assume a base grid as black mesh (corresponding to the grid for main-rotor, tail-rotor, or fuselage in the present code) with scalar or vector values defined at its nodal points. We also have a searching grid as red mesh (corresponding to inner background grid) whose spatial extent overlaps that of the black mesh in some arbitrary way. The data transfer operation is to interpolate flow information from nodal values of the black mesh with target cell points 1-2-3-4 (blue circles) onto searching point (red circle) of the red mesh, which is shown as point A, B, and C in Fig. 9. To speed up the searching, the searching points are checked at first whether they are located inside or outside of the box which consists of maximum and minimum values of base grid. This is expressed as 'in-box' in the figure. If a searching point is outside 'in-box' (point A), the point is categorized as 'out-of-box'. If the point is located within 'in-box', the search algorithm begins to work to find the target cell points. If successful (point C), the value at the point 'in-box' is interpolated using target cell points, 1, 2, 3 and 4. If not (point B), it means that the point is 'in-box but out-of-range' and it may be ignored. If a searching point lies on the face (or edge) between two (or more) target cell elements, it can be considered to be inside either cell for interpolation purposes. The cell is assumed to be small enough to be given as 4 lines as a boundary. The judgment of cell including the target point is executed by checking the relative position of two sets of three line vectors, ($v_{1\to2}$, $v_{1\to4}$, $v_{1\to C}$) and ($v_{3\to2}$, $v_{3\to4}$, $v_{3\to C}$), where $v_{1\to2}$ means the position line vector from point 1 to pint 2. The precision of this checking routine in the program is set to the minimum of double precision of real number as 10-15. The judgment for 3-demensional interpolation needs more checking routines, and becomes more complicated because of the 8 boundary faces of cell volume is not planar. Detailed explanations on 3-dimensional searching are described in Appendix A.

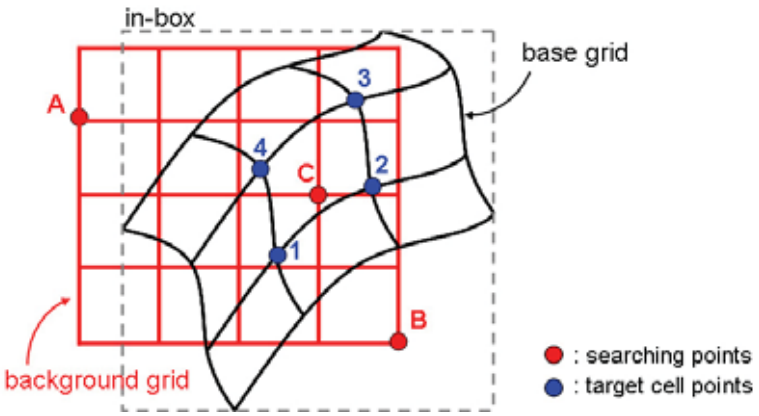


Figure 9. Arbitrary overlap of two meshes

3.1 Linear Searching Algorithm

The simplest searching algorithm for the structured grid (i.e. linear index mesh) is linear searching (LS) algorithm by brute force approach. If we consider (i, j) to be the index of base grid and (I, J) to be the index of background grid as shown in Fig. 10, LS algorithm continues checking if the searching point (I, J) is in the cell defined by nodal points (i, j), (i+1, j), (i+1, j+1) and (i, j+1). If the cell includes the searching point, the value at the point 'in-box' is interpolated using cell points. Then, the next searching point of background grid should be searched. If no cell is found to include the searching point, it is ignored and procedure continues to the next searching points.

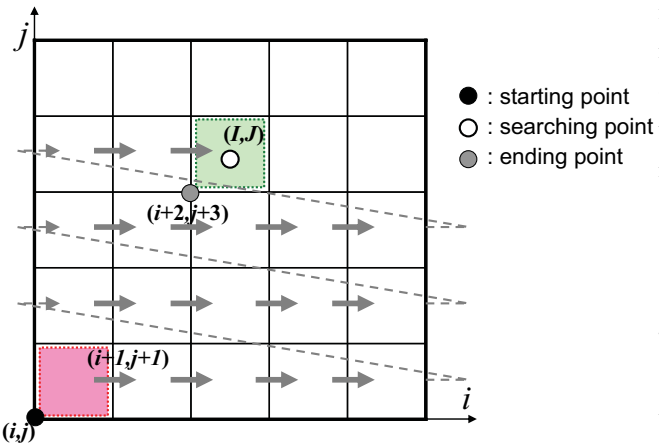


Figure 10. Diagram of linear searching algorithm

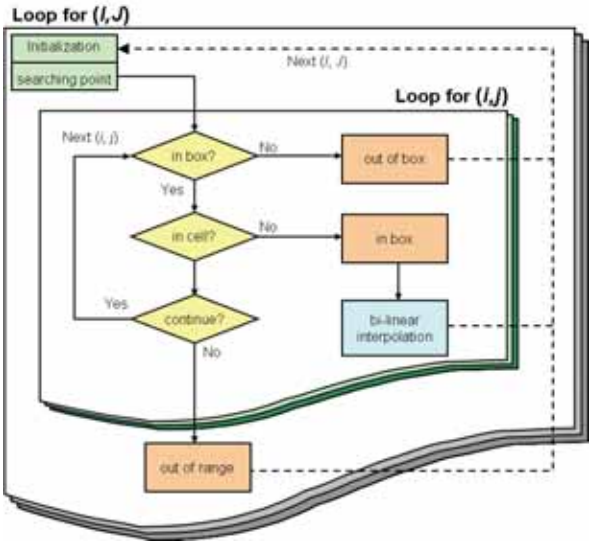


Figure 11. Flowchart of linear searching algorithm

Figure 11 shows a flowchart of the LS algorithm. At the beginning, the grid indexed (I, J) of the background grid point is assigned to be a searching point, and it is checked for the loop of each base grid cell in turns. During the loop of base grid (i, j), the base cell is checked whether it includes the searching point or not. The searching point (I, J) of background grid should be checked in the loop, and each searching point should be nested by the base grid loop. This algorithm is easy for coding and secure for any kind of geometry because all cells of base grid are checked. However, unnecessary index searching may be executed because the increase of index (e.g. from i to i+1) has no relation to the direction approaching the searching point from the current position. Especially, if the searching point is located 'in-box but out-of-range', the whole loop of base grid should be repeated in vain. For a complicated geometry such as helicopter fuselage, the base grid box may include huge number of points 'in-box but out-of-range', which gives rise to undesirable computing cost during interpolation. Moreover, helicopter in maneuver, which changes its position and orientation every moment, requires a massive interpolation. That is why better searching algorithms should be applied for the helicopter simulation in maneuver.

3.2 Alternating Index Searching (AIS) Algorithm

As another searching algorithm for the structured grid, Alternating Index Searching algorithm is proposed. It changes the direction of indexing until the base cell is judged to include the searching point (or until it is proven out of range). This algorithm can accelerate searching by jumping index according to the directivity for searching point by increasing/decreasing (i, j) index, which can make full use of spatial indexing order for the structured grid. The starting point of base grid (i, j) can be chosen arbitrarily such as (1, 1) or middle point of whole base grid. Another choice is to use the ending point of previous searching step because the next searching point is probably located near the previous one. The flowchart of procedure is shown in Fig. 12.

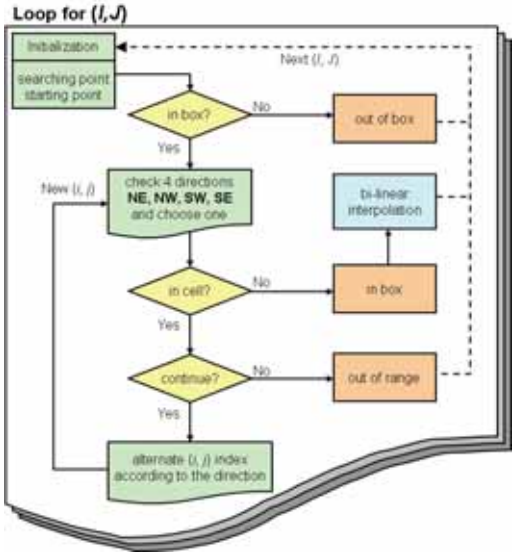
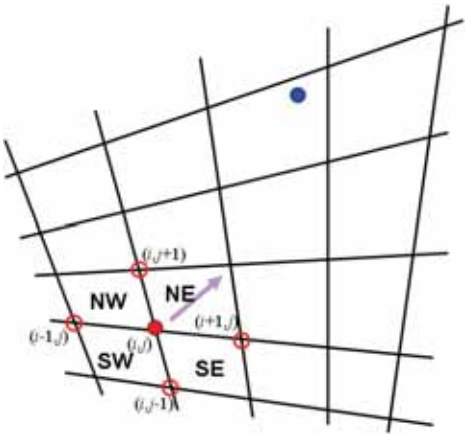
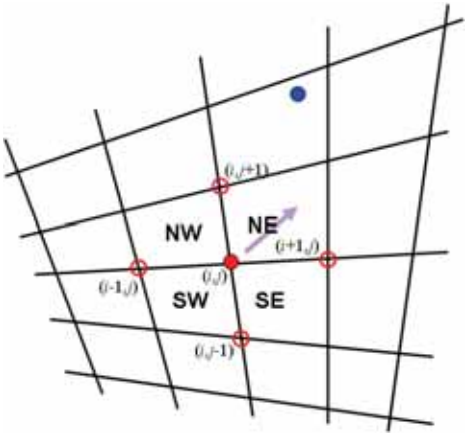


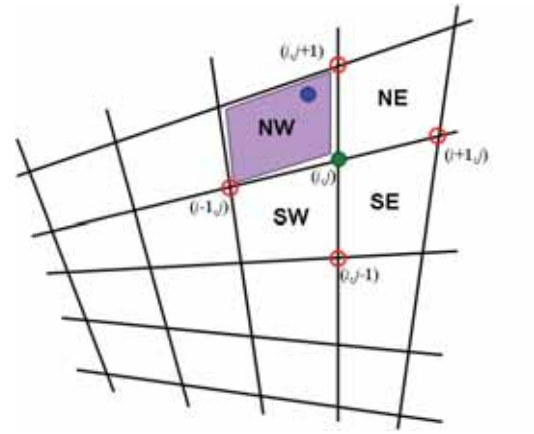
Figure 12. Flowchart of AIS algorithm



step (1)



step (2)



step (3)

● : starting point
● : searching point
● : ending point
○ : neighboring point

Figure 13. Steps of AIS algorithm

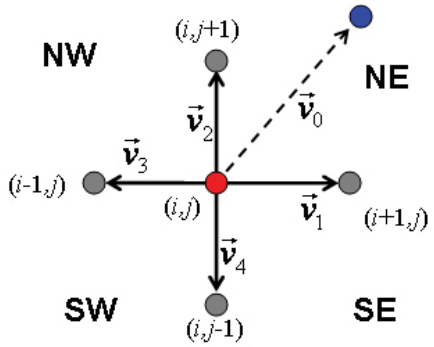


Figure 14. Searching direction in AIS algorithm

Figure 13 shows an example of steps for AIS algorithm starting from (i, j) to approach the searching point (I, J) and finally to find the cell including the searching point. In step (1), four neighboring cells (NE, NW, SW, and SE) of the base point (i, j) are checked to find the approaching direction, then the cell in the direction is checked if this cell includes the searching point. The approaching direction is determined from the relative position between searching vector, v0, and four neighboring spatial vectors, v1, v2, v3, and v4, as shown in Fig. 14. In the present example, the North-East (NE) direction is chosen because v0 lies in between v1 and v2. However, NE cell doesn't include the searching point, so the base point moves to NE by alternating index from (i, j) to (i+1, j+1). In step (2), the same routine is repeated to move the base cell to NE again. In step (3), the approaching direction changes to NW and the NW cell includes the searching point. Then, we can interpolate the value of searching point using 4 nodal points of NW cell: (i, j), (i-1, j), (i-1, j+1) and (i, j+1). Approaching directions in 3-dimensional searching algorithm are more complex, and detailed explanation is described in Appendix B.

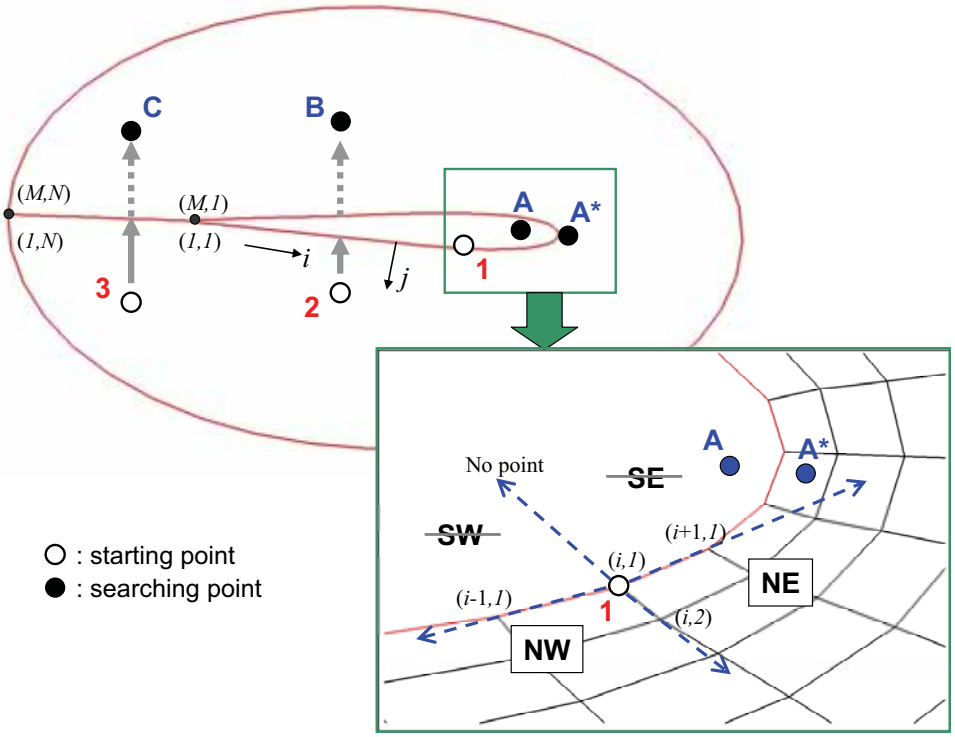


Figure 15. Special treatment for O-type base grid

Figure 15 shows three examples of special treatment for base grid around an airfoil when using the AIS algorithm. These cases come from the characteristics of boundary lines according to the grid types: the present explanation accounts for the O-type grid or partially C-type grid. Case 1 occurs when starting point lies in the boundary line of j=1. If searching point is A, the final answer should be 'in-box but out-of-range' because an approaching direction to SE cannot continue. But for the searching point A*, an extra treatment is added for correct searching even approaching direction SE is not available. The grid points on the boundary lines are checked in turns until the nearest point to the searching point A* is found. Then, from this point the searching algorithm is checked again. Case 2 occurs when approaching direction has to cross the j=1 boundary starting from point 2 to approach point B. Similarly, case 3 occurs when approaching direction has to cross i=1 or i=M lines, which is identical in O-type grid topology. These treatment routines are included in the present calculation.

3.3 Comparison of Algorithms using 2D Geometry in Motion

Searching efficiency of each algorithm is compared using 2-dimensional models with different grid density (coarse grid or fine grid) for both base grid and background grid. LS algorithm and AIS algorithm are used as shown in Table 2. The number of grid for each grid type is listed in Table 3, and four different base grids are shown in Fig. 16. Three different starting points are compared for the case of AIS algorithm such as initial point (1,1), middle point (Ni/2, Nj/2), and the previous ending point as shown in Fig. 17. Computing time during searching and step count (in other words, the number of alternating index operation) for each algorithm is compared. In order 1) to get an average value and 2) to eliminate effects of relative position and orientation between base grid and background grid, the base grid is rotated and translated 20 times as shown in Fig. 18, and the computing time and step count are averaged by this 20 time steps. The total number of interpolated points (N), averaged computing time (T), and averaged step count (S) for each algorithm are listed in Table 4. Comparison of averaged computing time is presented in Fig. 19, and the averaged speed-up of searching time relative to LS algorithm is shown in Fig. 20. For all cases, the results show the excellence of AIS algorithm, especially when starting from the previous end point. For example, the most efficient case by the AIS-P algorithm using fine airfoil grid and fine background grid can achieve about 40 times faster searching time than LS algorithm.

Table 2: Definition of algorithm

Algorithm	Exploration
LS	Linear Searching algorithm
AIS-I	Alternating Index Searching algorithm starting from initial point, (1,1)
AIS-M	Alternating Index Searching algorithm starting from middle point
AIS-P	Alternating Index Searching algorithm starting from previous goal point

Table 3: Specification of the number of grid

Grid type		coarse	fine
Base grid	Square	81×24	161×47
	Airfoil (with hole inside)	81×24	161×47
Background grid (including searching points)		450×80	899×159

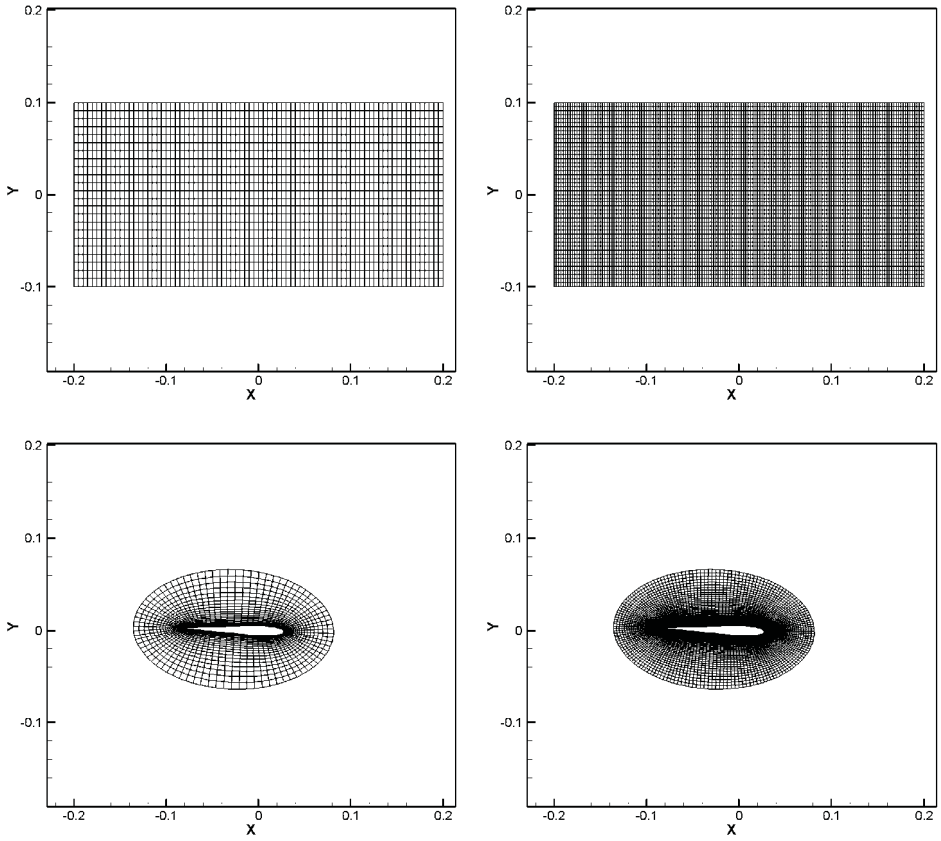


Figure 16. Different base grid according to grid density and geometry

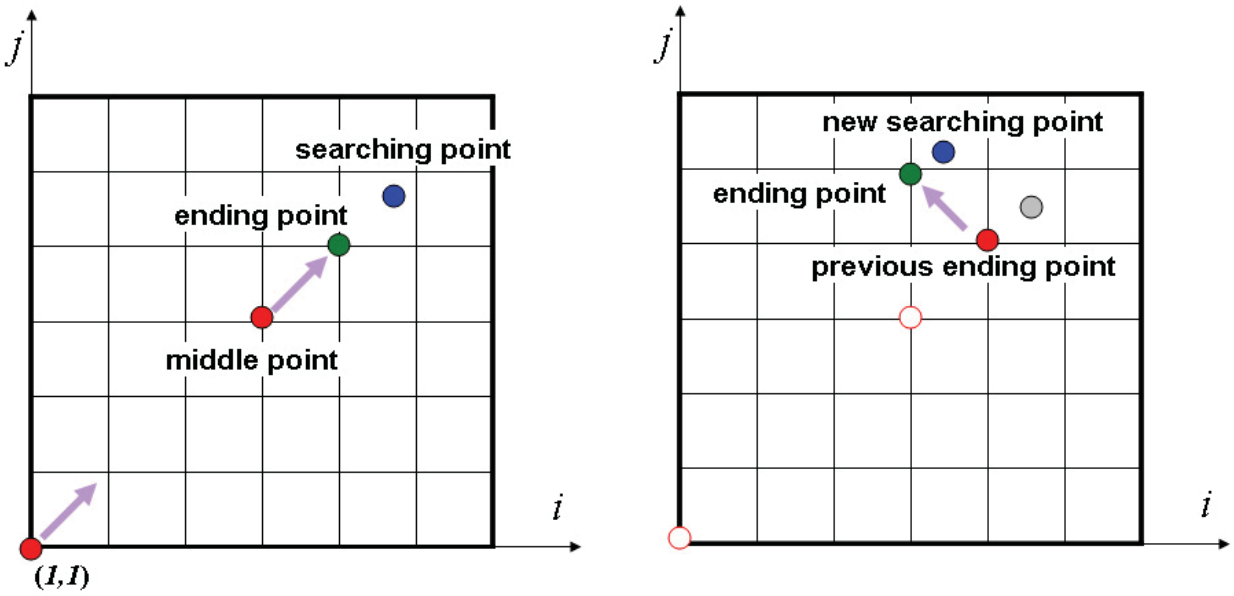


Figure 17. Three different starting points

(a) square (b) airfoil

Figure 18. Diagrams of 2D geometry in motion (rotation and translation)

Table 4: Comparison of total number of interpolated points(N), averaged computing time (T:10-6 sec), and averaged step count(S) for each algorithm and grid type.

Base grid / Back. grid		Square		Airfoil	
		coarse	fine	coarse	fine
coarse		N= 33059	33059	9441	9441
	LS	T(S)= 88386(1231)	316310(4923)	19361(1178)	68587(4702)
	AIS-I	16607(35.2)	32064(69.6)	2879(24.2)	5076(47.6)
	AIS-M	12444(20.6)	23706(40.6)	3034(21.4)	5338(41.8)
	AIS-P	4234(6.3)	7105(11.5)	1346(5.5)	1975(10.0)
fine		132357	132357	37737	37737
	LS	354072(1230.6)	1271496(4922)	78096(4702)	275950(4707)
	AIS-I	67054(35.1)	129160(69.6)	11846(24.1)	20970(47.6)
	AIS-M	50543(20.6)	95657(40.6)	12430(21.5)	21959(41.9)
	AIS-P	15610(5.4)	25114(9.9)	4835(4.1)	6847(7.1)

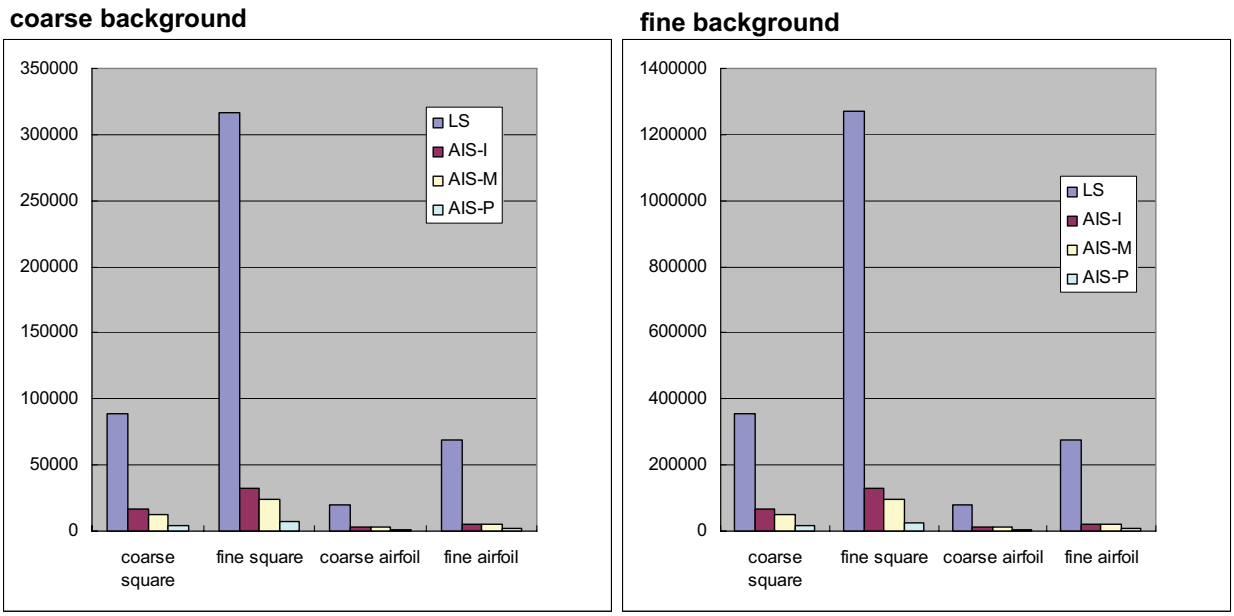


Figure 19. Comparison of averaged computing time for each algorithm and grid type.

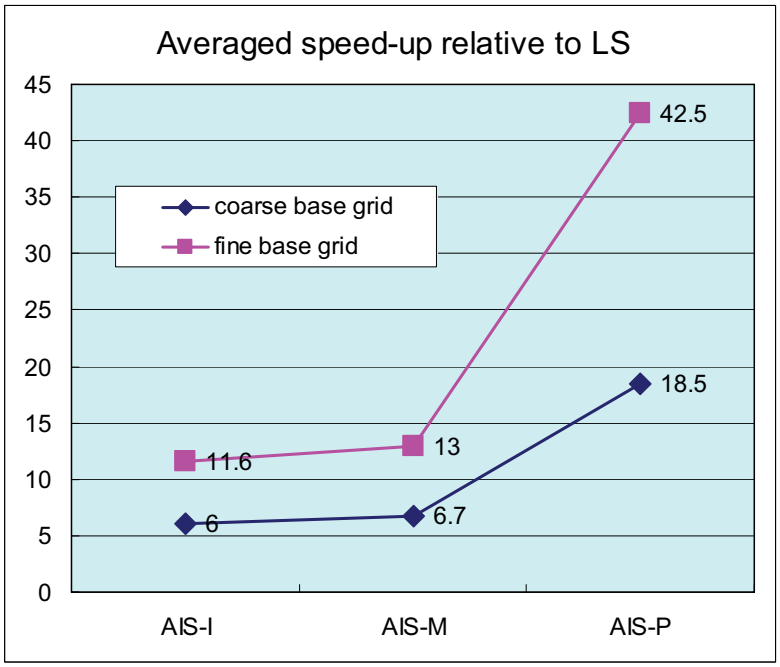


Figure 20. Comparison of averaged speed-up relative to LS algorithm for airfoil grid type

3.4 Comparison of Algorithms in 3D Calculations

Searching efficiency of each algorithm is compared using overlapped grid for full 3D helicopter simulation. Table 5 and Fig. 21 show the ratio of averaged computing time for each procedure during iteration of CFD calculation as shown in Fig.5 when using LS algorithm. Most time consuming routine is the calculation of wing grid which is composed of 9 parts (4 blades for main-rotor, 4 blades for tail-rotor and fuselage). If considering the computing time for each part, calculation of inner-background grid costs most, which is reasonable in terms of huge number of grid points. Computing time for interpolation from the blade grid to the background grid is about 11% when using LS algorithm. As more number of helicopter components should be included such as skid, stabilizer, and so on, the importance of interpolation efficiency may arise. Table 6 shows the ratio of averaged computing time when using AIS algorithm, and the computing time during interpolation from wing to inner background becomes 10 times faster compared to the computing time using LS algorithm. As a

result, the computing time for whole computation can be fastened up to 10 % by upgrading searching algorithm. It is a considerable enhancement because this speed-up is achieved without any modification of flow solving routine but only one interpolation routine among 4 different interpolation procedures.

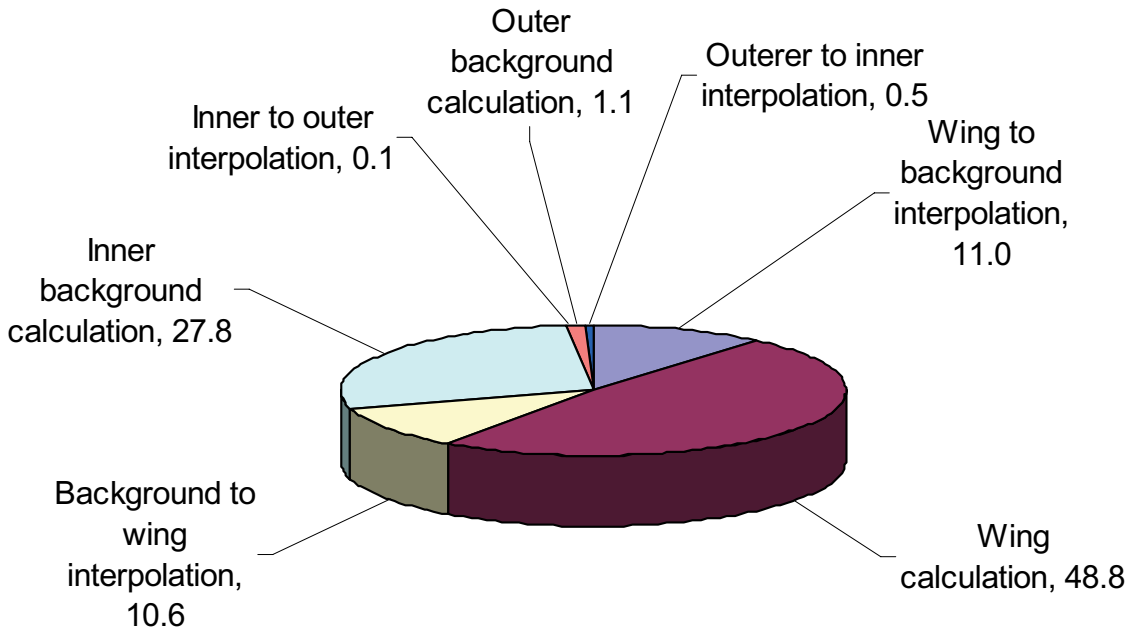


Figure 21. Diagram of averaged percentages of computing time at each procedure per iteration

Table 5: Averaged percentages of computing time for each procedure during iteration when using LS algorithm

operation	20 iteration	average	Ratio (%)
inner background to wing searching time	33.95	1.76	10.6
wing grid calculation time	155.75	7.79	48.8
wing to inner background searching time	35.14	1.70	11.0
inner background grid calculation time	88.76	4.44	27.8
inner to outer background interpolation time	0.17	0.01	0.1
outer background grid calculation time	3.56	0.18	1.1
outer to inner background interpolation time	1.61	0.08	0.5
Total	319.0	15.93	100

Table 6: Averaged percentages of computing time for each procedure during iteration when using AIS algorithm

operation	20 iteration	average	Ratio (%)
inner background to wing searching time	33.94	1.70	11.8
wing grid calculation time	155.75	7.79	54.2
wing to inner background searching time	3.54	0.18	1.2
inner background grid calculation time	88.76	4.44	30.9
inner to outer background interpolation time	0.17	0.01	0.1
outer background grid calculation time	3.56	0.18	1.2
outer to inner background interpolation time	1.61	0.08	0.6
Total	287.3	15.93	100
Speed up w.r.t. linear searching	31.7	14.38	9.93

4. INTERPOLATION FOR PARALLEL COMPUTATION

AIS algorithm shows a possibility of fast and efficient interpolation ways in the previous section when using single processor element (PE), which is a valuable result for single scalar calculation. When using multiple PEs, one of the most important things to be considered is to make a load balance between each PE during parallel computation. On account of the special geometric configuration of helicopter blade, which is characterized by a high aspect ratio, interpolation between rotor grid and background grid becomes more complicated when considering the load balance. Figure 22 shows the comparison of load balance for two different interpolation approaches. To apply AIS algorithm to rotor interpolation problems, the background grid should be divided in parts according to the number of PEs. Figure 22(a) shows a simplified diagram of two critical cases for load unbalance in parallel computation using AIS algorithm. When the azimuth angle of rotor is 0 degree, rotor belongs to the computational domains of only PE2 and PE3, which implies PE1 should wait without any activity until interpolation finish in PE2 and PE3. The case with the azimuth angle of 90 degrees is much worse considering load balance. The rotor blade belongs to PE2 only and all the other PEs are idling state until PE2 finishes interpolation, which makes the computing time with 3PE-parallelization to be the same as that with a single PE. A new interpolation algorithm approaches reversely as shown in Fig. 22(b), even for the interpolation to exchange data from the blade grid to the background grid. Instead of parallelizing the background grid points which should be checked to belong to target blade cell volume, the new searching algorithm parallelizes the blade grid volume which contains the searching background grid point. This parallelization is independent of azimuth angle of blade, which guarantees the almost same load balance among PEs during time marching i.e. temporal iteration according to the azimuth angle. We call this new algorithm as Reverse Linear Searching (RLS) algorithm, and Fig. 23 shows a flowchart of this algorithm. By using the RLS algorithm, we can make full use of the two merits as follows: (1) the fastest searching is possible by one-dimensional search in each direction for background grid, and (2) the good load balance among PEs is maintained for blade grid parallelization.

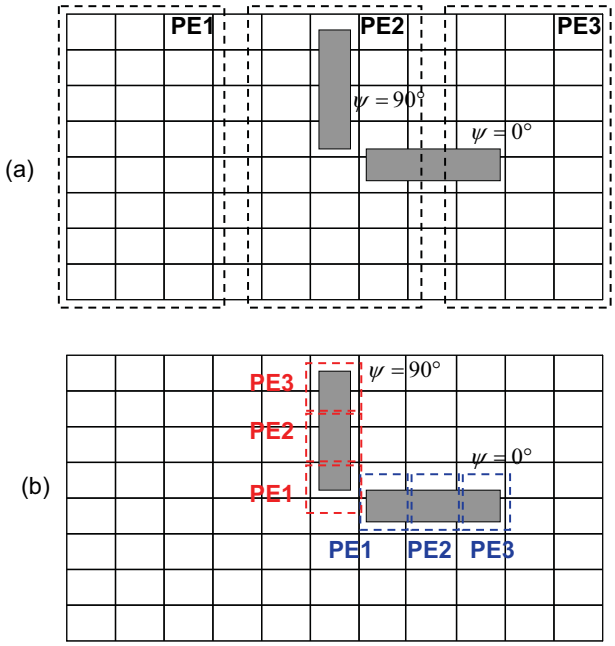


Figure 22. Load balance for parallelization using (a) AIS algorithm and (b) RLS algorithm

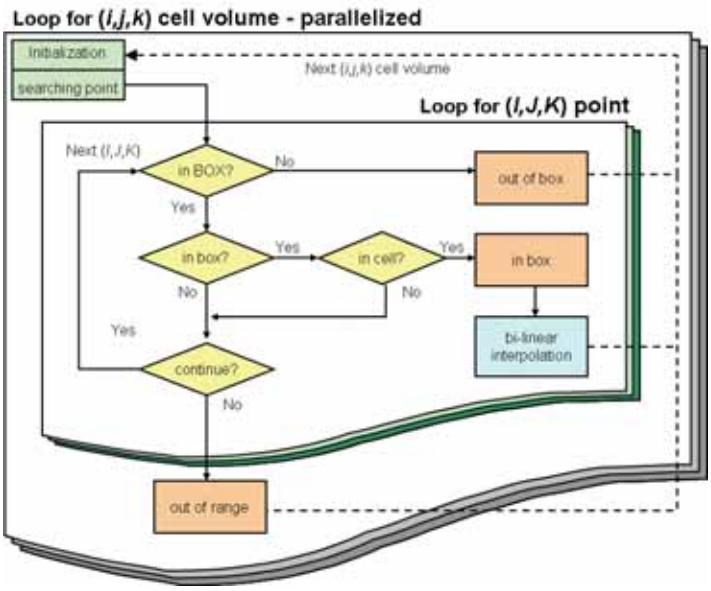


Figure 23. Flowchart of RLS algorithm

The computing times during interpolation by AIS and RIS algorithm are compared using overlapped grid CFD code with 10PEs, and the results are shown in Figs. 24 and 25. The blade grid which is located in two different azimuth angles (0° and 45°) is interpolated in two different background grids (coarse and fine grids). In general, the interpolation at 45° azimuth angle needs more computing time because of the complicated geometric relation between blade grid and Cartesian background grid than that at 0 ° azimuth angle. And the computing time for fine

background grid needs more interpolation time than that for coarse background grid, which is easily expected. Also the load balance among each PE is quite different according to the interpolation algorithms and blade geometry, as explained in the previous section. The result of AIS algorithm in parallel computation, as shown in Fig. 24, shows that the load balance is not well-distributed, and the computing time for some PEs which are actually used for the interpolation are remarkably high than that for the others which are in idling state. Especially for the blade at 0° azimuth angle, most interpolation time is concentrated to two or three PEs while the others are in idling state. The interpolation for the blade at 45° azimuth angle shows better load distribution, but still more than half PEs are in idling state. On the contrary, the result of RIS algorithm shows much better load balance among PEs for the blade at both 0° and 45° azimuth angles, as shown in Fig. 25. From the result, RIS algorithm is necessary to make full use of the parallel computation in the sense of load balance. On account of the well-distributed load balance when using RIS algorithm, the overall interpolation routine becomes faster than that of AIS algorithms which is apparently expressed as the bar height in the figures.

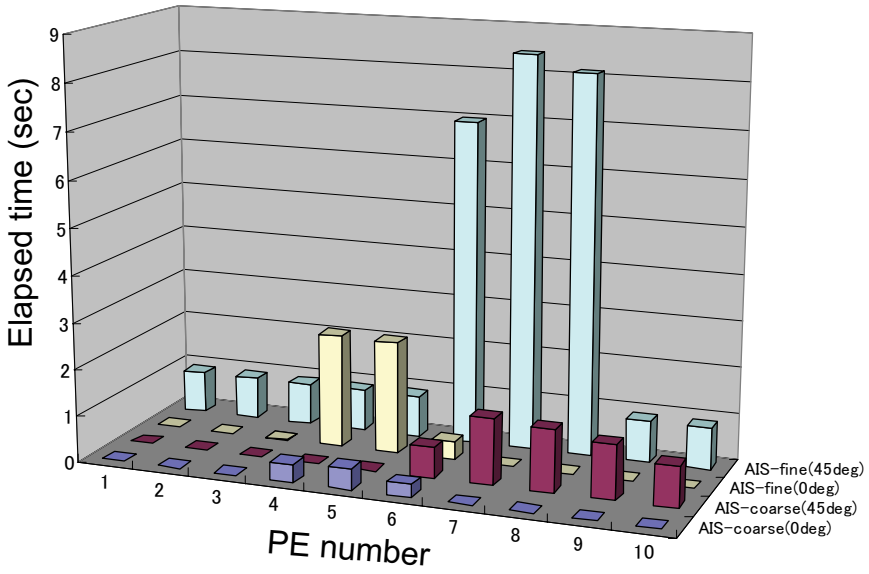


Figure 24. Comparison of interpolation time for each PE when using AIS algorithm

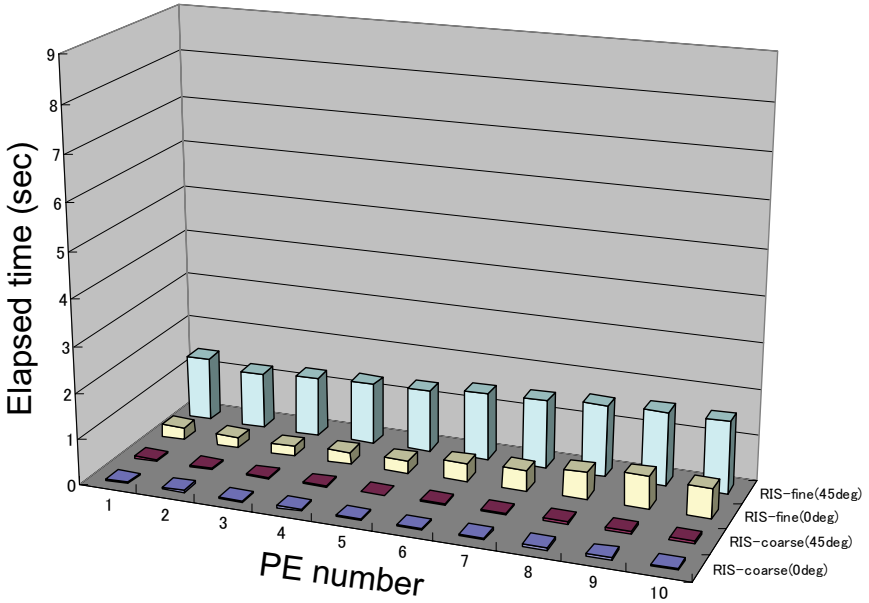


Figure 25. Comparison of interpolation time for each PE when using RIS algorithm

5. SUMMARY

New searching algorithms are implemented for the interpolation between two different grids, Cartesian grid and curvilinear grid, of which the overlapped grid system consists for the massive computation of the full helicopter configuration. Alternating Index Searching (AIS) algorithm is proposed and tested to achieve the speed-up of computing time compared to the conventional Linear Searching (LS) algorithm. Reverse Linear Searching (RLS) algorithm can execute the fastest interpolation by one-dimensional search for Cartesian background grid, and the remarkably enhanced load balance among PEs can be maintained by parallelizing the blade grid itself. Considering the interpolation time for overlapped grid method, RLS algorithm shows a promising possibility for efficient computation in 3D helicopter simulation with parallel computer.

ACKNOWLEDGEMENT

Authors would like to thank Dr. Paulus Lahur in Research Center of Computational Mechanics, Inc. (RCCM) for his useful comments and discussion on interpolation and searching algorithms.

REFERENCES

1) Plimpton, S. J., Hendrickson, B., and Stewart, J. R., "A Parallel Rendezvous Algorithm for Interpolation Between Multiple Grids", J. Parallel Distrib. Comput. 64(2): 266-276 (2004)
2) Yang, C., Aoyama, T., and Saito, S., "Numerical Analysis of Blade-Vortex Interaction Noise in Maneuvering Flight Using Moving Overlapped Grid Method", AHS 62nd Annual Forum, Pheonix, Arizona, May (2006)
3) Yang, C., Aoyama, T., and Saito, S., "Numerical Study on BVI Noise Reduction Using Active Flap Control", 31st ERF, No. 24, Florence, Italy, September (2005)
4) Yang, C., Lahur, P., and Aoyama, T., "Comparison of Interpolation Methods for Numerical Analysis on Helicopter Noise", Aerospace Numerical Simulation Symposium (ANSS), Tokyo, Japan, June (2006)
5) http://elsa.onera.fr/index.html
6) Hoinville, E., and Renaud, T., "CFD Simulation of Helicopter Rotor in the Vortex Ring State Regime", Proceedings of the American Helicopter Society 63rd annual forum, Virginia Beach, VA, May 1-3 (2007)
7) http://halfdome.arc.nasa.gov/cfd/CFD4/New_Page/Overflow-D2.htm
8) Bonet, J. and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," Int. J. Numer. Meth. Eng., Vol 31, 1-17, (1991)
9) Makinen, S. et al, "OVERFLOW-RCAS CFD-CSD Coupling." Proceedings of the American Helicopter Society 62th Annual Forum, Phoenix, Arizona, May 9–11 (2006)
10) Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Different Schemes", Journal of Computational Physics, Vol.43, pp. 357-372 (1981)
11) Aoyama, T., Kawachi, K., Saito, S. and Kamio, J., "Unsteady Analysis of Transonic Helicopter Rotor Noise", 19th European Rotorcraft Forum (1993)
12) Yamamoto, S. and Daiguji, H., "Higher-order-accurate Upwind Schemes for Solving the Compressible Euler and Navier-Stokes Equations", Journal of Computers & Fluids, 22, pp.259-270 (1993)
13) Shima, E. and Jounouchi, T., "Role of CFD in Aeronautical Engineering (No.14) -AUSM Type Upwind Schemes-", Proceedings of the 14th NAL Symposium on Aircraft Computational Aero-dynamics (NAL SP-34), pp.7-12, Tokyo, Japan, Jul (1996)
14) Farassat, F., "Theory of Noise Generation from Moving Bodies with an Application to Helicopter Rotors", NASA TR R 451 (1975)
15) Brentner, K. S., "Helicopter Rotor Noise Prediction: Background, Current Status, and Future Direction", Seminar in University Tennessee Space Institute, Chattanooga, Tennessee (1997)
16) Johnson, W., "Rotorcraft Aerodynamics Applications of Comprehensive Analysis", Heli-Japan 98, Paper No.S5, Gifu, Japan, April (1998)
17) Ochi, A., Aoyama, T., Saito, S., Shima, E. and Yamakawa, E., "Aerodynamic and Aeroacoustic Analysis of BVI by Moving Overlapped Grid Method", 24th European Rotorcraft Forum, Paper No. AC04 (1998)

APPENDIX A

Compared to the relatively simple judgment to find the planar cell including the target point in 2D case as explained in Sec.3, judgment for the volume call in 3D case becomes more complicated. Figure A1 shows a diagram of judgment for the inclusion of the searching point into the cell volume. A cell volume starting from the red point (i, j, k) and its neighboring points (grey) may or may not include the target point (blue), as shown in the figure. The judgment procedure is executed by checking relative direction vector between the normal vector of each cell face and position vector of the target point. The left case of Fig.A1 shows that the target point lines on the same side with each surface normal vector of 8 cell surface, which implies the target cell is included in the cell. On the other hand, the right case of Fig.B1 shows that the target point lines on the same side with each surface normal vector of 7 cell surface, but it lies on the opposite side with the normal vector of top surface, which implies the target cell is not included in the cell.

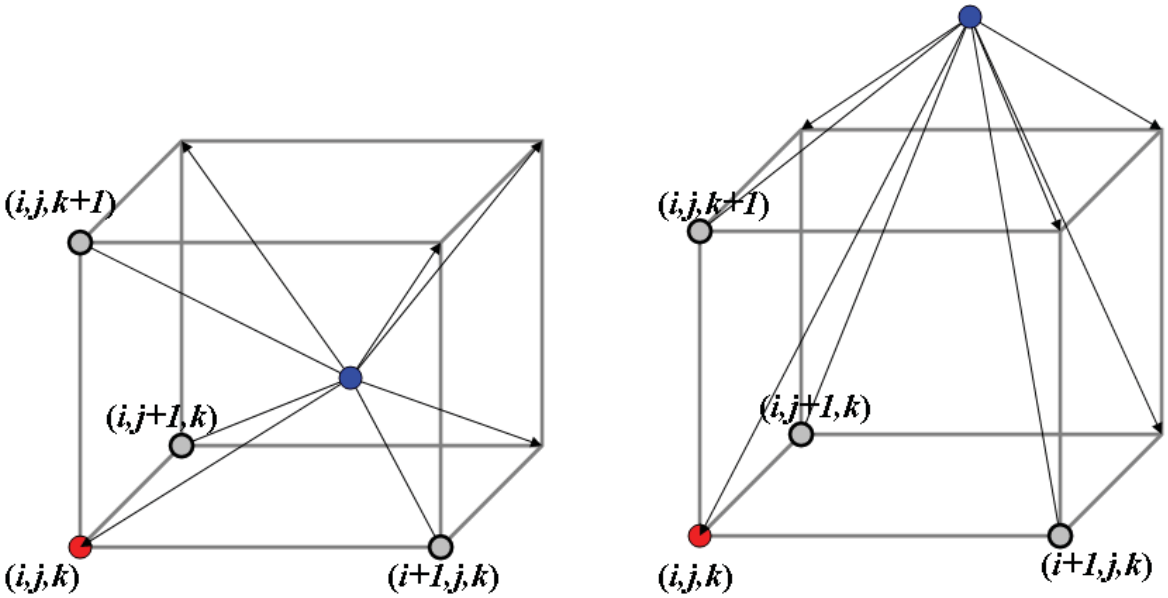
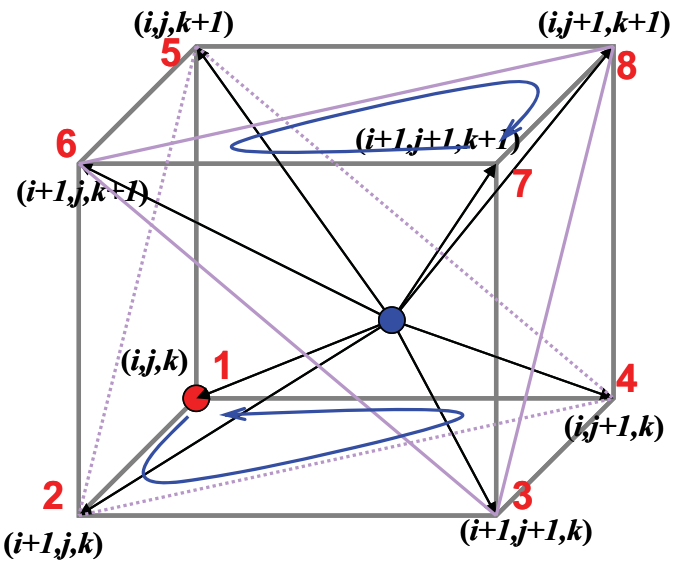


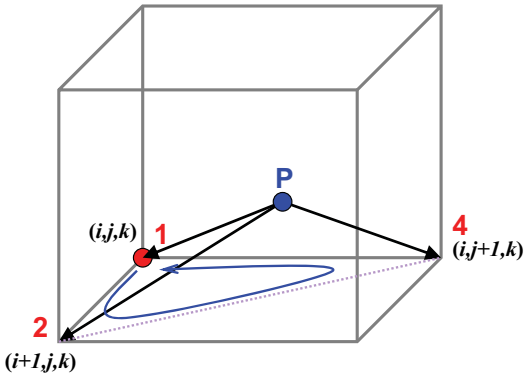
Figure A1. Diagram of judgment for inclusion of the searching point (blue) into the cell

Figure A2 shows the general procedure to check the inclusion of the searching point (blue) into the cell. Each cell surface is divided into two triangular planes, then total 12 triangular planes can be listed as shown in Fig.A2(a). At first, a pyramid can be defined by the three cell points and the target point P as shown in Fig.A2(b). If the target point is include in the cell volume, scalar triple product of each pyramid, which is calculated in ORIENT3D function of the program, should have the same sign. To save the computing time, the checking is done for the 6 couples of diagonal triangular planes as shown in the Fig.A2(b). If all 6 checking routine returns TRUE values, the target point can be judged as being included to the cell volume. If a searching point lies on the face (or edge) between two (or more) target cell volume, it can be considered to be inside either cell for interpolation purposes. The precision of this checking routine in the program is set to the minimum of double precision of real number as 10-15.



1	1-2-4
2	3-4-2
3	5-8-6
4	7-6-8
5	1-4-5
6	8-5-4
7	2-6-3
8	7-3-6
9	1-5-2
10	6-2-5
11	4-3-8
12	7-8-3

Point indices for the call volume and the numbering of 12 triangular plane



1	ORIENT3D(1,2,4,P) X ORIENT3D(7,6,8,P) > 0
2	ORIENT3D(3,4,2,P) X ORIENT3D(5,8,6,P) > 0
3	ORIENT3D(1,4,5,P) X ORIENT3D(7,3,6,P) > 0
4	ORIENT3D(8,5,4,P) X ORIENT3D(2,6,3,P) > 0
5	ORIENT3D(1,5,2,P) X ORIENT3D(7,8,3,P) > 0
6	ORIENT3D(6,2,5,P) X ORIENT3D(4,3,8,P) > 0

Definition of direction vectors and check the direction

Figure A2. Procedure to check the inclusion of the searching point (blue) into the cell

APPENDIX B

Compared to the 4 approaching direction vectors from the base point (i, j) to the four neighboring cells (NE, NW, SW, and SE) in Sec.3.2 for 2dimensional searching algorithm, 3 dimensional cases need more consideration. Figure A1 shows the cell indices which are used for 3 dimensional searching algorithms. Searching directions are defined from the base point (i, j, k) to eight neighboring cells (ENT, WNT, WST, EST, ENB, WNB, WSB, ESB), where E, W, S, N, T, and B mean east, west, south, north, top, and bottom, respectively.

Also, for the actual programming to determine 3D searching directions, special treatments are necessary to consider the situation when the base point is located on the boundary or on the periodic surface of O-type blade grid. Figures A3 (a)-(h) show these special treatments according to the different approaching direction vectors.

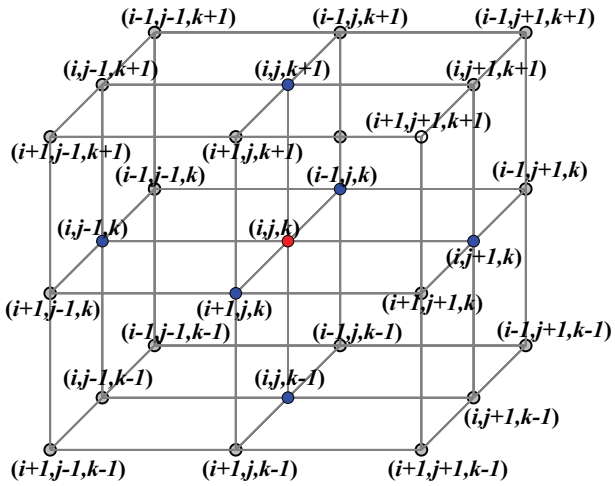


Figure B1. Cell indices used for interpolation

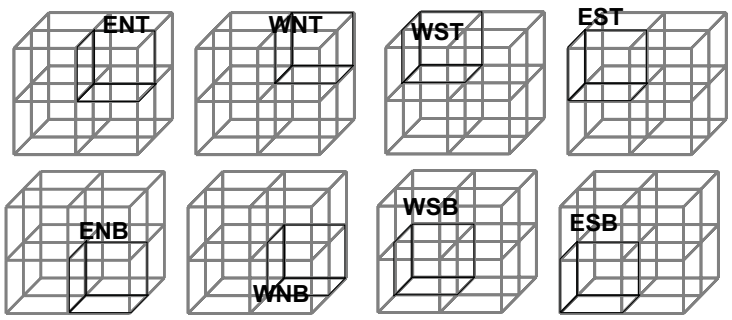
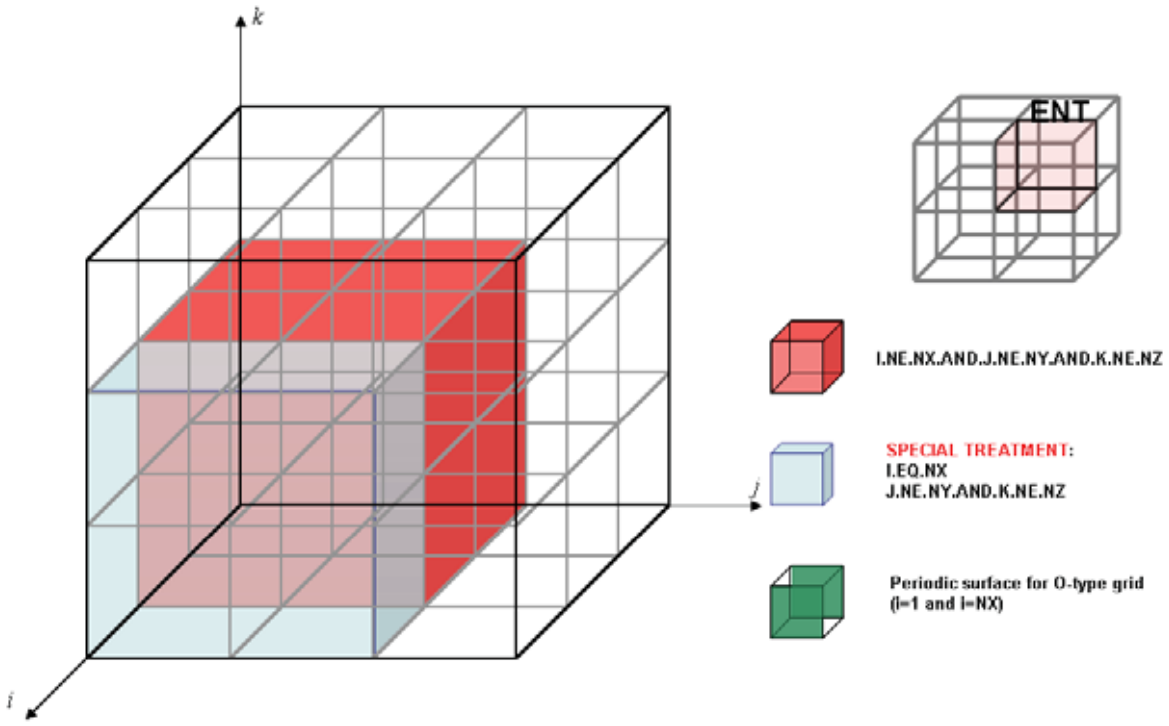
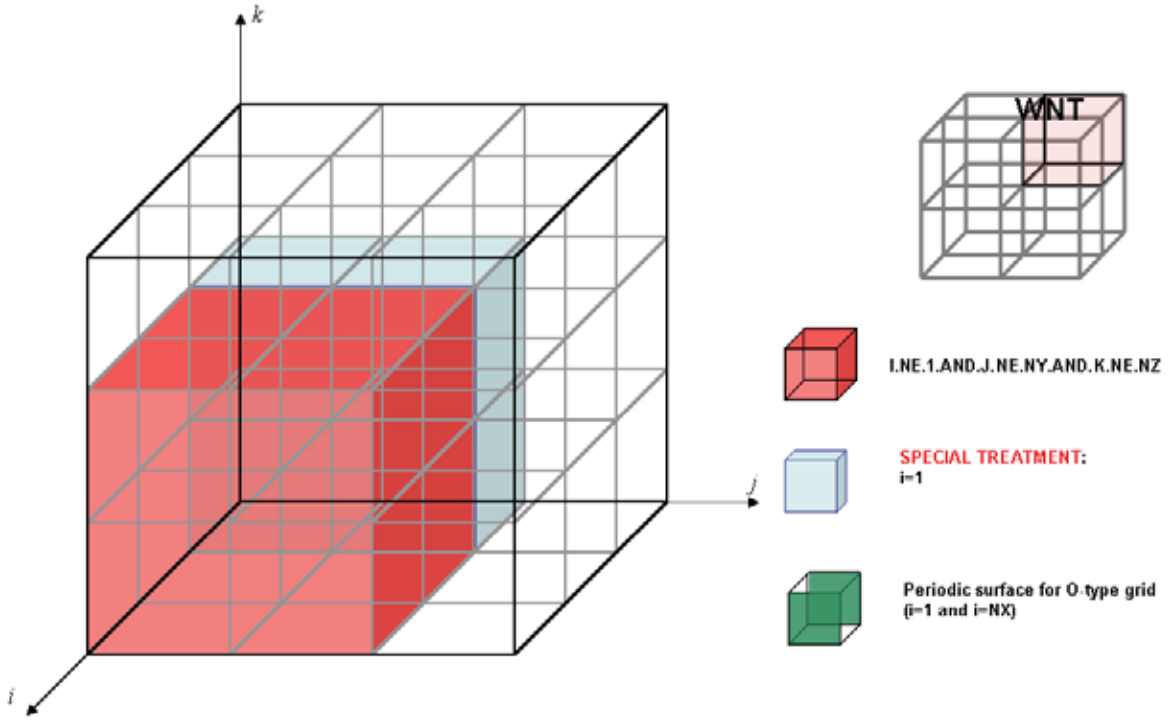


Figure B2. 3D searching direction in AIS algorithm

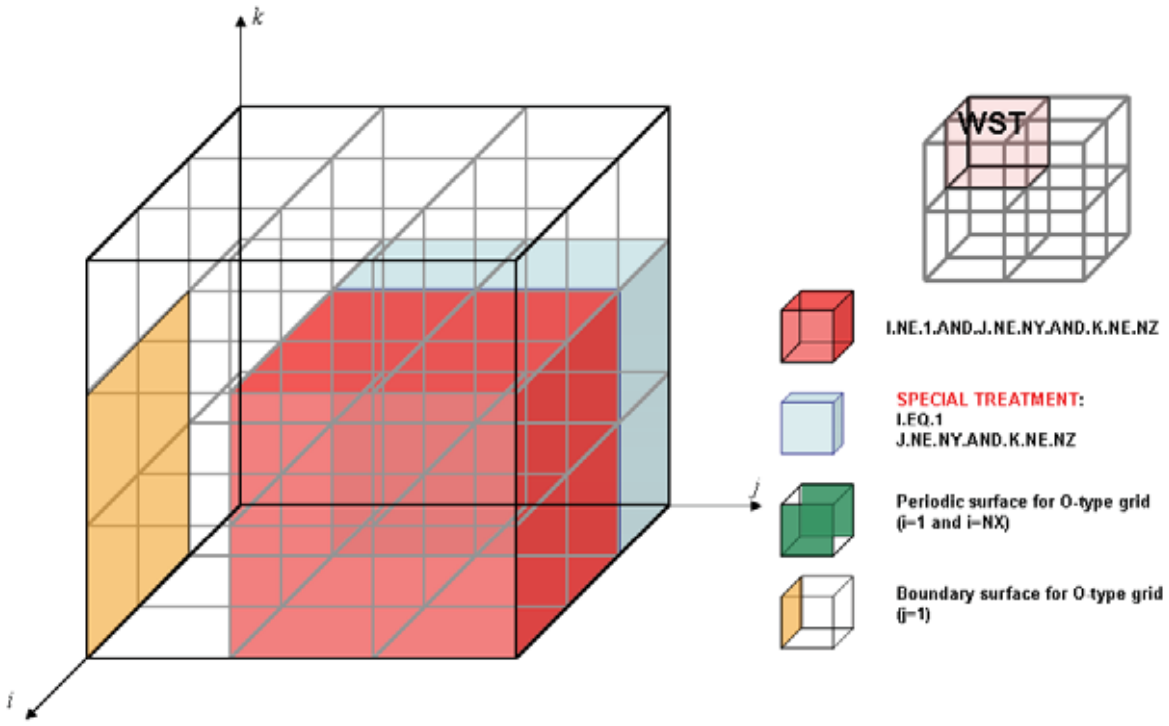


(a) ENT direction (i+1,j+1,k+1)

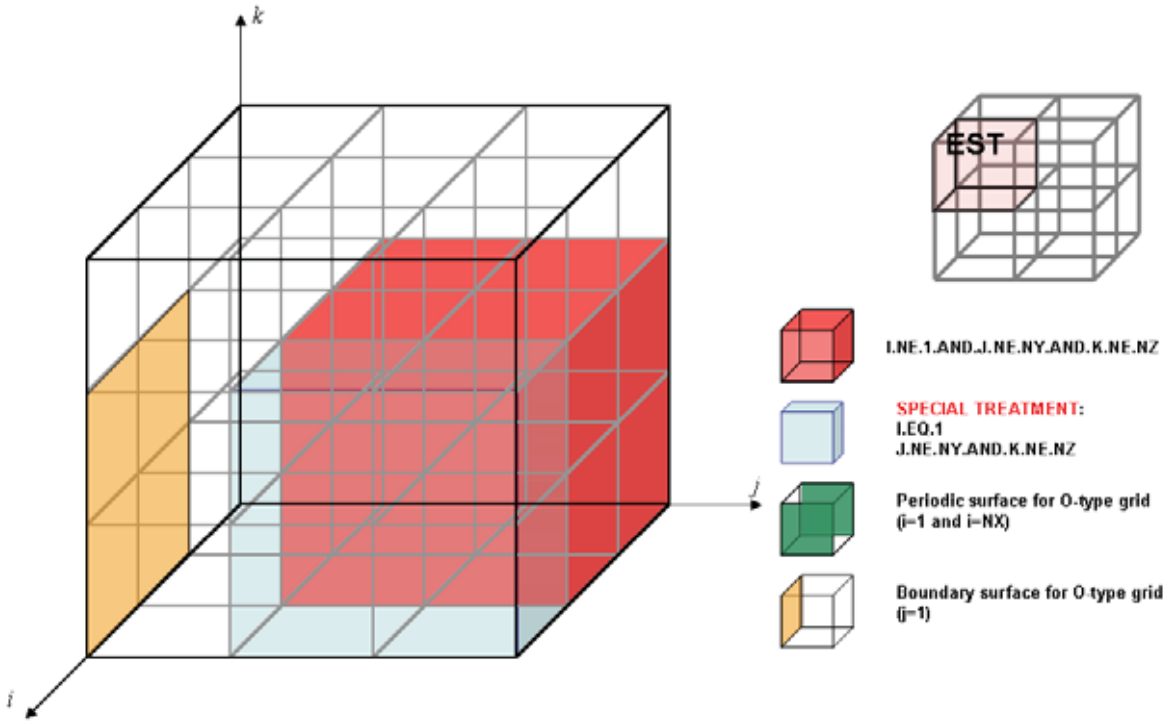


(b) WNT direction (i-1,j+1,k+1)

Figure B3. Special treatments according to approaching direction vectors

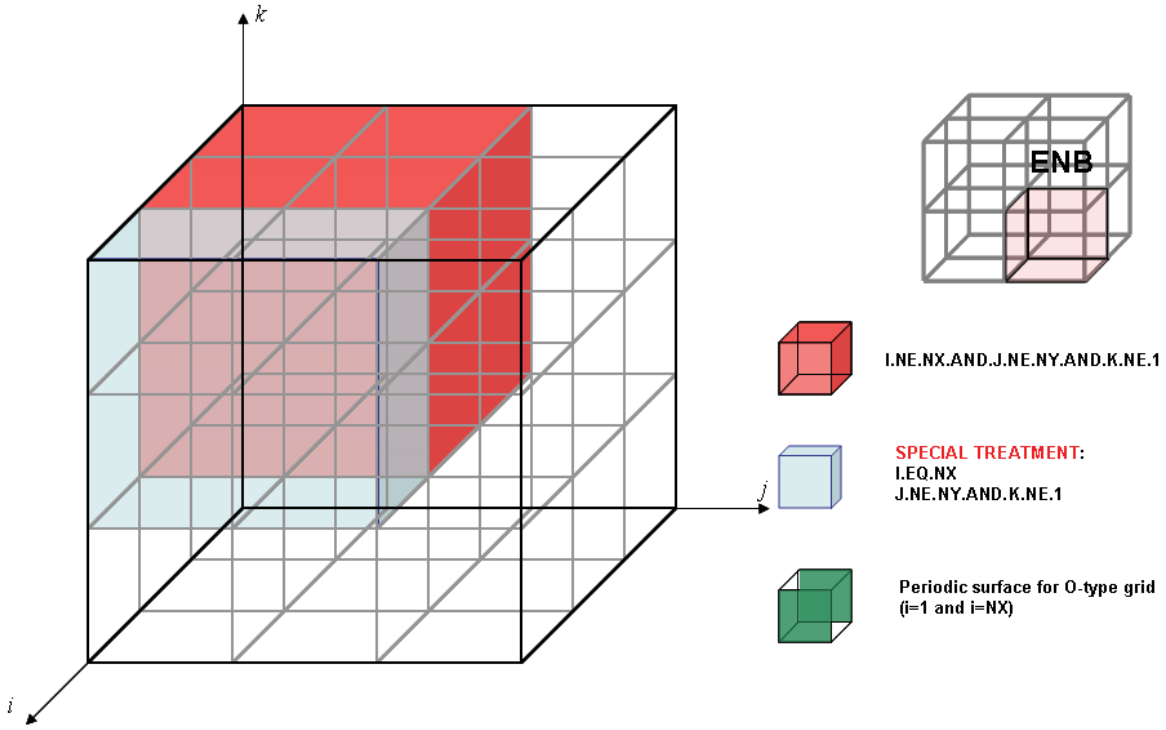


(c) WNT direction (i-1,j-1,k+1)

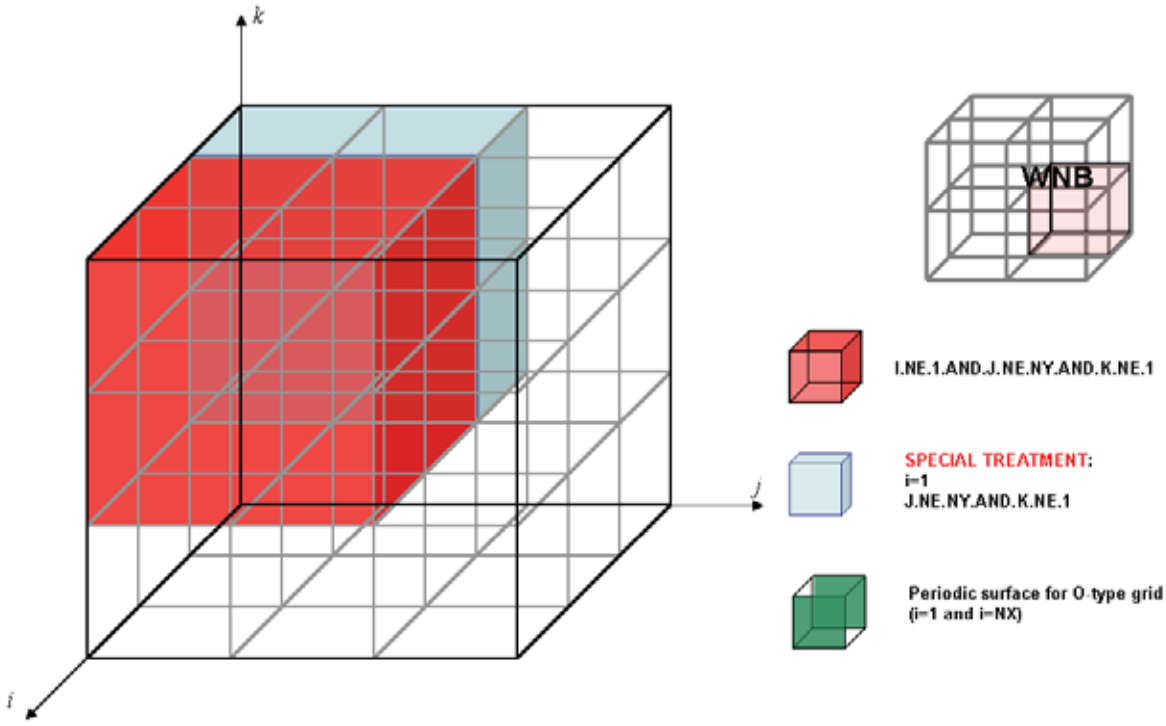


(d) EST direction (i+1,j-1,k+1)

Figure B3. (continued)

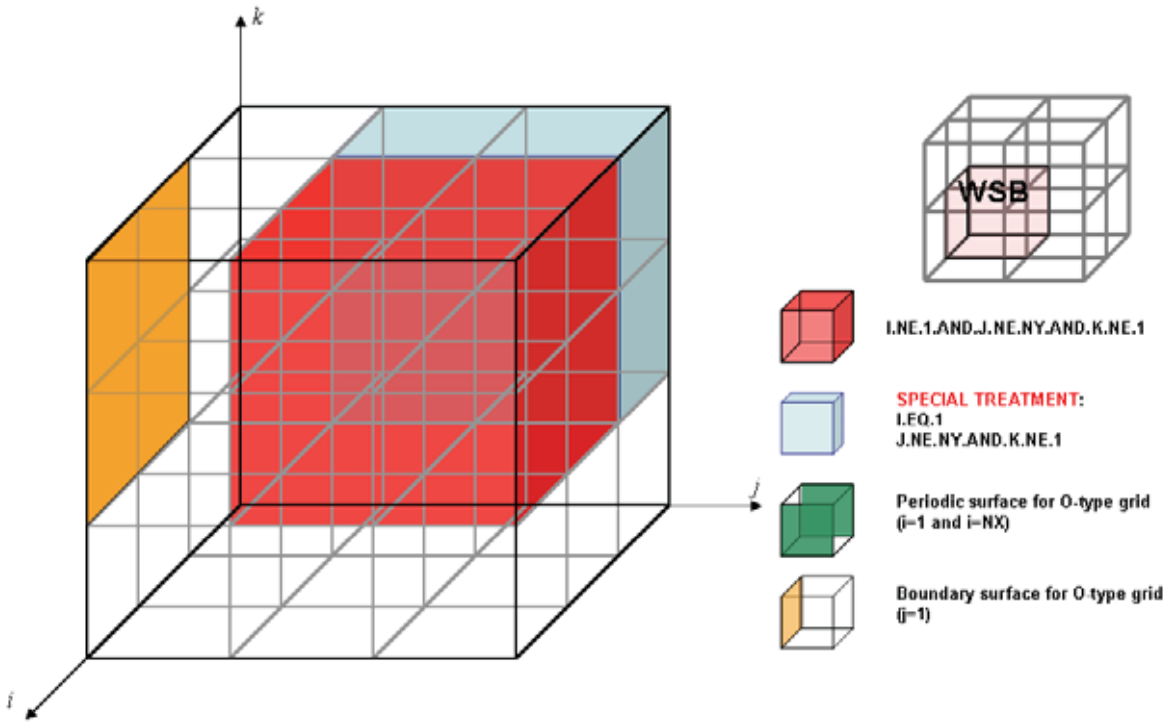


(e) ENB direction (i+1,j+1,k-1)

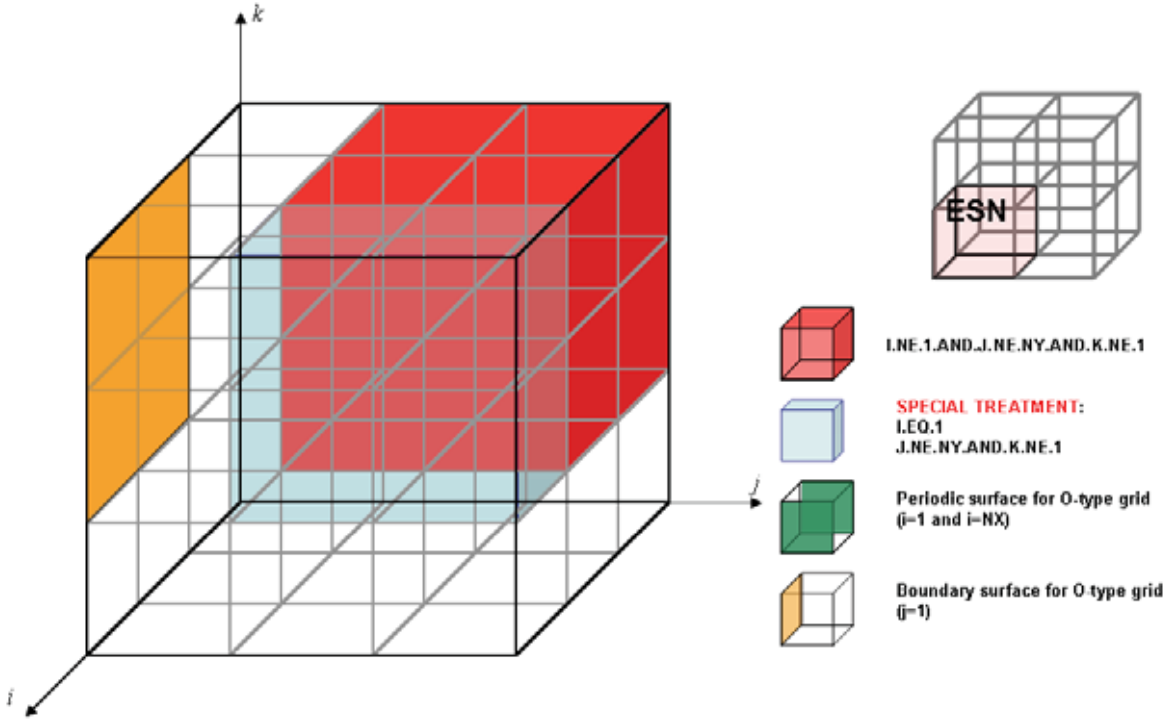


(f) WNB direction (i-1,j+1,k-1)

Figure B3. (continued)



(g) WNB direction (i-1,j-1,k-1)



(h) ESB direction (i+1,j-1,k-1)

Figure B3. (continued)

JAXA Research and Development Report JAXA-RR-07-024E

Date of Issue : February 29, 2008

Edited and Published by : Japan Aerospace Exploration Agency

7-44-1 Jindaiji-higashimach, Chofu-shi,Tokyo 182-8522,Japan

URL:http://www.jaxa.jp/

Printed by : NORTH ISLAND Co.,Ltd