中央大学理工学研究所論文集　第9号　2004年
Journal of the Institute of Science and Engineering. Chuo University

Wind Flow Simulation in City using Computational Fluid Dynamics and Parallel Computing

Kazuya NOJIMA* **and Mutsuto KAWAHARA***

abstract

In this study, simulation of wind flow in the city is investigated. For the reduction of the calculation time and storage requirement, the parallel computation which based on the domain decomposition method is employed. The wind flow is assumed as incompressible viscous flow. The Navier-Stokes equation is used as basic equation. For the spatial discretization, a mixed interpolation using the bubble function element is employed. The fractional step method introducing an intermediate velocity is applied. The computational methodology relies on the following ingredients: space discretization of the Navier-Stokes equations by the finite element approximations using tetrahedral element, time discretization by the second order accurate Crank-Nicolson scheme. The computation is carried out stably with the stabilising method. From the result of calculation time, it can be said that the parallel computing is effective for the reduction of the calculation time. The high performance computing is obtained by using parallel computing. In the case of calculation by the super computer (IBM pSeries 690), the parallel efficiency becomes higher than 100%. Thus the cause of this problem is investigated in this paper.

1 Introduction

The wind flow among buildings is an important problem in civil engineering. The estimation of the influence of the building to wind environment is necessary before the construction of the buildings. Especially, high-rise buildings cause very serious wind problems after the construction. Ordinary the estimations are carried out by model tests. However, the experiment of this problem and the field observation are very expensive and very time-consuming task. Recently the CFD is used for the estimation of the wind environment before the model test. CFD is able to estimate the wind flow with lower cost than measurement or model experiment. However, there are some difficulty to simulate the wind flow in city with CFD. The first problem is the modelling of the wind field. The modelling is very hard work because area in a city is very complicate. The second is large storage requirement. The computational domain are very large because huge number of the nodes are required in flow problems and computational fields of wind analysis are large. The third is that the phenomenon continues long time. The fourth is that the wind direction changes. Thus, it is necessary to simulate the various wind directions. In previous research, the modelling and meshing technique are proposed [1]. It uses the three-dimensional Delaunay triangulation and adaptive remeshing technique. In this research, the parallel computing for the large scale computation is employed. As for calculation technique, the stabilized scheme based on bubble function element is applied. As temporal and spatial discretization, Crank-Nicolson method and mixed interpolation is employed, respectively. In case of using the mixed interpolation, linear element is applied to pressure field and the bubble function element [2] [3] [4] [5] is applied to flow field.

* Dept. of Civil Engineering, Chuo Univ. 1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan

In order to obtain the high performance computation, the parallel computing is indispensable. The parallel computing technique based on the domain decomposition technique is developed in order to reduce the computational time and storage requirement. The parallel computing can reduce the computational time dramatically. Thus the parallel computing is introduced in this research.

2 Incompressible Navier-Stokes Equation

As basic equation of an incompressible viscous flow, the incompressible Navier-Stokes equation is employed. The incompressible Navier-Stokes equation is expressed by

$$\frac{\partial \boldsymbol{u}}{\partial \mathrm{t}} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \nabla \cdot \nu \left((\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T \right) = \boldsymbol{f}, \quad \text{in } \Omega, \tag{1}$$

$$\nabla \cdot \boldsymbol{u} = \boldsymbol{0}, \quad \text{in } \Omega, \tag{2}$$

where Ω is the computational domain enclosed by the boundary Γ. $\boldsymbol{u}$ and p represent the velocity and pressure, respectively. The term $\boldsymbol{f}$ is the source term, and ν is the kinematic viscosity. The function ν is equal to inverse of the Reynolds number. The boundary Γ is divided into subsets Γ_1 and Γ_2, where the boundary condition is prescribed as

$$\boldsymbol{u} = \hat{\boldsymbol{u}}, \quad \text{on } \Gamma_1, \tag{3}$$

$$\left\{ -p\boldsymbol{I} + \nu \left((\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T \right) \right\} \cdot \boldsymbol{n} = \hat{\boldsymbol{t}}, \quad \text{on } \Gamma_2, \tag{4}$$

in which $\boldsymbol{I}$ denotes the identity tensor, $\boldsymbol{n}$ is the outward normal on the boundary Γ, $\hat{u}$ is the given value of velocity and $\hat{t}$ is the given value of traction.

3 Temporal Discretization

For temporal discretization an implicit scheme, which can use long time increment and has stability, is applied.

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + u_j^* u_{i,j}^{n+\frac{1}{2}} + p_{,i}^{n+1} - \nu \left(u_{i,j}^{n+\frac{1}{2}} + u_{j,i}^{n+\frac{1}{2}} \right)_{,j} = f_i. \tag{5}$$

The continuity equation is implicitly expressed as

$$u_{,ii}^{n+1} = 0, \tag{6}$$

where

$$u_i^{n+\frac{1}{2}} = \frac{u_i^{n+1} + u_i^n}{2},$$

$$u_i^* = \frac{1}{2} \left(3u_i^n - u_i^{n-1} \right),$$

in which u_i^* is a linear approximation of advection and is obtained by the Adams-Bash-forth formula of second order. The scheme is a linear scheme which has accuracy of second order in time.

4 Fractional Step Method and Spatial Discretization

The Navier-Stokes equation can be solved by the fractional step method, where flow and pressure fields are separated by deriving the pressure Poisson equation from the momentum and continuity equation. The pressure Poisson equation is derived introducing an intermediate velocity which may not satisfy the continuity equation(2). The Galerkin formulation is used in space.

The mixed interpolation is applied to the spatial discretization. A bubble element is employed as an interpolation for the flow field. A linear element is employed as an interpolation for the pressure field.

If the pressure p^n of the previous time step is regarded as an approximated pressure, the pressure p^{n+1} is replaced with p^n in Eq.(5). The unknown velocity ${u_i}^{n+1}$ is replaced by the intermediate velocity ${\tilde{u_i}}^{n+1}$. The momentum equation(5) can be rewritten as

$$\frac{\tilde{u}_i^{n+1} - u_i^n}{\Delta t} + u_j^* \cdot \nabla \tilde{u}_{i,j}^{n+\frac{1}{2}} + p_{,i}^n - \nu \left(\tilde{u}_{i,j}^{n+\frac{1}{2}} + \tilde{u}_{j,i}^{n+\frac{1}{2}} \right)_{,j} = f_i, \quad \text{in } \Omega. \tag{7}$$

by taking residual between Eq.(7) and Eq.(5), the following equation can be obtained.

$$\begin{aligned} \frac{u_i^{n+1} - \tilde{u}_i^{n+1}}{\Delta t} &+ \frac{1}{2} u_j^* (u_{i,j}^{n+1} - \tilde{u}_{i,j}^{n+1}) + (p_i^{n+1} - p_i^n) \\ &- \frac{1}{2} \nu \left\{ (u_{i,j}^{n+1} - \tilde{u}_{i,j}^{n+1}) + (u_{j,i}^{n+1} - \tilde{u}_{j,i}^{n+1}) \right\} = 0, \quad \text{in } \Omega. \end{aligned} \tag{8}$$

The pressure Poisson equation is obtained taking divergence of Eq.(8) and substituting the resulting equation into Eq.(6). The unknown velocity ${u_i}^{n+1}$ is approximated by the intermediate ${\tilde{u_i}}^{n+1}$, and then the second and forth term of Eq.(8) is neglected. The pressure Poisson equation is obtained as

$$\Delta t (p_{,ii}^{n+1} - p_{,ii}^n) = \tilde{u}_{i,i}^{n+1}, \quad \text{in } \Omega. \tag{9}$$

5 Finite Element Formulation

5.1 Mixed Interpolation

As the spatial discretization of equations (1) where (r, s, t) are the iso-parametric coordinates, the mixed interpolation is applied. The bubble function element is shown in Fig. 1. The mixed interpolations for velocity and pressure can be expressed as follows:

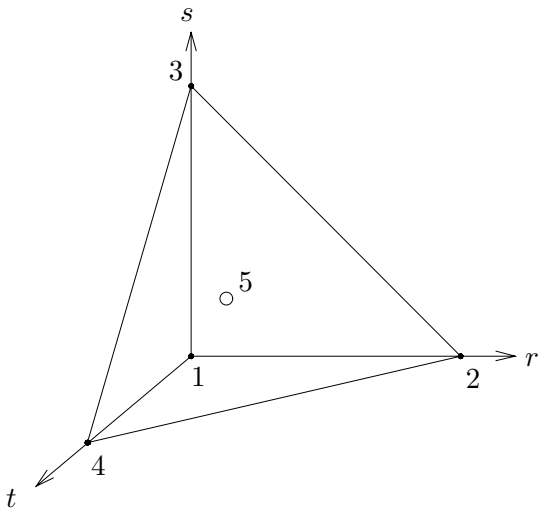

Fig. 1 Bubble Function Element

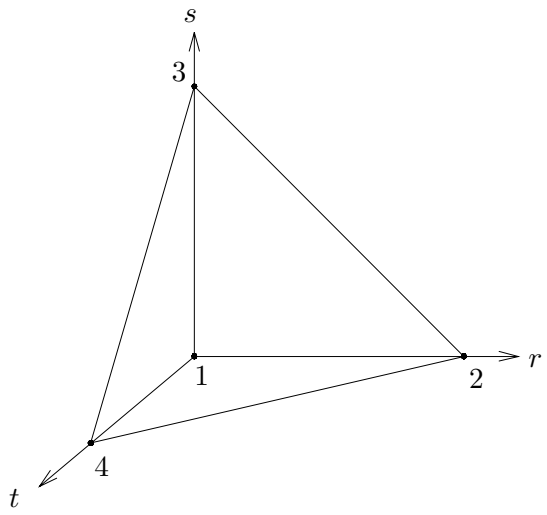

Fig. 2 Linear Element

$$\begin{aligned} u_{ie}^h &= \Phi_1 u_{i1} + \Phi_2 u_{i2} + \Phi_3 u_{i3} + \Phi_4 u_{i4} + \Phi_5 u_{i5}^{'}, \\ u_{i5}^{'} &= u_{i5} - \frac{1}{4}(u_{i1} + u_{i2} + u_{i3} + u_{i4}), \end{aligned} \tag{10}$$

$$\Phi_1 = 1 - r - s - t, \quad \Phi_2 = r, \quad \Phi_3 = s, \quad \Phi_4 = t, \quad \Phi_5 = \phi_B,$$
$$p_e^h = \Psi_1 p_1 + \Psi_2 p_2 + \Psi_3 p_3 + \Psi_4 p_4, \tag{11}$$

$$\Psi_1 = 1 - r - s - t, \quad \Psi_2 = r, \quad \Psi_3 = s, \quad \Psi_4 = t, \tag{12}$$

Here ϕ_B is the bubble function which satisfies C^0 continuity and $\Phi_\alpha(\alpha = 1 \sim 5)$ is the bubble element for velocity in five-node tetrahedral element. $\Psi_\lambda(\lambda = 1 \sim 4)$ is the linear element and $u_{i\alpha}$ and p_λ represent the nodal values at the α th node of each finite element. Eq. (10) is divided from the linear interpolation and the bubble function interpolation as follows:

$$u_{ie}^h = \bar{u}_{ie}^h + u_{ie}^{'h}, \quad \bar{u}_{ie}^h = \Phi_1 u_{i1} + \Phi_2 u_{i2} + \Phi_3 u_{i3} + \Phi_4 u_{i4}, \quad u_{ie}^{'h} = \phi_B b_{ie}, \quad b_{ie} = u_{i5}^{'}. \tag{13}$$

5.2 The Stabilized Control Method for The Bubble Function Element

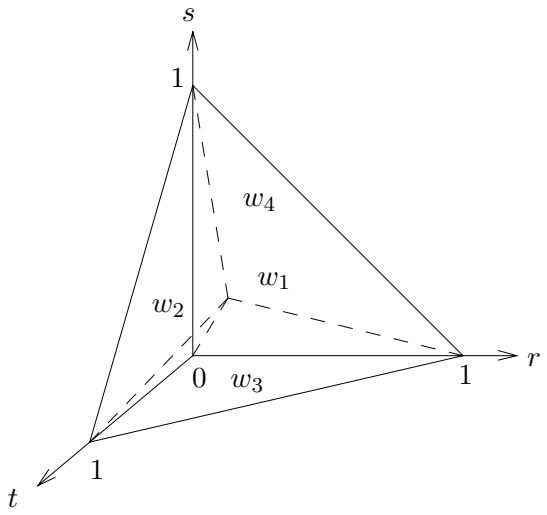

Fig. 3 Subdivision of Element

The linear bubble function is defined using the iso-parametric coordinates (r, s, t) as is shown in Fig. 3. four four tetrahedra $w_1 - w_4$ in Fig. 3 is divided at the center of gravity. The bubble function of C^0 continuity can be considered on each sub-tetrahedron as the following equation:

$$\phi_B^\xi = \begin{cases} 4^\xi \ (1 - r - s - t)^\xi & \text{in } \omega_1, \\ 4^\xi \ r^\xi & \text{in } \omega_2, \\ 4^\xi \ s^\xi & \text{in } \omega_3, \\ 4^\xi \ t^\xi & \text{in } \omega_4, \end{cases} \tag{14}$$

where ξ is the shape parameter of the bubble function. It is used so as to introduce the suitable stabilisation effect by changing the shape. The range of the parameter is a positive real number because Eq. (14) satisfies the condition that Eq.(14) is 0 on each boundary of element.

The criteria for the steady problem is used. in which the discretized form derived from the bubble element is equivalent to those from the SUPG. Thus, the stabilisation parameter τ_{eB} defined as

$$\tau_{eB} = \frac{\{\int_{\Omega_e} \phi_e^\xi d\Omega\}^2}{\nu \int_{\Omega_e} (\phi_{e,i}^\xi)^2 d\Omega V_e}, \tag{15}$$

where V_e is the element volume and ϕ_e^{ξ} is the bubble function defined on each element. From the criteria for the stabilized parameter in the SUPG, we selected the optimal parameter τ_{eS}

$$\tau_{eS} = \left[\left(\frac{2|u_i|}{h_e} \right)^2 + \left(\frac{4\nu}{h_e^2} \right)^2 \right]^{-\frac{1}{2}}, \tag{16}$$

where h_e^2 is width of element. Eq. (15) is not equal to Eq. (16). The bubble function is improved as the follow equation

$$\tau_{eB} = \frac{\{\int_{\Omega_e} \phi_e^{\xi} d\Omega\}^2}{(\nu + \nu') \int_{\Omega_e} (\phi_{e,i}^{\xi})^2 d\Omega V_e}, \tag{17}$$

$$\tau_{eB} = \tau_{eS}, \tag{18}$$

where $\nu^{'}$ the stabilized operator control parameter. Substituting Equations (15), (16) and (17) into finite element equation, the following term is added to the equation of motion:

$$\sum_{e=1}^{N_e} \nu^{'} \int_{\Omega_e} (\phi_{e,i}^{\xi})^2 d\Omega b_e, \tag{19}$$

where N_e and b_e are the total number of the elements and barycenter point, respectively.

6 Parallel Computing

6.1 Parallel Computer

The parallel computer is bundled some computers using network. Each computer has more than one processor and memory, respectively. In this research, two kinds of parallel computers are used. One is IBM pSeries 690 super computer and the other is PC cluster. IBM pSeries 690 has high speed network and high power CPU (Power4). Therefore, it is possible to compute with high parallel efficiency. On the other hand, the PC cluster uses network which is constructed with Giga-Bit Ethernet. In generally, PC cluster has low parallel efficiency because of slow network. However, PC cluster is much cheaper than super computer. In this research, massage passing interface (MPI) is used for massage passing library. The massage passing library is important for parallel computing.

6.2 Domain Decomposition

The parallel computing technique based on the domain decomposition technique is developed. The computational time of parallel computing is the sum of the general computational time, communication time and time to wait synchronisation. For the efficient parallel computing, the communication time and the time for waiting synchronisation must be reduced. In this research, non-overlap domain decomposition method is applied. Domain decomposition is carried out using METIS [6]. The METIS works fast and can be applied to the partitioning of finite element mesh which has large size. Parallel algorithm is applied to the element-by-element conjugate gradient method. The CG method is used as a solver for symmetric system of simultaneous linear equations. The Bi-CG stable method is used as a solver for unsymmetric system of simultaneous linear equations. The algorithm of CG method and Bi-CG stable method are improved for parallel computing. The improved points are in calculation of inner product and superposition. The algorithms of CG method and Bi-CG stable method are as follows:

CG method

1,Put x_0.
2,Compute $r_0 = b - Ax_0 = b - \underbrace{\sum_e A^{(e)} x_0}_{\text{(A)}}$.
3,$p_0 = r_0$.
4,Compute $q_k = Ap_k = \underbrace{\sum_e A^{(e)} p_k}_{\text{(A)}}$.
$\alpha = \underbrace{(r_k, r_k)}_{\text{(B)}} / \underbrace{(p_k, q_k)}_{\text{(B)}}$
$x_{k+1} = x_k + \alpha p_k$
$r_{k+1} = x_k - \alpha q_k$
5,if $r_{k+1} < \varepsilon$ STOP
6,Compute $\beta = \underbrace{(r_{k+1}, r_{k+1})}_{\text{(B)}} / \underbrace{(r_k, r_k)}_{\text{(B)}}$
7,Compute $p_{k+1} = r_{k+1} + \beta p_k$ go to 4.

Bi-CG Stable method

1,Put x_0.
2,Compute $r_0 = b - Ax_0 = b - \underbrace{\sum_e A^{(e)} x_0}_{\text{(A)}}$.
3,$p_0 = r_0$.
4,Compute $q_k^1 = Ap_k = \underbrace{\sum_e A^{(e)} p_k}_{\text{(A)}}$.
$\alpha = \underbrace{(r_0, r_k)}_{\text{(B)}} / \underbrace{(r_0, q_k^1)}_{\text{(B)}}$
$t_k = r_k - \alpha q_k^1$
5,Compute $q_k^2 = At_k = \underbrace{\sum_e A^{(e)} t_k}_{\text{(A)}}$.
$\zeta = \underbrace{(t_k, q_k^2)}_{\text{(B)}} / \underbrace{(q_k^2, q_k^2)}_{\text{(B)}}$
$x_{k+1} = x_k + \alpha p_k + \zeta t_k$
$r_{k+1} = t_k - \zeta q_k^2$
6,if $r_{k+1} < \varepsilon$ STOP
7,Compute $\beta = \alpha \underbrace{(r_0, r_{k+1})}_{\text{(B)}} / \zeta \underbrace{(r_0, r_k)}_{\text{(B)}}$
8,Compute $p_{k+1} = r_{k+1} + \beta p_k$ go to 4.

In the algorithm, A, x_k and b are the stiffness matrix, solution at kth iterations and right-hand-side vector, respectively. The functions r_k, p_k and q_k are vectors and the functions α, β and ζ are scalar.

The underlined terms need communication among processors in the algorithm. An example for the domain decomposition is shown in Fig. 4.

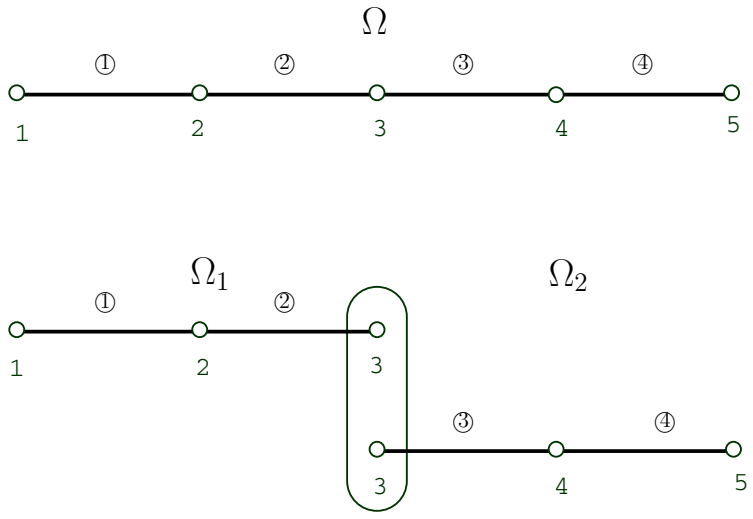

Fig. 4 Non-overlap Domain Decomposition : There is no overlap of elements.

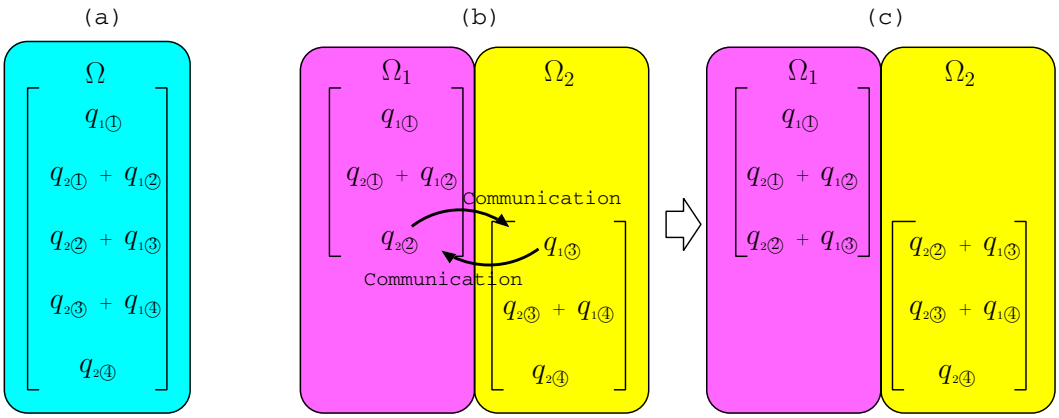

Fig. 5 Complement of Vectors on Each Sub-domain

(A) *Neighboring Communication*

This operation occurs on the interface of the sub-domain. The state value is communicated with each neighboring sub-domain. The complement of the element of vector q is required at the calculation of matrix-vector product as illustrated in Fig. 5. The matrix-vector product is expressed as follows

$$q_k = Ap_k = \sum_e A^{(e)} p_k \left(= \sum_e q_k^{(e)} \right),$$

where the superscribed e represents element number. The matrix-vector product is carried out by element-by-element process. The local product on each element such as

$$\begin{pmatrix} q^{(2)}_{k\ 2} \\ q^{(2)}_{k\ 3} \end{pmatrix} = \begin{pmatrix} a^{(2)}_{2,2} & a^{(2)}_{2,3} \\ a^{(2)}_{3,2} & a^{(2)}_{3,3} \end{pmatrix} \begin{pmatrix} p_{k\ 2} \\ p_{k\ 3} \end{pmatrix},$$

where subscribed number represents the number of node, is superposed onto global vector. In the case of single computer, an element of the global vector is expressed as follows,

$$q_{k\ 3} = q^{(2)}_{k\ 3} + q^{(3)}_{k\ 3}.$$

In the case of parallel computing, an element $q_{k\ 3}$ on each sub-domain is expressed as follows,

$$q^{'}_{k\ 3} = q^{(2)}_{k\ 3} \quad \text{on the domain1,}$$

$$q^{''}_{k\ 3} = q^{(3)}_{k\ 3} \quad \text{on the domain2.}$$

Therefore to get correct value, these values, $q^{'}_{k\ 3}$ and $q^{''}_{k\ 3}$, must be added together as follows,

$$q_{k\ 3} = q^{'}_{k\ 3} + q^{''}_{3}.$$

In this calculation, the communication between neighboring sub-domains is required.

(B) *Global Communication*

This operation occurs within all sub-domains. The inner product is associated with all sub-domains.

$$\boldsymbol{rr} = \sum_{i=1}^{n} r_i r_i,$$

where n and subscribed i are total number of nodes and the number of node, respectively. In the case of single computer, an element of the function is expressed as follows,

$$\boldsymbol{rr} = r_1 r_1 + r_2 r_2 + r_3 r_3 + r_4 r_4 + r_5 r_5$$

In the case of parallel computing, the function on each sub-domain is expressed as follows,

$$\boldsymbol{rr}' = r_1 r_1 + r_2 r_2 + r_3 r_3 \quad \text{on the domain1,}$$

$$\boldsymbol{rr}'' = r_3 r_3 + r_4 r_4 + r_5 r_5 \quad \text{on the domain2.}$$

The sum of $\boldsymbol{rr}'$ and $\boldsymbol{rr}''$ is as follows

$$\boldsymbol{rr}''' = \boldsymbol{rr}' + \boldsymbol{rr}'' = r_1 r_1 + r_2 r_2 + m \times r_3 r_3 + r_4 r_4 + r_5 r_5,$$

where m is total number of sub-domains, which are adjacent to the communication point. In this case, m is equal to 2. To obtain the $\boldsymbol{rr}$, the $\boldsymbol{rr}'$ and $\boldsymbol{rr}''$ are modified as follows,

$$\boldsymbol{rr}' = r_1 r_1 + r_2 r_2 + \frac{r_3 r_3}{m} \quad \text{on the domain1,}$$

$$\boldsymbol{rr}'' = \frac{r_3 r_3}{m} + r_4 r_4 + r_5 r_5 \quad \text{on the domain2.}$$

Then the $\boldsymbol{rr}$ is calculated correctly by adding all local inner products, $\boldsymbol{rr}'$ and $\boldsymbol{rr}''$.

7 Numerical Example

7.1 Computational Model

For computational example, computed the wind environment around the buildings in Korakuen campus of Chuo University is analized. The finite element mesh is constructed by Delaunay method. The mesh is shown in Fig. 6. The scale of the computational domain of this model is 182 m radius and 234 m height. The wind direction and the boundary condition are set as shown in Fig. 7. Non-slip condition is assumed on the wall of the building and on the ground. The total number of the nodes and the elements become 253,896 and 1447,491, respectively.

The domain is divided into some sub domains by the domain decomposition technique. Five cases which are two, four, eight and sixteen sub-domains shown in Figs.8, 9, 10 and 11 are calculated. The total number of nodes and the total number of communication points on each sub-domain are shown in Tabs. 1 and 2. Two computers, which are IBM pSeries 690 and PC Cluster, are used. The specification of each computer is shown in Tab. 6.

7.2 Results

The computational result of state of wind flow is shown in Fig. 12. This figure shows the pressure iso-surface and streamlines. It is verified that the calculation is carried out stably. Table 3 shows comparison of the computational time of each parallel computing with the computing by single processor at one time loop. The total calculation times by both computers are reduced as the number of CPUs increases. Tab. 4 shows the parallel efficiency. In general, The parallel efficiency becomes smaller as the number of CPUs increases. On the other hand, the parallel efficiency of PC cluster is falls. However in case of IBM pSeries 690, the parallel efficiency increases as the number of CPUs increases.

The iterative number in CG method is almost same in each case as shown in Tab. 5. Therefore,

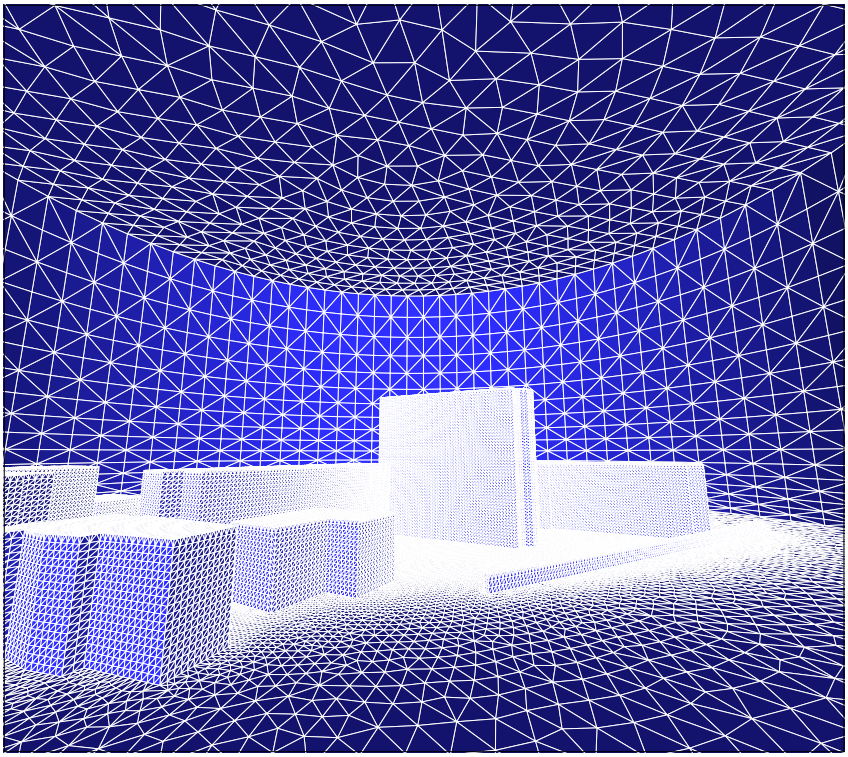

Fig. 6 Finite Element Mesh

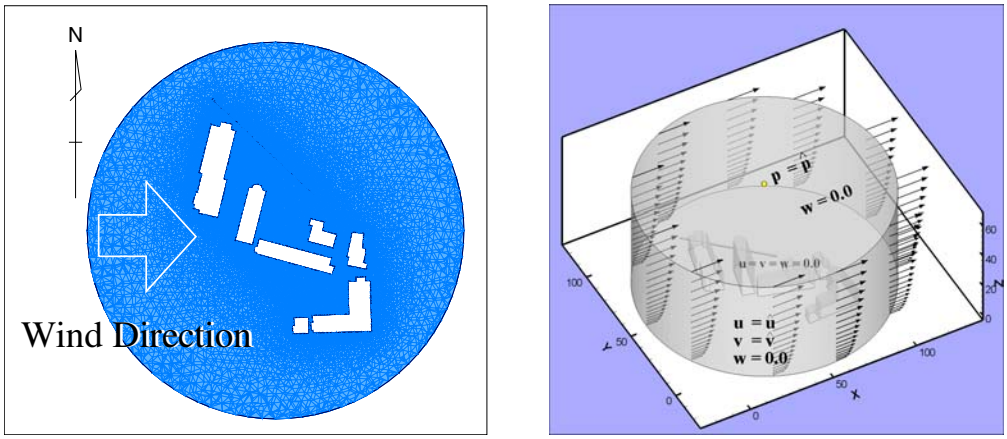

Fig. 7 Wind Direction and Boundary Conditions

the influence of the iterative number to the CPU time is considered to be small. The point which should be noted is the size of cash-memory. As shown in Tab. 6, IBM pSeries 690 has much larger size of the level 1 and level 2 cash-memories than PC cluster. Moreover IBM pSeries 690 has level 3 cash-memory. In the case that the domain is divided into more than seven sub-domains, the data which is required for an operation becomes so small enough that it is on the cash memory. Then the operation can be carried out with high speed because the access to the main memory, which is low-speed, is not necessary. In this case, influence of reduction of the access in main memory on the calculation time is larger than that of the increment of the amount of communications. Thus the parallel efficiency becomes higher than 100%. It can be said that the balance of cash-memory size

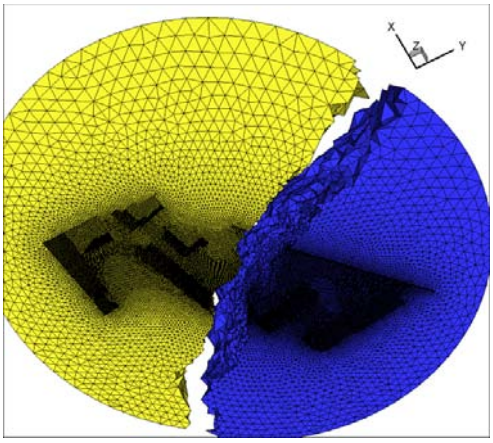

Fig. 8 2 Sub-domains

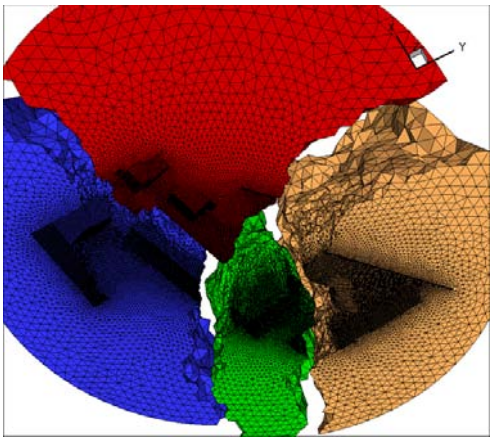

Fig. 9 4 Sub-domains

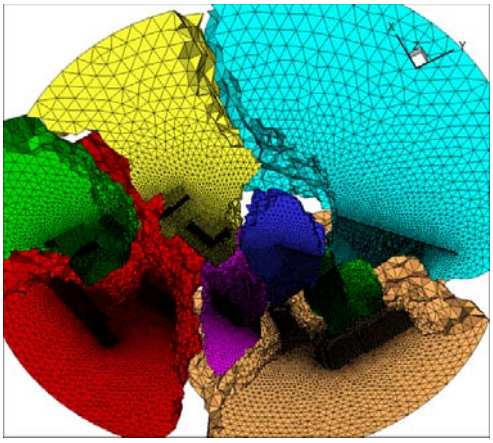

Fig. 10 8 Sub-domains

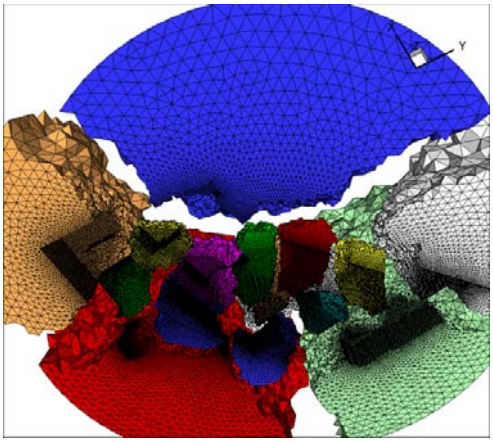

Fig. 11 16 Sub-domains

and the number of subdivisions is important for efficient parallel computing.

8 CONCLUSION

In this research, wind flow outside of a group of buildings is investigated. For the reduction of the calculation time and storage requirement, the parallel computation which based on the domain decomposition method is employed. The wind flow is assumed as incompressible viscous flow. The Navier-Stokes equation is used as basic equation. For the spatial discretization, a mixed interpolation using the bubble function element is employed.

It is mentioned that the computation is carried out successfully and stably from the computational result. From the result of calculation time, it can be said that the parallel computing is effective for the reduction of the calculation time. In this research, a special case in which the parallel efficiency becomes higher than 100% was found. As the reason why parallel efficiency is over 100%, it is considered that the IBM pSeries 690 has a high-speed network between each processor and has a large cash-memory. Thus the balance of cash-memory size and the number of subdivisions is important for efficient parallel computing.

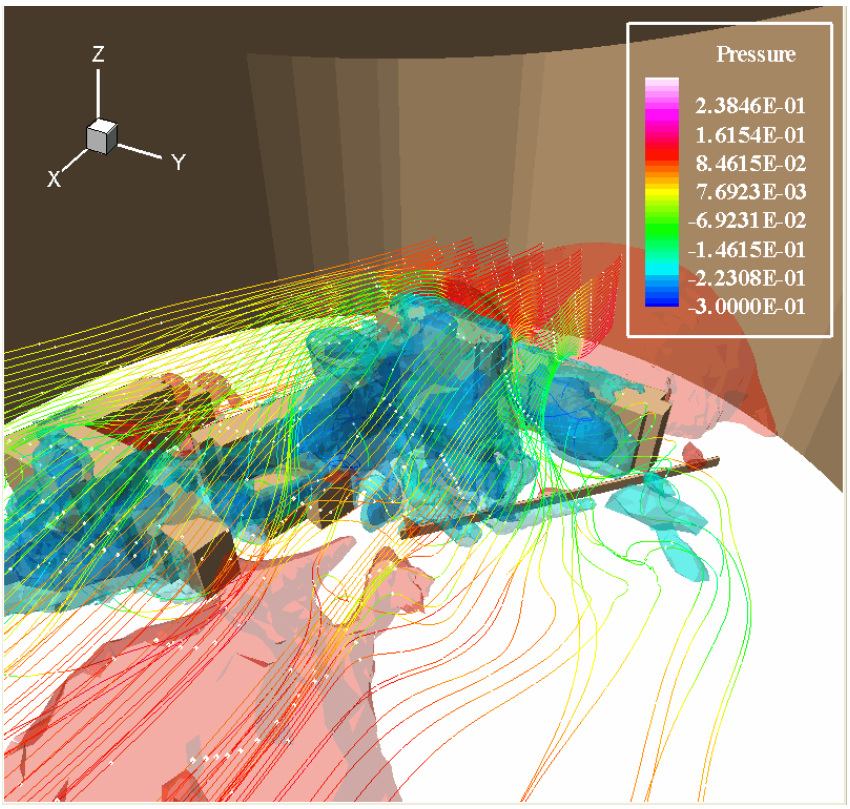

Fig. 12 Result : Pressure Distribution

Table 1 Number of Nodes in Each Sub-domain

	1 CPU	2 CPU	4 CPU	8 CPU	16 CPU
Domain 1	253,896	128,238	64,861	33,514	17,342
Domain 2	-	128,398	65,610	32,333	16,501
Domain 3	-	-	65,043	32,694	16,562
Domain 4	-	-	64,727	33,339	17,675
Domain 5	-	-	-	32,959	16,778
Domain 6	-	-	-	32,717	17,037
Domain 7	-	-	-	33,623	16,846
Domain 8	-	-	-	33,079	16,514
Domain 9	-	-	-	-	17,044
Domain 10	-	-	-	-	16,878
Domain 11	-	-	-	-	16,184
Domain 12	-	-	-	-	17,213
Domain 13	-	-	-	-	15,862
Domain 14	-	-	-	-	16,916
Domain 15	-	-	-	-	17,411
Domain 16	-	-	-	-	17,343
Average	253,896	128,318	65,060	33,026	16,882

Table 2 Number of Communication Points on Boundary of Each Sub-domain

	2 CPU	4 CPU	8 CPU	16 CPU
Domain 1	2,740	2,923	2,899	1,652
Domain 2	2,740	4,313	2,441	1,838
Domain 3	-	2,266	2,850	1,830
Domain 4	-	3,332	2,681	2,614
Domain 5	-	-	2,554	1,688
Domain 6	-	-	3,022	2,405
Domain 7	-	-	2,723	2,327
Domain 8	-	-	2,010	2,304
Domain 9	-	-	-	1,704
Domain 10	-	-	-	2,268
Domain 11	-	-	-	1,915
Domain 12	-	-	-	2,488
Domain 13	-	-	-	1,811
Domain 14	-	-	-	1,863
Domain 15	-	-	-	1,814
Domain 16	-	-	-	3,211
Average	2,740	2,648	3,026	2,107

Table 3 Comparison of the computational time of the parallel computing with that of single processor

		single processor	2 processors	4 processors	8 processors	16 processors
IBM pSeries 690	Cal.	$6m43.448s$	$3m56.609s$	$1m27.629s$	$35.957s$	$16.142s$
	Com.Min	–	$1.545s$(No.1)	$2.742s$(No.1)	$1.480s$(No.7)	$1.877s$(No.10)
	Com.Max	–	$8.142s$(No.2)	$23.057s$(No.3)	$9.117s$(No.6)	$7.030s$(No.7)
	T.T.	$6m43.448s$	$3m58.154s$	$1m29.371s$	$37.437s$	$17.019s$
PC Cluster	Cal.	$7m26.850s$	$3m40.479s$	$1m97.286s$	$1m\ 1.662s$	–
	Com.	–	$3.979s$(No.2)	$3.545s$(No.3)	$10.949s$(No.7)	–
	Com.	–	$15.238s$(No.1)	$25.608s$(No.4)	$26.887s$(No.8)	–
	T.T.	$7m26.850s$	$3m44.458s$	$2m\ 0.821s$	$1m12.611s$	–

Cal. : General Calculation Time, Com. : Communication Time, T.T. : Total Time
(No.n) : Number of Sub-domain

Table 4 Comparison of the parallel efficiency

	single processor	2 processors	4 processors	8 processors	16 processors
IBM pSeries 690	100.00%	84.70%	112.86%	134.71%	148.16%
PC Cluster	100.00%	99.54%	92.65%	76.93%	–

Table 5 Iterative Number in CG and Bi-CG Stable Method

Number of processor	CG method	Bi-CG Stable
1CPU (IBM pSeries 690)	1044	11
2CPU (IBM pSeries 690)	1042	11
4CPU (IBM pSeries 690)	1046	11
8CPU (IBM pSeries 690)	1048	11
16CPU (IBM pSeries 690)	1048	11
1CPU (PC Cluster)	1041	11
2CPU (PC Cluster)	1044	11
4CPU (PC Cluster)	1037	11
8CPU (PC Cluster)	1040	11

Table 6 Specification of Computer

		Power4(IBM pSeries 690)	Pentium4(PC Cluster)
CPU		Power4 1.3G Hz	Pentium4 1.8G Hz
Cash Memory / CPU	Level 1	32K byte	8K byte
	Level 2	1.5M byte	256K byte
	Level 3	32M byte	–
Main Memory / CPU		2G byte	1G byte
Network		10G bytes/sec	1G bits/sec

参考文献

[1] Kazuya Nojima and Mutsuto Kawahara, An analysis of wind environment around buildings with unstructured mesh generation technique, Computational Fluid and Solid Mechanics 2003, Vol. 1, pp.1066-1071,(2003)

[2] T. Yamada: A Bubble Element for Inviscid Flow",Finite Elements in Fluids Vol.9, 1995, pp.1567-1576.

[3] T. Yamada: A Bubble Element for the Compressible Euler Equations ", IJCFD, Vol.9, pp.273-283, 1998.

[4] Junichi Matsumoto, Tsuyoshi Umetsu and Mutsuto Kawahara, Incompressible Viscous Flow Analysis and Adaptive Finite Element Method Using Linear Bubble Function,(1999)

[5] Junichi Matsumoto and Mutsuto Kawahara, Shape Identification for Fluid-Structure Interaction Problem Using Improved Bubble Element, IJCFD, Vol. 15, pp.33-45,(2001)

[6] http://www-users.cs.umn.edu/ karypis/metis/