MHD Simulation of Earth's Magnetosphere: Visualization and Parallelization

Tatsuki Ogino (Solar-Terrestrial Environment Laboratory, Nagoya University)

1. Introduction

The three-dimensional global magnetohydrodynamic (MHD) simulation of interaction between the solar wind and earth's magnetosphere could reproduce the form of the average magnetosphere where power balanced about 20 years ago, and developed. And it continues development conjointly to the rapid progress of a computer and IT technology, and, these days, has developed even into the grade which can argue about the dynamics of a magnetosphere as compared with a satellite and a ground observations. In this way, it came to investigate from a simulation directly the response of the magnetosphere ionosphere to an upstream solar wind or change of the magnetic field (IMF) between planets and the substorms and magnetic storms which are the big turbulence phenomenon in a magnetosphere. In order to perform the global MHD simulation of these solar wind magnetosphere interactions with sufficient accuracy, while the calculation method needs to be improved of one side, use of the greatest supercomputer and use of a parallel computing method also with efficient it are indispensable.

As a candidate of such parallel computing common program language, it has been said that there are High Performance Fortran (HPF) and Message Passing Interface (MPI). Although the American Collaborative-Research person had said that HPF excelled as common program language, there was also criticism that performance did not fully come out by many large-sized programs. HPF/JA (Japanese extended edition of HPF by JAHPF) can be used from 2000, and the fluid code and the MHD code by which full vectorization full parallelization is carried out by VPP Fortran can be rewritten now to HPF/JA comparatively easily. Moreover, it was shown that the program of the HPF/JA obtains performance equivalent to VPP Fortran. However, the present condition is that the spread of HPF(s) is not progressing in spite of a success of HPF/JA. In this way, the expectation for MPI which remained at the end as a global standard parallelization language will grow. In this lecture and training, the production and the directions for the parallel computing three-dimensional MHD code using MPI are mainly explained, while comparing with the three-dimensional MHD code of the Earth's magnetosphere written by VPP Fortran and HPF/JA.

Visualization is indispensable in order to understand complicated simulation results, such as solar wind magnetosphere interaction. Especially two important and troublesome functions are animation production and three-dimensional visualization. But an animation helps an understanding of magnetosphere dynamics by showing time change. Three-dimensional visualization demonstrates power to clear the feature of the streamline of magnetosphere, magnetic

field line, and electron-current structure. Furthermore, everyone can see immediately the simulation result in which the information disclosure by the Internet which has become the center of attention recently does not have self-contradiction which is not simply made. Moreover, it is becoming a powerful means when you understand a phenomenon better.

In such situation, by the appearance of VRML(Virtual Reality Modeling Language), even if it did not have a three-dimensional image-processing special-purpose machine and the software only for three-dimensional image processing, when anyone had even the viewer of VRML, the situation which can be regarded as he liking a three-dimensional image was realized. Although it is dependent on the throughput of its own computer, if anyone uses browsers, such as Netscape and Internet Explorer, viewers, such as Cosmoplayer of VRML2.0 correspondence, can be used gratuitously. Because a personal computer is also a high speed recently, if high-speed CPU and a graphics accelerator are stacked and still more sufficient memory (256MB over) is carried, sufficient performance for visualization can be demonstrated. Moreover, if you want to see a high-precision three-dimensional image comfortably, use of Webspace or Cosmoworlds is still more effective.

2. Three-Dimensional Global MHD Model of Solar Wind-Magnetosphere Interaction

In the three-dimensional MHD model of the solar wind-magnetosphere interaction, the time development is solved by various methods by using a couple of the MHD equation and the Maxwell equation as an initial value and boundary value problem. The 2 step Lax-Wendroff method which solve a difference equation deduced from a partial differential equation is one of the examples. As various devices in the calculation method for raising spatial resolution, introduction of an un-uniform lattice approach, the non-structural lattice method, the adaptive mesh refinement approach, and the multiplex lattice approach between space-time etc. have been carried out. Below, we describe the modified leap-frog method which is one of the high precision algorithms used for the three-dimensional MHD simulation. [1-4]

2-1. Basic equation

The MHD standard equation functioning as the foundation of a MHD model and Maxwell equation are shown below.

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\mathbf{v}\rho) + D\nabla^2 \rho \qquad (1)$$

$$\frac{\partial \mathbf{v}}{\partial t} = -(\mathbf{v} \cdot \nabla)\mathbf{v} - \frac{1}{\rho}\nabla p + \frac{1}{\rho}\mathbf{J} \times \mathbf{B} + \mathbf{g} + \frac{1}{\rho}\Phi \qquad (2)$$

$$\frac{\partial p}{\partial t} = -(\mathbf{v} \cdot \nabla)p - \gamma\, p\nabla \cdot \mathbf{v} + D_p \nabla^2 p \qquad (3)$$

$$\frac{\partial \mathrm{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B} \qquad (4)$$

$$\mathbf{J} = \nabla \times (\mathbf{B} - \mathbf{B}_d) \qquad (5)$$

Equation (1) - (4) is an equation showing magnetic field change called the equation of continuation, an equation of motion, the equation of the pressure change which can be found from a principle of conservation of energy, and an induction equation, respectively. However, ρ is plasma density, $\mathbf{v}$ is Rate vector, p is plasma pressure, $\mathbf{B}$ is magnetic field vector. It asks for these eight parameters as an unknown. However, according to the numerical error of a finite difference method, since J to $\mathbf{B}_d$ becomes a limited value, the numerical error has been removed using a formula (5). $\mathbf{B}_d$ here is an electric doublet magnetic field as a peculiar magnetic field of a earth. Moreover, $\Phi \equiv \mu \nabla^2 \mathbf{v}$ is a viscous clause. $\eta = \eta_o$ $(T/T_o)^{-3/2}$ is the electrical resistance depending on temperature. $T = p/\rho$ here is plasma temperature, and T_o is a value in the ionosphere and it takes it in the range of η_o =0.0005-0.002. About a gravity clause, $g = -g_o/\zeta^3$ ($\zeta^2 = x^2 + y^2 + z^2$、 $g_o = 1.35\times10^{-7}$ ($9.8 m/s^2$)) is acceleration due to gravity, and γ =5/3 are the ratios of specific heat in three-dimensional space. Moreover, The diffusion coefficient D of particles, The diffusion coefficient D_p of pressure, Each coefficient of μ was artificially given, in order to control numerical vibration of the short wavelength resulting from an initial value or a rapid magnetic field change, and it was set to $D = D_p = \mu/\rho_{sw}$ =0.001 (however, ρ_{sw} is the density of solar wind). And, each parameter was standardized by the following. Distance is the earth radius $R_e = 6.37\times10^6$ m. Density is the value $\rho_s = mn_s$ (10^{10} m^{-3}) in the ionosphere. Magnetic field is the present electric doublet magnetic field intensity $B_s = 3.12\times10^4$ nT in the equator. Speed is the Alph Ben speed $V_A = 6.80\times10^6$ m/s in the equator. Time is the Alph Ben passage time $t_s = R_e/V_A$ =0.937s.

2-2. Coordinate system and boundary condition

The solar direction x-axis positive, the evening direction Y-axis positive, the solar earth's magnetosphere coordinate system made direction z-axis of the magnetic north pole positive as shown in Fig. 1 is used for a simulation. A MHD equation and the Maxwell equation are difference-ized between space-time. And time development of eight physical variables in a MHD equation system, the plasma density ρ, speed $\mathbf{v}$, pressure p, and the magnetic field $\mathbf{B}$ is solved.

Here, the earth's magnetosphere model of 1/4 domain which assumed symmetry in every morning and evening and north and south is considered.

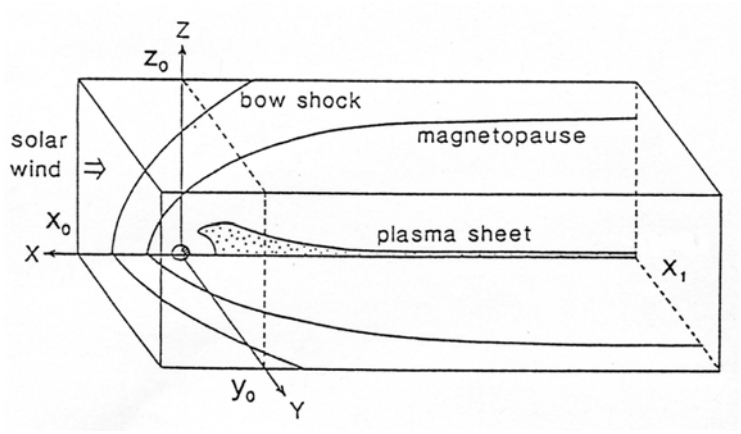

Fig.1 The solar Earth's magnetosphere coordinate system in the three-dimensional MHD simulation

Therefore, the following boundary condition is imposed to each physical quantity $\phi = (\rho, \mathbf{v}, p, \mathbf{B})$

(1) ϕ = const, at the time of the fixed boundary condition $x = x_0$

(2) $\partial\phi/\partial x = 0$, at Free boundary condition $x = x_1$

(3) $\partial\phi/\partial y = 0$, $\partial\phi/\partial z = 0$ at the free boundary condition which had an angle of 45 degrees to the X-axis $y = y_0$, $z = z_0$

(4) The mirror boundary condition to $z = 0$

$$\frac{\partial\rho}{\partial z} = \frac{\partial p}{\partial z} = \frac{\partial v_x}{\partial z} = \frac{\partial v_y}{\partial z} = \frac{\partial B_z}{\partial z} = 0 \qquad (6)$$
$$v_z = B_x = B_y = 0$$

(5) The mirror boundary condition to $y = 0$

$$\frac{\partial\rho}{\partial y} = \frac{\partial p}{\partial y} = \frac{\partial v_x}{\partial y} = \frac{\partial v_z}{\partial y} = \frac{\partial B_x}{\partial y} = \frac{\partial B_z}{\partial y} = 0 \qquad (7)$$
$$v_y = B_y = 0$$

(6) All the physical quantity is constant to $\xi = \left(x^2 + y^2 + z^2\right)^{1/2} \le \xi_a (= 3.5)$.

The internal solution ϕ_{in} of an initial state and the exterior ϕ_{ex} obtained from a simulation result are connected using the following formula the whole time step by introducing the smooth

form function $f \equiv a_0 h^2 \left(a_0 h^2 + 1\right)$.

$$\phi = f\phi_{ex} + (1 - f)\phi_{in} \qquad (8)$$

It is $h = \left(\xi / \xi_a\right)^2 - 1$ to $a_0 = 100$, $\xi \geq \xi_a$. It is $h = 0$ to $\xi < \xi_a$.

2.3 Initial condition

In an initial condition, "a mirror dipole magnetic field of zero in the upper stream from a symmetry plane" and "the ionosphere of spherical symmetry with which gravity and plasma pressure balanced statically" are supposed, and the structure of the magnetosphere near a stationary state is searched for by beginning to pass the Solar wind which has fixed density, speed, and temperature from the upper stream of a simulation box. The reason for using a mirror dipole magnetic field in early stages is for not including a magnetic field ingredient parallel to a flow upstream. As mentioned above, it is as a boundary condition, the upper stream takes into consideration the form of a Bow shock where a fixed end, the side, and an up-and-down side are formed in the front of a Magnetosphere, and The free end which gave x axis and the angle of 45 degrees, the lower stream is a free end to a direction perpendicular to a field, in the field of $y = 0$ passing through the center of the earth, or $z = 0$, a magnetic field, the vector of speed, and the boundary condition of a mirror image without inconsistency are imposed. Moreover, time change of solar wind or the parameter of IMF is carried out, and a response and turbulence phenomenon of magnetosphere ionosphere are investigated. The concrete function of an initial condition is given as follows.

Density

$$\begin{aligned} \rho_0 &= \xi^{-3} \qquad \rho_0 \geq 0.2\rho_{sw} \\ \rho_0 &= 0.2\rho_{sw} \qquad \rho_0 < 0.2\rho_{sw} \end{aligned} \qquad (9)$$

Plasma pressure

$$\begin{aligned} p_0 &= p_{00}\xi^{-2} \qquad p_0 \geq p_{sw} \\ p_0 &= p_{sw} \qquad p_0 < p_{sw} \end{aligned} \qquad (10)$$

Gravity

$$\mathbf{g} = -\frac{g_0}{\xi^3}(x, y, z) \qquad (11)$$

Dipole peculiar magnetic field

$$\mathbf{B}_d = \frac{1}{\xi^5}\left(-3xz,\ -3yz,\ x^2+y^2-2z^2\right) \qquad (12)$$

It is $p_{00} = (\gamma-1)g_0/\gamma = 5.4\times10^{-7}$ at $g_0 = 1.35\times10^{-6}$.

The parameter of Solar wind is Density $\rho_{sw} = 5\times10^{-4}$ (equivalent to $5/\mathrm{cm}^{-3}$), $\mathbf{v}_{sw} = (v_{sw},0,0)$

$v_{sw} = 0.0441-0.118$ $(300-800\mathrm{km/s})$, $p_{sw} = 3.56\times10^{-8}$ $(T_{sw} = 2\times10^5\mathrm{K})$ at $x = x_0$. And, the magnetic field between planets is $B_{IMF} = 0$ or $\pm1.5\times10^{-4}(\pm5\,\mathrm{nT})$. B_{IMF} shows the ingredient of uniform IMF carried by the Solar wind.

2.4 Introduction of modified Leap-Frog method

As a numerical computation method, the Modified leap-frog method as shown in Fig. 2 is used. 1 time of the beginning is solved by the two step Lax-Wendroff method, the (ℓ-1) next times are solved by the leap-frog method, and the procedure of a series of is repeated. When adopting the central space difference of secondary accuracy, it is chosen out of alignment calculation of numerical accuracy and preliminary simulation as ℓ = 8, because the larger one of the value of ℓ is numerically desirable in the stable range. The Modified leap-frog method has adopted a part of numerical stabilization effect of the two step Lax-Wendroff method. It is numerical attenuation of the leap-frog method, the numerical attenuation which adopted more small effects of distribution, and a kind of combination calculation method which balance is good for distribution and was able to be taken. It has an advantage which is easier to understand the influence of the numerical error given to a result, because it can be made in agreement with the two calculation methods of having got to know character well, by changing Parameter ℓ.The concrete calculation scheme of the Modified leap-frog method is shown below. The partial differential equation of the following form is introduced first.

$$\frac{\partial f}{\partial t} = -\frac{\partial f}{\partial x} - \frac{\partial f}{\partial y} - \frac{\partial f}{\partial z} - f \qquad (13)$$

(1) First step

$$f^{t}_{i+\frac{1}{2},\,j+\frac{1}{2},\,k+\frac{1}{2}} = \frac{1}{8}\left(f^{t}_{i,j,k} + f^{t}_{i+1,j,k} + f^{t}_{i,j+1,k} + f^{t}_{i+1,j+1,k} + f^{t}_{i,j,k+1} + f^{t}_{i+1,j,k+1} + f^{t}_{i,j+1,k+1} + f^{t}_{i+1,j+1,k+1}\right) \tag{14}$$

$$\begin{aligned} f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} = {} & f^{t}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \frac{1}{2}\Delta t\, f^{t}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} \\ & - \frac{\Delta t}{8\Delta x}(f^{t}_{i+1,j,k} + f^{t}_{i+1,j+1,k} + f^{t}_{i+1,j,k+1} + f^{t}_{i+1,j+1,k+1} - f^{t}_{i,j,k} - f^{t}_{i,j+1,k} - f^{t}_{i,j,k+1} - f^{t}_{i,j+1,k+1}) \\ & - \frac{\Delta t}{8\Delta y}(f^{t}_{i,j+1,k} + f^{t}_{i+1,j+1,k} + f^{t}_{i,j+1,k+1} + f^{t}_{i+1,j+1,k+1} - f^{t}_{i,j,k} - f^{t}_{i+1,j,k} - f^{t}_{i,j,k+1} - f^{t}_{i+1,j,k+1}) \\ & - \frac{\Delta t}{8\Delta z}(f^{t}_{i,j,k+1} + f^{t}_{i+1,j,k+1} + f^{t}_{i,j+1,k+1} + f^{t}_{i+1,j+1,k+1} - f^{t}_{i,j,k} - f^{t}_{i+1,j,k} - f^{t}_{i,j+1,k} + f^{t}_{i+1,j+1,k}) \end{aligned} \tag{15}$$

(2) Second step

$$\begin{aligned} f^{t+\frac{1}{2}}_{i,j,k} = \frac{1}{8}(& f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} + f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} \\ & + f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}) \end{aligned} \tag{16}$$

$$f^{t+1}_{i,j,k} = f^{t}_{i,j,k} - \Delta t\, f^{t+\frac{1}{2}}_{i,j,k} \tag{17}$$

$$\begin{aligned} & - \frac{\Delta t}{4\Delta x}(f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} \\ & \qquad - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}) \\ & - \frac{\Delta t}{4\Delta y}(f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} + f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} \\ & \qquad - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} - f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} - f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}}) \\ & - \frac{\Delta t}{4\Delta z}(f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} \\ & \qquad - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} - f^{t+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} - f^{t+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} - f^{t+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}}) \end{aligned} \tag{18}$$

(19)

The concrete procedure of two-step Lax-Wendroff method in the 3-dimensional MHD code is as follows;

1. $f(i,j,k)$ is given for $2 \le i \le nx1,\ 2 \le j \le ny1$ and $2 \le k \le nz1$
2. $f(i,j,k)$ for $i = 1, nx2,\ j = 1, ny2,$ and $k = 1, nz2$ is determined from boundary condition

3. 1st interpolation

$$p(i,j,k)=\frac{1}{8}(f(i,j,k)+f(i+1,j,k)+f(i,j+1,k)+f(i+1,j+1,k) + f(i,j,k+1)+f(i+1,j,k+1)+f(i,j+1,k+1)+f(i+1,j+1,k+1)) \quad (20)$$

$$u(i,j,k)=p(i,j,k) \quad (21)$$

4. Calculation of 1st step

$$\begin{aligned} u(i,j,k)=u(i,j,k)&-\frac{1}{2}\Delta t\, p(i,j,k) \\ &-\frac{\Delta t}{8\Delta x}(f(i+1,j,k)+f(i+1,j+1,k)+f(i+1,j,k+1)+f(i+1,j+1,k+1) \\ &\quad -f(i,j,k)-f(i,j+1,k)-f(i,j,k+1)-f(i,j+1,k+1)) \\ &-\frac{\Delta t}{8\Delta y}(f(i,j+1,k)+f(i+1,j+1,k)+f(i,j+1,k+1)+f(i+1,j+1,k+1) \\ &\quad -f(i,j,k)-f(i+1,j,k)-f(i,j,k+1)-f(i+1,j,k+1)) \\ &-\frac{\Delta t}{8\Delta z}(f(i,j,k+1)+f(i+1,j,k+1)+f(i,j+1,k+1)+f(i+1,j+1,k+1) \\ &\quad -f(i,j,k)-f(i+1,j,k)-f(i,j+1,k)-f(i+1,j+1,k)) \end{aligned} \quad (22)$$

5. 2nd interpolation

$$p(i,j,k)=\frac{1}{8}(u(i-1,j-1,k-1)+u(i,j-1,k-1)+u(i-1,j,k-1)+u(i,j,k-1) + u(i-1,j-1,k)+u(i,j-1,k)+u(i-1,j,k)+u(i,j,k)) \quad (23)$$

6. Calculation of 2nd step

$$\begin{aligned} f(i,j,k)=f(i,j,k)&-\Delta t\, p(i,j,k) \\ &-\frac{\Delta t}{4\Delta x}(u(i,j,k)+u(i,j-1,k)+u(i,j,k-1)+u(i,j-1,k-1) \\ &\quad -u(i-1,j,k)-u(i-1,j-1,k)-u(i-1,j,k-1)-u(i-1,j-1,k-1)) \\ &-\frac{\Delta t}{4\Delta y}(u(i,j,k)+u(i-1,j,k)+u(i,j,k-1)+u(i-1,j,k-1) \\ &\quad -u(i,j-1,k)-u(i-1,j-1,k)-u(i,j-1,k-1)-u(i-1,j-1,k-1)) \\ &-\frac{\Delta t}{4\Delta z}(u(i,j,k)+u(i-1,j,k)+u(i,j-1,k)+u(i-1,j-1,k) \\ &\quad -u(i,j,k-1)-u(i-1,j,k-1)-u(i,j-1,k-1)-u(i-1,j-1,k-1)) \end{aligned} \quad (24)$$

This calculation scheme serves as the two step Lax-Wendroff method, when setting with $u(i,j,k)=p(i,j,k)$. But if the value calculated from the former step is used for $u(i,j,k)$ as it is

and time width is doubled two with Δt from $\frac{1}{2}\Delta t$, it will become the Leap-frog method. The Modified leap-frog method is method of calculation by using the two-step Lax-Wendroff method for the 1st time, and the Leap-frog method for (ℓ-1) time to continue. (Fig. 2)

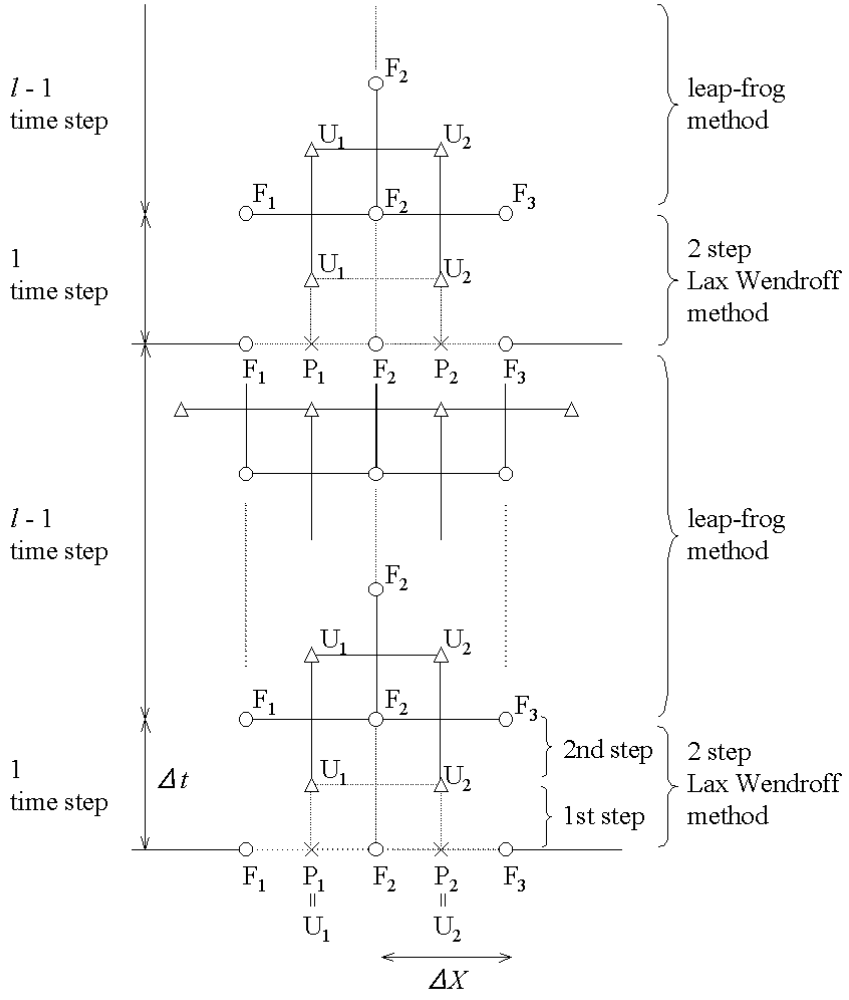

Fig.2 Diagram of Modified Leap-Frog numerical scheme.

Next, it argues about the numerical stability of the Modified leap-frog method using a transfer equation (25).

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \qquad (25)$$

When the difference between space-time is written to be $u_i^j = u(x_i, t_j)$, $u_{i\pm1}^{j\pm1} = u(x_i \pm \Delta x, t_j \pm \Delta t)$ and using Fourier series (26), the amplification rates $A = u_i^{j+1} / u_i^j$ of the two step Lax-Wendroff method is set to (27) and (28).

$$u_i^j = u_o^j e^{ikx}, \quad u_{i\pm1}^j = u_o^i e^{ik(x\pm\Delta x)} = u_i^j e^{\pm k\Delta x} \equiv u_i^j e^{\pm i\kappa} \tag{26}$$

$$A_{2LW} = 1 + i\delta \sin\kappa + \delta^2(\cos\kappa - 1) \tag{27}$$

$$\left|A_{2LW}\right|^2 = 1 + (\delta^4 - \delta^2)(\cos\kappa - 1)^2 \tag{28}$$

Therefore, at the time of $0 \le \delta \equiv \Delta t / \Delta x \le 1$, $\left|A_{2LW}\right| \le 1$ is materialized to all of $\kappa = k\Delta x$ and it becomes stable numerically.

The leap-frog method is set to (29), at $\left|A_{LF}\right| = 1$, and marginally stable.

$$A_{LF}{}^{\frac{1}{2}} = \pm\sqrt{1 + \delta^2 \sin^2 \tfrac{\kappa}{2}} - i\delta \sin \tfrac{\kappa}{2} \tag{29}$$

Therefore, the amplification rate of the Modified leap-frog method is given by the following formula.

$$A_{MLF} = A_{2LW}{}^{1/\ell} A_{LF}{}^{(\ell-1)/\ell} \tag{30}$$

The cycle dependability of the absolute value and phase velocity of an amplification rate for the Modified leap-frog method (MLF), 2 step Lax-Wendroff method (2LW), and Runge-Kutta-Gill method (RKG) is shown in Fig. 3. The absolute value of an amplification rate and cycle dependability of phase velocity for the Modified leap-frog method (MLF) at changing ℓ is shown in Fig. 4. By the Modified leap-frog method, it is understood that numerical accuracy is sharply improved also to phase velocity like an absolute value.

The result of having applied three kinds of calculation methods, the Modified leap-frog method, the two step Lax-Wendroff method, and the leap-frog method, to the wave equation is shown in Fig. 5. The result applied to the simulation of the MHD Bow shock which is a nonlinear phenomenon of electromagnetism hydrodynamics is shown in Fig. 6. When solving propagation of a pulse wave by a finite difference method with an alignment wave equation, a pulse wave collapses and the sequence of a wave appears behind, because numerical attenuation is large and phase velocity is slow at a wave with a short wavelength. It is found that the numerical attenuation and distribution are sharply improved by the modified leap-frog method. In the case of a nonlinear phenomenon, vibration occurs behind a Bow shock by numerical distribution by the two step Lax-Wendroff

method, it is controlled small and the form of the Bow shock is acquired well by the modified leap-frog method. On the other hand, by the leap-frog method, vibration becomes deep and a Bow shock is seen have separated into the train of impulses. Although this does not have numerical attenuation, numerical distribution is a phenomenon depending on the numerical characteristic of the existing leap-frog method, and it is physically meaningless.

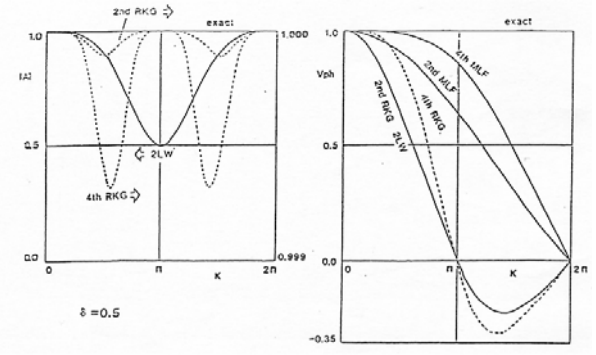

Fig.3 Simulation of the wave equation for 3 different kinds of numerical schemes.

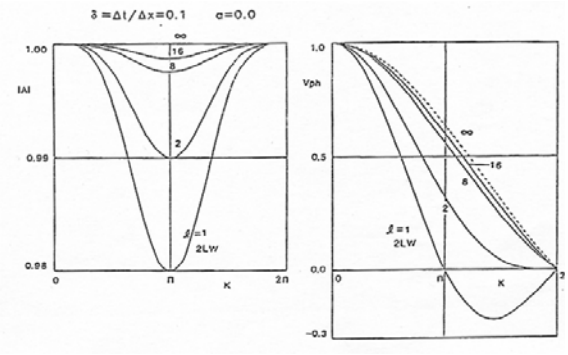

Fig.4 Simulation of the MHD shock wave for 3 different kinds of numerical schemes.

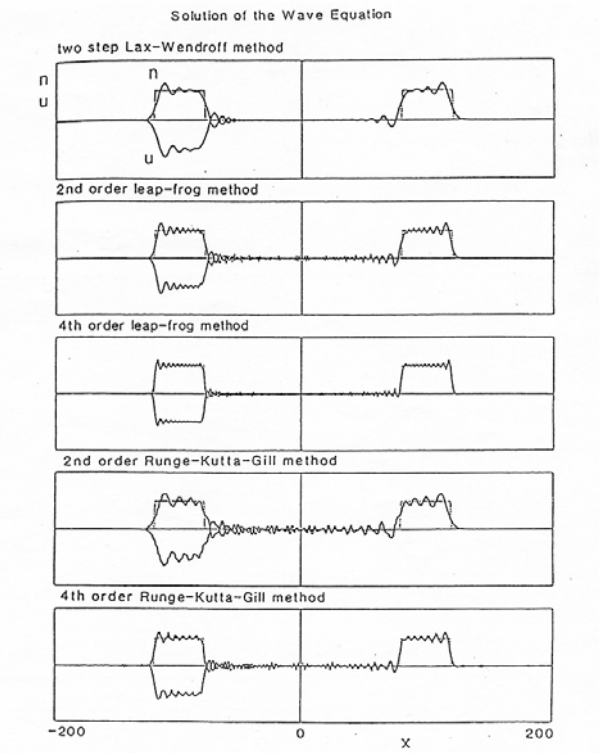

Fig.5 Comparison of the simulation result of the wave equation at the time of using the various calculation methods. Numerical vibration appears at the time of the standup of a pulse wave, and falling as a result of numerical distribution. Secondary accuracy and the 4th accuracy show the order of the accuracy of space difference.

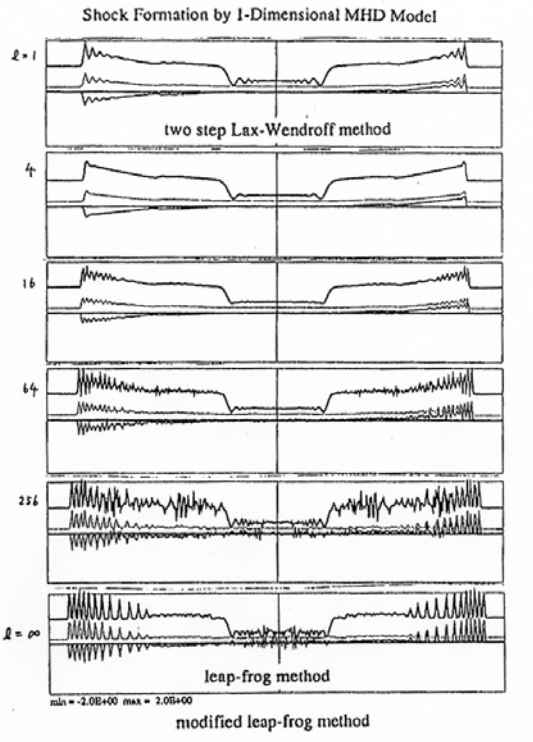

Fig.6 The simulation result of the one-dimensional MHD Bow shock at the time of changing Parameter ℓ by The Modified Leap-Frog method. It becomes the 2-Step Lax-Wendroff method at the time of $\ell=1$. It becomes the Leap-Frog method at the time of $\ell=\infty$. In the Leap-Frog method, it is obtained that a Bow shock decomposes into a train of impulses by a numerical cause.

3. Parallel Computing MHD Code Using MPI

Fujitsu VPP500 can be used now in 1995. The three-dimensional MHD code (earthb) which was carried out the full vectorization for performing the global MHD simulation of a Solar wind and an Earth's magnetosphere interaction was rewritten to VPP Fortran. Complete rewriting of the code was carried and the three-dimensional MHD code (pearthb) by which calculation efficiency was high and full vectorization full parallelization was mostly carried out by VPP Fortran was made [Ogino, 1997]. Then, the three-dimensional MHD code was rewritten from VPP Fortran to HPF/JA in 2000 (hearthb) [Ogino, 2000, Ogino 2002]. As a result, full vectorization full parallelization also of the three-dimensional MHD code (hearthb) written by HPF/JA was able to be carried out like the three-dimensional MHD code of VPP Fortran and calculation speed was also able to obtain the performance equivalent to VPP Fortran. Although it is almost equivalent as a vector parallelization MHD code also at VPP Fortran or HPF/JA, the vectorization MHD code (earthb) is completely a different thing. The biggest difference is that minimization of program size was impossible for in vector parallelization MHD code by the character of parallel computing to minimizing program size in vectorization MHD code. For this reason, when carrying out the MHD simulation of the same number of lattice points, compared with a vectorization code, an about 3 to 4 times as many computer memory as this is required for a vector parallelization code. [Ogino, 2000, Ogino 2002]

It was set to 2002, and cooperation of a surrounding man was able to be obtained, rewriting to MPI from VPP Fortran was able to be carried from HPF/JA to MPI, and the three-dimensional MHD code (meartb) by which full vectorization full parallelization was mostly carried out by MPI was able to be made. Although it is a labor of rewriting, if there is the three-dimensional MHD code by which full vectorization full parallelization was carried out by VPP Fortran or HPF/JA, it is possible to rewrite in the Fortran code of MPI use comparatively easily. Of course, it may arise that it must devise in order to maintain full vectorization full parallelization depending on a code. It is carried by next supplementary explanation how MPI should be used by the problem in MPI, or large-scale calculation.

In the case of vectorization or parallelization, the law of Amdahl is one of those which show a what time speed improvement rate is obtained compared with a non-vectorizing code or a non-parallelizing code. According to the law, in order to obtain a high speed improvement rate using many PE (Processing Element), it is very important that a parallelization rate is close to 100% infinite. Therefore, in order to make an efficient simulation code, it is depends on how full

vectorization and full parallelization are realized. In fact because it is vectorized by inside do roop, what is necessary will be just to parallelize by outside do roop, with the vectorization maintained. "Parallelize from do roop which computation time has required" is said. In case of the method, although parallelization efficiency increases until some extent, it is almost impossible to acquire the parallelization efficiency near 100%. Considering experience of conventional vectorization and parallelization, it is sure that it is most important to decide the structure of a program exactly. The foundations of the parallel computing in the case of using a distributed memory type parallel computer are simple things, and before calculating, they should just bring together all variables required for calculation in each PE. And what is necessary is just to lessen the number of times of transmission as much as possible by transmitting to a package as much as possible. That is, the structure of a program assigns the contents of use of an efficient array to the flow chart which shows the flow of the calculation centering on the variable (direction) of domain decomposition. Because it will usually be necessary to take many work arrays in a parallel computing program, when deciding the structure of a program, it is simultaneously necessary to make quantity of a work array into the minimum.

Most fundamental structures of three kinds of the Fortran program have not changed so that it may prove that the three-dimensional MHD code VPP Fortran of concrete vector parallel computing (pearthb), HPF/JA (hearthb), and MPI (meartb) is seen. The portion of the great portion of program can be rewritten by inserting VPP Fortran, HPF/JA, and the parallelization directions line of MPI, without hardly changing the structure of a program. If a portion peculiar to each compiler is added to it as a unit, the great portion of program will be done. The problems left behind behind will be a boundary condition and input and output, if it says roughly. In the usual case, these do not pose so serious a problem, either. However, in quite the target case to large-sized calculation in MPI cautions are required for a boundary condition and input and output. It is safer to think that the method usually written on the introductory writing and explanatory of MPI is hardly accepted although it depends on a case.

3.1. Production of MHD code by MPI (Example)

The Yukiya Aoyama work "parallel programming key　MPI version" is one of those were explained intelligibly and concretely about how to use MPI (Message Passing Interface). In it, although Mr. Aoyama has declared "what is carried without the necessary minimum method in order to cancel the inconsistency (side effects) accompanying parallelization" about the message switching subroutine, I completely think that it is philosophical view. If this is concretely written to me, it will be "Before calculating, it bundles up as much as possible, and the number of times of transmission is lessened as much as possible, and all variables required for calculation are calculated by bringing

them together in each PE (Processing Element)."

Then, the important portion of production of the MHD code by MPI will be seen concretely. There are two fundamental Fortran programs in the directory mearthb of a MPI version. Both are for these schools.

mearthb_send.f : Blocking communication mpi_send and mpi_recv are used.

mearthb.isend.f : The improved version using non-blocking communication mpi_isend and mpi_irecv.

Because the three-dimensional MHD code is transplanted to MPI from HPF/JA, the directions line of HPF/JA shown by !hpf$ remains as it is, but in MPI, they are all treated as a comment line. And, all the portions changed by the transplant to MPI are pinched in the comment line of CC MPI START and CC MPI END. mearthb_send.f is explained to below. As a numerical computation method, the Modified leap-frog method is used and domain decomposition is carried in the direction of k (z). The parameter of calculation of a program etc. is omitted here, because a setup of the calculation parameter of next 4.1 explains in detail. The number of PE(Processing Element) is npe=2, and isize is PE number, irank is a number of a rank (PE), and it is set to isize=npe=2, irank=0, and 1 in this case. ks and ke show the usual initial value and usual end value of k in irank. The relation between local k=k_local of each irank and global k=k_global is given by k_global=k_local+kss. Therefore, correspondence of the global variable of the direction of k (z) and the local variable after parallelization is as follows.

```
k=1,nz2     ->  k=ks,ke
k=1,nz2-1   ->  k=ks,ke1
k=2,nz2-1   ->  k=ks1,ke1
```

And, recvcount and displs show the size and the head address of the data of each rank when gathering.

```
CC MPI START
      include 'mpif.h'
      integer istatus(mpi_status_size)
      common /para_info/ks,ks1,ke,ke1,kss,irank,isize
c for mpi_gatherv
      parameter (npe=2)
      integer recvcount(npe),displs(npe)
CC MPI END
```

There are two kinds of division methods of the direction of one dimension often used. By the division method 1, ko of the same number enters to a certain rank, and the rank after it is also the way ko-1

of the same number enters. The division method 2 takes ranks containing ko of the same number as much as possible, the rank after it is the method of reducing one by one. Here, the division method 1 is used. In this case, the same number which enters by a certain rank is set to ko=nzz=(nz2-1)/npe+1. It is nz2=nz+2 here including the boundary of both sides. Because one edge left for applying paste is required for the both sides divided in the direction of k (z) respectively, the array range of the direction of k (z) required for each PE is given by k= (0:nzz+1) and just to take a size with nzz+2. And, fg (nx2, ny2, nz2) is a work array used for Files read and write.

```
CC MPI START
      parameter(nzz=(nz2-1)/npe+1)
      dimension f(nx2,ny2,0:nzz+1,nb),u(nx2,ny2,0:nzz+1,nb),
     1          ff(nx2,ny2,0:nzz+1,nb),p(nx2,ny2,0:nzz+1,nbb),
     2          pp(nx2,ny2,0:nzz+1,3)
c for all_gather
      dimension fg(nx2,ny2,nz2)
CC MPI END
```

The concrete parameter using the division method 1 is given in the following portion. By this calculation, the value of ks, ke, kss, recvcount (npe), and displs (npe) to isize and irank is decided. mpi_gather is used for calculation of recvcount (npe). This portion is applicable to other programs as it is, and when it can understand it, it means that it had understood the main portions of the division method of the direction of one dimension of MPI.

```
CC MPI START
      call mpi_init(ier)
      call mpi_comm_rank(mpi_comm_world,irank,ier)
      call mpi_comm_size(mpi_comm_world,isize,ier)
c
      kk=nz2/isize
      kmod=mod(nz2,isize)
c
      ks=1
      kss=irank*kk+min(kmod,irank)
c
      if (irank.lt.kmod) kk=kk+1
      ke=ks+kk-1
```

```
          ks1=ks
          ke1=ke
          if (irank.eq.0) ks1=2
          if (irank.eq.isize-1) ke1=ke-1
c
          nword=(ke-ks+1)*nx2*ny2
          call mpi_gather(nword,1,mpi_integer,recvcount,
     *                         1,mpi_integer,0,mpi_comm_world,ier)
          displs(1)=0
          do i=2,isize
            displs(i)=displs(i-1)+recvcount(i-1)
          end do
c
CC MPI END
```

The following two are simple, it has declared performing read of data, write, input-output of a file by irank=0.

```
CC MPI START
          if (irank.eq.0) then
          open(11,file='./school/mearthb/meart01.data',
     1                    access='sequential',form='unformatted')
          end if
CC MPI END

CC MPI START
          if (irank.eq.0)
     * write (6,12) iii,last,nx,ny,nz,n1,n2,n3,n4,n5,n6,eat0,rmu0,aru,
     1 eud,rrat,hx,hy,hz,t,t1,ro01,pr01,gra,dx2,dy2,dz2,dx4,dy4,dz4,
     2 bis,(cp(i),i=1,11),(cj(j),j=1,10)
CC MPI END
```

And, because domain decomposition is carried out in the direction of k (z), the global variable k= 1 of the direction of k (z) and nz2 must be changed into certainly local variable k=ks and ke.

```
CC MPI START
```

```
      do 22 k=ks,ke
CC MPI END
```

In the following portion, using blocking communication, the data of ks of irank is sent to irank-1 by mpi_send, and the data of an edge left for applying paste is sent to PE of a rank one younger by transmitting the data to ke+1 of irank from irank+1. mpi_barrier is for taking a synchronization.

```
CC MPI START
      do m=1,nb
        len=nx2*ny2
        if (irank.gt.0) then
          call mpi_send(f(1,1,ks,m),n2,mpi_real,irank-1,
     &                  100,mpi_comm_world,ier)
        end if
        if (irank.lt.isize-1) then
          call mpi_recv(f(1,1,ke+1,m),n2,mpi_real,irank+1,
     &                  100,mpi_comm_world,istatus,ier)
        end if
      end do
      call mpi_barrier(mpi_comm_world,ier)
CC MPI END
```

Next, all the data was brought together in irank=0 by mpi_gatherv, and is written out to the file by irank=0. It is necessary to take a synchronization by mpi_barrier before the beginning at this time.

```
      do 173 m=1,nb
CC MPI START
c     do m=1,nb
      call mpi_barrier(mpi_comm_world,ier)
        call mpi_gatherv(f(1,1,1,m),nword,mpi_real,fg(1,1,1),
     *                   recvcount,displs,mpi_real,0,
     *                   mpi_comm_world,ier)
c     end do
      call mpi_barrier(mpi_comm_world,ier)
CC MPI END
CC MPI START
```

```
      if(irank.eq.0) then
CC MPI END
      do 1732 k=1,nz2
        write(ntap) fg(1:nx2,1:ny2,k)
 1732   continue
CC MPI START
      end if
CC MPI END
  173   continue
```

And, when continuing a MHD simulation, it is judged whether numerical instability arose by always acting as a monitor of the maximum of the absolute value of speed. If 1 is exceeded with the value which vmax standardized, the data at that time will be written out and calculation will be made to be ended. mpi_allreduce is used for calculation of the vmax.

```
CC MPI START
      call mpi_allreduce(vmax,vmax1,1,mpi_real,mpi_max,
     *                   mpi_comm_world,ier)
      vmax=vmax1
CC MPI END
```

As seen above, when the program of HPF/JA is rewritten to MPI, if the addition and replacement of the template (module) inserted by CC MPI START and CC MPI END are done, it can rewrite almost as it is. Rewriting to MPI from VPP Fortran is also almost the same. In this way, most three-dimensional MHD codes rewritten by MPI may think that vectorization and parallelization are carried out. Of course, it may arise that a new device is sometimes required of the portion of a boundary condition. And, it is even if the period until middle-scale is efficiently calculable, in the case of quite large-scale calculation (over about 100 million lattice points are used by three-dimensional MHD), the further cautions and consideration are required.

3.2. Supplementary explanation about MPI use

By MPI use, it explains supplementarily about Production of the high efficiency program of parallel computing, how to treat a boundary condition, and read and write of a file.

(1) A wild card and package transmission and reception

How to transmit and receive the data of ks of irank-1 collectively by mpi_sendrecv to ke+1 of

irank+1 is shown by using a wild card and blocking communication. Since the use of a wild card can issue the special work produced in processing of the both ends of domain decomposition out of do loop, calculation efficiency is improved considerably. And, it becomes in many cases earlier than the case where transmission and reception are separated using mpi_send and mpi-recv. Adoption of this method actually took out the maximum efficiency with all the used computers.

```
CC MPI START
      iright = irank + 1
      ileft  = irank - 1
      if(irank.eq.0) then
        ileft = MPI_PROC_NULL
      else if(irank.eq.isize-1) then
        iright = MPI_PROC_NULL
      end if
      do m=1,nb
      call mpi_sendrecv(f(1,1,ks,m),n2,mpi_real,ileft,100,
     &                  f(1,1,ke+1,m),n2,mpi_real,iright,100,
     &                  mpi_comm_world,istatus,ier)
      end do
CC MPI END
```

(2) Use of non-blocking communication

In a former paragraph, although blocking communication mpi_send and mpi_recv were used, if non-blocking communication mpi_isend and mpi_irecv are used, the calculation efficiency of a program will improve. Please have a look, because the directions for use are easy and the example of use is in mearthb.isend.f. In this case, because non-blocking communication is used, it is necessary to make a set mpi_wait which waits for completion of transmission and reception simultaneously with transmitting mpi_isend and a receiving mpi_irecv command, and to use it. However, when non-blocking communication is applied to a practical use code, to certainly check is required because the improvement on calculation efficiency cannot be obtained.

(3) Periodic boundary condition

A periodic boundary condition is transmitted to PE of the first number, and k=ks of irank=0 from PE of the last number, and k=ke-1 of irank=isize-1 as follows. Moreover, what is necessary is just to transmit to PE of the last number, and k=ke of irank=isize-1 from PE of the first number, and k=ks +1 of irank=0. As the concrete example, please refer to the program which solves a

three-dimensional wave equation by the Modified leap-frog method and mwave3.f.send of Directory mwave, and mwave3.f.isend as shown below.

```
CC MPI START
            if (irank.eq.isize-1) then
              call mpi_send(f(1,1,ke-1,m),n2,mpi_real,0,
     &                      110,mpi_comm_world,ier)
            elseif (irank.eq.0) then
              call mpi_recv(f(1,1,ks,m),n2,mpi_real,isize-1,
     &                      110,mpi_comm_world,istatus,ier)
            end if
c
            if (irank.eq.0) then
              call mpi_send(f(1,1,ks+1,m),n2,mpi_real,isize-1,
     &                      115,mpi_comm_world,ier)
            elseif (irank.eq.isize-1) then
              call mpi_recv(f(1,1,ke,m),n2,mpi_real,0,
     &                      115,mpi_comm_world,istatus,ier)
            end if
          call mpi_barrier(mpi_comm_world,ier)
CC MPI END
```

(4) How do dispel a special boundary condition by MPI?

Like the following example, when you rearrange the variable of the direction of k (z) of division conversely, please consider how the efficient program of MPI is made. It is referred to as nz2=nz+2 and nz3=nz+3 here. (Hint: Collect the minimum required variable into irank=0 by mpi_gatherv and rearrange. Then, method of distributing to each PE can be considered by mpi_scatterv.)

```
do k=1,nz2
f(2:nx1,1,k,1:nb) = f(2:nx1,2,-k+nz3,1:nb)
end do
```

(5) What do we do with file input-output of read and write?

In three-dimensional MHD code mearthb_send.f of the Earth's magnetosphere, the work array fg (nx2, ny2, nz2) of data input-output was prepared, data was altogether brought together in irank=0 by mpi_gatherv, and it has written out to the file by irank=0. This poses a big problem, when increasing an array and enlarging program size. First, because it is settled in restriction of the

buffer of transmission and reception or then all data is brought together in fg (nx2, ny2, nz2) of irank=0, there are memory restrictions of irank=0. In the usual calculation (the lattice point in the three-dimensional MHD code is less than about 100 million pieces), writing here does not pose a serious problem. However, when enlarging program size extremely, not using the work array fg, and bringing data together in irank=0 altogether also needs to stop. In this case, each name is distinguished from each rank (PE), and a file is written out separately, and it will edit summarizing each file later etc. and will use.

(6) The conclusion of the program creation of MPI

As seen to this, I think that the program creation of MPI is easy, and think that there is no mistake. If it is also the program efficiently written by VPP Fortran or HPF/JA, it is still more so. And in many cases, full vectorization and full parallelization will also be attained easily. Supposing a problem arises, it sometimes generates in a boundary condition. And, efficient-ization of a MPI program is related also to the feature of a parallel computer and function which are used. Therefore, probably, it is good to ask a question and consult for program counselors such as a center, when a problem does not arise or efficient-ization does not come out. In parallel to it, It is a very realistic and effective method that the researcher using a MPI program also exhibits and shares the use knowledge of MPI. We prepare the next Homepage (http://center.stelab.nagoya-u.ac.jp/kaken/kakenhi.html) for this purpose, and the plan is made in order to share the knowledge of the parallel computing which each researcher obtained.

3.3. Efficiency of parallel computing method

In order to perform large-sized simulations, such as a three-dimensional MHD simulation, use of a supercomputer and the increase in efficiency of the calculation speed by vectorization or parallelization are indispensable. Comparison of the speed when carrying out the three-dimensional MHD code used by a lecture and training by SUN and VPP-5000 (VP Fortran, VPP Fortran, HPF/JA, MPI) is shown in Table 1. Computation time (sec) shows the time taken to advance a time step once by the Modified leap-frog method. Compared with SUN (GR720), it proves that about 70 times as many calculation speed as this has all come out by VPP5000 (2PE). In the actual three-dimensional MHD simulation of a Solar wind Earth's magnetosphere interaction, because about 10,000 repetition calculation is carried out, it proves that it takes about 20 hours to calculate in SUN (GR720), and about 20 minutes in VPP5000 (2PE, MPI). If PE is increased, the difference will become still larger further.

Table 1. Comparison of computer processing capability of 3-dimensional global MHD code with a quarter volume: earthb with (nx,ny,nz)=(180,60,60).

Computer Processing Capability
A Quarter Model of the Earth's Magnetosphere (nx,ny,nz)=(180,60,60); earthb

computer	number of PEs	compiler	sec	(MFLOPS)	GF/PE	(date)
Fujitsu GR720	(1PE)	Fortran 90 (frt)	7.72998	(136)	0.14	(2002.08.01)
Fujitsu VPP-5000	(1PE)	VP Fortran	0.19342	(5,428)	5.43	(2002.08.01)
Fujitsu VPP-5000	(2PE)	VPP Fortran	0.10509	(9,990)	5.00	(2002.08.01)
Fujitsu VPP-5000	(2PE)	HPF/JA	0.11064	(9,489)	4.74	(2002.08.01)
Fujitsu VPP-5000	(2PE)	MPI	0.09899	(10,606)	5.30	(2002.08.01)
Fujitsu VPP-5000	(2PE)	MPI (isend)	0.09774	(10,797)	5.40	(2002.08.08)

---

frt: Fujitsu VPP Fortran 90 HPF: High Performance Fortran
MPI; Massage Passing Interface
: MFLOPS is an estimated value in comparison with the computation by
1 processor of CRAY Y-MP C90.

In order to show the validity of a parallel computing method, comparison of the calculation speed using Fujitsu VPP5000/64 in the three-dimensional MHD code of the Solar wind Earth's magnetosphere interaction written by VPP Fortran, HPF/JA and MPI is shown. Equivalent calculation speed has come out by VPP Fortran, HPF/JA, and MPI, and it proves that the calculation efficiency is also quite high. These parallel computing MHD codes have performed full vectorization and full parallelization, and high efficiency beyond a 400 Gflops grade is also realized in large-scale calculation of VPP Fortran and HPF/JA. In this way, I regard as the ability to understand that the supercomputer of the maximum scale has the calculation speed of about 1000 times over to the fastest thing of a workstation or PC.

When an array becomes large like (800x200x478, 800x200x670) and the scale of calculation becomes large, MPI has not still necessarily realized sufficient efficient calculation speed in the usual operation mode. In Table 2, although high efficiency is acquired as for the MPI Fortran job, this shows the test result in the single mode. The reason high efficiency is not necessarily acquired by the usual operation mode has some causes, such as the problem of the MPI Fortran code, the problem of matching with vector parallel-processing supercomputer VPP5000, and the problem on which only computation time measurement of MPI differs from others, and it is under investigation. If a problem clarifies, above-mentioned Homepage etc. is due to show. Although the total of cpu hour

of use is used for computation time measurement in VPP Fortran and HPF/JA, the lapsed time of the job is only used in MPI, because there is no method of figuring out the total of cpu hour of use. Therefore, in measurement of MPI, if the multiplex job is performed simultaneously, it will come out that much late. Anyway, because MPI has been just started to use, there are many problems which should be solved. Since MPI is the last common parallel computing method, it thinks that sharing of the use knowledge of MPI is important also for the problem solving.

Table 2. Comparison of computer processing capability between VPP Fortran and HPF/JA and MPI in a 3-dimensional global MHD code of the solar wind-magnetosphere interaction by using Fujitsu VPP5000/64.

Number of PE	Number of grids	VPP Fortran cpu time Gflops Gf/PE	HPF/JA cpu time Gflops Gf/PE	MPI cpu time Gflops Gf/PE
1PE	200x100x478	119.607 (0.17) 0.17 (scalar)		
1PE	200x100x478	2.967 (6.88) 6.88	3.002 (6.80) 6.80	
2PE	200x100x478	1.458 (14.01) 7.00	1.535 (13.30) 6.65	1.444 (14.14) 7.07
4PE	200x100x478	0.721 (28.32) 7.08	0.761 (26.85) 6.71	0.714 (28.60) 7.15
8PE	200x100x478	0.365 (55.89) 6.99	0.386 (52.92) 6.62	0.361 (56.55) 7.07
16PE	200x100x478	0.205 (99.38) 6.21	0.219 (93.39) 5.84	0.191 (107.19) 6.70
24PE	200x100x478	0.141 (144.49) 6.02	0.143 (143.02) 5.96	0.1302(157.24) 6.55
32PE	200x100x478	0.107 (191.23) 5.98	0.110 (186.13) 5.82	0.1011(202.50) 6.33
48PE	200x100x478	0.069 (297.96) 6.21	0.074 (276.96) 5.77	0.0679(301.51) 6.28
56PE	200x100x478	0.064 (319.53) 5.71	0.068 (299.27) 5.34	0.0639(320.39) 5.72
64PE	200x100x478	0.0662(308.91) 4.83	0.0627(324.57) 5.07	0.0569(359.80) 5.62
1PE	500x100x200	2.691 (7.94) 7.94	2.691 (7.94) 7.94	
2PE	500x100x200	1.381 (15.47) 7.73	1.390 (15.37) 7.68	1.355 (15.77) 7.89
4PE	500x100x200	0.715 (29.97) 7.47	0.712 (29.99) 7.50	0.688 (31.03) 7.76
8PE	500x100x200	0.398 (53.65) 6.71	0.393 (54.38) 6.80	0.372 (57.50) 7.19
16PE	500x100x200	0.210 (101.87) 6.37	0.202 (105.74) 6.61	0.193 (110.70) 6.92
24PE	500x100x200	0.160 (133.70) 5.57	0.150 (142.40) 5.93	0.135 (158.26) 6.59
32PE	500x100x200	0.131 (163.55) 5.11	0.120 (175.50) 5.48	0.1084(197.10) 6.15
48PE	500x100x200	0.100 (214.48) 4.46	0.091 (231.69) 4.82	0.0811(263.44) 5.49
56PE	500x100x200	0.089 (239.48) 4.28	0.086 (244.85) 4.37	0.0688(310.54) 5.55
64PE	500x100x200	0.0956(222.95) 3.48	0.0844(249.49) 3.90	0.0687(310.99) 4.86

2PE	800x200x478	10.659 (15.33) 7.66	10.742 (15.21) 7.60	10.428 (15.67) 7.83
4PE	800x200x478	5.351 (30.53) 7.63	5.354 (30.52) 7.63	5.223 (31.28) 7.82
8PE	800x200x478	2.738 (59.67) 7.46	2.730 (59.85) 7.48	2.696 (60.61) 7.58
12PE	800x200x478	1.865 (87.58) 7.30	1.911 (85.49) 7.12	1.771 (92.25) 7.68
16PE	800x200x478	1.419 (115.12) 7.19	1.389 (117.66) 7.35	1.342 (121.81) 7.61
24PE	800x200x478	0.975 (167.54) 6.98	0.976 (167.45) 6.98	0.905 (180.59) 7.52
32PE	800x200x478	0.722 (226.33) 7.07	0.717 (227.72) 7.12	0.690 (236.63) 7.39
48PE	800x200x478	0.534 (305.70) 6.36	0.515 (317.26) 6.61	0.469 (348.38) 7.25
56PE	800x200x478	0.494 (330.95) 5.91	0.464 (352.49) 6.29	0.433 (377.73) 7.74
64PE	800x200x478	0.465 (351.59) 5.49	0.438 (373.41) 5.83	0.389 (420.45) 6.57
4PE	800x200x670	7.618 (30.06) 7.52	8.001 (28.62) 7.16	7.433 (30.81) 7.70
8PE	800x200x670	3.794 (60.36) 7.54	3.962 (57.81) 7.23	3.683 (62.17) 7.77
12PE	800x200x670	2.806 (81.61) 6.80	3.005 (76.21) 6.35	2.696 (84.95) 7.08
16PE	800x200x670	1.924 (119.00) 7.44	2.012 (113.85) 7.12	1.854 (123.53) 7.72
24PE	800x200x670	1.308 (175.10) 7.30	1.360 (168.44) 7.02	1.254 (182.61) 7.60
32PE	800x200x670	0.979 (233.85) 7.31	1.032 (221.88) 6.93	0.955 (239.77) 7.49
48PE	800x200x670	0.682 (335.62) 6.99	0.721 (317.80) 6.62	0.662 (346.21) 7.21
56PE	800x200x670	0.595 (384.61) 6.87	0.628 (364.87) 6.52	0.572 (400.59) 7.15
16PE	1000x500x1118	9.668 (123.52) 7.72	9.619 (125.50) 7.84	
32PE	1000x500x1118	5.044 (236.73) 7.40	4.992 (241.83) 7.56	
48PE	1000x500x1118	3.550 (336.40) 7.01	3.479 (346.97) 7.23	
56PE	1000x500x1118	2.985 (400.04) 7.14	2.935 (411.36) 7.35	
32PE	1000x1000x1118	9.979 (239.33) 7.48	9.813 (243.37) 7.61	
48PE	1000x1000x1118	7.177 (332.79) 6.93	7.028 (339.85) 7.08	
56PE	1000x1000x1118	5.817 (410.55) 7.33	5.794 (412.23) 7.36	

: Mflops is an estimated value in comparison with the computation by
1 processor of CRAY Y-MP C90.

4. Execution of Three-Dimensional MHD Code of Solar Wind-Magnetosphere Interaction

The parameter used in three-dimensional global MHD simulation code of the Solar wind of 1/4 domain and an Earth's magnetosphere interaction used by computer training explains. And the calculation execution method and example of the parallel computing three-dimensional MHD code

of three versions whichi are the memory saving type vectorization code calculated by normal computers, such as a workstation, VPP Fortran, HPF/JA, and MPI are shown. Moreover, the graphics output of a simulation result using a PostScript file, and the methods and those examples of the three-dimensional visualization using VRML (Virtual Reality Modeling Language) are shown.

4.1. Setup of calculation parameter

A setup of the parameter currently used in vectorized three-dimensional MHD code earthb (earthb10.f) is summarized to the next. The contents of a setting of the parameter in the vector parallelization three-dimensional MHD code are the same.

main program : earthb10.f

earthb10.f　using modified leap-frog scheme

3D MHD simulation of 1/4 earth's magnetosphere

Cartesian coordinate　finite resistivity　45 degree boundary

(nx,ny,nz)=(180,60,60)　: grid number without boundary

nxp=30　: parameter to determine earth position

last=1024　: number of time steps

iiq0=8　: a unit of modified leap-frog scheme

iip0= 32　: adjust upstream boundary condition

iis0= 1024　: sampling step of data

thx=4.00　: parameter to adjust time step

(xl,yl,zl)=(90.5,30.5,30.5)Re: length in each direction

hx=xl/float(nx+1)=0.5Re　: grid interval in x direction

hy=yl/float(ny+1)=0.5Re　: grid interval in y direction

hz=zl/float(nz+1)=0.5Re　: grid interval in z direction

t=0.5*hx*thx　: time interval

t(real)=t*ts　: real time to one time step advance

=0.5*0.5*4.00*0.937　: ts is normalization value in time

=0.937 sec

x=0.5*hx*float(2*i-nx2-1+2*nxp) : x position versus grid number

y=0.5*hy*float(2*j-3)　: y position versus grid number

z=0.5*hz*float(2*k-3)　: z position versus grid number

where nx2=nx+2, ny2=ny+2 and nz2=nz+2

ro01=5.0E-4 (5/cc) : mass density of solar wind
pr01=3.56E-8 : pressure of solar wind
vsw=0.044 (300km/s) : speed of solar wind
bis=CP(11)=1.5E-4 (5nT) : amplitude of IMF

eatt : resistivity
rmuu : viscosity
eud0 : friction or collision term

1-dimensional array variable f(i1)=f(i,j,k,m)

n1=nx+2,n2=n1*(ny+2),n3=n2*(nz+2)
nb=8,nbb=11,n4=n3*nb,n5=n3*nbb

i1=i+n1*(j-1)+n2*(k-1)+n3*(m-1)

m=1 : rho, plasma density
m=2 : Vx, x-component of velocity
m=3 : Vy, y-component of velocity
m=4 : Vz, z-component of velocity
m=5 : P, plasma pressure
m=6 : Bx, x-component of magnetic field
m=7 : By, y-component of magnetic field
m=8 : Bz, z-component of magnetic field

4.2. Example of calculation execution

The example of calculation execution in the vectorized three-dimensional MHD code earthb (earthb10.f), vector parallelization three-dimensional MHD code, MPI (mearthb), HPF/JA (hearthb), and VPP Fortran (peartb) are shown below. The shell of compile and execution is placed into each directory by execution of the vector parallelization three-dimensional MHD code, and the example of the execution command is written to the readme file.

4.2.1. <<execution of main program>>

1. f77 -O earthb10.f
2. a.out &

where file must be defined in open statement like

```
c          open(10,file='earthb10.data',
c       1             access='sequential',form='unformatted')
           open(11,file='earthb11.data',
        1             access='sequential',form='unformatted')
c
```

or

1. f77 -o earthb10 -O earthb10.f
2. earthb10 &

4.2.2. <<compile and execution using by supercomputer, Fujitsu VPP5000>>

(1) MPI (Massage Passing Interface): mearthb

All the comand shells are in "readme" file.

(1a) TSS

mpifrt progmpi.f :compile to make execution file, a.out

jobexec -vp 2 ~/school/mearthb/a.out :execution of a.out by 2 PEs

(1b) Batch

qsub -q c -eo -o pconpmpi2.out pcompmpi2.sh :compile

qsub mpi_lim02e.sh :execution of progmpi by 2 PEs

(2) HPF/JA (High Performance Fortran): hearthb

qsub -q c -eo -o pconphpf2.out pcomphpf2.sh :compile

qsub -q z -eo -lPv 2 -o pexechpf.out pexechpf.sh :execution by 2 PEs

qsub -q c -eo -o comp.out comp.sh :compile vector mode only

qsub -q x -eo -o exec.out exec.sh :execution by 1 PE

(3) VPP Fortran (Fortran 90): pearthb

qsub -q c -eo -o pcomp90.out pcomp90.sh :compile

qsub -q z -eo -lPv 2 -o pexec90.out pexec90.sh :execution by 2 PEs

4.3.Graphics processing

In order to carry graphics processing systematically, the following three conditions need to be fulfilled.

1. Establishment of Method independent of Kind of Computer
2. Everything, such as software, are controlled by themselves.
3. Establishment of the methods of using as systematically (in common) as possible, such as a program.

It will be said that the software, the language and specification depending on a computer are not used and the graphics processing application software which only a specific contractor sells is not used. The conclusion which is the effective method for image processing and systematic use of a graphics output that it was making a PostScript graphics file directly using Fortran etc. was reached. The systematic method of image processing of the computer simulation which we are carrying now is described as an item below.

(1) Save simulation data in IEEE Binary form.
(2) Make a PostScript graphics file directly by a Fortran program.
Make interface Subroutine Package for making a PostScript file.
(3) Make the graphics files (gif etc.) compressed with file conversion tools (xv, pstogif, etc.) from the PostScript file.
(4) Exhibit compression graphics files (gif etc.) by WWW.

When Fortran could be used and the capital letter and the small letter could be distinguished in it by this method, it became possible to make a PostScript graphics file, and to take out a graphics output without being based on the kind of computer. And, Interface Subroutine Package for the C language is also prepared so that the C language may also be made.

http://gedas.stelab.nagoya-u.ac.jp/simulation/jst2k/hpf02.html

4.3.1. Graphics program to make PostScript files

1. gm150b.f (main) + gsub150.f (subroutine)
noon-midnight meridian and equatorial plots (black and white)
2. gm220b.f (main) + gsub220.f (subroutine)
energy distribution of cross section
3. gm480b.f (main) + gsub480.f (subroutine)
3-dimensional magnetic field lines

<<execution of PostScript graphics program>>

1. f77 -c -O gsub150.f
2. f77 -O gm150b.f gsub150.o
3. a.out > gm150b.ps &
4. gs gm150b.ps
5. lp gm150b.ps

1. f77 -c -O gsub220.f
2. f77 -O gm220b.f gsub220.o
3. a.out > gm220b.ps &

1. f77 -c -O gsub480b.f
2. f77 -O gm480b.f gsub480b.o
3. a.out & : output is written in fort.10
4. mv fort-10 gm480b.ps

The figures showing the structure of the Earth's magnetosphere obtained from the three-dimensional global MHD simulation of a Solar wind and an Earth's magnetosphere interaction are shown below.

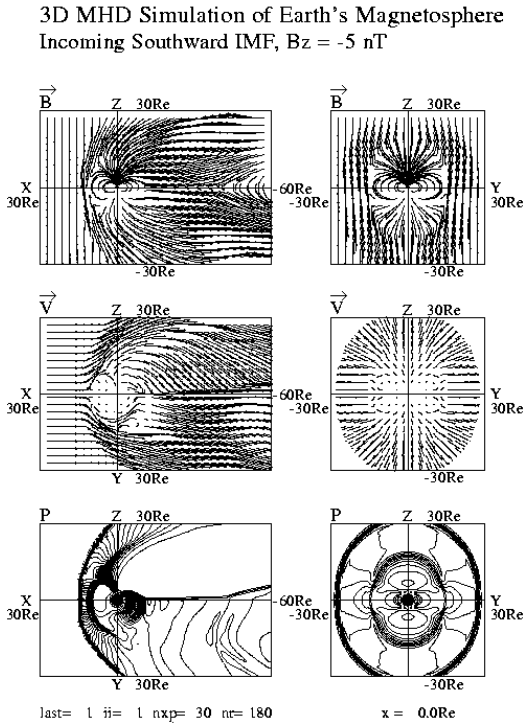

Fig.7 Magnetospheric configurations in the noon-midnight meridian and equator and cross sectional patterns (black and white:gm150b.ps).

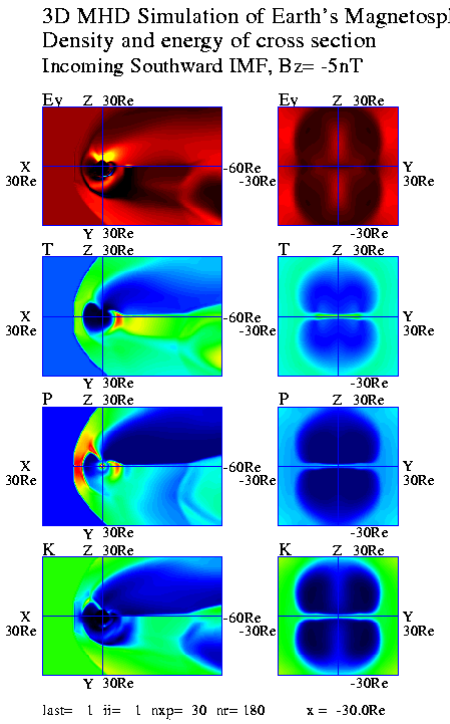

Fig.8 Magnetospheric configurations in the noon-midnight meridian and equator and cross sectional patterns (color:gm220b.ps).

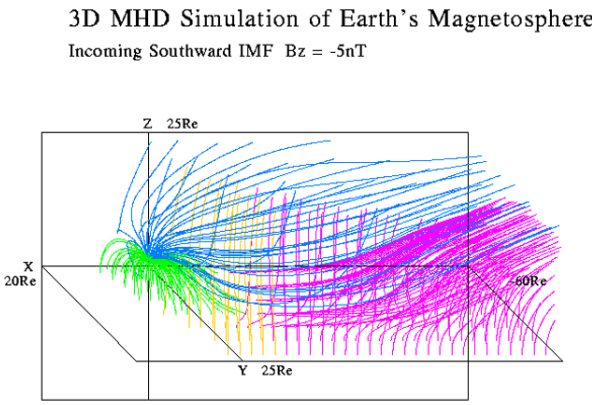

Fig.9 3-dimensional configuration of magnetic field lines in the earth's magnetosphere (gm480b.ps).

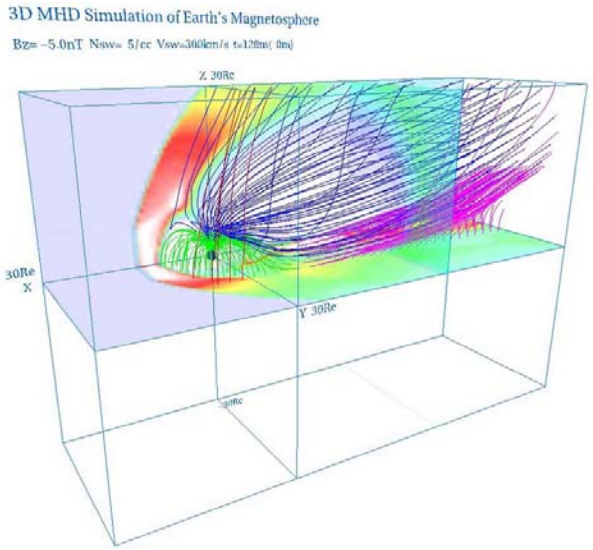

Fig.10 3-dimensional visualization of the earth's magnetosphere by using VRML (zvrml01.wrl).

5. Three-Dimensional Visualization by VRML

By the appearance of VRML (Virtual Reality Modeling Language), when there was even a viewer of VRML, everyone was able to come to look at a three-dimensional image. Although it depends for the speed of three-dimensional image processing (rotation, expansion reduction, etc.) on the throughput of its own computer, Three-dimensional visualization can be realized always anywhere with viewers, such as cosmo player for VRML2.0 by Netscape and Internet Explorer, etc., How is production of a VRML file realized? We prepare Fortran Interface Subroutine Package for VRML file production, and are making the VRML file (*. wrl) from three-dimensional simulation data directly using the FORTRAN program. Although there is a difference between three-dimensional and two-dimensional, this is the same as the method of making a PostScript graphics file. The viewer of VRML has walk mode which moves a viewpoint and examine mode which carries out movement/rotation/expansion reduction of the subject. It is very effective in seeing the relation of detailed structures, such as a Magnetic reconnection.

5.1. Subroutine package using Fortran program

ftp://gedas.stelab.nagoya-u.ac.jp/sramp/simulation/vrml/

VRML (Virtual Reality Modeling Language) and PostScript Fortran programs

1. vrml

   3-dimensional visualization Fortran program by using VRML

2. PostScript

   Fortran test program to make PostScript graphic files

3. PostScript2

   Fortran test program to make PostScript graphic files with subroutine

5.2. Application for Three-Dimensional MHD Simulation of Earth's Magnetosphere

ftp://gedas.stelab.nagoya-u.ac.jp/sramp/simulation/earthb/

3-dimensional graphics program by VRML files

<Virtual Reality Modeling Language>

1. zvrmagb.f (main) + zvrsubb.f (subroutine)
   3-dimensional magnetic field lines
2. zvrcrob.f (main) + zvrsubb.f (subroutine)
   cross sectional pattern by pixel image

<<execution of VRML graphics program>>

1. f77 -c -O zvrsubb.f
2. f77 -O zvrmagb.f zvrsubb.o
3. a.out & : output is written in fort.10
4. mv fort.10 fort.102

1. f77 -c -O zvrsubb.f
5. f77 -O zvrcrob.f zvrsubb.o
6. a.out & : output is written in fort.10
7. mv fort.10 fort.101
8. cat fort.101 fort.102 > zvrml01.wrl

The example which carried out three-dimensional visualization of the Earth's magnetosphere structure acquired in the simulation using VRML is shown in Fig. 10.

6. Concluding Summary

To the time of vector computers, such as FUJITSU VP-2600, Hitachi S820, NEC SX-3, and CRAY Y-MP, the three-dimensional global MHD simulation of a Solar wind and an Earth's magnetosphere interaction was able to be carried out using all the computers which carry a Fortran compiler using

the three-dimensional MHD code which full vectorization. By the flexibility of this Fortran program, we were able to move the three-dimensional MHD code among the world anywhere. And we were able to carry the Collaborative Research with many researchers in the world through distribution of the MHD code etc. However, as soon as a vector parallel machine and a super parallel machine appear in a computer simulation community, in order to raise the efficiency of parallelization of a Fortran program, you have to take various different techniques depending on a computer. Many simulation researchers lost common program language and had to use the Fortran program language of a dialect. The efficiency of parallelization was able to be raised only from specific maker's model.

HPF (High Performance Fortran) and MPI (Message Passing Interface) are parallel computing common program language which overcomes such a blockade situation. The calculation of the supercomputer in the code written by HPF can hardly expect high efficiency in Japan and the United States. The improved version HPF/JA (improved version of HPF by which development improvement was carried out in Japan) can realize high efficiency. But it is also limited to the supercomputer of FUJITSU and NEC in Japan, and cannot acquire high efficiency by the machine of Hitachi now. In this way, the expectation for MPI as a common parallel computing method was still larger. The concrete method of the three-dimensional MHD code production using MPI was exemplified under such a situation. Then it has been shown concretely that the efficient calculation more than VPP Fortran and HPF/JA is realizable. MPI will be expected as a method widely used for parallel computing by many kinds of computers including image processing etc. in the future.

Visualization is indispensable, in order to understand the simulation result of a Solar wind and an Earth's magnetosphere interaction etc. and to let you understand it better further. And, production of an animation animation and three-dimensional visualization/three-dimensional image analysis demonstrate very powerful power. The complicated behavior can be made quite obvious by an animation, and three-dimensional image analysis can be further taken now immediately to anyone by the appearance of an Internet three-dimensional language and VRML. That is, it can be said that sharing of three-dimensional visualization contents can be realized now through a network. The electromagnetism fluid code using the common computer language of HPF or MPI which can carry out parallel computing of the space plasma phenomenon efficiently, a particle code, and a hybrid code are made and spread using the domestic new generation parallel-processing supercomputer which is proud of the performance of a world maximum high speed. And bringing new knowledge to the nonlinear physics of Solar wind Magnetosphere Ionosphere dynamics or space plasma from the large scale simulation prior to the world and the large scale simulation which cooperated those codes is expected.

Acknowledgments

The computer simulation of this paper is made with the supercomputer Fujitsu VPP5000/64 of Information Technology Center of Nagoya University. And, at rewriting to HPF/JA and MPI from VPP Fortran, I appreciate the direction of the Tomoko Tsuda assistant in the Information Technology Center of Nagoya University which obtained many advice, and FUJITSU, LTD. Moreover, at rewriting of the prototype to MPI, I am thankful to the Masaki Okada assistants of the National Institute of Polar Research which obtained care instruction, and Hitachi.

References

[1] T. Ogino, A three-dimensional MHD simulation of the interaction of the solar wind with the earth's magnetosphere: The generation of field-aligned currents, J. Geophys. Res., 91, 6791-6806 (1986).

[2] T. Ogino, R.J. Walker and M. Ashour-Abdalla, A global magnetohydrodynamic simulation of the magnetosheath and magnetopause when the interplanetary magnetic field is northward, IEEE Transactions on Plasma Science, Vol.20, No.6, 817-828 (1992).

[3] T. Ogino, Two-Dimensional MHD Code, (in Computer Space Plasma Physics), Ed. by H. Matsumoto and Y. Omura, Terra Scientific Publishing Company, 161-215, 411-467 (1993).

[4] T. Ogino, R.J. Walker and M. Ashour-Abdalla, A global magnetohydrodynamic simulation of the response of the magnetosphere to a northward turning of the interplanetary magnetic field, J. Geophys. Res., Vol.99, No.A6, 11,027-11,042 (1994).

[5] T. Ogino, Electromagnetism hydrodynamic simulation of a Solar wind and a Magnetosphere interaction, The Japan Society of Plasma Science and Nuclear Fusion Research, CD-ROM special plan (description dissertation) Vol.75, No.5, CD-ROM 20-30, 1999. http://gedas.stelab.nagoya-u.ac.jp/simulation/mhd3d01/mhd3d.html

[6] T Ogino, Computer simulation of a Solar wind Magnetosphere interaction, Nagoya University mainframe center news, Vol.28, No.4, 280-291, 1997

[7] T Ogino, Computer simulation and visualization, The Ehime University synthesis information processing center public relations, Vol.6, 4-15, 1999. http://gedas.stelab.nagoya-u.ac.jp/simulation/simua/ehime985.html

[8] T Ogino, HPF from VPP Fortran, Nagoya University mainframe center news description, 372-405, Vol.31, No.4, (2000). http://gedas.stelab.nagoya-u.ac.jp/simulation/hpfja/hpf013.html

[9] T. Ogino, Global MHD Simulation Code for the Earth's Magnetosphere Using HPF/JA, Special Issues of Concurrency: Practice and Experience, 14, 631-646, 2002.

http://gedas.stelab.nagoya-u.ac.jp/simulation/hpfja/mhd00.html

[10] Tomoko Tsuda, Use of new supercomputer VPP5000/56, Nagoya University mainframe center news, Vol.31, No.1, 18-33, 2000.

[11] Tomoko Tsuda, Parallelization by MPI, Information Technology Center of Nagoya University, MPI school data, February, 2002

[12] High Performance Fortran Forum, 「High Performance Fortran 2.0 Official manual」、Springer, 1999.

[13] FUJITSU LTD., HPF programming - parallel programming (volume on HPF), XP/V the 1.0th version, June, 2000 .

[14] FUJITSU LTD., VPP FORTRAN programming, UXP/V, the 1.2nd edition of UXP/M, April, 1997.

[15] FUJITSU LTD., MPI programming - Fortran and C-, the 1.3rd edition and April, 2001.

[16] FUJITSU LTD., The object for MPI use guidance document V20, UXP/V First edition, September, 1999

[17] Yukiya Aoyama (IBM Japan Corp.), Parallel programming key MPI version, key series 2, January, 2001.

[18] Hitachi Ltd., Super technical server SR8000 program transplant report, National Institute of Polar Research, March, 2002.

[19] Information Technology Center of Nagoya University, Guidance of supercomputer VPP5000 use, April, 2002.
The pdf file of guidance of VPP5000 use : URL
http://nucc.cc.nagoya-u.ac.jp/CENT/vpp_tebiki.pdf
Supercomputer VPP5000/64 use guidance : URL
http://www2.itc.nagoya-u.ac.jp/sys_riyou/vpp/vpptebiki.htm