JAEA-Data/Code
2010-025

テラヘルツ電磁波発振シミュレーション・プログラムの並列処理

Parallel Processing for the Simulation of Electromagnetic Terahertz Wave Emission

樋口 健二　平塚 篤　圓戸 辰郎　太田 幸宏
町田 昌彦

Kenji HIGUCHI, Atsushi HIRATSUKA, Tatsuro ENDO
Yukihiro OTA and Masahiko MACHIDA

システム計算科学センター

Center for Computational Science & e-Systems

February 2011

Japan Atomic Energy Agency

日本原子力研究開発機構

JAEA-Data/Code

独立行政法人日本原子力研究開発機構　研究技術情報部　研究技術情報課
〒319-1195　茨城県那珂郡東海村白方白根 2 番地 4
電話 029-282-6387, Fax 029-282-5920, E-mail:ird-support@jaea.go.jp

JAEA-Data/Code 2010-025

テラヘルツ電磁波発振シミュレーション・プログラムの並列処理

日本原子力研究開発機構　システム計算科学センター

樋口　健二、平塚　篤[*1]、圓戸　辰郎[※1]、太田　幸宏[※2]、町田　昌彦

(2010 年 11 月 16 日　受理)

1980 年代に出現したベクトル計算機に始まるスーパーコンピュータのアーキテクチャは、約 10 年間隔で変遷を遂げてきた。すなわち、1990 年代のベクトル並列計算機、2000 年前後のスカラ並列計算機、2010 年前後のマルチコア並列計算機である。システム計算科学センターは、1984 年にベクトル計算機 FIJITSU VP-100 を導入以来、常に最新のスーパーコンピュータを用いた原子力コードの高速化技術開発を進めることでこれらの変化に対応してきた。今回、テラヘルツ発振高温超伝導現象のシミュレーション・プログラムの開発においても、並列化を行うとともに、マルチコア・システム上でチューニングを行い、実効効率の高い並列処理を実現した。ここでは、高温超伝導現象の数値解析を高速に計算するため、シミュレーション空間を分割し、各空間を並列処理した。並列処理は、MPI(Message Passing Interface)通信によって実装し、複数プロセスにより各空間に対する計算を行った。当該プログラムについては、東京大学情報基盤センターの HA8000 クラスタシステム上で性能評価を行った。その結果、高温超伝導現象の数値解析問題のマルチコア・システム上での高速化を確認した。

システム計算科学センター（上野駐在）：〒110-0015　東京都台東区東上野 6-9-3
※1 技術開発協力員
※2 特定課題推進員
＊1 株式会社ケーシーエス

JAEA-Data/Code 2010-025

Parallel Processing for the Simulation of Electromagnetic Terahertz Wave Emission

Kenji HIGUCHI, Atsushi HIRATSUKA*1, Tatsuro ENDO※1, Yukihiro OTA※2 and Masahiko MACHIDA

Center for Computational Science & e-Systems
Japan Atomic Energy Agency
Higashiueno, Taito-ku, Tokyo

(Received November 16, 2010)

The change of supercomputer architecture has been made at intervals of a decade, i.e.; vector processor in nineteen eighties, vector parallel processor in nineties, scalar parallel processor in twenties and multi-core systems in twenty ten. To cope with this change, Center for Computational Science & e-Systems has been engaged continuously in the development of high performance computing techniques for nuclear codes on the latest supercomputers succeeding to the FUJITSU VP-100 introduced to JAEA in 1984. In this report we describe the parallelization and performance evaluation of the simulation program for emission of terahertz electromagnetic wave from high-Tc cuprate superconductors, where high performance was achieved as a result of tuning on multi-core system. In other words, the simulation space was divided for parallel processing, where MPI (Message Passing Interface) communication was implemented. In the evaluation of the parallelized program on HA8000 Cluster System at Information Technology Center of the University of Tokyo, we have confirmed high performance for the analysis of terahertz electromagnetic wave emission from superconductors.

Keywords: Supercomputer, Parallel Processing, Multi-Core Processor, Performance Evaluation, Superconductor, Terahertz Electromagnetic Wave

※1Collaborating Engineer
※2Special Topic Researcher
＊1KCS Corp.

目次

Contents

図リスト

表リスト

This is a blank page.

1. はじめに

1.1 概要

本報告は、テラヘルツ発振高温超伝導における数値シミュレーションを行うプログラムに対する並列化および並列処理についての報告である。

ある金属や化合物を超低温に冷却した際、急激に電気抵抗が下がる現象が起こる。この現象を超伝導と呼ぶ。特に、液体窒素の沸点 77 [K]以上で超伝導の現象を起こすことを高温超伝導と呼ぶ。また、絶縁体を超伝導素子で挟むことで、トンネル効果によって素子間を電流が流れる現象が起こるが、その現象をジョセフソン効果と呼ぶ。この状況で、磁場・定電流を与えることで、超伝導素子からテラヘルツ波が発振される [1]。この現象を数値シミュレーションすることで、発振素子の最適な設計、さらには素子開発の応用が可能となり、その成果に対して期待が寄せられている。

システム計算科学センターでは、松本秀樹教授（東北大学）によって開発されたテラヘルツ発振シミュレーション [2]の基本プログラムをベースに、大規模な数値シミュレーション・システムを構築した。ここでは、数値計算を高速化するため、プログラムを並列化し、実用化へ向けた開発を進めた。

本報告では、上記シミュレーションのための計算モデル、プログラムの並列化への指針、および並列化されたプログラムの挙動解析結果について述べる。また、プログラムの並列化を行うにあたって性能評価を行っており、その結果を示すとともに、並列化したプログラムのマルチコア・システム上における最適化手法についても報告する。

並列化を行ったプログラムの性能評価は、東京大学 HA8000 クラスタシステム（以降、HA8000）で行った。HA8000 は、マルチコア・プロセッサ（4 コア搭載）4 個（計 16 コア）を持つノード 952 台を、ネットワークで接続したクラスタ型計算機である。

1.2 HA8000 クラスタシステム

1.2.1 クラスタシステムの構成

HA8000 の概要について述べる。Table 1 に HA8000 の仕様を示す。また、Fig. 1 にシステム全体構成図を示す。HA8000 は、ノード数 952 台を持つクラスタシステムである [3]。各クラスタで計算を並列に行う。ノード間は、高速なネットワークによって構成され、MPI 通信により同期計算が可能である。

1.2.2 ノードの構成

クラスタシステムを形成するノードの仕様を Fig. 2 に示す。HA8000 を構成する各ノードは、AMD 社製の AMD Quad-Core Opteron™ 8356 プロセッサ 4 個で構成される [4]。AMD Quad-Core Opteron™ プロセッサは、4 コアのマルチコアプロセッサであり、1 ノードは合計 16 コアで構成される。プロセッサの構成を Table 2 に示す。

各ノード間は Myrinet-10G インターコネクトで接続されている。Myrinet-10G は双方向同時通信を可能とし、1 リンク 1 方向あたり 1.25 [GB/s]のバンド幅を持つ。HA8000 は、1 ノード当たり Myrinet-10G を 2 本持つタイプ A 群(5 [GB/s]接続)と、1 ノード当たり Myrinet-10G を 2 本もつタ

イプ B 群(2.5 [GB/s]接続)を持つ。

AMD Quad-Core Opteron™プロセッサの各コアは、1 クロックで 4 回の浮動小数点計算ができ、1 コア当たりの演算性能は 2.3 [GHz] × 4 = 9.2 [Gflops]である。1 ノード当たり 16 コアを搭載しているため、1 ノードの理論演算性能は 9.2 [Gflops] × 16 = 147.2 [Gflops]となる。

ノードのメインメモリは、8 [GB]×4（各プロセッサごとに 8 [GB]）の NUMA(Non-Uniform Memory Access) 構成をとる。すなわち、ノード内のすべてのプロセッサは、ノード内のすべてのメモリにアクセス可能であるが、アクセスするメモリの距離による性能差が発生する。Fig. 3 にその構成図を示す。当然ながら、各コアは、CPU に直結するメインメモリへのアクセスが、最も高速である。

1.2.3 AMD Quad-Core Opteron™ プロセッサのキャッシュメモリ

AMD Quad-Core Opteron™プロセッサは、レベル 1（L1）キャッシュメモリから、レベル 3（L3）キャッシュメモリまでの 3 段階のキャッシュメモリを持つ。L1 および L2 キャッシュは、各コア ごとに搭載しているが、L3 キャッシュは 4 コアで共有している。Table 3 にキャッシュメモリの仕様を示す。また、Fig. 4 にキャッシュメモリの構成を示す。

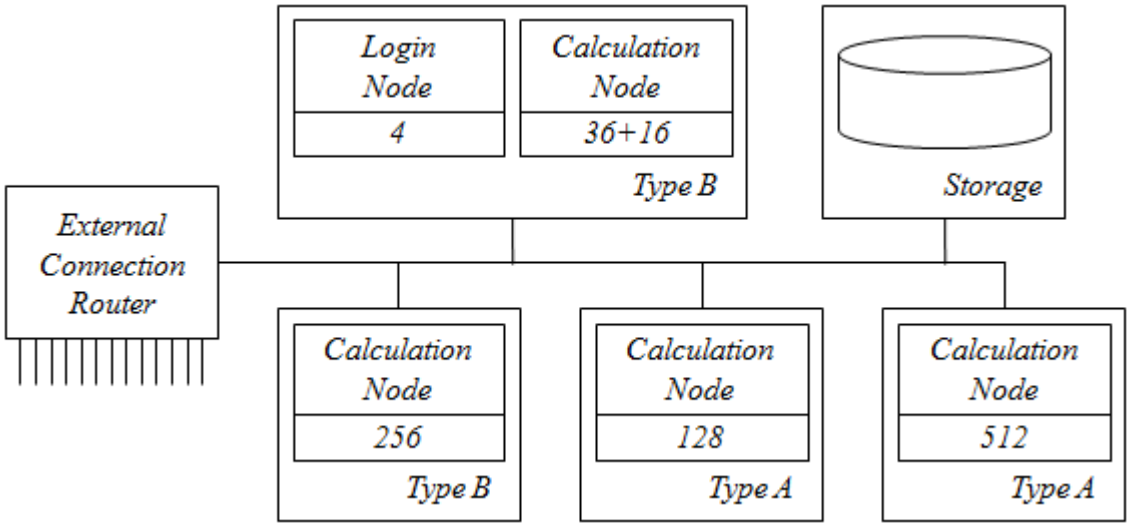


Fig. 1 HA8000 クラスタシステム

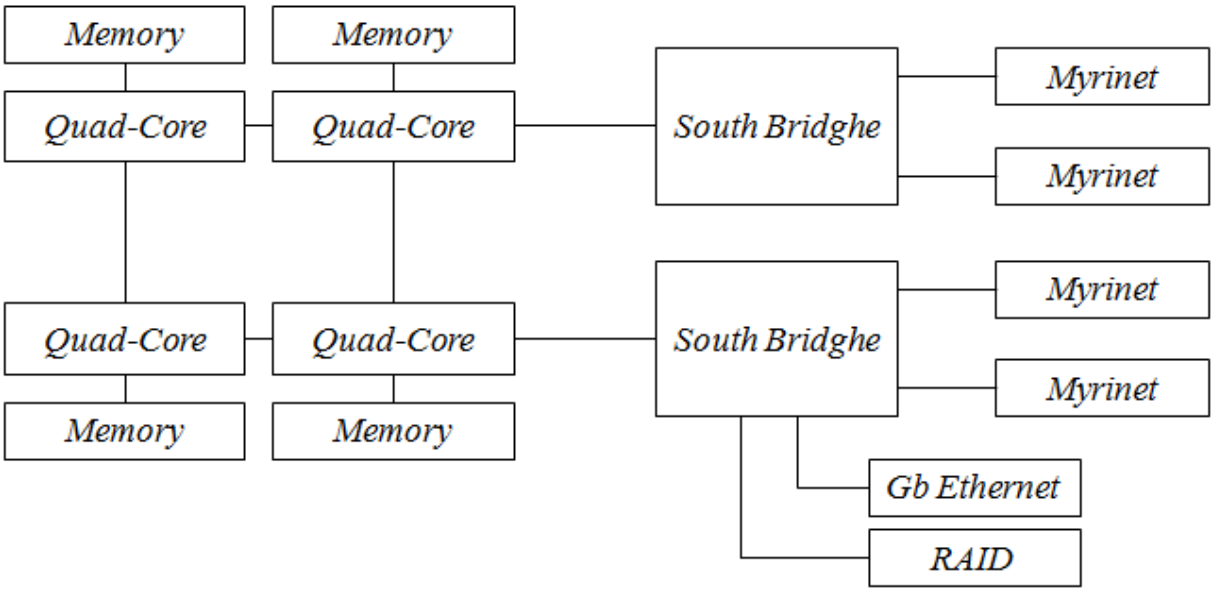


Fig. 2 ノード構成（Type A）

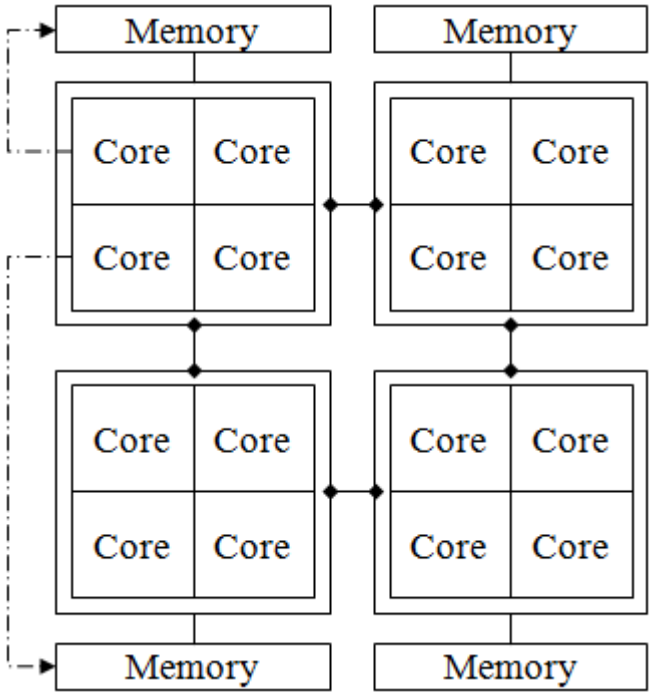


Fig. 3 NUMA

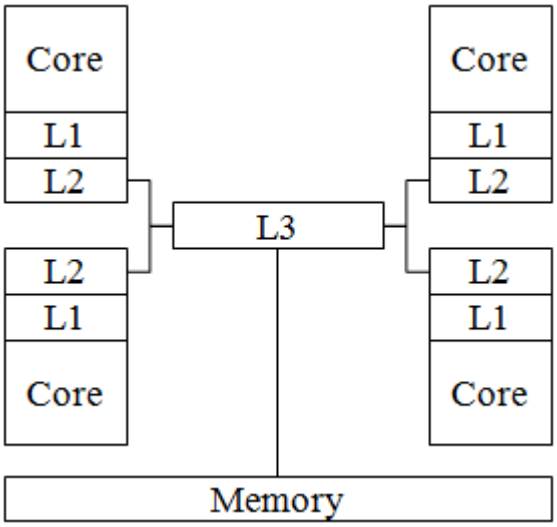


Fig. 4 AMD Quad-Core Opteron™ Processor のメモリ構成

Table 1 HA8000 クラスタシステムの仕様

Item	Specification
Theoretical Performance	140.1344 [Tflops]
Memory	31.25 [TB]
Number of Nodes	512 + 128 (Type A) 256 + 56 (Type B) 952 (All)
Network Performance	5 [GB/s] : Bidirectional (Type A) 2.5 [GB/s] : Bidirectional (Type B)
Storage Capacity	1 [PB](RAID6)

Table 2 HA8000 クラスタシステムのノードの仕様

Node / Processor	Item	Specification
Node	Theoretical Performance	9.2 [GHz] × 16 = 147.2 [Gflops]
	Number of Processor	4 Processor 16 Core
	Memory	8 [GB] × 4 = 32 [GB]
Processor	Operating Frequency	AMD Quad Core Opteron™ 8386 2.3 [GHz]
	Theoretical Performance	2.3 [GHz] × 4 = 9.2 [Gflops]

Table 3 AMD Quad-Core Opteron™ Processor のキャッシュメモリの仕様

Cache Level	Size
L1	128 [KB/Core]
Data Instruction	32 [KB] 32 [KB]
L2	512[KB/Core]
L3	2 [MB]

2. 計算モデルと数値解法

2.1 物理モデル

超伝導金属絶縁体システムは、真空領域、金属および絶縁体領域から成る。金属および絶縁体領域に与えられた電場による真空領域への電磁波の伝播は、マクスウェルの方程式でモデル化した。また、金属および絶縁体領域を Junction 域と呼ぶ。それらの領域の外側を覆うように、*x* 方向境界は有限長の吸収壁を仮定し、*y* 方向、*z* 方向の境界は有限長の吸収壁、または周期条件を仮定した。

Fig. 5 に、モデル化された超伝導金属絶縁体を示す。

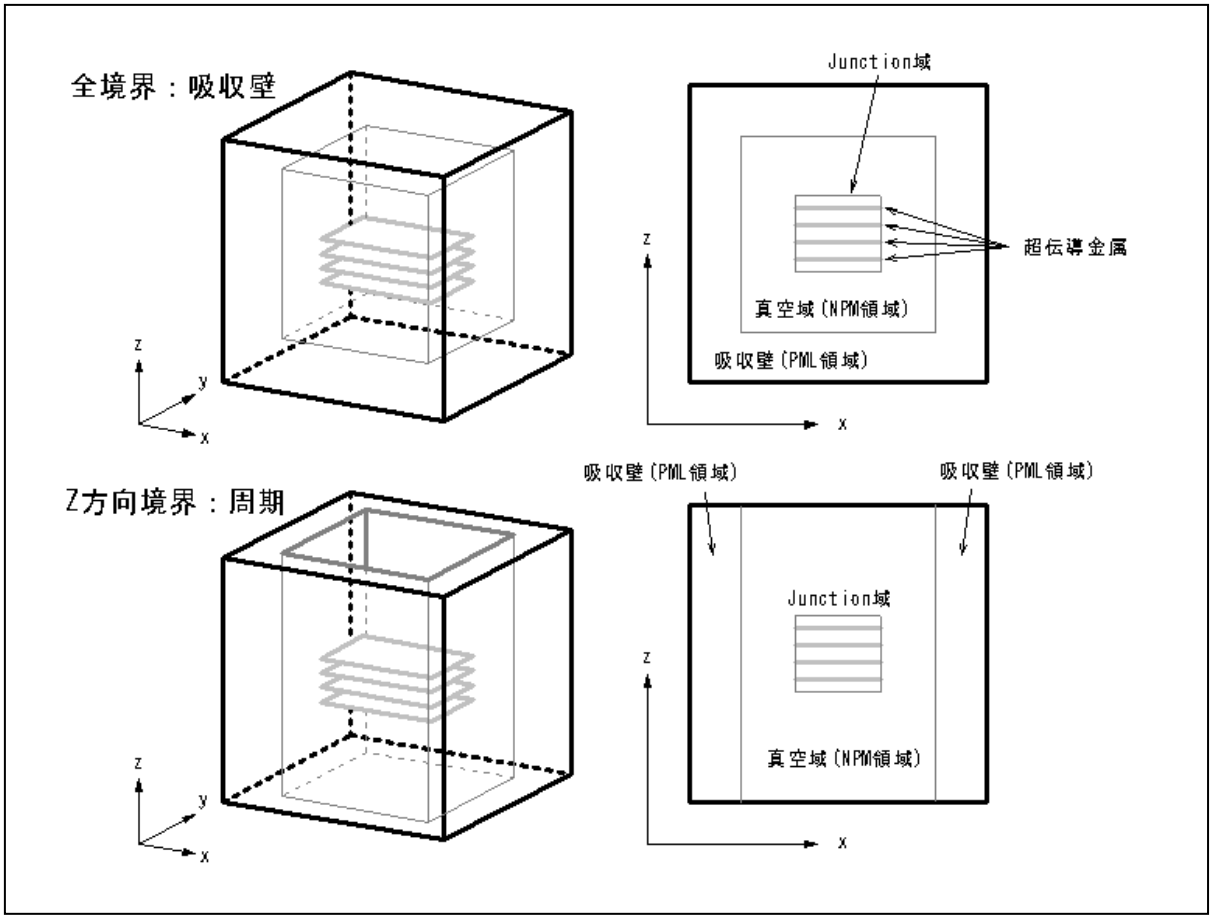


Fig. 5 テラヘルツ発振シミュレーションモデル

2.2 計算モデル

2.2.1 超伝導金属絶縁体

超伝導金属絶縁体システムをマクスウェルの方程式によりモデル化した。真空中の電場（電場の強度）を $E(t,x,y,z)$、磁場（磁束密度）を $B(t,x,y,z)$、電荷密度を ρ とすると、マクスウェルの方程式から次のようにモデル化できる。

$$\nabla \cdot B = 0 \tag{1}$$

$$\nabla \cdot D = \rho \tag{2}$$

$$\nabla \times E + \frac{\partial B}{\partial t} = 0 \tag{3}$$

$$\nabla \times B - \mu_0 \frac{\partial D}{\partial t} = \mu_0 J \tag{4}$$

ここで、μ_0 は真空中の透磁率、D は電束密度である。また、誘電率を ε とすると、電束密度は次のように表せる。

$$D = \varepsilon E \tag{5}$$

J は電流密度であり、初期値の z 成分を J_{z_0}、金属板間の初期の位相差を ϕ_0、動的な位相差を v_ϕ とし、次のように仮定する。

$$J = \begin{bmatrix} J_x \\ J_y \\ J_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ J_{z_0}(x,y,z) - \sin\left(v_\phi(t,x,y,z) + \phi_0(x,y)\right) \end{bmatrix} \tag{6}$$

演算子 ∇ は次のように定義する。

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \end{pmatrix} \tag{7}$$

2.2.2 吸収壁における電場と磁場

本システムでは、真空領域の外周に吸収壁を仮定している。この吸収壁では、電場と磁場の減衰を想定している。 xyz 軸の各方向の伝導率(あるいは減衰率)を σ、各方向の吸収壁境界からの距離を L とおき、次のように定義する。

$$\sigma(L)=\begin{cases}\sigma_{\max}\left(\dfrac{L_x}{W_x}\right)^m\\ \sigma_{\max}\left(\dfrac{L_y}{W_y}\right)^m\\ \sigma_{\max}\left(\dfrac{L_z}{W_z}\right)^m\end{cases} \tag{8}$$

W_x, W_y, W_z は吸収壁の厚さであり、$\sigma_{\max}, m$ は任意に与える。式(3)に減衰項を加え、x 成分を取り出すと、次のようになる。

$$\frac{\partial B_x}{\partial t}=-\left(\frac{\partial E_z}{\partial y}-\frac{\partial E_y}{\partial z}\right)-\left(\sigma_x^{(y)}B_x{}^{(y)}+\sigma_x^{(z)}B_x{}^{(z)}\right) \tag{9}$$

式(9)の右辺第 3 項が減衰項に対応する。$\sigma_x^{(y)}B_x{}^{(y)}$ は電場 y 方向変位に対する反射磁場、$\sigma_x^{(z)}B_x{}^{(z)}$ は電場 z 方向変位に対する反射磁場と解釈される。ここで、B_x を y 方向変位成分 $B_x{}^{(y)}$ と z 方向変位成分 $B_x{}^{(z)}$ とを用いて次のように分割する。

$$B_x(t,x,y,z)=B_x{}^{(y)}(t,x,y,z)+B_x{}^{(z)}(t,x,y,z) \tag{10}$$

このとき、吸収壁内での磁場は次式により決定される。

$$\begin{aligned}\frac{\partial}{\partial t}B_x(t,x,y,z)&=\frac{\partial}{\partial t}B_x{}^{(y)}(t,x,y,z)+\frac{\partial}{\partial t}B_x{}^{(z)}(t,x,y,z)\\ \frac{\partial B_x{}^{(y)}}{\partial t}&=-\frac{\partial E_z}{\partial y}-\sigma_x^{(y)}B_x{}^{(y)}\\ \frac{\partial B_x{}^{(z)}}{\partial t}&=\frac{\partial E_y}{\partial z}-\sigma_x^{(z)}B_x{}^{(z)}\end{aligned} \tag{11}$$

式(10)と(11)を組み合わせることで、式(9)が再現されることに注意せよ。こうして、得られた $B_x{}^{(y)}$ と $B_x{}^{(z)}$ とを使って、ある時間における吸収壁領域の B_x は次式のように定まる。

$$B_x(t,x,y,z)=B_x{}^{(y)}(t,x,y,z)+B_x{}^{(z)}(t,x,y,z) \tag{12}$$

磁場の他の成分 B_y, B_z についても同様に、以下のようになる。

$$\frac{\partial}{\partial t}B_y(t,x,y,z)=\frac{\partial}{\partial t}B_y{}^{(x)}(t,x,y,z)+\frac{\partial}{\partial t}B_y{}^{(z)}(t,x,y,z) \tag{13}$$

$$\frac{\partial}{\partial t}B_z(t,x,y,z)=\frac{\partial}{\partial t}B_z{}^{(x)}(t,x,y,z)+\frac{\partial}{\partial t}B_z{}^{(y)}(t,x,y,z) \tag{14}$$

また、電場Dについても同様に、次のように定義する。

$$\begin{aligned}
\frac{\partial}{\partial t}D_x(t,x,y,z)&=\frac{\partial}{\partial t}D_x{}^{(y)}(t,x,y,z)+\frac{\partial}{\partial t}D_x{}^{(z)}(t,x,y,z)\\
\frac{\partial}{\partial t}D_y(t,x,y,z)&=\frac{\partial}{\partial t}D_y{}^{(x)}(t,x,y,z)+\frac{\partial}{\partial t}D_y{}^{(z)}(t,x,y,z)\\
\frac{\partial}{\partial t}D_z(t,x,y,z)&=\frac{\partial}{\partial t}D_z{}^{(x)}(t,x,y,z)+\frac{\partial}{\partial t}D_z{}^{(y)}(t,x,y,z)
\end{aligned} \tag{15}$$

2.3 数値解法

2.3.1 方程式の解法

前節では、吸収壁を仮定した電場、磁場のモデル化を行った。吸収壁の領域では、減衰率が正値をとり、その内部の領域(Negative Permeability Meta-material 領域、以降 NPM 領域)では、減衰率がゼロをとると仮定する。電場、磁場の空間の微分は、3 次元直交座標系に等間隔にとった離散点の中心差分で近似する。近似された方程式を時間で積分する際、4 次のルンゲ・クッタ法を用いる。説明のため、一般的な方程式を以下のように定義する。

$$\begin{aligned}
y' &= f(t,y)\\
y(t_0) &= y_0\\
t_{n+1} &= t_n + h
\end{aligned} \tag{16}$$

この方程式に対して、4 次のルンゲ・クッタ法は、次のように方程式の解を求める。

$$\begin{aligned}
y_{n+1} &= y_n + h\sum_{k=1}^{4} a_k f_k\\
f_1 &= f(t_n, y_n)\\
f_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2} f_1)\\
f_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2} f_2)\\
f_4 &= f(t_n + h, y_n + hf_3)\\
a_1 &= \frac{1}{6}, a_2 = \frac{1}{3}, a_3 = \frac{1}{3}, a_4 = \frac{1}{6}
\end{aligned} \tag{17}$$

吸収壁の領域境界に対して、電場、磁場ともに連続条件を与える。この条件はx方向境界では次のようになる。

$$\frac{\partial E_y}{\partial x} = \frac{\partial E_z}{\partial x} = 0$$
$$\frac{\partial B_y}{\partial x} = \frac{\partial B_z}{\partial x} = 0 \tag{18}$$

y 方向境界も同様に、次のようになる。

$$\frac{\partial E_x}{\partial y} = \frac{\partial E_z}{\partial y} = 0$$
$$\frac{\partial B_x}{\partial y} = \frac{\partial B_z}{\partial y} = 0 \tag{19}$$

z 方向境界も同様である。

$$\frac{\partial E_x}{\partial z} = \frac{\partial E_y}{\partial z} = 0$$
$$\frac{\partial B_x}{\partial z} = \frac{\partial B_y}{\partial z} = 0 \tag{20}$$

2.3.2 中心差分による方程式の解法

微分方程式を解くにあたり、x 成分 $\boldsymbol{Bx}$ の時間変化式を考える。式(3)より

$$\frac{\partial B_x}{\partial t} = -\left(\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} \right) \tag{21}$$

であり、空間の微分項 $\frac{\partial E_z}{\partial y}, \frac{\partial E_y}{\partial z}$ を中心差分で近似すると、次のように記述できる。(Fig. 6 参照)

$$\frac{\partial}{\partial y} E_z(x,y,z) \approx \frac{1}{\Delta y}\left(E_z(x, y+\frac{\Delta y}{2}, z) - E_z(x, y-\frac{\Delta y}{2}, z) \right) \tag{22}$$
$$\frac{\partial}{\partial z} E_y(x,y,z) \approx \frac{1}{\Delta z}\left(E_y(x, y, z+\frac{\Delta z}{2}) - E_y(x, y, z-\frac{\Delta z}{2}) \right) \tag{23}$$

これらの式を 4 次のルンゲ・クッタ法で求解するが、$\boldsymbol{Bx}$ の時間変化式を解くために、$\boldsymbol{E_z}, \boldsymbol{E_y}$ に着目する。式(5)から $\boldsymbol{Ez}, \boldsymbol{Ey}$ は次のようになる。

$$E_z(t) = \frac{1}{\varepsilon} D_z(t) \tag{24}$$
$$E_y(t) = \frac{1}{\varepsilon} D_y(t) \tag{25}$$

ここで、$\boldsymbol{D}_z$ の時間変化式は式(4)から、次のようになる。

$$\frac{\partial D_z}{\partial t} = \frac{1}{\mu_0}\left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y}\right) - J_z \tag{26}$$

空間の微分項 $\frac{\partial B_y}{\partial x}, \frac{\partial B_x}{\partial y}$ は、次のように記述できる。(Fig. 7 参照)

$$\frac{\partial}{\partial x} B_y(x,y,z) \approx \frac{1}{\Delta x}\left(B_y(x+\frac{\Delta x}{2},y,z) - B_y(x-\frac{\Delta x}{2},y,z)\right) \tag{27}$$

$$\frac{\partial}{\partial y} B_x(x,y,z) \approx \frac{1}{\Delta y}\left(B_x(x,y+\frac{\Delta y}{2},z) - B_x(x,y-\frac{\Delta y}{2},z)\right) \tag{28}$$

同様に $\boldsymbol{D}_y$ の時間変化式は式(4)から、次のようになる。

$$\frac{\partial D_y}{\partial t} = \frac{1}{\mu_0}\left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x}\right) \tag{29}$$

この空間の微分項 $\frac{\partial B_x}{\partial z}, \frac{\partial B_z}{\partial x}$ についても同様に、次のようになる。（Fig. 8 参照）

$$\frac{\partial}{\partial z} B_x(x,y,z) \approx \frac{1}{\Delta z}\left(B_x(x,y,z+\frac{\Delta z}{2}) - B_x(x,y,z-\frac{\Delta z}{2})\right) \tag{30}$$

$$\frac{\partial}{\partial x} B_z(x,y,z) \approx \frac{1}{\Delta x}\left(B_z(x+\frac{\Delta x}{2},y,z) - B_z(x-\frac{\Delta x}{2},y,z)\right) \tag{31}$$

電場、磁場の他の成分についても同様のことが言える。各要素の離散点座標を Table 4 に示す。

2.3.3 セルのインデックス

数値解析のために方程式を離散化するが、3 次元の計算領域を離散化するにあたり、次のように定義する。離散化された計算領域内で、ある格子点 (x_i, y_j, z_k) から *xyz* 正方向にそれぞれ長さ $\Delta x, \Delta y, \Delta z$ を有する直方体をセルと呼び、その位置を格子点の位置と同様に (i, j, k) とする。また、格子点座標 (x_i, y_j, z_k) をセル格子点と呼ぶ。セル (i, j, k) の領域は

$$\begin{aligned} x_i &\le x < (x_i + \Delta x) \\ y_j &\le y < (y_j + \Delta y) \\ z_k &\le z < (z_k + \Delta z) \end{aligned} \tag{32}$$

である。セル領域内に存在する離散点を、セルと同じインデックスで表す。セル(i, j, k)にある各離散点(i, j, k)の位置を Fig. 9 に示す。

2.3.4 電場と磁場の差分方程式

Table 4 に従い、電場の差分方程式を求めると、次式のようになる。

$$\begin{aligned}
\frac{\partial}{\partial t} D_{x\,i,j,k}^{(y)} &= \frac{1}{\mu_0}\left(\frac{1}{\Delta y}(B_{z\,i,j+1,k} - B_{z\,i,j,k})\right) - \beta_{x\,i,j,k}^{(y)} \frac{D_{x\,i,j,k}^{(y)}}{\varepsilon_{i,j,k}} \\
\frac{\partial}{\partial t} D_{x\,i,j,k}^{(z)} &= \frac{1}{\mu_0}\left(-\frac{1}{\Delta z}(B_{y\,i,j,k+1} - B_{y\,i,j,k})\right) - \beta_{x\,i,j,k}^{(z)} \frac{D_{x\,i,j,k}^{(z)}}{\varepsilon_{i,j,k}}
\end{aligned} \tag{33}$$

$$\begin{aligned}
\frac{\partial}{\partial t} D_{y\,i,j,k}^{(x)} &= \frac{1}{\mu_0}\left(-\frac{1}{\Delta x}(B_{z\,i+1,j,k} - B_{z\,i,j,k})\right) - \beta_{y\,i,j,k}^{(x)} \frac{D_{y\,i,j,k}^{(x)}}{\varepsilon_{i,j,k}} \\
\frac{\partial}{\partial t} D_{y\,i,j,k}^{(z)} &= \frac{1}{\mu_0}\left(\frac{1}{\Delta z}(B_{xi\,,j,k+1} - B_{x\,i,j,k})\right) - \beta_{y\,i,j,k}^{(z)} \frac{D_{y\,i,j,k}^{(z)}}{\varepsilon_{i,j,k}}
\end{aligned} \tag{34}$$

$$\begin{aligned}
\frac{\partial}{\partial t} D_{z\,i,j,k}^{(x)} &= \frac{1}{\mu_0}\left(\frac{1}{\Delta x}(B_{y\,i+1,j,k} - B_{y\,i,j,k})\right) - \beta_{z\,i,j,k}^{(x)} \frac{D_{z\,i,j,k}^{(x)}}{\varepsilon_{i,j,k}} - \frac{1}{2} J_{z\,i,j,k} \\
\frac{\partial}{\partial t} D_{z\,i,j,k}^{(y)} &= \frac{1}{\mu_0}\left(-\frac{1}{\Delta y}(B_{x\,i,j+1,k} - B_{x\,i,j,k})\right) - \beta_{z\,i,j,k}^{(y)} \frac{D_{z\,i,j,k}^{(z)}}{\varepsilon_{i,j,k}} - \frac{1}{2} J_{z\,i,j,k}
\end{aligned} \tag{35}$$

βは減衰率である。各変数は、次のとおりである。

$$\begin{aligned}
D_{x\,i,j,k} &= D_{x\,i,j,k}^{(y)} + D_{x\,i,j,k}^{(z)} \\
D_{y\,i,j,k} &= D_{y\,i,j,k}^{(x)} + D_{y\,i,j,k}^{(z)} \\
D_{z\,i,j,k} &= D_{z\,i,j,k}^{(x)} + D_{z\,i,j,k}^{(y)}
\end{aligned} \tag{36}$$

$$E_{i,j,k} = \frac{D_{i,j,k}}{\varepsilon_{i,j,k}} \tag{37}$$

$$J = \begin{bmatrix} 0 \\ 0 \\ J_{z_{0i,j,k}} - \sin\left(\nu_{\phi\,i,j,k} + \phi_{0\,i,j}\right) \end{bmatrix} \tag{38}$$

磁場の差分方程式は、次のようになる。

$$\begin{aligned}
\frac{\partial}{\partial t}B_{x_{i,j,k}}^{(y)} &= -\frac{1}{\Delta y}(E_{z_{i,j,k}} - E_{z_{i,j-1,k}}) - \sigma_{x_{i,j,k}}^{(y)} B_{x_{i,j,k}}^{(y)} \\
\frac{\partial}{\partial t}B_{x_{i,j,k}}^{(z)} &= \frac{1}{\Delta z}(E_{y_{i,j,k}} - E_{y_{i,j,k-1}}) - \sigma_{x_{i,j,k}}^{(z)} B_{x_{i,j,k}}^{(z)}
\end{aligned} \tag{39}$$

$$\begin{aligned}
\frac{\partial}{\partial t}B_{y_{i,j,k}}^{(x)} &= \frac{1}{\Delta x}(E_{z_{i,j,k}} - E_{z_{i-1,j,k}}) - \sigma_{y_{i,j,k}}^{(x)} B_{y_{i,j,k}}^{(x)} \\
\frac{\partial}{\partial t}B_{y_{i,j,k}}^{(z)} &= -\frac{1}{\Delta z}(E_{x_{i,j,k}} - E_{x_{i,j,k-1}}) - \sigma_{y_{i,j,k}}^{(z)} B_{y_{i,j,k}}^{(z)}
\end{aligned} \tag{40}$$

$$\begin{aligned}
\frac{\partial}{\partial t}B_{z_{i,j,k}}^{(x)} &= -\frac{1}{\Delta x}(E_{y_{i,j,k}} - E_{y_{i-1,j,k}}) - \sigma_{z_{i,j,k}}^{(x)} B_{z_{i,j,k}}^{(x)} \\
\frac{\partial}{\partial t}B_{z_{i,j,k}}^{(y)} &= \frac{1}{\Delta y}(E_{x_{i,j,k}} - E_{x_{i,j-1,k}}) - \sigma_{z_{i,j,k}}^{(x)} B_{z_{i,j,k}}^{(x)}
\end{aligned} \tag{41}$$

$$\begin{aligned}
B_{x_{i,j,k}} &= B_{x_{i,j,k}}^{(y)} + B_{x_{i,j,k}}^{(z)} \\
B_{y_{i,j,k}} &= B_{y_{i,j,k}}^{(x)} + B_{y_{i,j,k}}^{(z)} \\
B_{z_{i,j,k}} &= B_{z_{i,j,k}}^{(x)} + B_{z_{i,j,k}}^{(y)}
\end{aligned} \tag{42}$$

2.3.5 位相差v_ϕの差分方程式

位相差$v_{\phi_{i,j,k}}$は、超伝導金属板上でのみ時間変化する。超伝導金属板上にある$v_{\phi_{i,j,k}}$の範囲をFig. 10に示す。超伝導金属板上にある$v_{\phi_{i,j,k}}$の範囲を、次にように定義する。

$$\begin{aligned}
-iNxe \le i \le iNx - 1 \\
-iNye \le j \le iNy - 1 \\
-iNze + 1 \le k \le iNz - 1
\end{aligned} \tag{43}$$

本システムでは、$v_{\phi_{i,j,k}}$の差分方程式を次のように仮定し、$v_{\phi_{i,j,k}}$の時間変化$\frac{\partial}{\partial t}v_{\phi_{i.j,k}}$を、次の条件で離散化する。

$k = -iNze + 1$のとき：

$$\frac{\partial}{\partial t}v_{\phi_{i.j,k}} = E_{z_{i,j,k}} - \alpha\left(\left(\frac{\varepsilon_L}{\varepsilon}E_{z_{i,j,k+1}} - E_{z_{i,j,k}}\right) - \left(E_{z_{i,j,k}} - E_{z_{i,j,k-1}}\right)\right) \tag{44}$$

$-iNze + 2 \le k \le iNz - 2$のとき：

$$\frac{\partial}{\partial t}v_{\phi_{i.j,k}} = E_{z_{i,j,k}} - \alpha\left(\left(E_{z_{i,j,k+1}} - E_{z_{i,j,k}}\right) - \left(E_{z_{i,j,k}} - E_{z_{i,j,k-1}}\right)\right) \tag{45}$$

$k = iNz - 1$のとき：

$$\frac{\partial}{\partial t} v_{\phi_{i.j,k}} = E_{z_{i,j,k}} - \alpha\left(\left(E_{z_{i,j,k+1}} - E_{z_{i,j,k}}\right) - \left(E_{z_{i,j,k}} - \frac{\varepsilon_L}{\varepsilon} E_{z_{i,j,k-1}}\right)\right) \tag{46}$$

また、空間微分について、同様に次のように仮定し、離散化する。

$$\begin{aligned} \frac{\partial}{\partial x} v_{\phi_{i.j,k}} &= B_{y_{i,j,k}} - \eta\left(\left(B_{y_{i,j,k+1}} - B_{y_{i,j,k}}\right) - \left(B_{y_{i,j,k}} - B_{y_{i,j,k-1}}\right)\right) \\ \frac{\partial}{\partial y} v_{\phi_{i.j,k}} &= B_{x_{i,j,k}} - \eta\left(\left(B_{x_{i,j,k+1}} - B_{x_{i,j,k}}\right) - \left(B_{x_{i,j,k}} - B_{x_{i,j,k-1}}\right)\right) \end{aligned} \tag{47}$$

ここで、α, η は入力で与えられる定数である。ここで、$\frac{\partial}{\partial t} v_{\phi_{i,j,k}}$ を解くにあたり、以降の表現の簡単化のため、次のような表現を用いる。まず、z 方向の離散点数 N として、次のように仮定する。

$$N = (iNz - 1) - (-iNze + 1) + 1 \tag{48}$$

離散化された位相差 $v_{\phi_{i,j,k}}$ をベクトル v_ϕ として、次のように定義する。

$$v_\phi = \begin{bmatrix} v_{\phi_{i,j,-iNze+1}} \\ \vdots \\ v_{\phi_{i,j,k}} \\ \vdots \\ v_{\phi_{i,j,iNx-1}} \end{bmatrix} \tag{49}$$

同様に、電場 E_z も次のように定義する。

$$E_z = \begin{bmatrix} E_{z_{i,j,-iNze+1}} \\ \vdots \\ E_{z_{i,j,k}} \\ \vdots \\ E_{z_{i,j,iNx-1}} \end{bmatrix} \tag{50}$$

このとき、位相差の微分方程式は、次のように表現できる。この方程式を解くことで位相差を求めることができる。

$$\frac{\partial v_\phi}{\partial t} = M_e E_z - \alpha \frac{\varepsilon_L}{\varepsilon} C_e \tag{51}$$

ただし、行列 M_e と C_e は次のとおりである。

$$M_e = \begin{bmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 \\ -\alpha & 1+2\alpha & -\alpha & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & -\alpha & 1+2\alpha & -\alpha \\ 0 & \cdots & 0 & -\alpha & 1+2\alpha \end{bmatrix} \tag{52}$$

$$C_e = \begin{bmatrix} E_{z_{i,j,-iNze}} \\ 0 \\ \vdots \\ 0 \\ E_{z_{i,j,iNz}} \end{bmatrix} \tag{53}$$

2.3.6 位相差v_ϕから求める Junction 域の磁場

Junction 域の磁場B_x, B_yは、前小節で与えられるv_ϕから中心差分により近似した$\frac{\partial v_\phi}{\partial x}, \frac{\partial v_\phi}{\partial y}$から、次のように得られる。

$$\begin{aligned} \frac{\partial}{\partial x} v_{\phi_{i.j,k}} &= B_{y_{i,j,k}} - \eta\left(\left(B_{y_{i,j,k+1}} - B_{y_{i,j,k}}\right) - \left(B_{y_{i,j,k}} - B_{y_{i,j,k-1}}\right)\right) \\ \frac{\partial}{\partial y} v_{\phi_{i.j,k}} &= B_{x_{i,j,k}} - \eta\left(\left(B_{x_{i,j,k+1}} - B_{x_{i,j,k}}\right) - \left(B_{x_{i,j,k}} - B_{x_{i,j,k-1}}\right)\right) \end{aligned} \tag{54}$$

Junction 域のB_zはゼロである。式(54)を解くにあたり、前小節と同様に表現する。Junction 域の *z* 方向離散点数を *N* として、次のように仮定する。

$$N = (iNz - 1) - (-iNze + 1) + 1 \tag{55}$$

このとき、磁場の *x* 成分と *y* 成分をそれぞれ、B_x、B_yとすると、次のように表現できる。

$$\begin{aligned} \frac{\partial v_\phi}{\partial t} &= M_b B_x - \eta C_{B_x} \\ \frac{\partial v_\phi}{\partial t} &= M_b B_y - \eta C_{B_y} \end{aligned} \tag{56}$$

ただし、行列 M_b、C_{Bx}、C_{By} は次のとおりである。

$$M_b = \begin{bmatrix} 1+2\eta & -\eta & 0 & \cdots & 0 \\ -\eta & 1+2\eta & -\eta & & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & & -\eta & 1+2\eta & -\eta \\ 0 & \cdots & 0 & -\eta & 1+2\eta \end{bmatrix} \tag{57}$$

$$C_{B_x} = \begin{bmatrix} B_{x_{i,j,-iNze}} \\ 0 \\ \vdots \\ 0 \\ B_{x_{i,j,iNz}} \end{bmatrix} \tag{58}$$

$$C_{B_y} = \begin{bmatrix} B_{y_{i,j,-iNze}} \\ 0 \\ \vdots 0 \\ 0 \\ B_{y_{i,j,iNz}} \end{bmatrix} \tag{59}$$

求めたいものは、磁場の B_x と B_y であるため、式(56)を次のように式変形する。

$$\begin{aligned} B_x &= M_b^{-1}\left(\frac{\partial \nu_\phi}{\partial t} + \eta C_{B_x}\right) \\ B_y &= M_b^{-1}\left(\frac{\partial \nu_\phi}{\partial t} + \eta C_{By}\right) \end{aligned} \tag{60}$$

式(60)から磁場を求めることができる。位相差 ν_ϕ から導出される B_x, B_y の離散点の定義を Fig. 11 に示す。

2.3.7 方程式の初期値問題

モデル化した方程式の初期値を、次のように設定する。

$$\begin{cases} E(t_0) = E_0 \\ B(t_0) = 0 \\ \nu_\phi(t_0) = \Phi_0 \end{cases} \tag{61}$$

この条件をもとに、*ntmax* ステップの時間発展計算を行い、方程式の解を求める。さらに、$(\Delta t \times t_{nt\max})$時間経過した場の値を、新たな初期値として設定する。

$$\begin{cases} E^{\text{iter=2}}(t_0) = E^{\text{iter=1}}(t_{nt\max}) \\ B^{\text{iter=2}}(t_0) = B^{\text{iter=1}}(t_{nt\max}) \\ v_\phi{}^{\text{iter=2}}(t_0) = v_\phi{}^{\text{iter=1}}(t_{nt\max}) \end{cases} \tag{62}$$

更新された初期値をもとに、*ntmax* ステップの計算を同様に行う。これを一巡とし、複数回実行して場の値の収束をみる。ここでの収束に至る繰り返し回数とステップ数は、任意に設定する。この計算により、バイアス電流の減少に伴う場の挙動を解析する。Fig. 12 に全体のループ構造を示す。また、ルンゲ・クッタ法による時間積分の処理を Fig. 13 に示す。

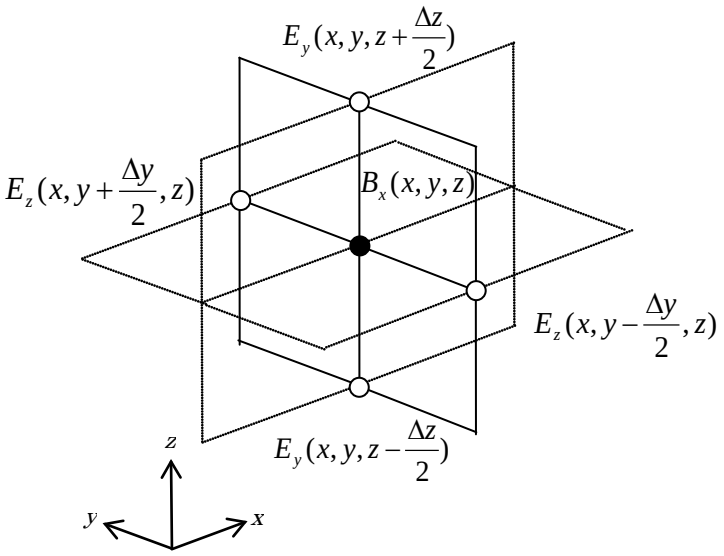


Fig. 6 *xyz* 平面における B_x, E_y, E_z

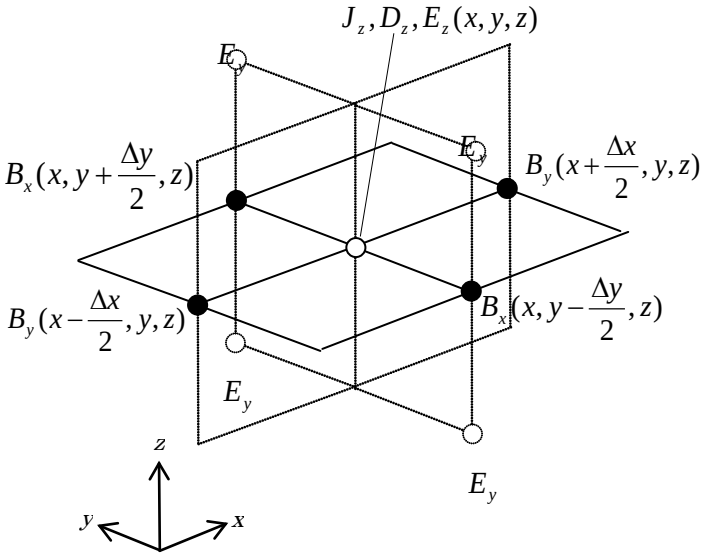


Fig. 7 *xy* 平面における E_z, B_x, B_y

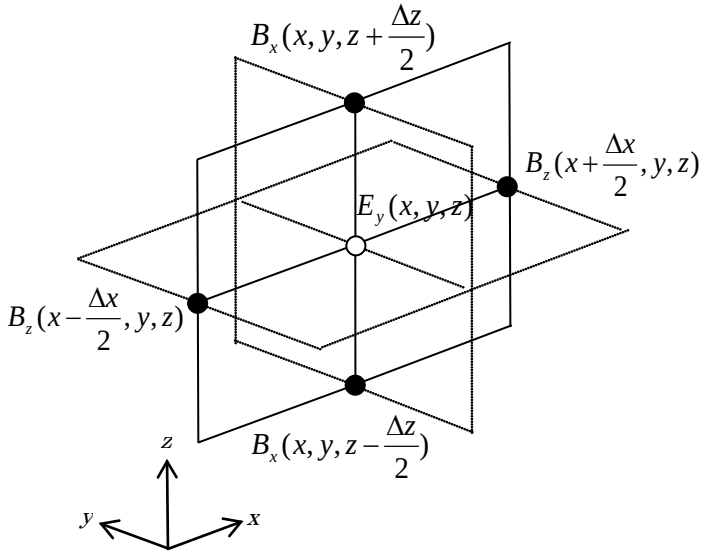


Fig. 8 *xz* 平面における E_y, B_x, B_z

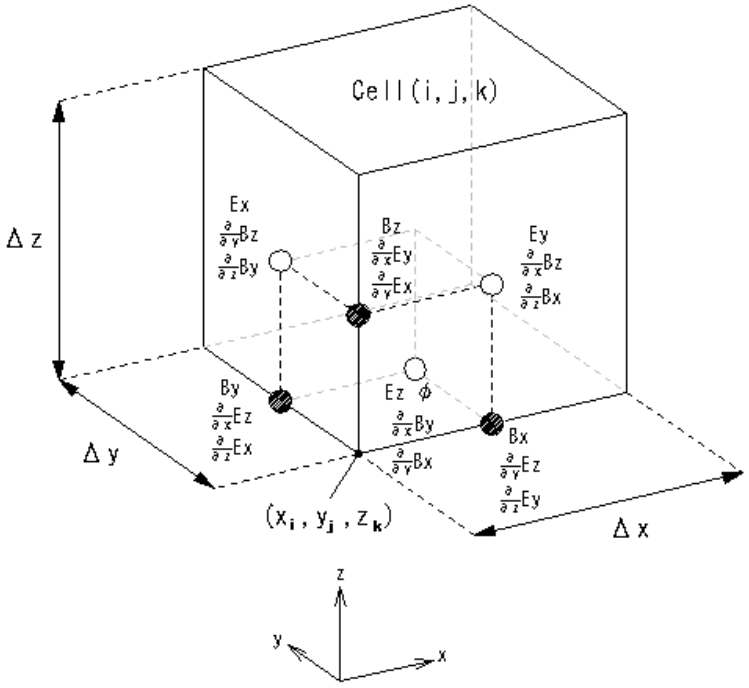


Fig. 9 セル格子 (i, j, k) 内の離散点 (x_i, y_j, z_k) の定義

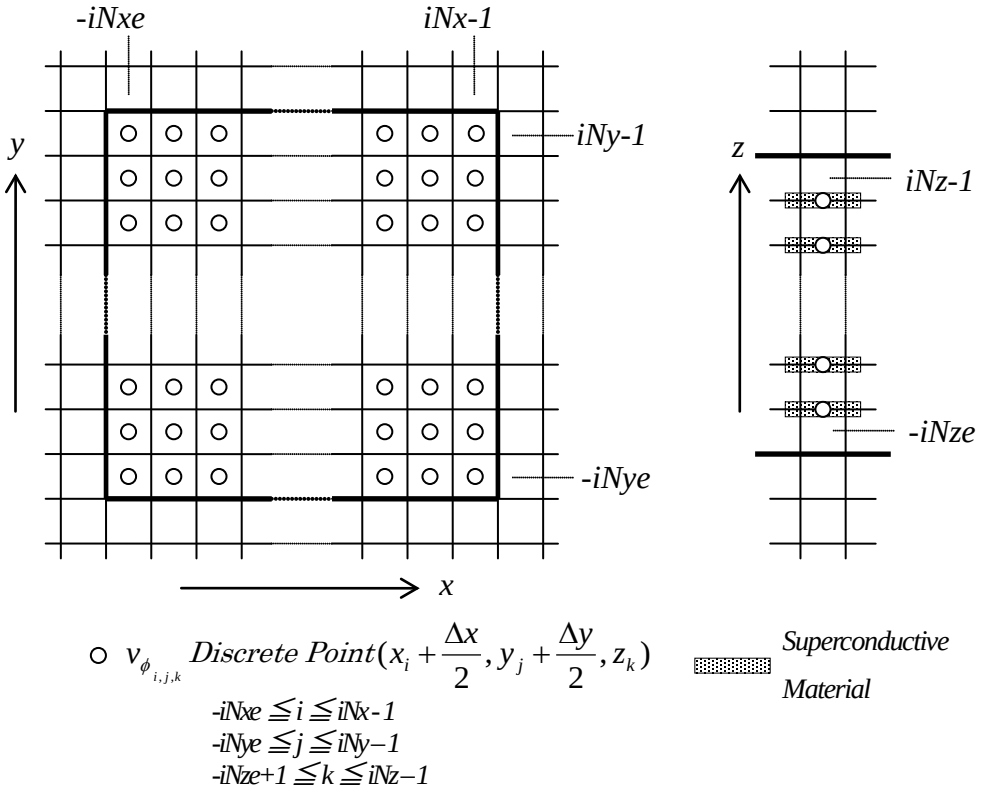
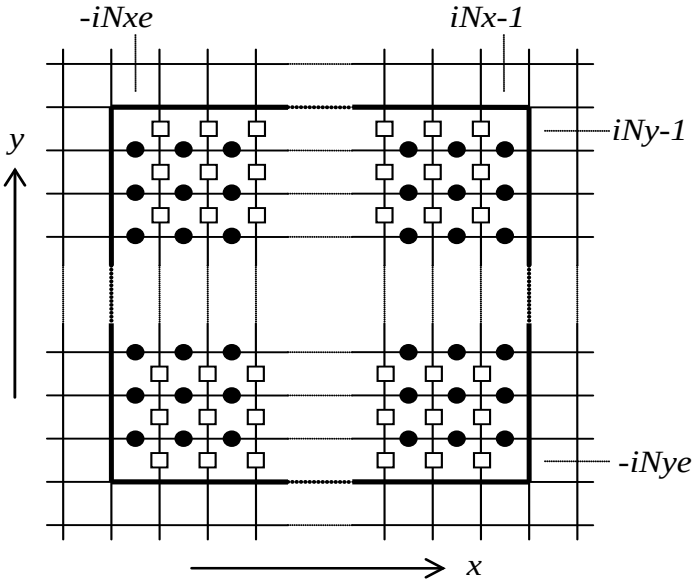


Fig. 10 超伝導素子金属板上の離散点 v_ϕ



● *Discrete Point* $B_{x_{i,j,k}}\left(x_i+\frac{\Delta x}{2}, y_j, z_k\right)$ *from* $\frac{dv_\phi}{dy}$

$-iNxe \leqq i \leqq iNx-1$
$-iNye+1 \leqq j \leqq iNy-1$
$-iNze+1 \leqq k \leqq iNz-1$

□ *Discrete Point* $B_{y_{i,j,k}}\left(x_i, y_j+\frac{\Delta y}{2}, z_k\right)$ *from* $\frac{dv_\phi}{dx}$

$-iNxe+1 \leqq i \leqq iNx-1$
$-iNye \leqq j \leqq iNy-1$
$-iNze+1 \leqq k \leqq iNz-1$

Fig. 11 xy 平面における位相差v_ϕから得られる離散点B_x, B_y

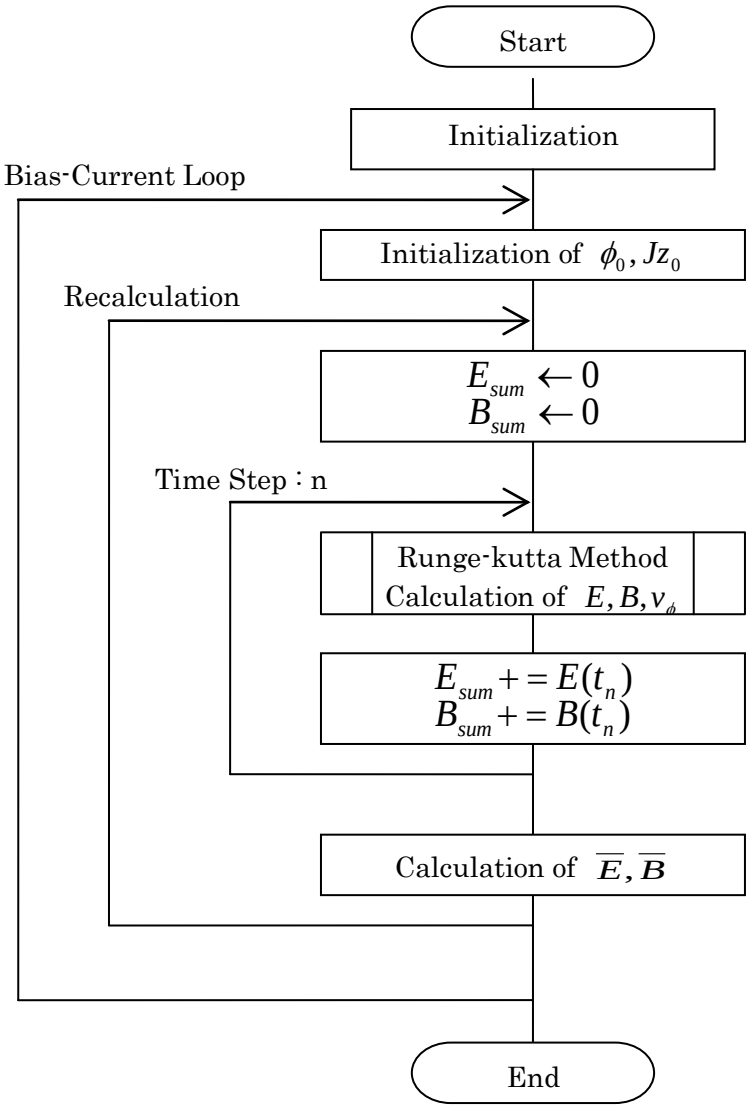


Fig. 12 プログラムのフローチャート

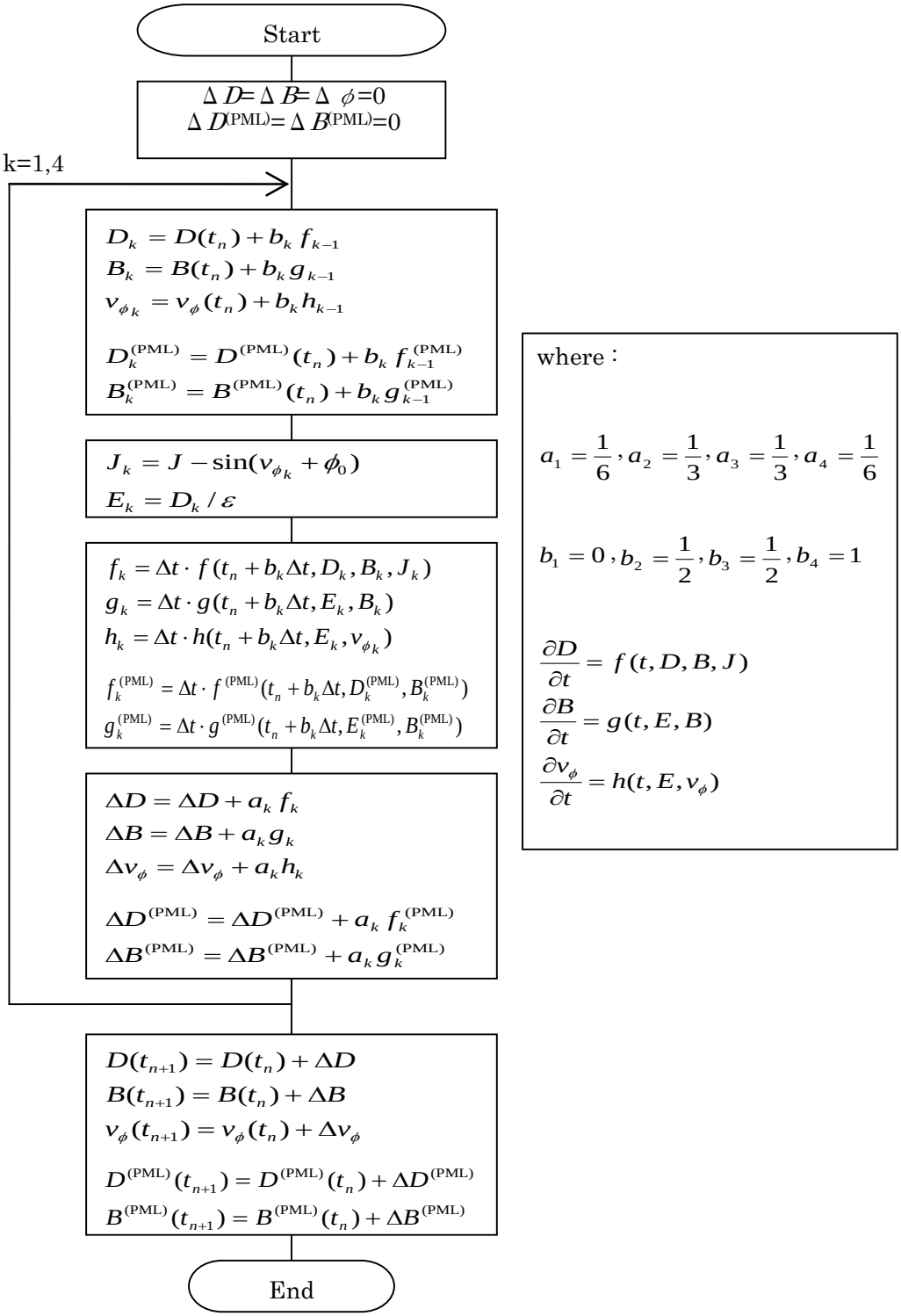


Fig. 13 ルンゲ・クッタ法のフローチャート

Table 4 各変数における離散点の定義

Variables	*Coordinate of Discrete Point(*)*		
	x	y	z
$E_x(D_x), \frac{\partial B_z}{\partial y}, \frac{\partial B_y}{\partial z}$	x_i	$y_j + \frac{\Delta y}{2}$	$z_k + \frac{\Delta z}{2}$
$E_y(D_y), \frac{\partial B_z}{\partial x}, \frac{\partial B_x}{\partial z}$	$x_i + \frac{\Delta x}{2}$	y_j	$z_k + \frac{\Delta z}{2}$
$E_z(D_z), \frac{\partial B_y}{\partial x}, \frac{\partial B_x}{\partial y}, \nu_\phi, J$	$x_i + \frac{\Delta x}{2}$	$y_j + \frac{\Delta y}{2}$	z_k
$B_x, \frac{\partial E_z}{\partial y}, \frac{\partial E_y}{\partial z}$	$x_i + \frac{\Delta x}{2}$	y_j	z_k
$B_y, \frac{\partial E_z}{\partial x}, \frac{\partial E_x}{\partial z}$	x_i	$y_j + \frac{\Delta y}{2}$	z_k
$B_z, \frac{\partial E_y}{\partial x}, \frac{\partial E_x}{\partial y}$	x_i	y_j	$z_k + \frac{\Delta z}{2}$

(*) x_i, y_j, z_k are grid coordinates.

2.4 空間座標の定義

2.4.1 原点座標

Junction 域の 3 次元直交座標系の x 軸に平行な幅 W、y 軸に平行な奥行 D からなる厚さ Δz 未満の長方形金属板と、x-y 平面に平行に Δz 間隔で $iNNz$ 枚重ねた絶縁体の中心を、3 次元直交座標系の原点と定義する。$iNNz$ は任意に与えられる正の整数である。

2.4.2 金属板領域と超伝導層の座標

超伝導金属板の幅 W(x 方向長)、奥行き D(y 方向長)は次式で与える。

$$\begin{aligned} W &= iNNx \times \Delta x \\ D &= iNNy \times \Delta y \end{aligned} \tag{63}$$

ここで、Δx は x 方向セル幅、Δy は y 方向セル幅であり、$iNNx$、$iNNy$ は任意に与える正の整数である。金属板から絶縁体の Δz 幅の層を超伝導層と呼ぶ。超伝導層の z 方向長 H は次のとおりである。

$$H = \Delta z \times (iNNz + 1) \tag{64}$$

超伝導層の中心を原点とするため、$iNNx$、$iNNy$、$iNNz$ が偶数か奇数かでセル格子点座標が変わる。$iNNx=7$、$iNNy=6$、$iNNz=3$ のときの例を Fig. 14 に示す。

2.4.3 真空領域長

真空領域長は、絶縁体システム中心点（原点）から正方向、負方向に対称にそれぞれ長さ $\mathbf{L}$ は次式で与える。

$$\mathbf{L} = 2l^{(\mathrm{PML})} + 2l^{(\mathrm{VA})} \tag{65}$$

ここで $l^{(\mathrm{PML})}$ は、原点から吸収壁境界までの距離、$l^{(\mathrm{VA})}$ は吸収壁長（吸収壁の厚さ）であり、$l^{(\mathrm{PML})}$、$l^{(\mathrm{VA})}$ ともに(格子点間隔)×(任意の正の整数) として与える。ただし、金属板の大きさ、レイヤ数に依存して原点がセル格子点からずれる場合には、原点から吸収壁境界までの距離 $l^{(\mathrm{PML})}$ にある吸収壁境界座標が、セル格子点と一致するよう以下のように考える。

x 方向の吸収壁境界までの距離を $l^{(\mathrm{PML})}{}_x$ として x 方向について説明する。原点 $x=0$ がセル格子点と一致するとき（金属板の x 方向セル数 $iNNx$ が偶数のとき）、次のようになる。

$$l^{(\mathrm{PML})}{}_x = iNPMx \times \Delta x \tag{66}$$

原点 $x=0$ がセル格子点と一致しないとき（$iNNx$ が奇数のとき）、次のようになる。

$$l^{(\mathrm{PML})}{}_x = iNPMx \times \Delta x - \frac{\Delta x}{2} \tag{67}$$

ここで *iNPMx* は任意の正の整数である。与える *iNNx* が奇数のとき、偶数に比べて真空領域長を Δx 短くとるということであり、 y 方向の真空領域長についても、同様に定義する。*iNNx* が偶数のときと、奇数のときの真空領域長の違いを Fig. 15 に示す。次に、z 方向についてだが、 z 方向の原点は、超伝導層数（金属板数+1）が、偶数のときに原点とセル格子点が重なる。原点 $z = 0$ がセル格子点と一致するとき（金属板数 *iNNz* が奇数のとき）、次式のようになる。

$$l^{(\mathrm{PML})}{}_z = iNPMz \times \Delta z \tag{68}$$

原点 $z = 0$ がセルの中央にあるとき（iNNz が偶数のとき）、次式のようになる。

$$l^{(\mathrm{PML})}{}_z = iNPMz \times \Delta z - \frac{\Delta z}{2} \tag{69}$$

iNNz が偶数のときと、奇数のときの真空領域長の違いを Fig. 16 に示す。

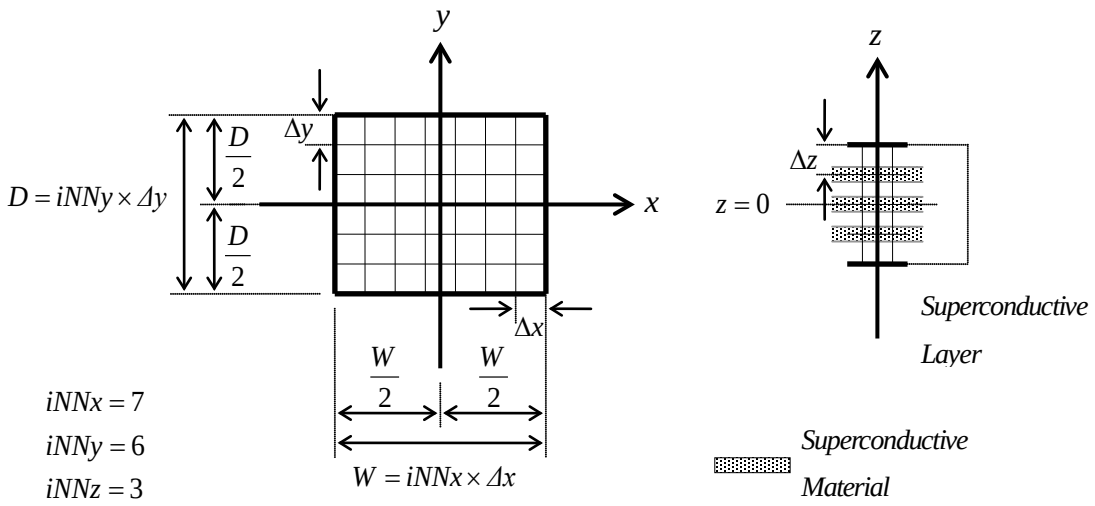


Fig. 14 金属板上での座標系（*iNNx=7*、*iNNy=6*、*iNNz=3*）

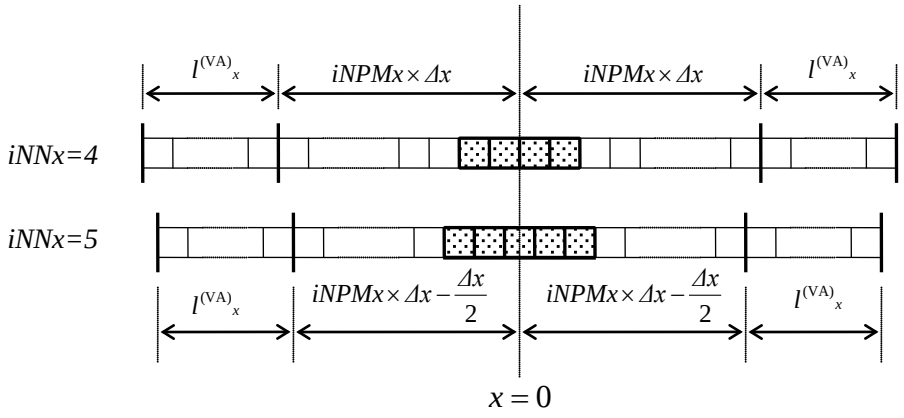


Fig. 15 *x* 方向における真空領域の長さの定義（*iNNx* が奇数、および偶数の場合）

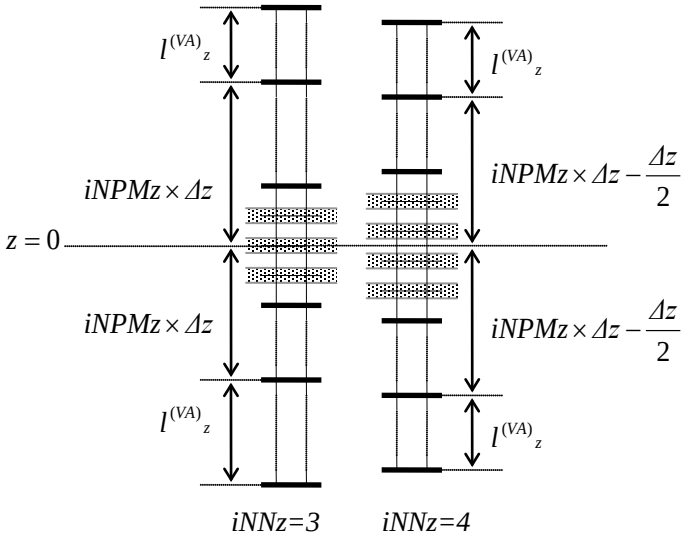


Fig. 16 *z* 方向における真空領域の長さの定義（*iNNz* が奇数、および偶数の場合）

2.5 セルインデックスの定義

2.5.1 セルの原点

次に定義される、あるセルの領域に原点が存在するセルの座標を 0 と定義する。

$$\begin{aligned} x_i \le x < (x_i + \Delta x) \\ y_j \le y < (y_j + \Delta y) \\ z_k \le z < (z_k + \Delta z) \end{aligned} \tag{70}$$

セル$(0,0,0)$のセル格子点座標(x_0, y_0, z_0)は、超伝導金属板の形状パラメータ(x 方向セル数 *iNNx*、y 方向セル数 *iNNy*、板の枚数 *iNNz*)を用いて次のように表せる。

$$\begin{aligned} x_0 &= -\frac{\Delta x}{2} \times \mathrm{mod}(iNNx,2) \\ y_0 &= -\frac{\Delta y}{2} \times \mathrm{mod}(iNNy,2) \\ z_0 &= -\frac{\Delta z}{2} \times \mathrm{mod}(iNNz+1,2) \end{aligned} \tag{71}$$

ここで、$\mathrm{mod}(a,b)$ は、$a \div b$ の整数剰余を表し、*iNNx* が偶数のとき $x_0 = 0$ 、奇数のとき $x_0 = -\frac{\Delta x}{2}$ になる。z 方向は、超伝導層(*iNNz+1*)の対称性を基準にしている。

2.5.2 超伝導金属板の $x-y$ 平面におけるセルの座標

x 軸正方向に向かって、超伝導金属板上端境界の格子点の添え字を *iNx*、下端境界の格子点の添え字を *–iNxe* とすると、次式のようになる。

$$\begin{aligned} iNx &= \mathrm{int}\left(\frac{iNNx+1}{2}\right) \\ iNxe &= iNx - \mathrm{mod}(iNNx,2) \end{aligned} \tag{72}$$

Fig. 17 に *iNNx* が偶数のときと奇数のときの *iNx*、*iNxe* の違いの例を示す。

y 軸についても同様にそれぞれ *iNy*、*–iNye* とすると、次式のようになる。

$$\begin{aligned} iNy &= \mathrm{int}\left(\frac{iNNy+1}{2}\right) \\ iNye &= iNy - \mathrm{mod}(iNNy,2) \end{aligned} \tag{73}$$

2.5.3 超伝導層の z 方向のセルの座標

z 軸正方向に向かって、超伝導層上端境界の格子点の添え字を *iNz*、下端境界の格子点の添え字を *–iNze* とすると、次式のようになる。

$$\begin{aligned} iNz &= \mathrm{int}\left(\frac{iNNz}{2}\right)+1 \\ iNze &= iNz - \mathrm{mod}(iNNz+1,2) \end{aligned} \tag{74}$$

ここで、最上位に位置する金属板がある格子点の添え字は *iNz-1*、最下位に位置する金属板がある格子点の添え字は*-iNze+1* になる。Fig. 18 に *iNNz* が偶数のときと、奇数のときの *iNz*、*iNze* の違いの例を示す。

2.5.4 吸収壁の境界のセルの座標

x 軸正方向に向かって、吸収壁の上端境界の格子点の添え字を *iNPMx*、下端境界の格子点を *-iNPMxe* とすると、次式のようになる。*iNPMx* は任意に与える正の整数である。

$$iNPMxe = iNPMx - \mathrm{mod}(iNNx,2) \tag{75}$$

y 軸方向についても同様に、吸収壁の上端境界の格子点の添え字を *iNPMy*、下端境界の格子点の添え字を *–iNPMye* とすると、次式のようになる。*iNPMy* は任意に与える正の整数である。

$$iNPMye = iNPMy - \mathrm{mod}(iNNy,2) \tag{76}$$

z 軸方向についても同様に、吸収壁の上端境界の格子点の添え字を *iNPMz*、下端境界の格子点の添え字を *–iNPMze* とすると、次式のようになる。*iNPMz* は任意に与える正の整数である。

$$iNPMze = iNPMz - \mathrm{mod}(iNNz+1,2) \tag{77}$$

2.5.5 真空領域の境界のセルの座標

x 軸正方向に向かって、真空領域の上端境界の格子点の添え字を *iNvx*、下端境界の格子点を *-iNvxe* とすると、次式のようになる。

$$\begin{aligned} iNvx &= iNPMx + iNvax \\ iNvxe &= iNPMxe + iNvax \end{aligned} \tag{78}$$

ここで *iNPMx*、*iNPMxe* は前小節で述べた吸収壁の境界の格子点の添え字であり、*iNvax* は吸収壁の長さ(厚さ)のセル長である。

y 軸方向についても同様に、真空領域の上端境界の格子点の添え字を *iNvy*、下端境界の格子点の添え字を *–iNvye* とすると、次式のようになる。

$$\begin{aligned} iNvy &= iNPMy + iNvay \\ iNvye &= iNPMye + iNvay \end{aligned} \tag{79}$$

z 軸方向についても同様に、真空領域の上端境界の格子点の添え字を *iNvz*、下端境界の格子点の添え字を *–iNvze* とすると、次式のようになる。

$$\begin{aligned} iNvz &= iNPMz + iNvaz \\ iNvze &= iNPMze + iNvaz \end{aligned} \tag{80}$$

y 方向境界条件に周期条件が与えられたとき、*iNvay=0* であり、*iNvy=iNPMy* になる。 *z* 方向についても同様に、*iNvaz=0* であり、*iNvz=iNPMz* になる。Fig. 19 に、計算領域と格子点の座標を示す。Fig. 19 の例では、*iNNx=4*、*iNNy=4*、*iNNz=3*、*iNPMx=6*、*iNPMy=6*、*iNPMz=6*、*iNvax=3*、*iNvay=3*、*iNvaz=3* となっている。

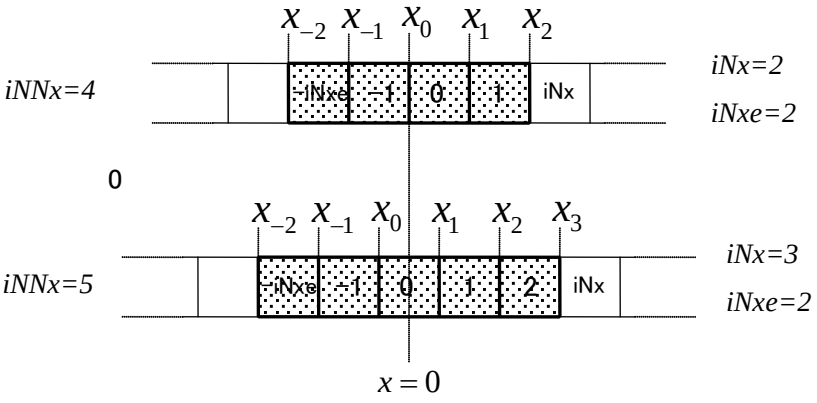


Fig. 17 金属板上の離散点のインデックス定義（*iNNz* が奇数、および偶数の場合）

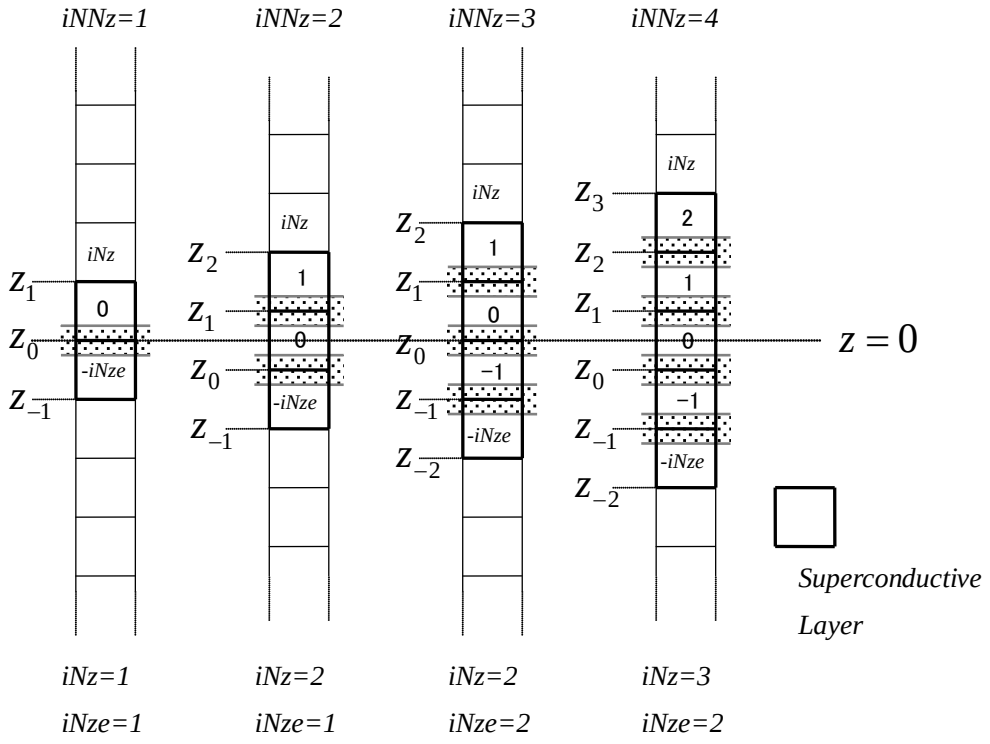


Fig. 18 *z* 方向の離散点のインデックス定義（*iNNz* が奇数、および偶数の場合）

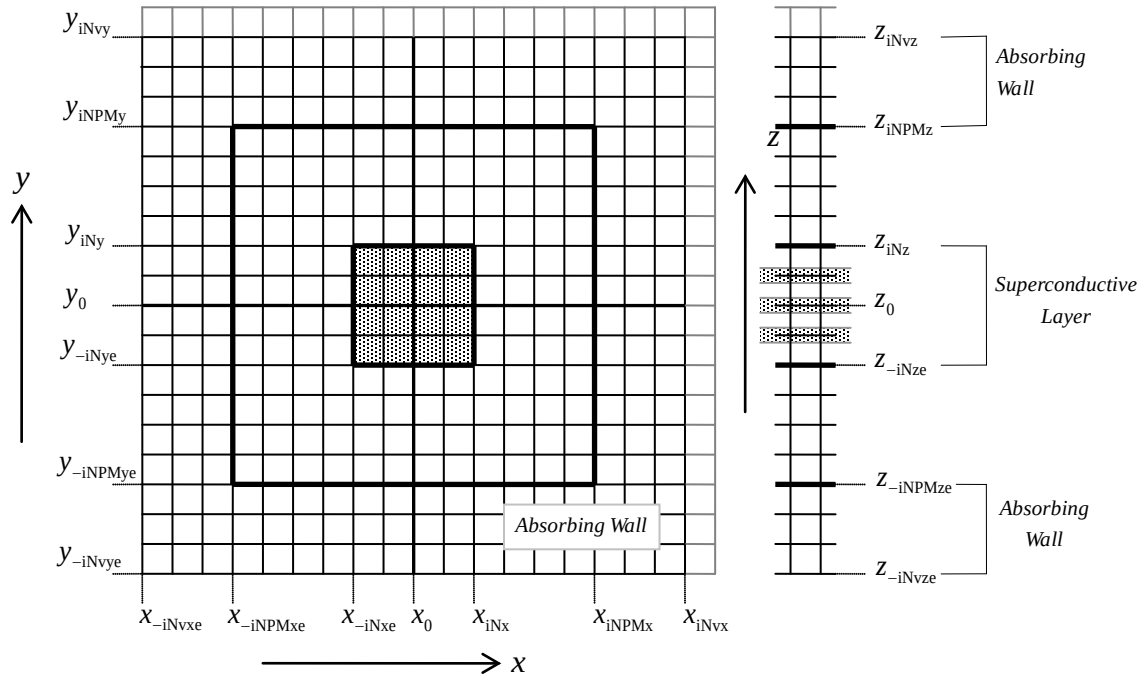


Fig. 19 *xy* 平面と *z* 軸方向の離散点の定義

3. 並列化

3.1 並列化モデル

3.1.1 並列化の指針

プログラム開発においては、逐次版に対して、MPI による並列化を行う手順を取った。ここでは、全体の計算領域をプロセス数に応じて矩形分割し、分割された直方体領域を各プロセスが処理する。また、Fig. 1 に示すように、全ての領域に対する場の計算と Junction 域に対する計算を交互に行う。すなわち、全てのプロセッサにより全体領域に対する場の計算を行った後、全てのプロセッサにより Junction 域に対する計算を行う。それぞれの計算領域を矩形分割することで、プロセス間での計算負荷の均一化を図った。

3.1.2 空間の分割（計算領域の分割）

1) 場の計算を行う際のセル領域（全体の計算点の分割）

場の計算を行う際の計算領域を分割する方法について述べる。Fig. 20 にあるように、全体の計算領域を *x* 方向、*y* 方向、*z* 方向に任意に分割してできた直方体領域にあるセル群を 1 つのプロセスに割り当てる。プロセス間のデータ参照は、すべて MPI 通信で行う。

分割の一例として、*x* 方向 100、*y* 方向 100、*z* 方向 100 からなる計算領域空間を *x* 方向 5 分割、*y* 方向 4 分割、*z* 方向 2 分割（すなわち、5×4×2＝40 分割=40 プロセス）したとすると、各プロセスに割り当てられる計算領域長は *x* 方向 20(=100÷5)、*y* 方向 25(=100÷4)、*z* 方向 50(=100÷2)となる。この際、計算領域長は、必ずしも分割数の整数倍でなければならないという制限は無い。計算領域の一辺が分割数で割り切れない場合は、各分割領域間での一辺の長さの差が最小（必ずその差分は 1 以下）になるように分割する。一辺が分割数で割り切れないケースでは、各プロセスが担当するセル数は、プロセスによって異なる。

2) 場の計算を行う際の座標系とセルの座標の定義

計算領域の座標系と、各プロセスが保持する空間のセルの座標について述べる。各プロセスの計算領域は、全体の計算領域空間と同じように 3 次元方向に配置される。また、その座標系と座標は、全体の計算領域空間と同じ値をとる。あるプロセスの担当する *x* 方向のインデックスを i、*x* 方向セル長を $ixSize$、*x* 方向の分割数を N_x とすると、そのプロセスの担当する最下端 $ixHead_{(i)}$ と最上端セルのインデックスを $ixTail_{(i)}$ は、次のようになる。

$$\begin{aligned} ixHead_{(i)} &= ixTail_{(i-1)} + 1 \\ ixTail_{(i)} &= ixHead_{(i)} + ixSize_{(i)} - 1 \\ 1 &\le i \le N_x \end{aligned} \tag{81}$$

ただし、$ixHead_{(0)}$ は、全体の計算領域空間の最下端セルのインデックスであり全体の領域長 N_{vx} は、次の式を満たす。

$$N_{vx} = \sum_{i=1}^{N_x} ixSize_{(i)} \tag{82}$$

例として、全体の領域長 16 の計算領域を 4 分割した場合の、各 Field ノードが保持するセル座標の範囲を、Fig. 21 に示す。図中の *rank#0* … *rank#3* は、計算ノードの識別名である。

3) Junction 域の分割

Junction 域を計算する際の分割について述べる。Junction 域の計算では、Junction 域を *x* 方向、*y* 方向、*z* 方向に分割し、分割してできた直方体領域にあるセル群を各プロセスに割り当てて行う。Junction 域の分割数は *x* 方向、*y* 方向、*z* 方向それぞれ全体の計算領域空間の分割数と同じである。ただし、Junction 域の一辺の長さが分割数に満たない場合はその長さを分割数とする。

例として、Fig. 5 に全体領域を計算する場合と Junction 域を計算する場合のセルを 2 次元モデルで示す。この例では、計算領域長は、*x* 方向 12、*y* 方向 12 であり、これを *x* 方向に 3 分割、*y* 方向に 3 分割して 9 つのプロセスに割り当てている。Junction 域の大きさは、*x* 方向 6、*y* 方向 6 の 36 点であり、各プロセスに割り当てられるセル数は、1 つのプロセスあたり 2×2=4 となる。

3.1.4 *N* 対 *N* プロセス間通信モデル

開発したプログラムで実装したプロセス間通信について述べる。*N* 対 *N* のプロセス間通信時に発生するデッドロックを排除するために、送信側非同期、受信側同期として実装した。Fig. 22 に、データ通信処理のシーケンスを示す。Fig. 23 に、隣接プロセス間での境界値交換問題を例にとり、FORTRAN による実装例を示す。

3.1.5 エラー通知モデル

MPI 通信におけるエラー処理方法について述べる。MPI 通信で実装されている mpid（MPI 通信実行のために事前に起動しておくプログラム）における例外シグナル（並列処理に使われているいずれかのプロセッサに障害が起きたことなどを通知する機能）の取扱いは、MPI としての明確な規格が存在しないため、実装は MPI を提供するベンダによって異なる。子プロセスの TERMINATION シグナルに連鎖して全ての子プロセスを正しく終了させる実装もあれば、TERMINATION シグナルのハンドラを割愛した実装もある。本プログラムでは、コードの移植性を考慮し、TERMINATION シグナルが検出された場合、全てのプロセスを終了させる仕様とした。すなわち、例外を検出した場合、発生した例外事象を他プロセスへ通知して各プロセスが自発的に処理を終了する仕組みを用意した（予期せぬ例外には、MPI__Abort 関数を使用して、終了処理を MPI の実装に従う）。ここでは、エラー通知の方法を交通信号の概念でモデル化した。例外事象を検出したプロセスは、信号を赤にスイッチする。全てのプロセスは交差点で必ず信号待ちをして、信号が青なら交差点を渡る。信号が赤なら直ちにプロセスの出口へ向かう。シグナルは、「進め」「止まれ」のトグルスイッチである。逐次処理において Fig. 24 のような分岐処理がある場合のプロセス間エラー通知モデルの実装例を、Fig. 25 に示す。AMD 社提供の mpid プログラムからのシグナル処理の流れを、Fig. 26 に示す。

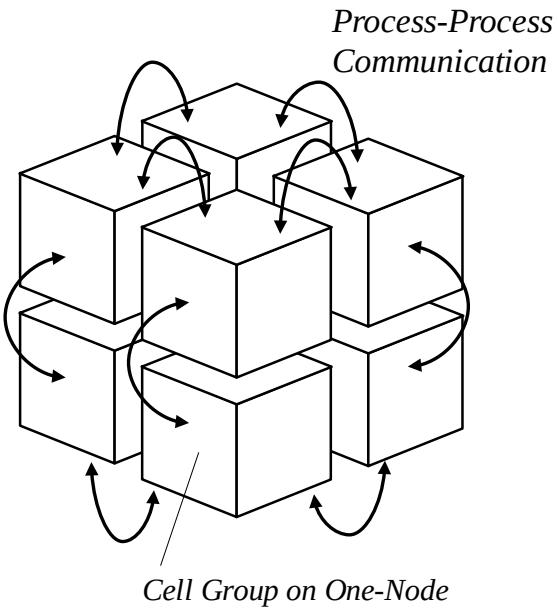


Fig. 20 分割された空間のプロセス間通信の様子

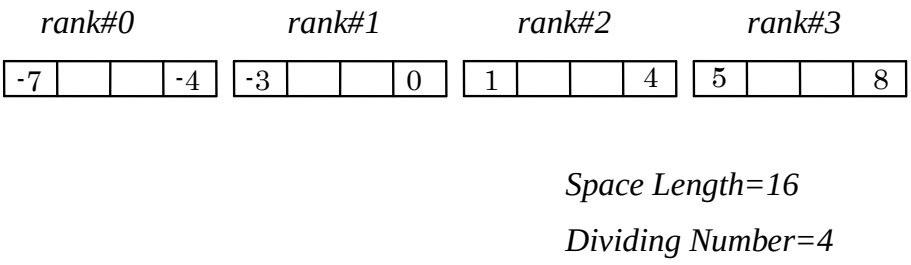


Fig. 21 空間を分割した例（16 の長さを持つ空間を 4 個に分割した場合）

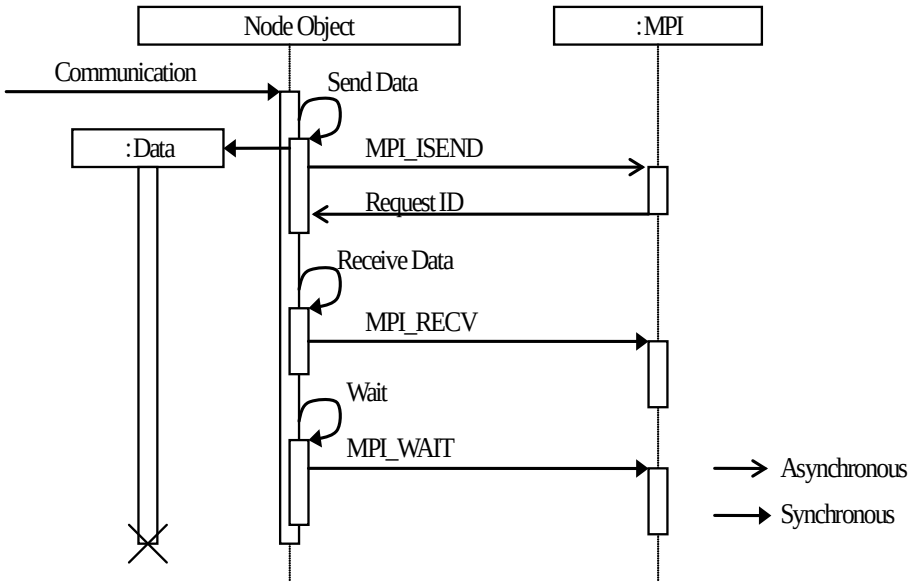


Fig. 22 プロセス間通信の実装方法

```
!!! Send Data (Asynchronous)
do i=1,N_neighbors
   call MPI_ISEND(send_edge(1,i), edge_length(i), MPI_REAL8, &
                  nbr_rank(i), tag, comm ,requests(i), ierr )
enddo

!!! Receive Data (Synchronous)
do i=1, N_neighbors
   call MPI_RECV (recv_edge(1,i), edge_length(i), MPI_REAL8, &
                  nbr_rank(i), tag, comm, status, ierr)
enddo

!!! Wait
call MPI_WAITALL(N_neighbors, requests, status, ierr)
```

Fig. 23 プロセス間通信の実装例

```
!!! Irregular Procedure
call DoSomething(ierror)
if ( ierror .ne. 0 )   return

...
```

Fig. 24 逐次処理における例外処理判定の実装例

```
!!! Irregular Procedure
call DoSomething(ierror)
if ( ierror .ne. 0 )   call TurnSignal(ISIG_STOP)

!!! Wait signal
call WaitSignal(signal)
if ( signal .eq. ISIG_STOP )   return

...
```

Fig. 25 複数プロセスにおける例外処理判定の実装例

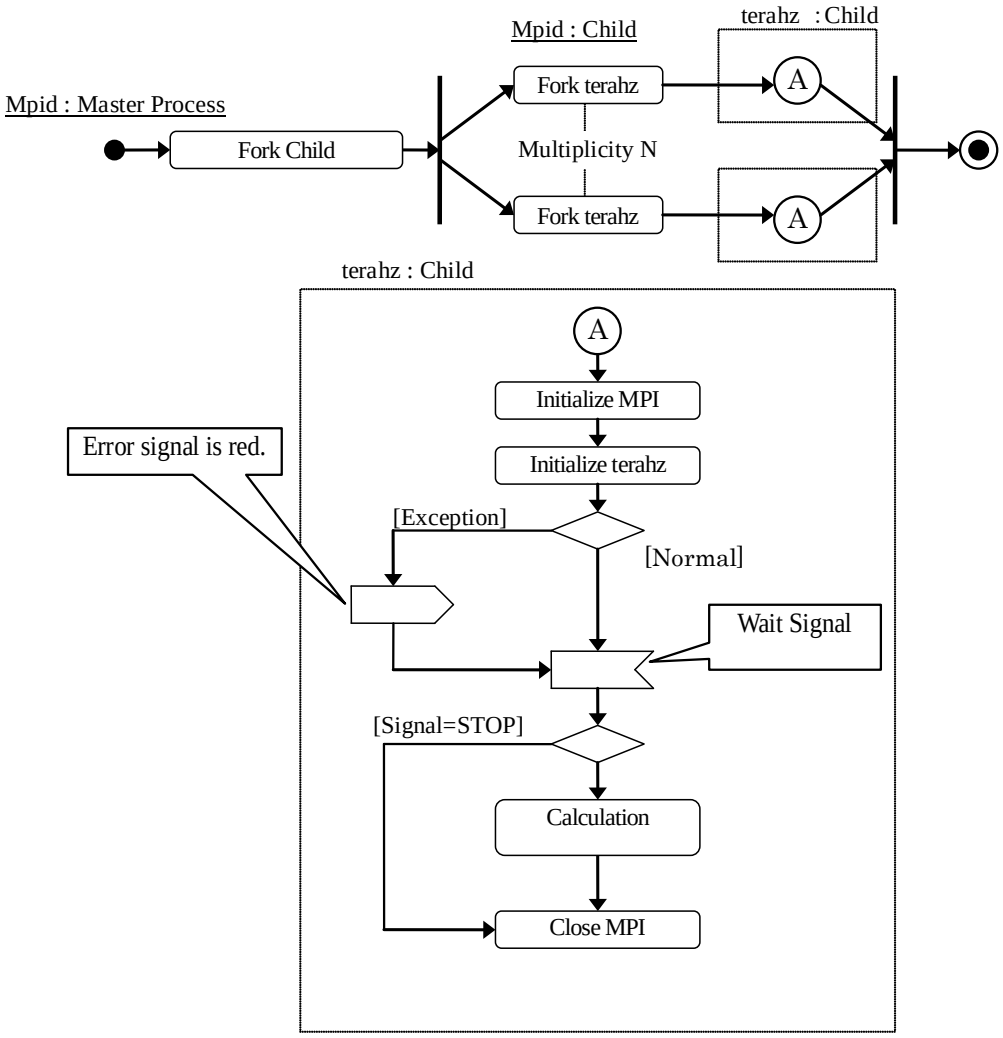


Fig. 26 MPI アプリケーションの実装例

3.2 プログラム構造

3.2.1 フローチャートの記述について

各小節で示すフローチャートでは、全体の流れを把握し易くするために、いくつかの例外処理分岐の記述を省略している。図中の太字の関数（サブルーチン）名は、MPI-API（MPI 通信用関数）を表す。また、太枠の外部手続きは、太枠の外部手続き後にプロセス間データ通信が発生することを表す。

3.2.2 メインプログラム

メインプログラムの処理を Fig. 27 に示す。

3.2.3 プロセスインスタンスの初期化処理

プロセスインスタンスの初期化処理の処理を Fig. 28 に示す。

3.2.4 計算処理ドライバ

計算処理ドライバの処理を Fig. 29 に示す。

3.2.5 計算部分メイン制御処理

計算部分メイン制御処理の処理を Fig. 30 に示す。

3.2.6 時間積分処理

時間積分処理の処理を Fig. 31 に示す。

3.2.7 時間積分処理における k ステップのデータの流れ

ルンゲ・クッタ法による時間積分の処理における k ステップのデータフローダイアグラムを、Fig. 32 から Fig. 35 に示す。

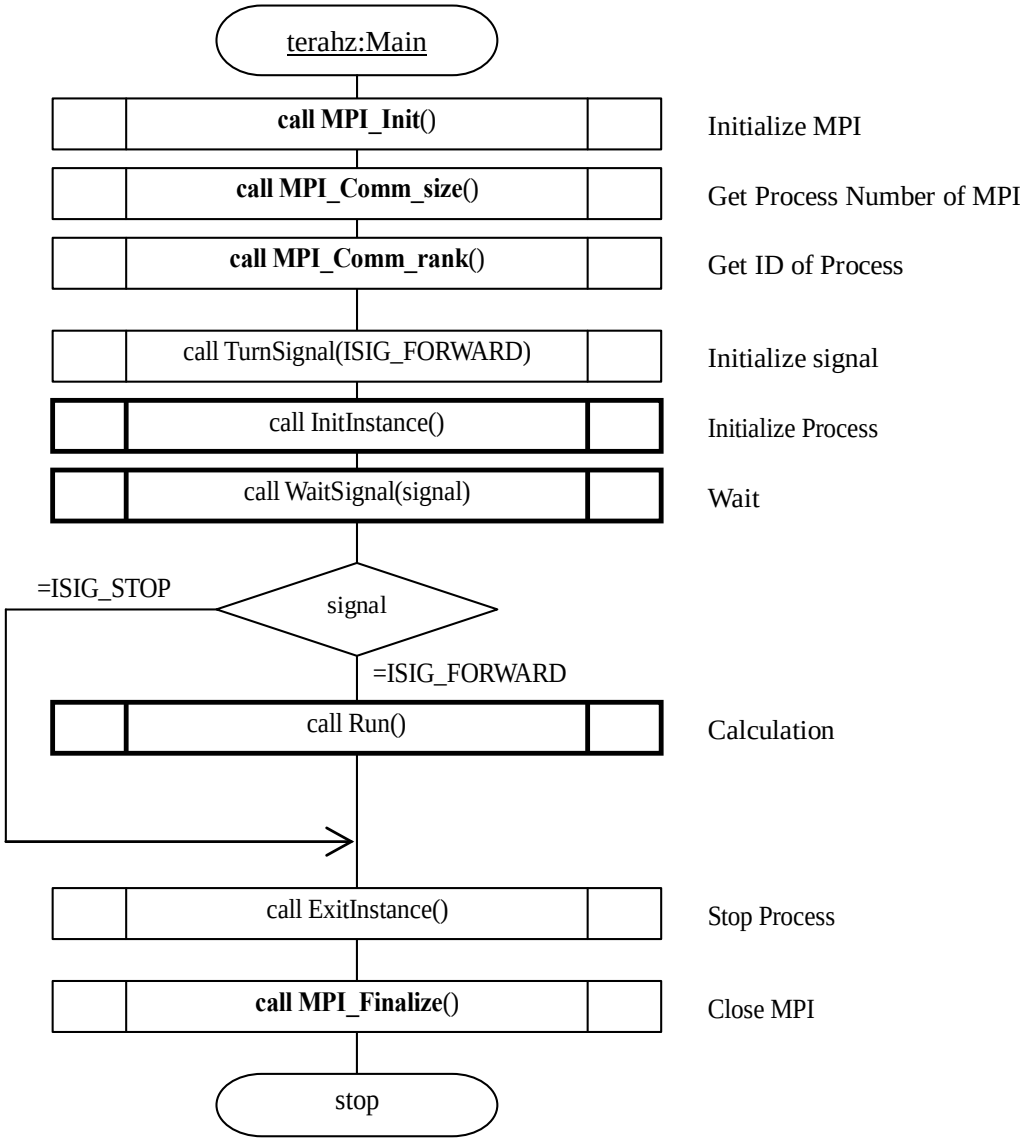


Fig. 27 プロセス間通信のフローチャート

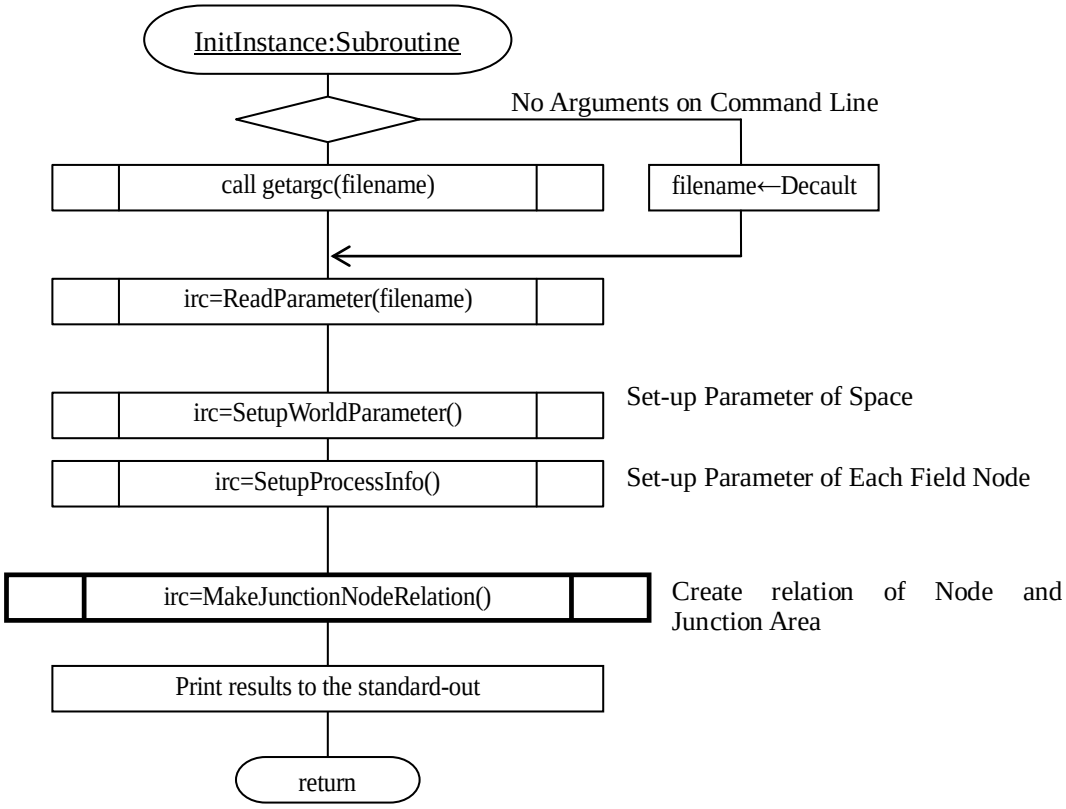


Fig. 28 プロセスの初期化処理

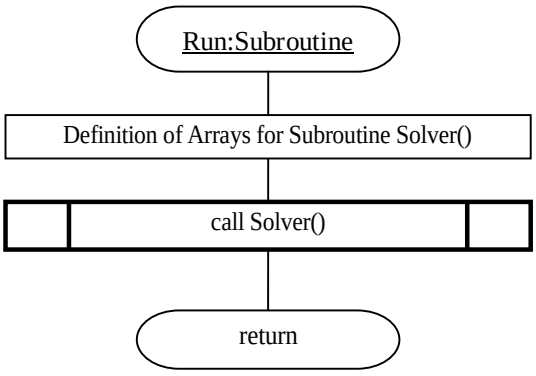


Fig. 29 計算処理のフローチャート

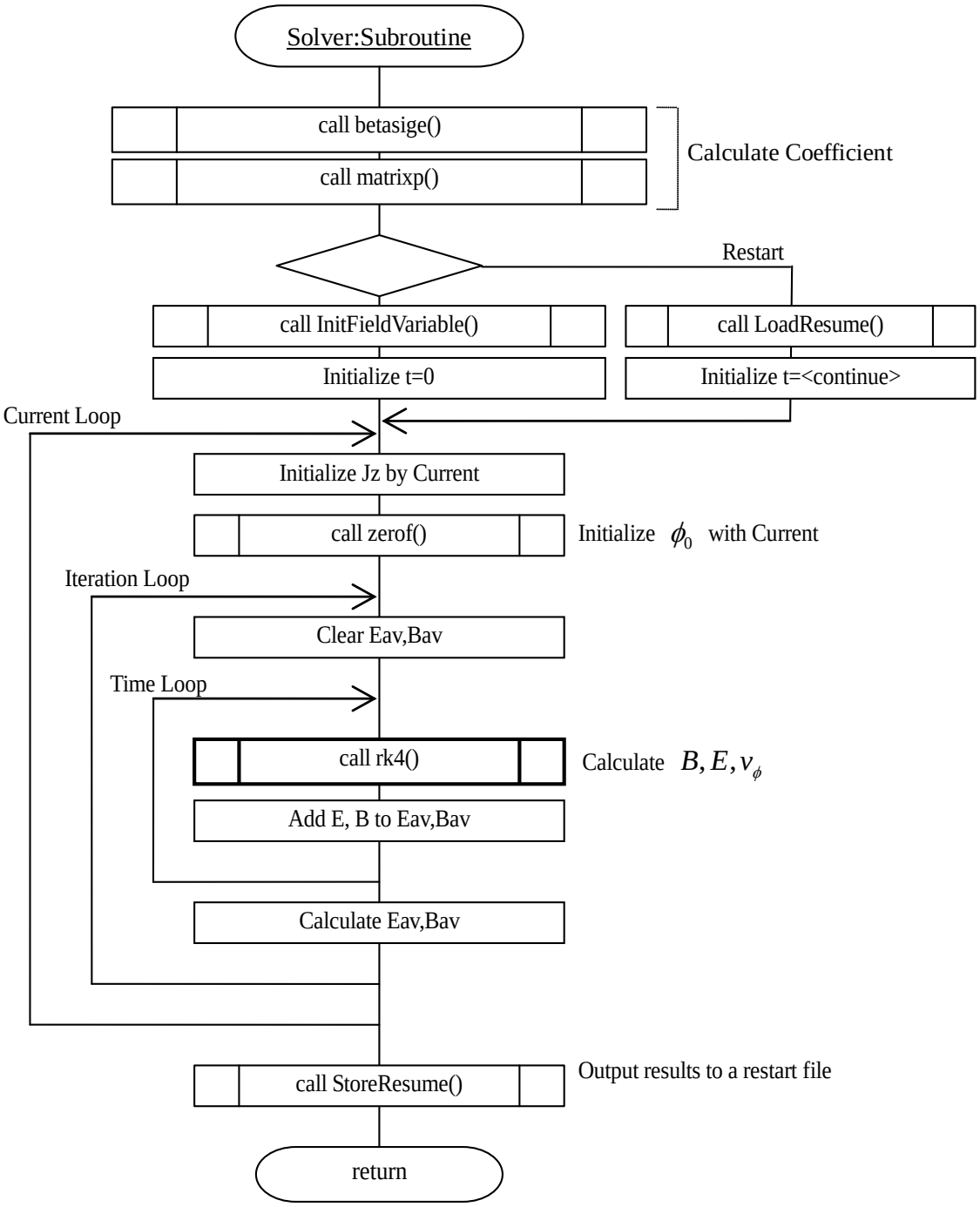


Fig. 30 数値解析のフローチャート

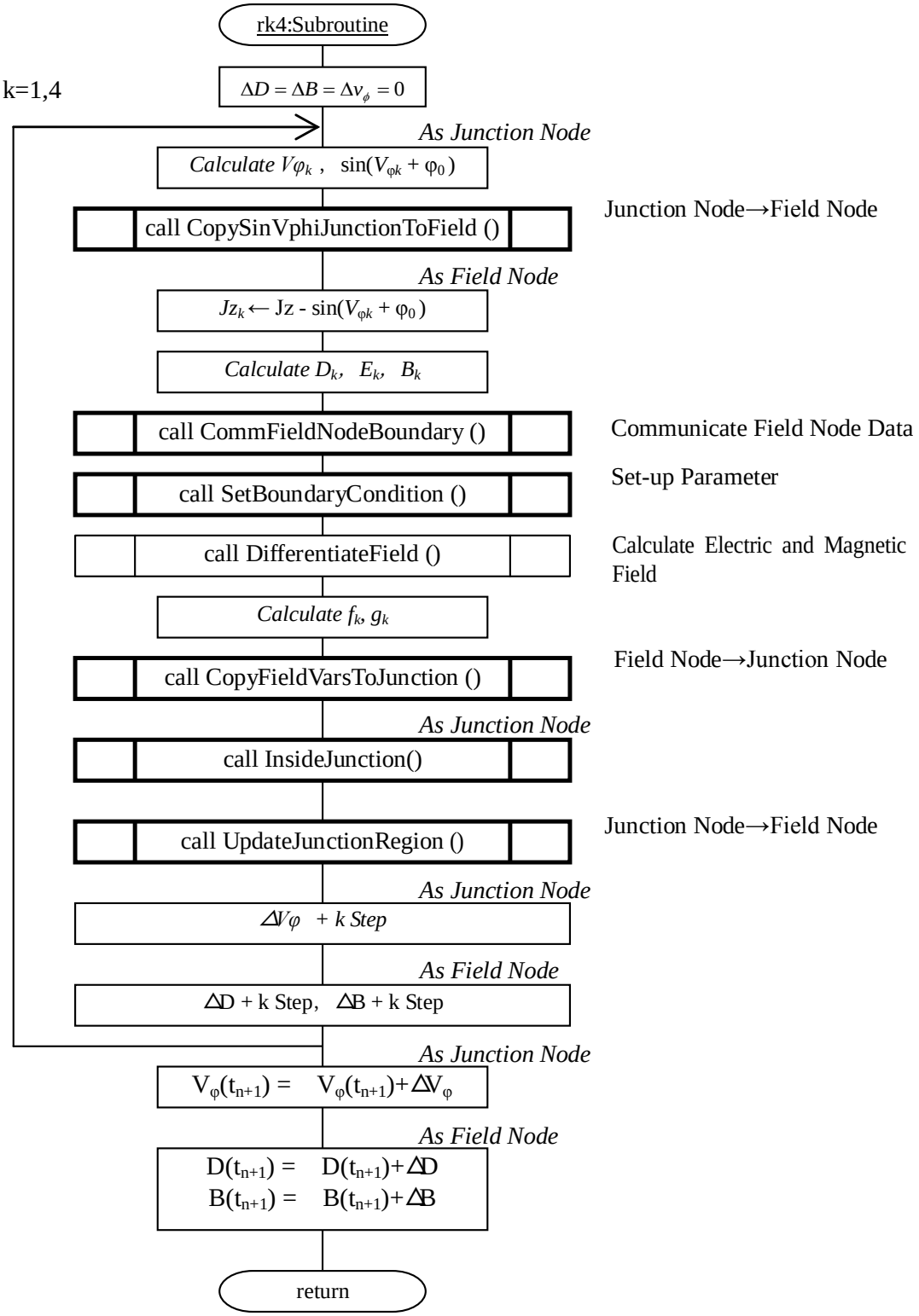


Fig. 31 時間積分のフローチャート

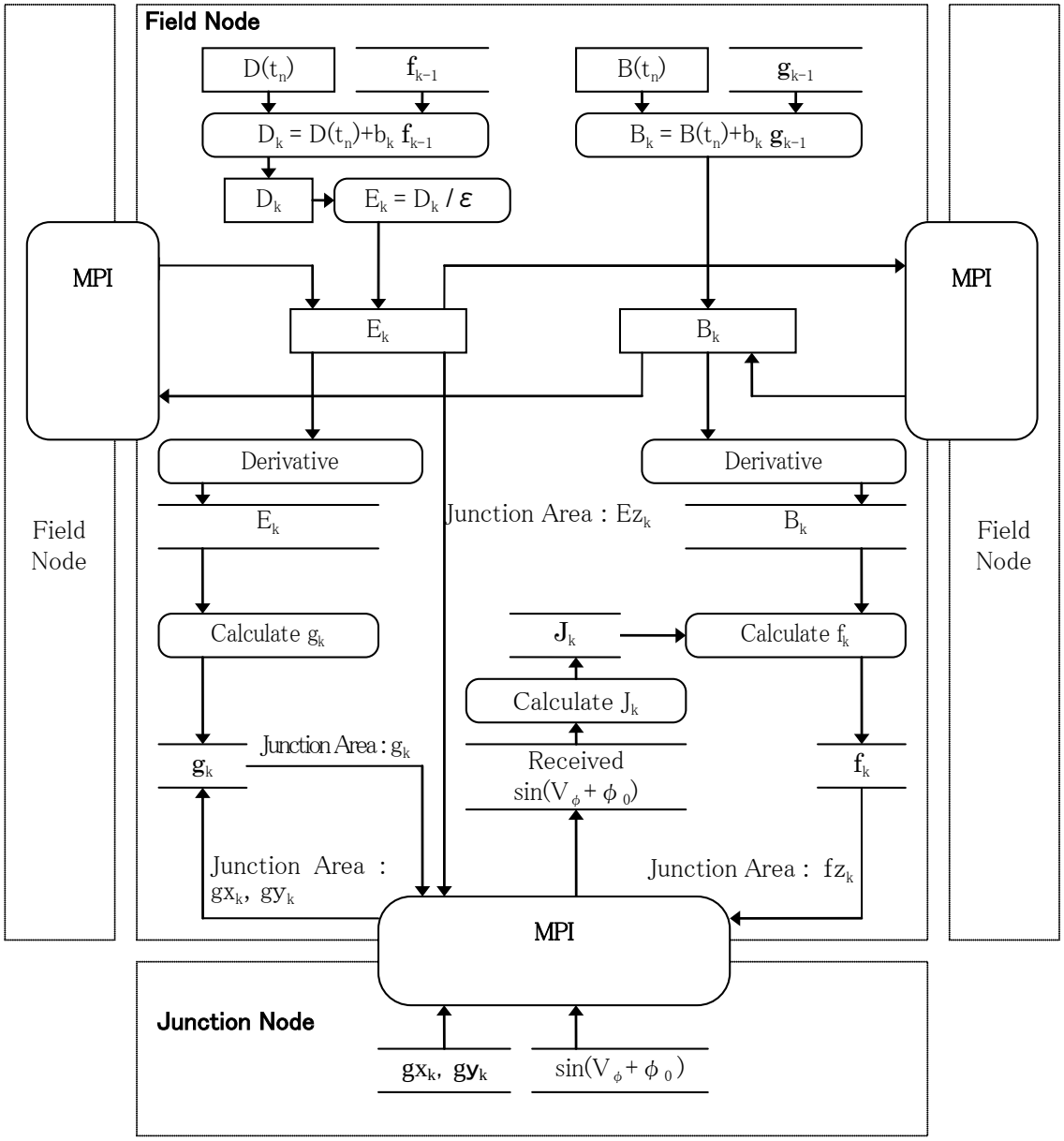


Fig. 32 ルンゲ・クッタ法のデータフロー

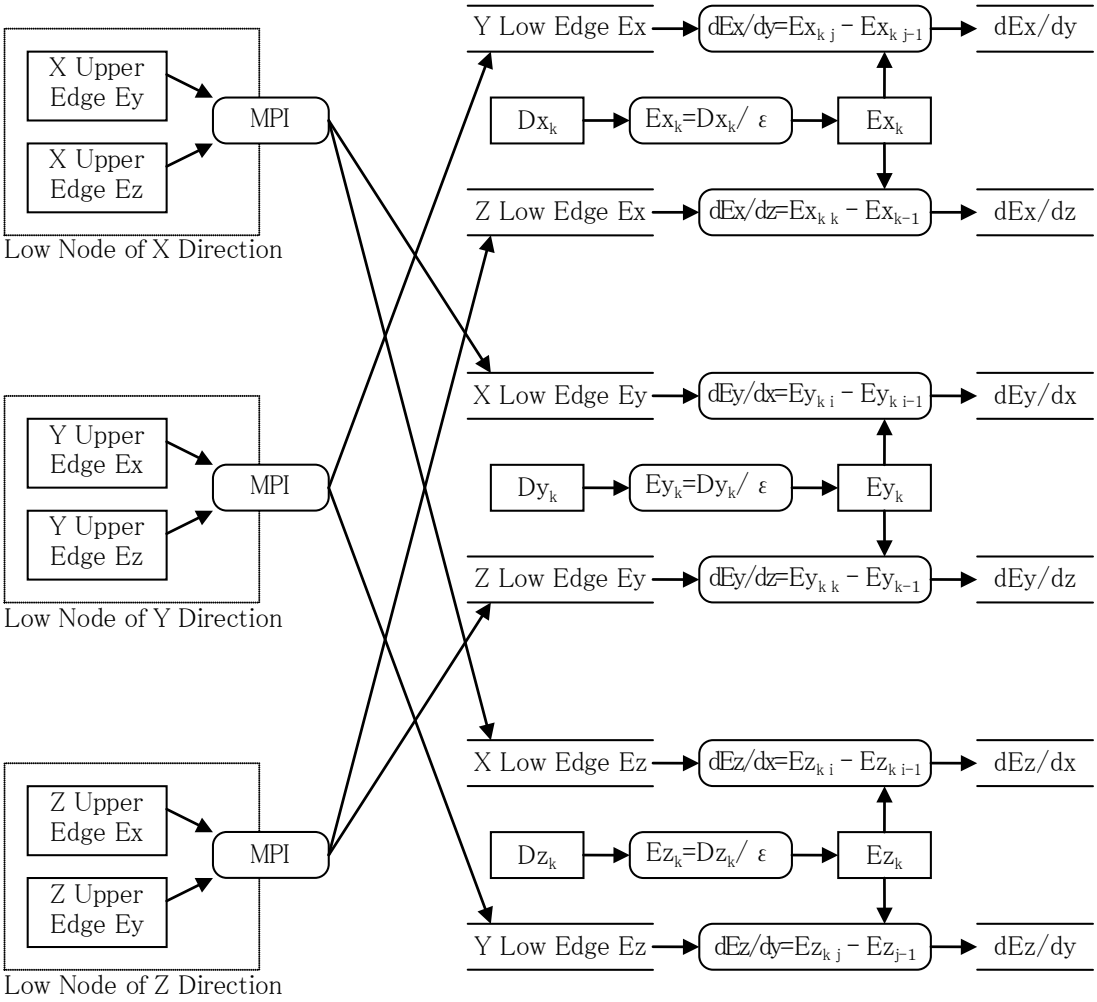


Fig. 33 ルンゲ・クッタ法による空間微分（電場）

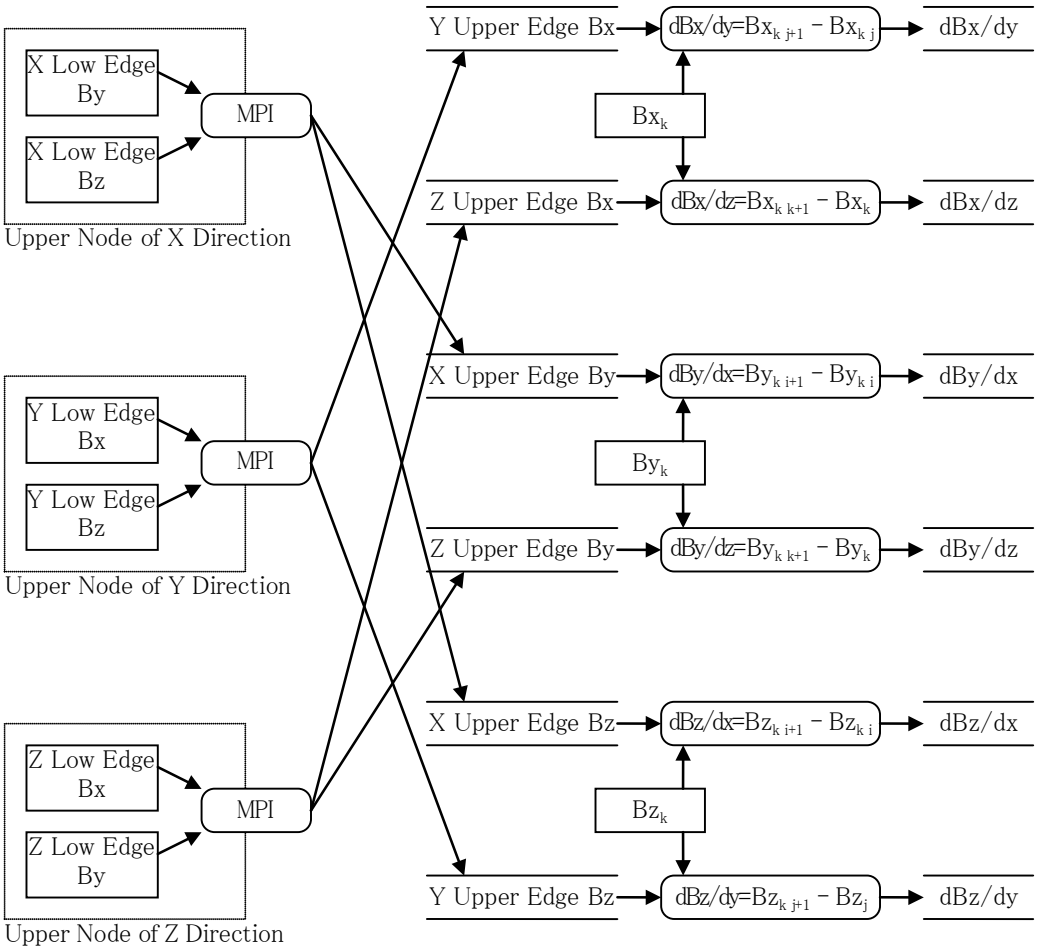


Fig. 34 ルンゲ・クッタ法による空間微分（磁場）

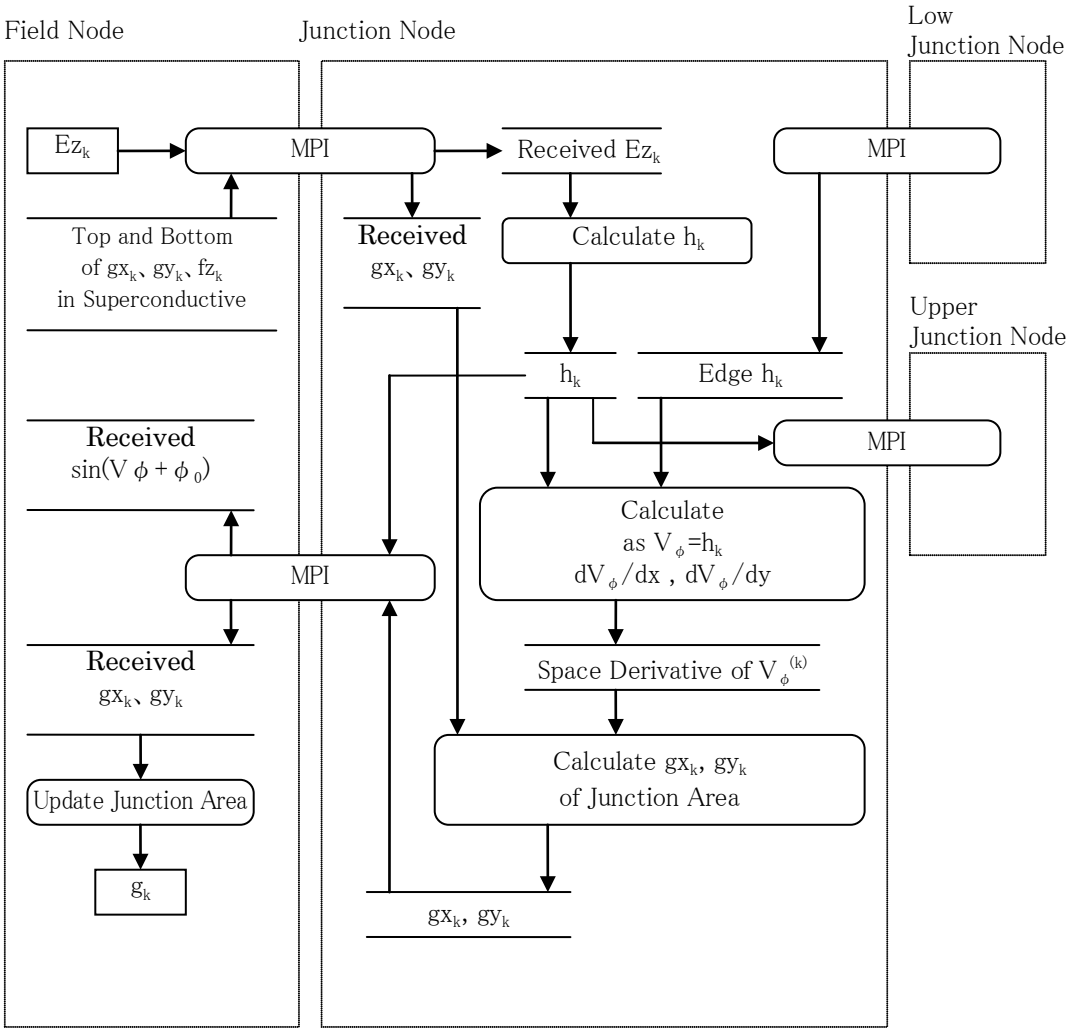


Fig. 35 ルンゲ・クッタ法による空間微分（超伝導素子）

3.3 プロセス間通信

3.3.1 場の計算を行う際の境界値の通信

場の計算を行う際のデータ通信について述べる。中心差分により電場、磁場の空間微分を求める際、境界面を共有するプロセス間で境界値をデータ通信する必要がある。次の電場 *D* の差分方程式から、

$$\begin{aligned}
\frac{\partial}{\partial t}Dx_{i,j,k} &= \frac{1}{\mu_0}\left(\frac{1}{\Delta y}(Bz_{i,j+1,k}-Bz_{i,j,k})-\frac{1}{\Delta z}(By_{i,j,k+1}-By_{i,j,k})\right)-J_{i,j,k}\\
\frac{\partial}{\partial t}Dy_{i,j,k} &= \frac{1}{\mu_0}\left(\frac{1}{\Delta z}(Bx_{i,j,k+1}-Bx_{i,j,k})-\frac{1}{\Delta x}(Bz_{i+1,j,k}-Bz_{i,j,k})\right)-J_{i,j,k}\\
\frac{\partial}{\partial t}Dz_{i,j,k} &= \frac{1}{\mu_0}\left(\frac{1}{\Delta x}(By_{i+1,j,k}-By_{i,j,k})-\frac{1}{\Delta y}(Bx_{i,j+1,k}-Bx_{i,j,k})\right)-J_{i,j,k}
\end{aligned} \tag{83}$$

磁場 *B* の中心差分が必要とするプロセスが持つ空間の境界値は、*xyz* 各方向の上位に位置するプロセスが保持する空間の最下端の値である。また、次の磁場 *B* の差分式から、

$$\begin{aligned}
\frac{\partial}{\partial t}Bx_{i,j,k} &= -\left(\frac{1}{\Delta y}(Ez_{i,j,k}-Ez_{i,j-1,k})-\frac{1}{\Delta z}(Ey_{i,j,k}-Ey_{i,j,k-1})\right)\\
\frac{\partial}{\partial t}By_{i,j,k} &= -\left(\frac{1}{\Delta z}(Ex_{i,j,k}-Ex_{i,j,k-1})-\frac{1}{\Delta x}(Ez_{i,j,k}-Ez_{i-1,j,k})\right)\\
\frac{\partial}{\partial t}Bz_{i,j,k} &= -\left(\frac{1}{\Delta x}(Ey_{i,j,k}-Ey_{i-1,j,k})-\frac{1}{\Delta y}(Ex_{i,j,k}-Ex_{i,j-1,k})\right)
\end{aligned} \tag{84}$$

電場 *E* の中心差分が必要とするプロセスが持つ空間の境界値は、*xyz* 各方向の下位に位置するプロセスが保持する空間の最上端の値である。

x-y 平面でみた電場 *E* の境界値の通信を Fig. 36 に、磁場 *B* の境界値の通信を Fig. 37 に示す。周期条件のときの境界値の通信を Fig. 38 から Fig. 39 に示す。

3.3.2 Junction 域を計算する場合の境界値の通信

Junction 域に流れる電流の位相 v_ϕ の空間差分方程式は次のとおりである。

$$\begin{aligned}
\frac{\partial}{\partial x}v_{\phi i,j,k} &= \frac{1}{\Delta x}(v_{\phi i,j,k}-v_{\phi i-1,j,k})\\
\frac{\partial}{\partial y}v_{\phi i,j,k} &= \frac{1}{\Delta y}(v_{\phi i,j,k}-v_{\phi i,j-1,k})
\end{aligned} \tag{85}$$

このとき、$\frac{\partial v_\phi}{\partial x}, \frac{\partial v_\phi}{\partial y}$ を中心差分で求める。Fig. 40 に示すように、より大きな *i* と *j* を担当す

るプロセスの最上端の値を、そのプロセスが持つ空間の境界値として通信する。

3.3.3 プロセス間通信のタイミングと内容

場の計算から Junction 域への計算に移行する際のデータ通信について述べる。Junction 域の計算に必要なセル情報（ルンゲ・クッタ法の *k* ステップ目の *Ezk* 値、*Bxk* 値、*Byk* 値）は、プロセス間でデータ転送を行い、計算終了後、結果(*Bxk* 値、*Byk* 値)をフィードバックする。

例として、場の計算から Junction 域への計算間のメッセージの相関図を Fig. 41 に示す。Fig. 41 の例では、場の計算を終えた *rank#0* は、Junction 域を計算するために *rank#0* に参照値を送信し、計算結果を受信する。すなわち、rank#0 は、場の計算を終了した *rank#0, rank#1, rank#3, rank#4* から参照値を受信し、計算結果を *rank#0, rank#1, rank#3, rank#4* に送信する。この関係をプロセスで見ると、*rank#0* プロセスは、場の計算を行う際、1 プロセスとの通信を行い、Junction 域の計算を行う際、 4 プロセスと通信を行う。

なお､開発したプログラムのファイル一覧を付録 A に､入力ファイルの設定方法を付録 B に、HA8000 へのジョブ投入時に使用するファイルの設定方法を付録 C に示す。

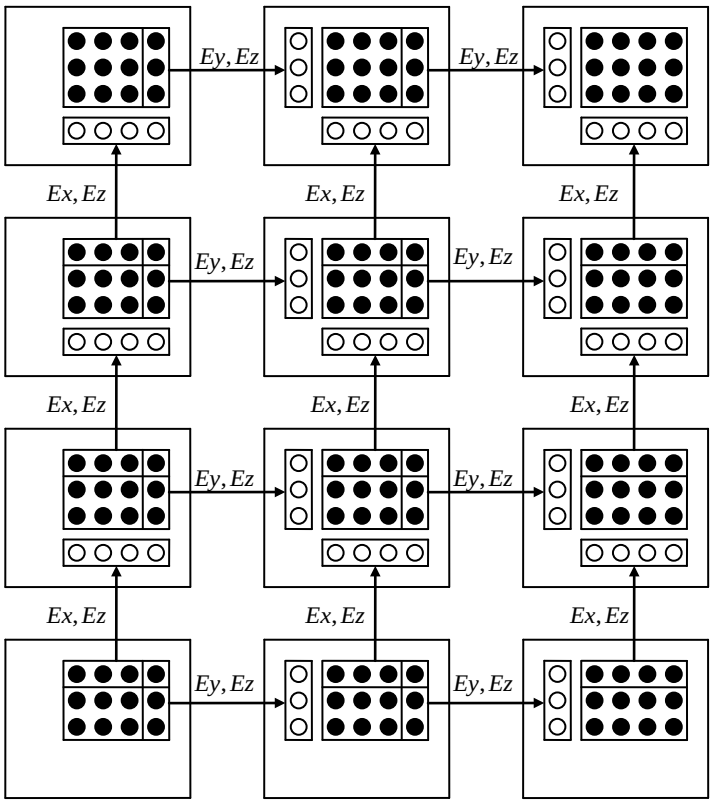


Fig. 36 *xy* 平面における電場解析時の各プロセス間の境界領域送受信方法

白丸：送受信データ、黒丸：各プロセスにおける電場の値

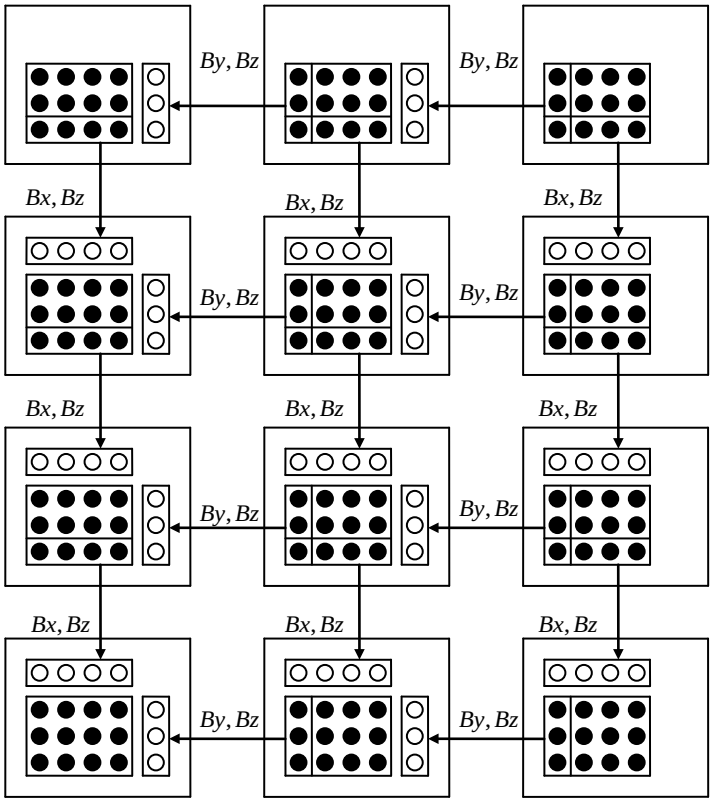


Fig. 37 *xy* 平面における磁場解析時の各プロセスの境界領域送受信方法

白丸：送受信データ、黒丸：各プロセスにおける磁場の値

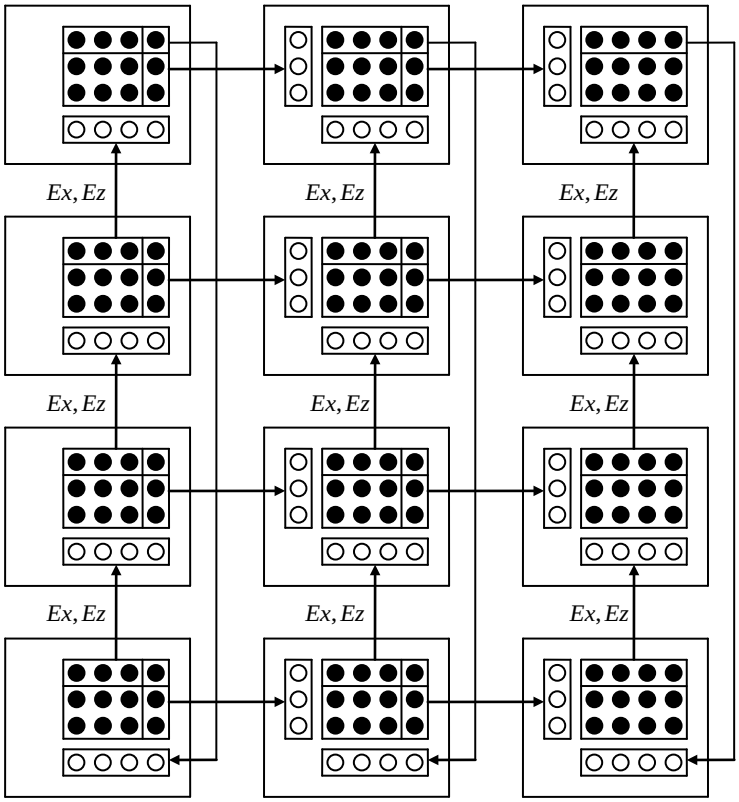


Fig. 38 *xy* 平面における電場解析時の各プロセスの境界領域送受信方法（境界壁がない場合）

白丸：送受信データ、黒丸：各プロセスにおける電場の値

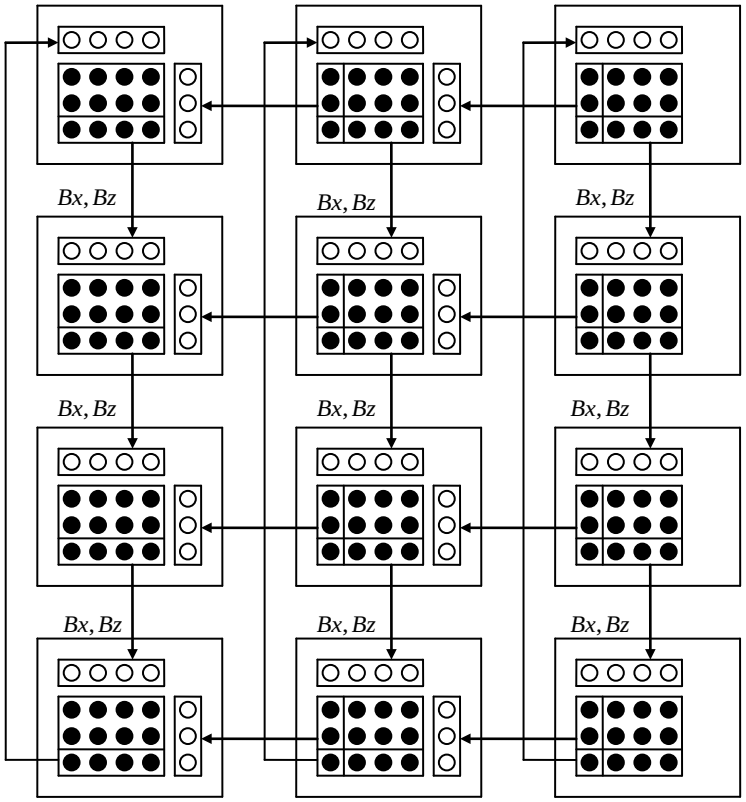


Fig. 39 *xy* 平面における磁場解析時の各プロセスの境界領域送受信方法（境界壁がない場合）

白丸：送受信データ、黒丸：各プロセスにおける磁場の値

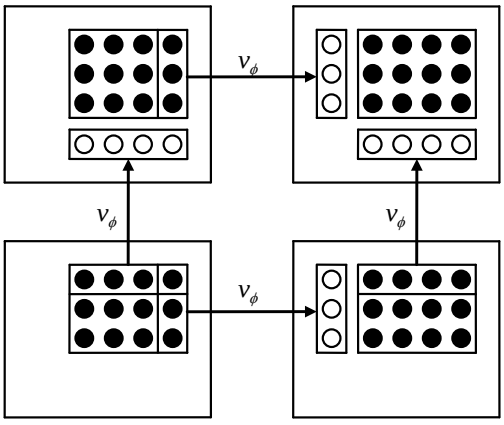


Fig. 40 *xy* 平面における位相差解析時の各プロセスの境界領域送受信方法

白丸：送受信データ、黒丸：各プロセスにおける位相差の値

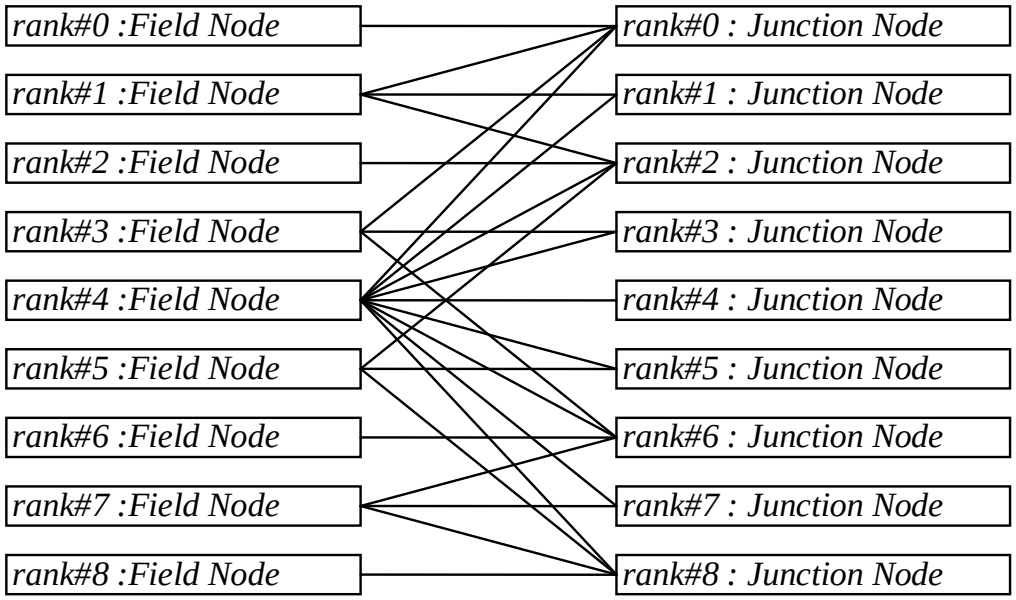


Fig. 41 Field ノード（場の計算を行うプロセス）と Junction ノード（Junction 域の計算を行うプロセス）の関係図の一例

4. 性能評価

4.1 HA8000 の基本性能の評価について

開発したプログラムの性能を考察するため、次の項目について、HA8000 の基本的な性能評価を行った。結果を 4.2 節～4.6 節に示す。

・浮動小数点演算
・メモリ帯域
・ノード間通信帯域
・キャッシュメモリ

これらの評価は、単一ノードの演算性能及び複数ノードの演算性能を評価する基準となる。各性能を評価する際に使用するコードは、HA8000 用 HITACHI 製 FORTRAN コンパイラでコンパイルし（Table 5 に最適化オプションを示す）、Table 6 に示す関数で時間計測や MPI 通信を行った。また、CPU に直結されるメインメモリを利用するように設定した。性能評価で使用したクラスタのノード群は、Type B（ネットワークの通信帯域は 2.5 [GB / s]）である。計測時間を取得する関数のオーバーヘッドを調べた結果、約 1.35 [μs]であったことから、性能評価では、このオーバーヘッドの影響を無視できる程度の計算量になるよう、演算回数とデータ量を設定した。

並列処理時のコアへのプロセスの割り当て方法は複数あり、その例を Fig 42 に示す。図の左から、4 コアを 1 つのプロセスに割り当てた場合、2 コアを 1 つのプロセスに割り当てた場合、1 コアを 1 つのプロセスに割り当てた場合である。図の点線で囲まれた箇所が、1 つのプロセスである。2 つ以上のプロセスを扱う場合、プロセスごとに異なる CPU を指定することができる。例えば、図の中央にある 2 つのプロセスの場合、異なる CPU の 2 つのコアを指定することができる。以降、HA8000 の性能評価では、4 コア以上を扱う場合に、図の左端の割り当て方法（1 つの CPU 上にある 4 コアを 1 プロセスで利用）を使用した。

Table 5 性能評価に用いたコンパイルオプション一覧

オプション	効果
-Oss	最大最適化
-parallel=4	並列化における最大最適化

Table 6 性能評価に用いたライブラリ一覧

サブルーチン	処理内容
xclock()	時間計測
mpi_barrier()	MPI 通信の同期処理
mpi_reduce()	各ノードのデータを取得してルートプロセスが指定した処理を行う
mpi_scatter()	ルートプロセスがデータを各ノードに送信
mpi_gather()	ルートプロセスがデータを各ノードから受信

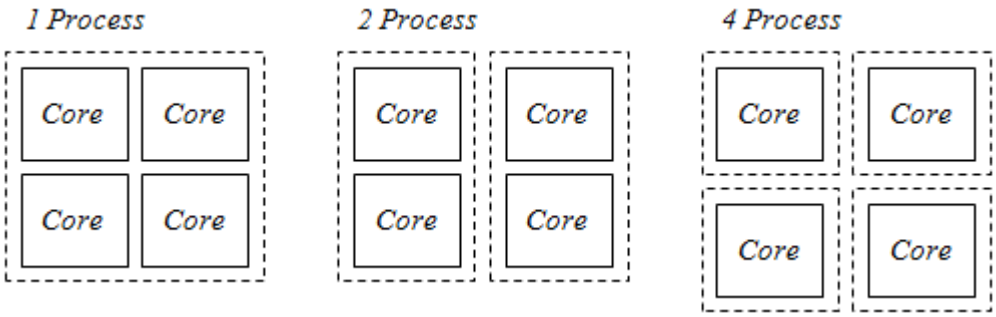


Fig 42 コアに対するプロセスの割り当て方法

4.2 浮動小数点演算の性能評価

4.2.1 評価の条件

HA8000 の浮動小数点演算の性能を評価するにあたり、次のように条件を設定した。

・メモリ帯域の影響を避けるため、浮動小数点演算に配列を取り扱わない
・並列性能の比較においては、ノード間通信の時間を演算時間に含める

これらの条件に従って作成したコードをFig. 43 に示す（性能測定箇所の抜粋である）。使用したライブラリをTable 6 に示す。このコードは、3 回の加算を行った時間を計測する。3 重ループ[1]のうち、外側にあるループの変数*iStart*と*iEnd*は、プロセス数に応じて変化する。例えば、2 つのプロセスを起動して、外側のループを 1 から 1000 まで繰り返す場合、一方のプロセスは*iStart*=1 と*iEnd*=500 であり、他方は*iStart*=501 で*iEnd*=1000 である。つまり、プロセス数が増えるほど演算回数は減り、プロセスの負荷は軽くなる。ここで、*jMax*や*kMax*は任意に設定する定数である。また、時間測定は、*stime*と*etime*の差を取得して計測した。

4.2.2 評価結果

浮動小数点演算に関する性能評価の結果を Table 7、Table 8 に示す。コア数に応じて、演算にかかった時間、単位時間あたりの浮動小数点演算回数、および理論性能との対比を表している。浮動小数点演算の性能は、次の計算を用いて値を求めた。F は単位時間あたりの浮動小数点演算回数、N は浮動小数点演算回数、t は演算にかかった時間である。今回の評価用コードでは、N を 3,000,000,000 回とした（3 回の加算を 1,000,000,000 回 ）。

$$F = \frac{N}{t} \tag{86}$$

コア数が増えるに従って、単位時間あたりの浮動小数点演算性能は向上し、演算時間は減少を示している。理論性能との比較では、ノード数が 1 の場合（16 コアまで）、98%以上の性能を得た。複数ノードを利用した演算時間は、ノード数の増加に対して減少傾向ではあるが、性能は 1 ノードのそれより低い 90%程度であった。性能が低い理由は、後述するように、ネットワーク通信によるオーバーヘッドの影響である。

[1] 単純な計算を行うループの場合、コンパイラの最適化によって計算結果が予測されることがあり、計算結果を予測困難にする目的でこのような形にしている。

```
ans = 0.0d0

call xclock( stime, 8 )!start

do i = iStart, iEnd
        do j = 1, jMax
                do k = 1, kMax
                        ans = ans + i + j + k
                enddo
        enddo
enddo

sbuf = ans
call mpi_barrier( mpi_comm_world, mpi_err )
call mpi_reduce( sbuf, rbuf, 1, mpi_double_precision, mpi_sum, 0, mpi_comm_world, &mpi_err )
ans = rbuf

call xclock( etime, 8 )!stop
```

Fig. 43 浮動小数点演算の性能評価用プログラム

Table 7 浮動小数点演算の性能評価の結果（1 コア～16 コア）

コア数	1	2	4	8	16
CPU 数	1	1	1	2	4
ノード数	1	1	1	1	1
時間 [sec.]	1.324	0.663	0.332	0.166	0.084
浮動小数点演算回数 [GFlops]	2.3	4.5	9.0	18.1	35.9
性能比 [%]	98.52	98.37	98.26	98.38	97.47

Table 8 浮動小数点演算の性能評価の結果（32 コア～256 コア）

コア数	32	64	128	256
CPU 数	8	16	32	64
ノード数	2	4	8	16
時間 [sec.]	0.042	0.021	0.011	0.006
浮動小数点演算回数 [GFlops]	70.7	140.8	273.0	539.6
性能比 [%]	96.02	95.66	92.72	91.65

4.3 メモリ帯域の性能評価

4.3.1 評価の条件

HA8000 のメモリ帯域の性能を評価するにあたり、次のように条件を設定した。

・動的にメモリを確保した場合、オーバーヘッドが大きいことから、メモリは静的に確保する
・物理アドレス計算が容易となるよう、1 次元配列を扱う
・並列性能の比較においては、ノード間通信時間を演算時間に含めるものとする

これらの条件に従って作成したコードを Fig. 44 に示す。このコードは各プロセスで配列を 2 つ（data と tmp_data）用意し、一方の配列からもう一方の配列へデータをコピーする。使用したサブルーチンを前節と同様に Table 6 に示す。*iStart* と *iEnd* はプロセスによって異なる。すべてのプロセスでデータのコピー終了後、同期処理を行う。時間計測方法は、前節と同様である。

4.3.2 評価結果

メインメモリ帯域に関する性能評価の結果を Table 9、Table 10 に示す。表は、コア数に応じて、データのコピーにかかった時間とメモリ帯域性能を示している。ただし、メモリ帯域の倍率については、Table 9 が 1 コアに対する倍率、Table 10 が 16 コアに対する倍率を示す。表に示す帯域は、次のように求めている。ここで、*B* はメモリ帯域、*M* は配列のサイズ（今回は double precision の配列）、*t* はコピーにかかった時間である。配列のサイズを 2 倍している理由は、2 つの配列を使用するからである。今回の評価では、*M* は 400,000,000 [byte]である。

$$B = \frac{2 \times M}{t}$$

コア数が増えるに従って、メモリ帯域は増加している様子が見られる。同一ノード内では、コア数の増加する割合より、メモリ帯域の増加する割合が大きい。複数ノードの場合のメモリ帯域の変化の様子は、ネットワーク通信のオーバーヘッドの影響を受けて、コア数の増加する割合より小さな増加傾向となった。

ここで、上述の同一ノード内で見られた性能を調査するため、異なる方法で 4 コアを使った性能を評価した。その結果を Table 11 に示す。ここでは、1 つのプロセスを起動して 1 つの CPU を使った場合、4 つのプロセスを起動して 1 つの CPU を使った場合、4 つのプロセスを起動して 4 つの CPU を使った場合で比較している。評価したコードは、同じく Fig. 44 である。1 つの CPU で 4 コアを 1 つのプロセスで利用する場合が最もメモリ帯域が広くなり、1 つの CPU で 1コアを 1 つのプロセスで利用する場合に最もメモリ帯域が狭くなった。この理由は、後述するキャッシュメモリの性能評価と併せて説明する。

```
call xclock( stime, 8 )!start

do i = iStart, iEnd
        iLen = ( i - 1 ) * iMax
        do j = 1, jMax
                jLen = ( j - 1 ) * jMax
                do k = 1, kMax
                        tmp_data( iLen + jLen + k ) = data( iLen + jLen + k )
                enddo
        enddo
enddo

call mpi_barrier( mpi_comm_world, mpi_err )

call xclock( etime, 8 )!stop
```

Fig. 44 メモリ帯域の性能評価用プログラム

Table 9 メモリ帯域の性能評価の結果（1 コア～16 コア）

コア数	1	2	4	8	16
CPU 数	1	1	1	2	4
ノード数	1	1	1	1	1
時間 [sec.]	0.223	0.010	0.052	0.025	0.013
メモリ帯域 [GB / s]	3.3	7.5	14.4	29.6	58.8
性能比	1.00	2.27	4.36	8.97	17.81

Table 10 メモリ帯域の性能評価の結果（32 コア～256 コア）

コア数	32	64	128	256
CPU 数	8	16	32	64
ノード数	2	4	8	16
時間 [sec.]	0.006	0.003	0.002	0.001
メモリ帯域 [GB / s]	118.7	227.5	445.7	869.1
性能比	2.02	3.87	7.58	14.78

Table 11 メモリ帯域の性能評価の結果（4 コアを使った場合）

コア数	1	1	4
プロセス数	1	4	4
時間 [sec.]	0.052	0.072	0.056
メモリ帯域 [GB / s]	14.4	10.3	13.4
1 コアのメモリ帯域に対する性能比	4.4	3.1	4.1

4.4 ノード間通信帯域の性能評価

4.4.1 評価の条件

HA8000 のノード間通信帯域の性能を評価するにあたり、次のように条件を設定した。

・動的にメモリを確保した場合、オーバーヘッドが大きいことから、メモリは静的に確保する
・双方向通信の性能評価をする

これらの条件に従って作成したコードを Fig. 45 に示す。使用したサブルーチンを前節と同様に Table 6 に示す。このコードは、ルートプロセスが用意した配列データを他のプロセスに送信して、受信したデータをルートプロセスに送信するコードである。ルートプロセスは、ある一定サイズのデータを用意する。送受信するデータの合計サイズは、プロセス数に応じて変化する。具体的には、次の式でサイズを求める。ここで、M は各プロセスが送受信するデータの合計サイズ、S はルートプロセスが用意するデータサイズ、P はプロセス数である。

$$M = 2 \times S \frac{P-1}{P} \tag{87}$$

また、データサイズは次のような条件を持つ。

$$S \leq M < 2 \times S \tag{88}$$

データの送信と受信のときに、それぞれ同期処理をする。時間計測方法は、前節と同様である。

4.4.2 評価結果

ノード間通信帯域に関する性能評価の結果を Table 12 に示す。表は、通信にかかった時間、通信帯域、および送受信データサイズを示している。表に示す通信帯域は、次のように求めている。ここで、B は通信帯域、その他の変数はこれまでに述べたものと同じである。

$$B = \frac{M}{t} \tag{89}$$

ノード間通信帯域は、約 1.2 [GB / s]が HA8000 のネットワーク通信の利用可能な帯域であることがわかる。

```
call xclock( stime, 8 )!start

call mpi_barrier( mpi_comm_world, mpi_err )
call mpi_scatter( data, num_buf * jMax * kMax, mpi_double_precision, rbuf,
          &num_buf * jMax * kMax, mpi_double_precision, 0, mpi_comm_world, mpi_err )

call mpi_barrier( mpi_comm_world, mpi_err )
call mpi_gather( rbuf, num_buf * jMax * kMax, mpi_double_precision,
          &data( iStart * jMax * kMax ), num_buf * jMax * kMax, mpi_double_precision,
          &0, mpi_comm_world, mpi_err )

call xclock( etime, 8 )!stop
```

Fig. 45 ネットワーク帯域の性能評価用プログラム

Table 12 ネットワーク帯域の性能評価の結果

ノード数	2	4	8	16	32
時間 [sec.]	0.494	1.051	1.059	1.109	1.174
ネットワーク帯域 [GB / s]	1.508	1.063	1.232	1.259	1.229
送信データ量 [MB]	800	1200	1400	1500	1550

4.5 キャッシュメモリの性能評価

4.5.1 評価の条件

HA8000 のキャッシュメモリの性能を評価するにあたり、次のように条件を設定した。

・L1、L2、および L3 キャッシュメモリを評価するため、配列のサイズを複数用意する
・共有メモリの L3 キャッシュメモリを評価するため、複数のコアを使った場合も測定する

これらの条件に従って作成したコードを Fig. 46 に示す。利用したサブルーチンを前節同様 Table 6 に示す。このコードは、一方の配列からもう一方の配列にデータをコピーする。ここでは、1 つの CPU に限定して、1 つのプロセスで 1 コア利用する場合、2 つのプロセスで 2 コアずつ利用する場合、4 つのプロセスで 4 コア（1 つのプロセスは 1 コア）利用する場合で評価した。また、今回に限り、キャッシュメモリの性能を正確に評価するため、コンパイルオプションで最適化フラグを指定していない。

4.5.2 評価結果

キャッシュメモリの性能に関する性能評価の結果を Fig. 47 に示す。図に示す結果は、1 プロセスあたりにおける、ループ内での 1 回のデータコピーにかかる時間である。横軸はコピーしたデータのサイズ、縦軸はコピーにかかった時間を表している。共通して言えることは、L1 キャッシュメモリと L2 キャッシュメモリに性能差は見られないということである。256 [KB]付近までの時間は、どの場合も一定であり、L1 キャッシュメモリにデータが収まる場合と、L2 キャッシュメモリを使っている場合で性能差は見られない。性能評価に利用したコードは、2 つの配列を使うため、L2 キャッシュメモリの半分の容量となる 256 [KB]あたりで、メモリが枯渇し性能が低下し始める様子が見られる。これ以降、プロセスの数で差が出ている。4 つのプロセスを使う場合、L3 キャッシュメモリの容量である 2 [MB]の 4 分の 1 となる 512 [KB]あたりで性能低下が起こる。2 つのプロセスを使う場合は、600 [KB]を越えた付近で性能が低下し始め、1024 [KB]付近で性能の低下に変化が見られなくなる。1 つのプロセスだけを使う場合は、1024 [KB]を越えた付近で性能が低下し始め、1792 [KB]付近で性能の低下に変化が見られなくなる。

複数のプロセスを利用した場合、使用するメモリをプロセスごとに確保するため、プロセスが使用するデータは、異なるメモリ空間に存在する。プロセスで必要なデータがキャッシュメモリに存在しない場合、プロセスごとにキャッシュメモリのヒットミスが起こる。プロセスを 1 つだけ利用した場合、確保するメモリ空間は 1 つである。そのため、図が示すとおり、プロセス数に応じて、L3 キャッシュメモリ内のデータのヒットミスが起こる様子に変化が生じる。

前節までに示した Table 11 の結果について、ここで述べる。まず、1 つの CPU で、1 つのプロセスと 4 つのプロセスを利用した場合の違いだが、これまでに述べたとおり、プロセス数が異なることから、キャッシュメモリのヒットミスの差が性能差の原因である。次に、1 つの CPU 上で 1 つのプロセスを利用した場合と、4 つの CPU 上でそれぞれ 1 コアずつ利用した場合の違いについて述べる。4 つの CPU を利用する場合、各 CPU 上でのキャッシュメモリのヒットミスは、それぞれの CPU の L3 キャッシュメモリ上で起こり、L3 キャッシュメモリを 1 つのプロセスがすべて利用できることから、メインメモリへのデータ参照回数は 1 つの CPU 上で 1 プロセスを利用する場合と差はない。よって、この場合の性能の違いは、複数プロセスを利用したことによるオーバーヘッドが原因であると言える。この場合のオーバーヘッドとは、複数プロセスの起動処理、終了処理、同期処理である。

```
call xclock( stime, 8 )!start

do i = 1, iMax
        tmp_data( i ) = data( i )
enddo

call xclock( etime, 8 )!stop
```

Fig. 46 キャッシュメモリの性能評価用プログラム

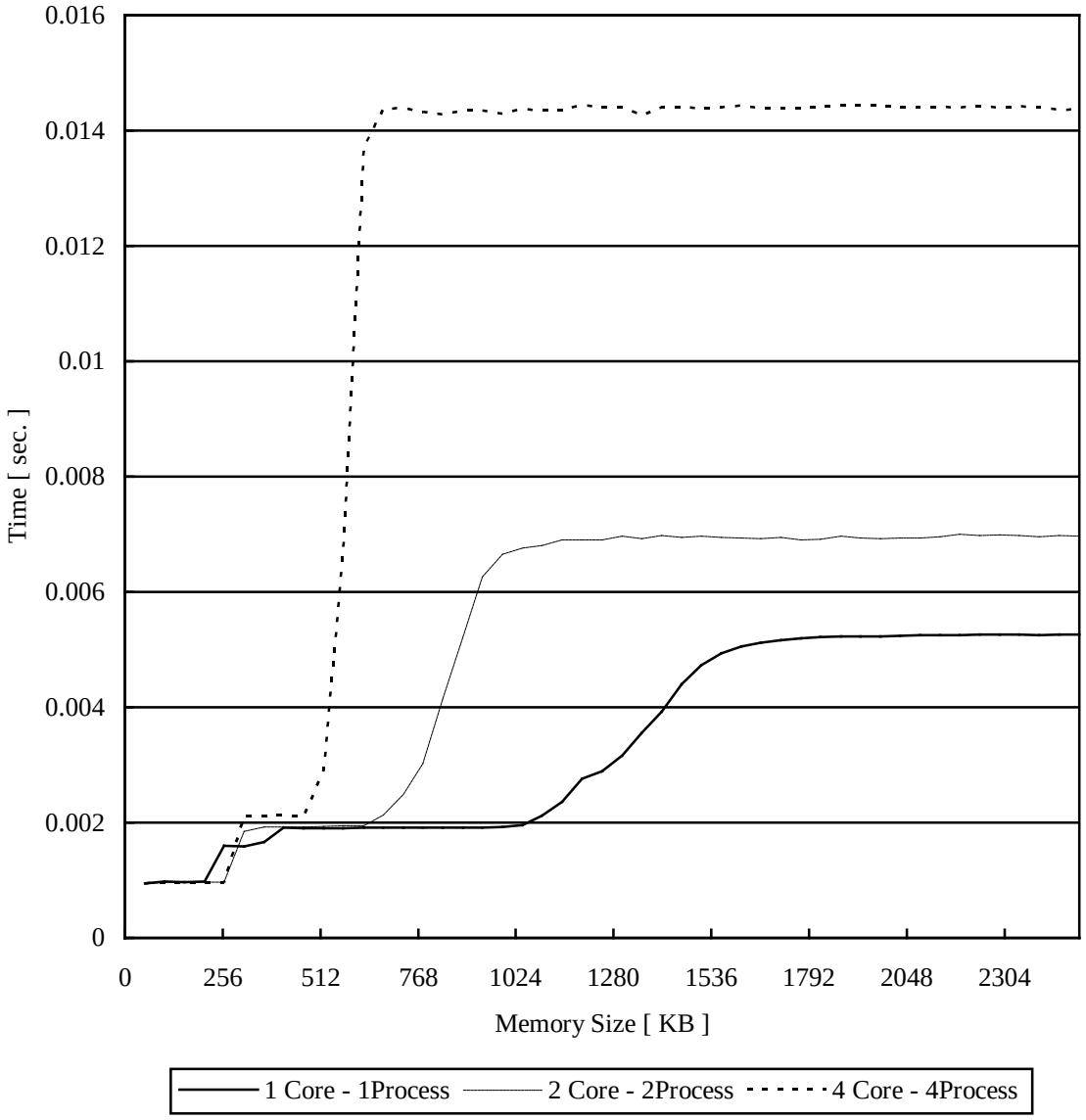


Fig. 47 キャッシュメモリの性能評価の結果

4.6 データサイズとメモリ帯域の関係

4.6.1 評価の条件

データサイズを変更した場合の HA8000 のメモリ帯域の変化を評価するため、次のように条件を設定した。

・メモリ帯域の性能評価と同じプログラムを利用する
・複数のコアを利用した場合も測定する

利用したプログラムを Fig. 44 に示す。また、これまでと同様に、使用したサブルーチンを Table 6 に示す。

4.6.2 評価結果

評価の結果を Fig. 48 に示す。図に示す結果は、横軸がデータサイズ、縦軸がメモリ帯域を表している。それぞれ、1 コア、1CPU 上で 4 プロセス、4CPU 上で 1 プロセスずつ利用した場合である。データサイズが小さい場合、前述のキャッシュメモリの影響により、性能差が大きい。キャッシュメモリの容量を超えた場合、性能が低下する。その後、メモリ帯域は上昇し、データサイズが十分大きくなった場合、前述までに示したメモリ帯域の性能評価のとおり、性能は一定となる。

1 コアと 4 コアを利用した場合の性能を比較するため、Fig. 49 に 1 コアに対する性能の倍率を示す。横軸は同様にデータサイズであり、縦軸は性能比である。2 [MB]あたりまでで見られるグラフの変化は、L2 キャッシュメモリとのアクセスの影響によるものと考えられる。また、8 [MB] あたりまでの変化は L3 キャッシュメモリとのアクセスの影響によるものであり、それ以降では、メインメモリとのアクセスによるものであると考えられる。4CPU を利用する場合は、L3 キャッシュメモリの量が 4 倍となるため、4 [MB]あたりから 8 [MB]まで性能差が緩やかに低下する。1 コアと 4 コアの性能は、データサイズが小さい場合は、L3 キャッシュメモリが性能に大きく影響を与え、データサイズが大きい場合は、メインメモリにおける帯域の性能に依存する。

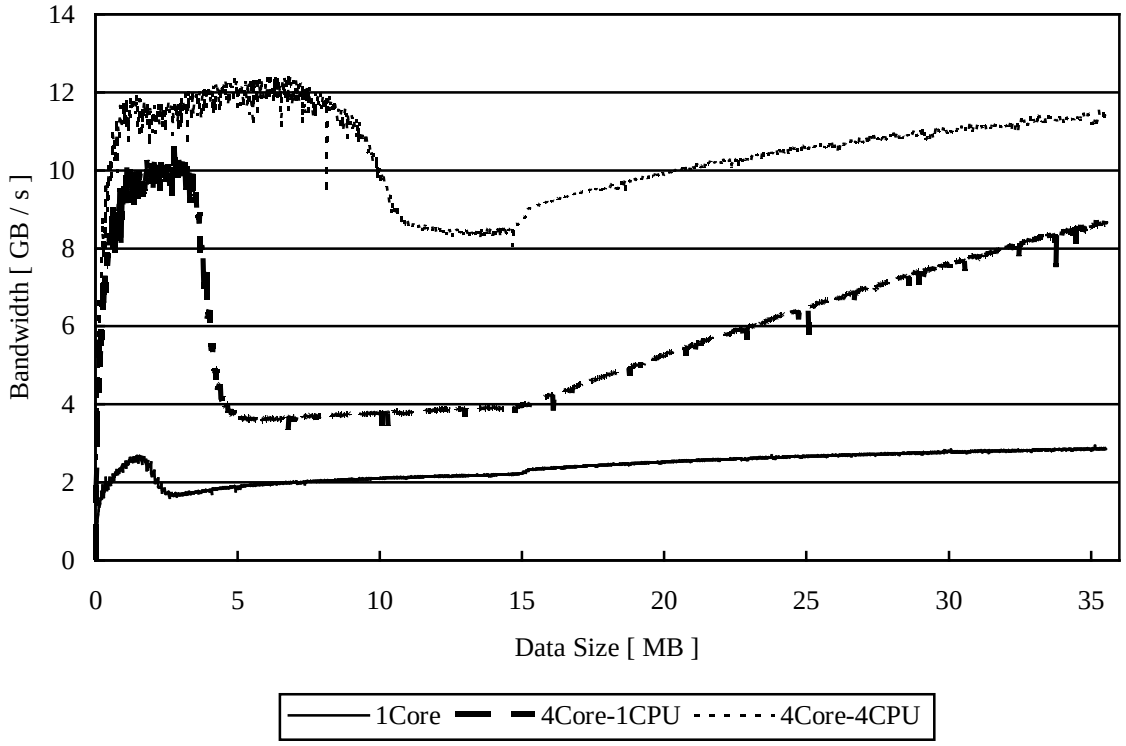


Fig. 48 データサイズを変更しつつ計測したメモリ帯域の性能評価の結果（1 コア、4 コア）

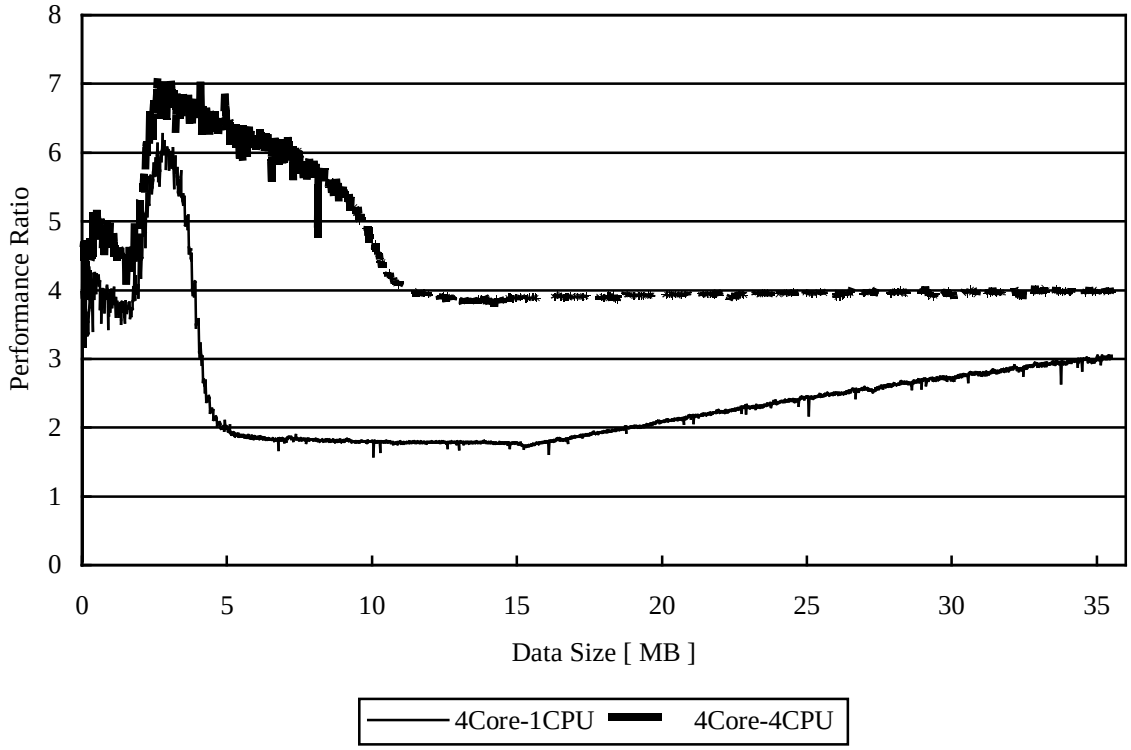


Fig. 49 1 コアのメモリ帯域の性能に対する 4 コアの場合の性能比

4.7 コード性能評価方法

4.7.1 評価概要

本報告で開発したコードの性能評価を行い、並列化の効果について考察する。性能評価を行うにあたり使用した環境は、HA8000 の性能評価をしたときと同様である。また、実行プログラムの構築には最大限の最適化を適用した。ただし、計算の正確さを確保するため、ソフトウェアパイプライン機能、配列のプリフェッチ機能を抑止した。Table 13 に使用した最適化オプションを示す。

4.7.2 評価手法

開発したプログラムの処理は、以下のような手順で行われる。

1) 初期化処理
2) 微分方程式の数値解析
3) 結果のファイル出力処理

「1)初期化処理」では MPI 通信の初期化、入力データ読み込み、定数の初期化、場の計算と Junction 域の計算間の接続情報の構築などを行う。プログラム起動時に一度だけ行われる。「2)微分方程式の数値解析」ではプログラムの主たる数値解析を行う。「3)結果のファイル出力処理」では、計算結果の出力、計算終了時にリスタートのための情報出力を行う。HA8000 のファイルシステムは、トラフィック（その時使用しているユーザの数など）の状況に依存して処理時間に数秒～数十秒の差が出るため「3) 結果のファイル出力処理」を除いた 1)～2)を計測範囲とした。「1)初期化処理」には入力データ読み込みでファイルアクセスがあるがファイルの大きさが 1 [Kbyte] 以下の小規模データ入力であり、影響は小さい。

Table 13 性能評価に用いたコンパイルオプション

オプション	効果
-Oss	最大最適化
-precise	命令のリオーダリングの禁止
-noprefetch	プリフェッチの禁止
-noswpl	ソフトウェアパイプラインの禁止
-noifswpl	if 文のソフトウェアパイプラインの禁止

4.8 コアの性能評価

1 ノード上での 1 コアとマルチコアの性能を評価する。プログラムを Table 14 に示した条件で実行し、1 コア、4 コアを使った場合について、それぞれ計算に要した時間を Table 15 に示す。4 コアの場合では、4 コアを 1 つのプロセスで利用した場合、1 コアを 1 つのプロセスによって 4 つの CPU 上で利用した場合を示す。Table 15 にある計算にかかる時間と MPI 通信にかかる時間の合計が「2)微分方程式の数値解析」の時間である。MPI 通信によるオーバーヘッドを把握するため、別々に計測した。また、計算時間と全体時間について、1 コアでの実行時間に対する性能向上の倍率を括弧内に示している。このとき、1 コアに対する倍率でコア数が同じにも関わらず、プロセスと CPU の数の違いで性能差が見られる結果となった。また、CPU を複数利用した場合、性能の倍率が理論値（コア数の倍率）を超えて高速化された。

ここで、上述の性能差の原因を調査するため、データサイズを変更して実行時間を計測した。その結果を Fig. 50 に示す。図の結果は、データを小さくしていった場合に、1 コア、1CPU で 4 プロセス、4CPU で 1 プロセスずつ利用した場合の「2)微分方程式の数値解析」の時間の変化を示す。それぞれ、ほぼ直線的に時間が変化した。ただし、データサイズがキャッシュメモリ上にすべて収まる場合の 8 [MB]未満では、曲線的な変化をしており、その状況を示すため、1 コアに対する性能差を Fig. 51 に示す。横軸は、同様にデータサイズであり、縦軸は性能比を示す。データサイズが十分小さい場合、性能差はおよそ 5 倍となった。その後は、L3 キャッシュメモリの容量の違いによって差が生じており、それ以降は、メインメモリの帯域の性能差となる。性能差の結果を見るに、概ねメモリ帯域の Fig. 49 に示した結果と同じような変化となっており、メモリ帯域の性能が、それぞれの性能差に影響を与えていると考えられる。

Table 14 性能評価に用いた入力値

領域長さ	X	128
	Y	128
	Z	20
境界条件	X	PML
	Y	PML
	Z	PML
シミュレーション時間		200

Table 15 1 コアと 4 コアを利用した場合のそれぞれの性能評価の結果

コア数		1	4	
CPU 数		1	1	4
分割された領域長さ (X, Y, Z)		(128, 128, 20)	(64, 64, 20)	
時間 [sec.]	初期化	0.189	0.142	0.120
	計算	171.363 (1.00)	72.125 (2.38)	36.862 (4.65)
	MPI 通信	0.078	0.280	0.523
	合計	172.630 (1.00)	72.547 (2.38)	37.515 (4.60)

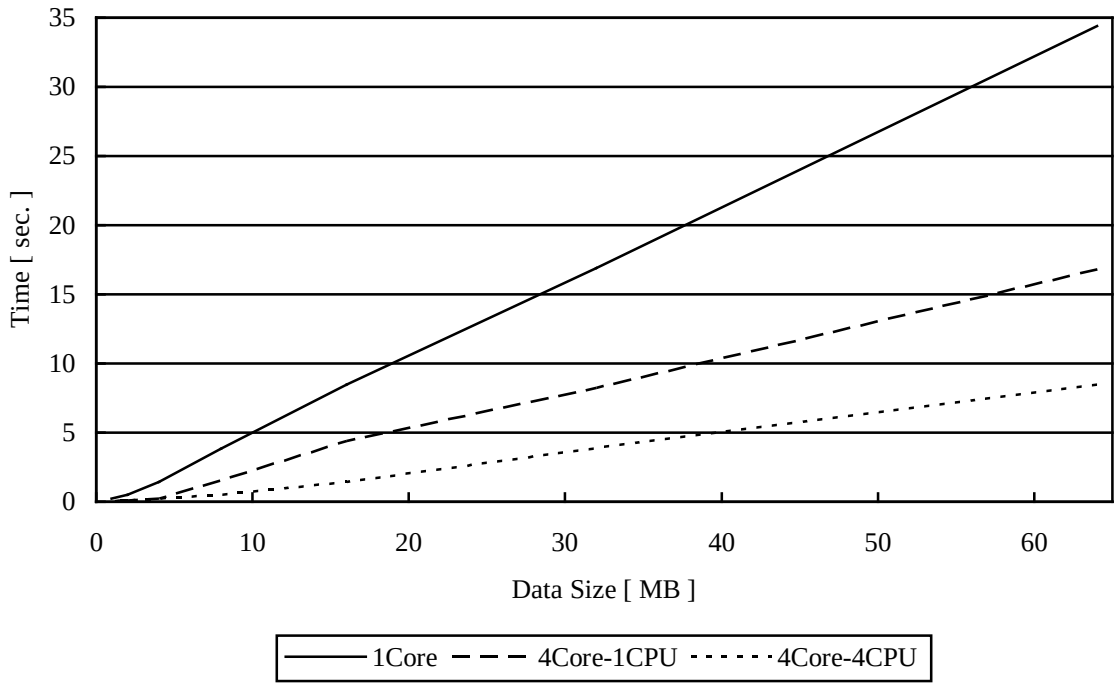


Fig. 50 データサイズを変更しつつ評価した 1 コアと 4 コアの性能差

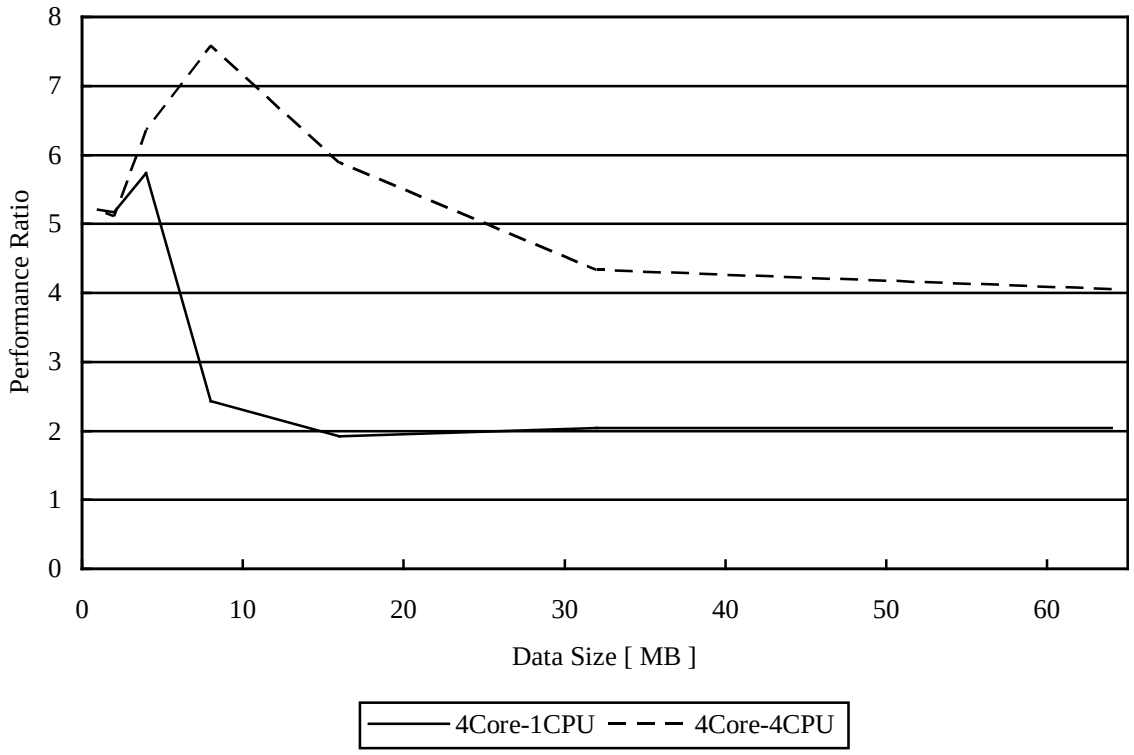


Fig. 51 1 コアに対する 4 コアの場合の性能比

4.9 低次（単ノード）の並列性能

1 ノード内での CPU 数の増加させた場合の性能評価を行う。ここでは、CPU ごとに 4 コアをすべて利用する。前節と同様、Table 14 の条件を用いて、1 つの CPU、2 つの CPU、4 つの CPU を利用してプログラムを実行した場合の実行時間を Table 16 に示す。また、1 つの CPU での実行時間を基準として、性能の倍率を示している。各 CPU 上では、4 つのプロセスで 4 コアを利用している。

結果から、1 ノード上でのデータ並列化による性能の向上度合いは、CPU の数に応じて向上すると言える。各プロセスは、動作する CPU に直結するメインメモリを利用しているため、異なる CPU のメインメモリを参照しない。前節の結果から、本報告で開発したプログラムの性能は、メモリへのデータ参照回数に依存するが、今回のように互いに影響を及ぼさない場合（CPU 上で動作する 4 つのプロセスは互いに影響を与えるが)、性能の倍率はコア数の増加に応じて推移する。

Table 16 低次並列時の性能評価の結果（4 コア、8 コア、16 コア）

コア数		4	8	16
CPU 数		1	2	4
分割された領域長さ (X, Y, Z)		(64, 64, 20)	(32, 64, 20)	(32, 32, 20)
時間 [sec.]	初期化	0.142	0.114	0.128
	計算	72.125 (1.00)	36.204 (1.99)	18.133 (3.98)
	MPI 通信	0.280	0.576	0.563
	合計	72.547 (1.00)	36.894 (1.97)	18.824 (3.85)

4.10 高次（複数ノード）の並列性能

複数のノード上での並列処理の性能評価を行う。Table 17 に示した条件で 16～1024 コアを利用して、プログラムを実行した時間を Table 18 に示す。16 コアを利用した場合を基準として、性能の倍率を示している。ここでは、オーバーヘッドと処理時間のスケールを考慮し、前節と同じ問題（入力データ）ではなく、問題サイズを変更して（大きくして）時間計測を行った。

Table 18 に示すように、並列化により処理時間は短縮されるものの、コア数に比例した台数効果は出ない。前節までに述べたとおり、本報告で開発したプログラムの性能は、メインメモリのデータ参照回数に大きく依存する。そこで、Table 19 に示すように、メモリ帯域の性能倍率とプログラムの性能倍率を比較する。性能倍率の数値は、それぞれ、Table 10 と Table 18 で求めた数値を用いた。この比較から、プログラムの性能は、メモリ帯域の性能に沿って推移していると判断できる。よって、これまで述べたとおり、本プログラムの性能は、メインメモリのデータ参照回数に依存し、複数ノードの場合、ネットワーク通信の影響が顕著である。

Table 17 性能評価に用いた入力値

領域長さ	X	320
	Y	320
	Z	320
境界条件	X	PML
	Y	PML
	Z	PML
シミュレーション時間		200

Table 18 高次並列時の性能評価の結果
上段：16 コア～128 コア
下段：256 コア～1024 コア

コア数		16	32	64	128
CPU 数		4	8	16	32
分割された領域長さ(X, Y, Z)		(160, 160, 80)	(160, 80, 80)	(80, 80, 80)	(40, 80, 80)
時間 [sec.]	初期化	1.13	0.79	0.80	1.13
	計算	1906.55 (1.00)	951.75 (2.00)	487.44 (3.91)	254.31 (7.50)
	MPI 通信	17.30	8.61	7.93	3.73
	合計	1924.98 (1.00)	961.15 (2.00)	469.17 (4.10)	258.81 (7.44)

コア数		256	512	1024
CPU 数		64	128	256
分割された領域長さ (X, Y, Z)		(40, 40, 80)	(20, 40, 80)	(20, 20, 80)
時間 [sec.]	初期化	1.80	3.32	10.79
	計算	127.83 (14.91)	65.44 (29.13)	32.76 (58.19)
	MPI 通信	5.28	5.18	4.57
	合計	134.91 (14.27)	73.94 (26.03)	48.12 (40.00)

Table 19 HA8000 のメモリ帯域性能と高次並列時の性能の比較

ノード数	1	2	4	8	16
メモリ帯域の性能	1.00	2.02	3.87	7.58	14.78
高次並列時の性能	1.00	2.00	4.10	7.44	14.27

5. 性能改善

前章で性能解析を行ったプログラムに対し、二つの性能改善を行った。本章では、その内容と結果を示す。

5.1 配列の要素間隔における性能劣化の改善

HA8000 では、各プロセッサの持つ 4 つのコアの L3 キャッシュメモリは、4 つのコアで共有されている。その共有メモリ上で各コアがデータにアクセスする場合に、データの配置によってバンクコンフリクトを起こす場合がある。ここでは、そのバンクコンフリクトを避けるための方法を述べる。また、その方法を実装したプログラムについて述べる。

HA8000 が採用しているプロセッサは AMD 社製の Opteron 8356[3)]である。このプロセッサの L3 共有メモリは、32 の独立したバンク（L3 キャッシュメモリを 32 個に分割し、各データ保存場所をバンクと呼ぶ）から構成されており、同じバンクにアクセスが集中しなければ、同時に 32 のデータアクセス命令を受け付けることができる。ただし、同じバンクにアクセスが集中すれば、並列にアクセスすることができなくなり、直列にアクセスすることとなる。この場合、プロセッサの持つ各コアはデータをコアごとに順番に受け取ることになり、並列性を大きく損なう。その状況をバンクコンフリクトと呼ぶ。L3 キャッシュメモリは 2 [MB]であり、32 個のバンクからなるので、1 個のバンクの大きさは、次のように求められる。

$$2048[KB] \div 32 = 64[KB] \tag{90}$$

64 [KB]ごとにバンクが構成されており、各コアが計算を行う上で、並列化しても同時に L3 キャッシュに 64 [KB]ごとにアクセスすればコンフリクトする。また、1 コアだけ使用する場合でも、同じ問題に対して対処が必要である。これは、CPU の演算時間よりもデータの転送のほうが時間はかかるためである。（CPU からデータ転送命令が多く出ても、キャッシュメモリに並列アクセスが可能ならば、データ取得はほぼ並列にできるが、同じバンクにアクセスする場合は CPU がデータ待ち状態となる）

例えば、double precision 型（8 [byte]）で要素数 256×1024 の配列に 4 コアでアクセスする場合、プログラムが単純にデータサイズを 4 分割すれば、コアがデータアクセスする際、64 [KB]の整数倍のサイズでアクセスしており、各コアが同じバンクにアクセスすることになる。各コアが同じ時間でプログラムを実行するとすれば、L3 キャッシュメモリにアクセスするたびにバンクコンフリクトを起こす。

バンクコンフリクトを回避するために配列間隔の取り方に注意を払うようにプログラムに関数を追加した。追加したプログラムを Fig. 52 に示す。なお、追加したプログラムは、16 の偶数倍となっている配列間隔を、奇数倍へと変換するプログラムである。（完全なバンクコンフリクト回避ではないが、これで概ね回避可能である。バンクコンフリクト回避のために、配列の要素を奇数倍にする方法はよく使われる。）

```
integer*4 function IArrayGap( inumber )

implicit none
integer*4 inumber
integer*4 itmp


itmp = ( inumber + 15 ) / 16
itmp = itmp + mod( itmp + 1, 2 )

IArrayGap = 16 * itmp


return
end
```

Fig. 52 バンクコンフリクト回避用プログラム

5.2 ループタイリングによるキャッシュヒット率の改善

5.2.1 改善内容

本報告で開発したコードの性能評価から、4 次のルンゲ・クッタ法による時間積分を行う際、非常に計算時間がかかることがわかる。この関数の高速化は、コード全体の高速化に大きく影響を与える。ここで、4 次のルンゲ・クッタ法の高速化について考える。

4 次のルンゲ・クッタ法の計算手順を Fig. 53 に示す。性能評価から、L3 キャッシュメモリの効率的な使用方法に着目した。図中の処理 C で算出される関数値は、電場 *E*、磁場 *B* の空間微分値（図中の処理 A、処理 B で算出）を参照する。この一連の処理において、処理 C が参照する空間微分値が L3 キャッシュメモリに存在していれば、処理 C の処理時間が短縮される。

Fig. 54 に空間微分値を算出する部分を抜粋したソースコードを示す。図中にある *ixSize*、*iySize*、*izSize* は *xyz* 空間のメッシュ点数であり、ループ中の演算の左辺は差分を求めて得た空間微分値である。差分の計算では、磁場の値(*bxk, byk, bzk*)が参照されている。処理 a、処理 b で参照される配列 *bzk* は、次ステップの処理 c、処理 d で参照されている。処理 a、処理 b で定義参照されるデータ長 N_1 は次のとおりである。

$$
\begin{aligned}
N_1 &= (A_1 + B_1 + C_1) \times 8[byte] \\
A_1 &= 2Array^{*1)} \times ixSize \times iySize \times izSize \\
B_1 &= 2Array^{*2)} \times ixSize \times iySize \times izSize \\
C_1 &= 2Array^{*3)} \times iySize \times izSize
\end{aligned}
\tag{91}
$$

**1) dxbyk, dxbzk*
**2) byk, bzk*
**3) xupper_byk, xupper_bzk*

N_1 が次の条件のとき、

$$N_1 \le 512[KB]$$

処理 c、処理 d で参照される *bzk* は 3 次キャッシュメモリに存在していることになり、処理時間の短縮が期待できる。

同様に処理 a から処理 d までの計算で参照されるデータ長 N_2 は次のとおりである。

$$
\begin{aligned}
N_2 &= (A_2 + B_2 + C_2 + D_2) \times 8[byte] \\
A_2 &= 4Array^{*1)} \times ixSize \times iySize \times izSize \\
B_2 &= 3Array^{*2)} \times ixSize \times iySize \times izSize \\
C_2 &= 2Array^{*3)} \times iySize \times izSize \\
D_2 &= 2Array^{*4)} \times ixSize \times izSize
\end{aligned}
\tag{92}
$$

**1) dxbyk, dxbzk, dybxk, dybzk*
**2) bxk, byk, bzk*
**3) xupper_byk, xupper_bzk*
**4) yupper_byk, yupper_bzk*

また、次の条件のとき、

$$N_2 \le 512[KB]$$

処理 e、処理 f で参照される *bxk,byk* は 3 次キャッシュに存在していることになり、処理時間の短縮が期待できる。

空間微分値の計算全体で参照されるデータ長を *N*、関数値の計算で参照されるデータ長を *M* とすると、次のとおりである。

$$\begin{aligned}
N &= (A+B+C+D+E)\times 8[byte] \\
M &= (A+F)\times 8[byte] \\
A &= 12Array^{*1)}\times ixSize\times iySize\times izSize \\
B &= 6Array^{*2)}\times ixSize\times iySize\times izSize \\
C &= 4Array^{*3)}\times iySize\times izSize \\
D &= 4Array^{*3)}\times ixSize\times izSize \\
E &= 4Array^{*3)}\times ixSize\times iySize \\
F &= 43Array^{*3)}\times ixSize\times iySize\times izSize
\end{aligned} \tag{93}$$

**1) 電場、磁場の空間微分値*
**2) 電場 x,y,z 方向 3 成分＋磁場 x,y,z 方向 3 成分*
**3) 電場、磁場の x 方向境界要素*
**4) 電場、磁場の y 方向境界要素*
**5) 電場、磁場の z 方向境界要素*
**6) 空間微分値以外のループ内の配列数*

空間微分計算から関数値計算までに参照されるデータ長 *L* が、

$$\begin{aligned}
L &= (A+B+C+D+E)\times 8[byte] \\
&= (61\times ixSize\times iySize\times izSize + 4\times (iySize\times izSize + ixSize\times izSize + ixSize\times iySize))\times 8[byte]
\end{aligned}$$

3 次キャッシャの容量に収まるよう、空間を複数の小領域に分割すること（ループタイリング）で、処理時間を短縮できる。

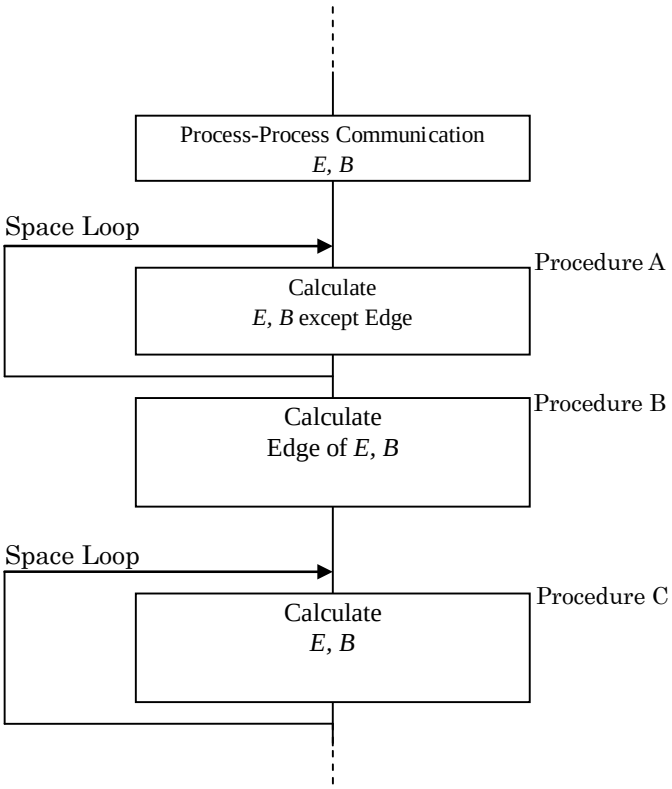


Fig. 53 ルンゲ・クッタ法の処理手順

```
do iz=1,nzsize                                                          a
   do iy=1,nysize
      do ix=1,nxsize-1
         dxbyk(ix,iy,iz)=(byk(ix+1,iy,iz)-byk(ix,iy,iz))/dx
         dxbzk(ix,iy,iz)=(bzk(ix+1,iy,iz)-bzk(ix,iy,iz))/dx
      end do
   end do
end do

do iz=1,nzsize                                                          b
   do iy=1,nysize
      dxbyk(nxsize,iy,iz)=(xupper_byk(iy,iz)-byk(nxsize,iy,iz))/dx
      dxbzk(nxsize,iy,iz)=(xupper_bzk(iy,iz)-bzk(nxsize,iy,iz))/dx
   end do
end do

do iz=1,nzsize                                                          c
   do iy=1,nysize-1
      do ix=1,nxsize
         dybzk(ix,iy,iz)=(bzk(ix,iy+1,iz)-bzk(ix,iy,iz))/dy
         dybxk(ix,iy,iz)=(bxk(ix,iy+1,iz)-bxk(ix,iy,iz))/dy
      end do
   end do
end do

do iz=1,nzsize                                                          d
   do ix=1,nxsize
      dybzk(ix,nysize,iz)=(yupper_bzk(ix,iz)-bzk(ix,nysize,iz))/dy
      dybxk(ix,nysize,iz)=(yupper_bxk(ix,iz)-bxk(ix,nysize,iz))/dy
   end do
end do

do iz=1,nzsize-1                                                        e
   do iy=1,nysize
      do ix=1,nxsize
         dzbxk(ix,iy,iz)=(bxk(ix,iy,iz+1)-bxk(ix,iy,iz))/dz
         dzbyk(ix,iy,iz)=(byk(ix,iy,iz+1)-byk(ix,iy,iz))/dz
      end do
   end do
end do

do iy=1,nysize                                                          f
   do ix=1,nxsize
      dzbxk(ix,iy,nzsize)=(zupper_bxk(ix,iy)-bxk(ix,iy,nzsize))/dz
      dzbyk(ix,iy,nzsize)=(zupper_byk(ix,iy)-byk(ix,iy,nzsize))/dz
   end do
end do
```

Fig. 54 rk4.DifferentiateField.F のサブルーチンの抜粋

5.2.2 ループタイリングの実装

高速化手法であるループタイリングを実装し、評価する。タイリングの空間分割方法は、各ノードに与えられる空間を *y* 軸と *z* 軸方向に分割している。

Table 20 に示す条件で実行したときの改善前と改善後の「2)微分方程式の数値解析」実行時間を、Table 21 に示す。また、改善後の高次並列化における実行時間と 16 コアでの実行時間に対する各性能倍率を、Table 22 に示す。全体的に速度向上が見られ改善がなされた。すなわち、ループタイリングが高速化に寄与した。

Table 20 性能評価に用いた入力値

領域長さ	X	320
	Y	320
	Z	320
境界条件	X	PML
	Y	PML
	Z	PML
シミュレーション時間		200

Table 21 ループタイリングを用いた場合の性能評価
実行時間上段：ループタイリングを用いない場合の時間
実行時間下段：ループタイリングを用いた場合の時間

コア数		16	32	64	128	256	512	1024
CPU 数		4	8	16	32	64	128	256
時間 [sec.]	非タイリング	1906.55	951.75	487.44	254.31	127.83	65.44	32.76
	タイリング	1553.32	797.02	407.81	207.45	105.83	55.77	27.92
性能比		1.23	1.19	1.20	1.23	1.21	1.17	1.17

Table 22 高次並列時の性能評価

上段：16 コア～128 コア

下段：256 コア～1024 コア

コア数		16	32	64	128
CPU 数		4	8	16	32
分割された領域長さ (X, Y, Z)		(160, 160, 80)	(160, 80, 80)	(80, 80, 80)	(40, 80, 80)
時間 [sec.]	初期化	1.09	0.80	0.75	1.02
	計算	1553.32 (1.00)	797.02 (1.95)	407.81 (3.81)	207.45 (7.49)
	MPI 通信	21.99	5.77	5.20	5.29
	合計	1576.40 (1.00)	803.59 (1.96)	413.76 (3.81)	213.76 (7.37)

コア数		256	512	1024
CPU 数		64	128	256
分割された領域長さ (X, Y, Z)		(40, 40, 80)	(20, 40, 80)	(20, 20, 80)
時間 [sec.]	初期化	1.76	3.66	7.48
	計算	105.83 (14.68)	55.77 (27.85)	27.92 (55.63)
	MPI 通信	5.01	5.50	4.37
	合計	112.60 (14.00)	64.93 (24.28)	39.77 (39.64)

6. おわりに

高温超伝導体素子のテラヘルツ発振をシミュレーションするプログラムの開発、並列化および性能評価について述べた。並列処理は、領域分割により、クラスタシステム HA8000 上で行った。ここでは、64 ノード（1024 並列）で計算した場合、単ノード（16 並列）と比較し、約 40 倍の高速化を実現した。この値は、次世代スパコンのような数十万台規模の超並列システムの利用を想定すると、やや低い値であるが、その理由は、本文で述べたように、HA8000 の基本性能の計測結果から、並列数増加に伴うメモリ帯域の性能低下にある。最適化においては、キャッシュメモリのヒット率の向上策を検討した。すなわち、ループタイリングによるチューニングを施した結果、64 ノードで計算した場合に単ノードと比較し約 49 倍の高速化を実現した。今回用いた入力モデルは、プログラム開発及び性能評価のための試験問題であり、プロダクション・ランにおいては、10^3 倍以上の規模の問題が想定されることから、より大きな計算粒度による、より高い並列化効率が期待できる。今後の作業において、この点（実問題の超並列処理への適応性）を評価する予定である。また、マルチコア・システムにおけるスレッド並列処理の効果についても調査する予定である。

謝辞

松本秀樹氏（東北大学）及び小山富男氏（同）には、テラヘルツ発振高温超伝導現象の基本原理についてご教授いただいたとともに、基礎プログラムをご提供いただいた。蕪木英雄氏（システム計算科学センター研究主席）及び村松健氏（システム計算科学センター長）には、査読において貴重な助言をいただいた。松本潔氏（情報システム利用推進室長）には、プログラム開発及び性能評価のための予算をお手配いただいた。以上の方々に、深謝いたします。

参考文献

1) L. Ozyuzer, A. E. Koshelev, C. Kurter, N. Gopalsami, Q. Li, M. Tachiki, K. Kadowaki, T. Yamamoto, H. Minami, H. Yamaguchi, t. Tachiki, K. E. Gray, W.-K. Kwok, U. Welp: “Emission of Coherent THz Radiation from Superconductors”, SCIENCE Vol. 318 1291(2007).

2) Tomio Koyama, Hideki Matsumoto, Masahiko Machida and Kazuo Kadowaki : “In-phase electrody -namics and terahertz wave emission in extended intrinsic Josephson junctions”, PHYSICAL REVIEW B79, 104522(2009).

3) 東京大学情報基盤センター: “HA8000　クラスタシステム　利用の手引き”, 東京大学情報基盤センター　スーパーコンピューティング部門, (2009)

4) Advanced Micro Devices Inc.: “Understanding Third - Generation AMD Opteron™ Processor Model Numbers”, available from http://www.amd.com/es-es/Processors/ProductInformation/0,,30_118_8796_15226, 00.html (accessed 2010-06-04)

付録 A ファイル一覧

開発したプログラムのファイル一覧を Table A.1 に示す。

Table A.1 ファイル一覧

ファイル名	内容
makefile	実行ファイル作成
param.ini	入力パラメータ設定
bin/makefile	実行ファイル作成（詳細設定）
header/Constants.h	各種定数の定義
header/Control.h	common 変数の定義
header/DefArraySize.h	配列サイズの定義
header/ElapsBuff.h	時間計測用変数の定義
header/ERRCODE.h	エラーコードの定義
header/FieldNode.h	Field ノードの common 変数の定義
header/Geometory.h	座標系の common 変数の定義
header/JunctionNode.h	Junction ノードの common 変数の定義
header/MPI_BoundaryComunicatior.h	MPI 通信用定数の定義
header/MPI_Stub.h	非 MPI 通信時の定数の定義
header/MPI_Wrapper.h	非 MPI 通信時のスタブの定義
header/ObservationFramework.h	配列ポインタ位置の定義
header/PrintParam.h	出力フォーマット定数の定義
header/ProcessInfo.h	プロセスの common 変数の定義
header/Utility.h	多目的関数の定義
source/betasige.F	配列準備
source/check.F	出力フォーマット指定
source/CheckCaller.F	データ領域内判定チェック
source/ClockTool.F	時間計測
source/CopyFieldVarsToJunction.F	Field ノードのデータを Junction ノードへコピー
source/CopySinVphiJunctionToField.F	位相差データをコピー
source/debug_module.F	デバッグ
source/Driver_ReadParameter.F	入力ファイル読み込み
source/InitFieldVariables.F	Field ノード初期化
source/InsideJunction.CalcVphiDerivative.F	位相差の微分
source/InsideJunction.CommVphiBoundary.F	境界領域上の位相差の計算
source/InsideJunction.F	Junction ノード計算
source/main.ExitInstance.F	メインプログラム（終了設定）
source/main.F	メインプログラム
source/main.InitInstance.F	メインプログラム（開始設定）
source/main.Run.F	メインプログラム（数値計算呼び出し）
source/MakeGridNodeInfo.F	ノード内のグリッド準備
source/matrixp.F	逆行列計算
source/MPI_BoundaryCommunicator.F	境界領域通信
source/MPI_Stub.F	MPI スタブ

source/MPI_Wrapper.F	スタブ呼び出しラッパー
source/NegotiateJunctionNodeRelation.F	Junction ノード間の計算
source/ObservationCallbacks.F	リスタートの初期化処理
source/ObservationFramework.F	リスタート関数
source/Output.F	出力データ処理
source/PMLUtility.F	PML 領域処理
source/PrintParam.F	出力データフォーマット
source/ReadParameter.F	データ読み込み
source/RegionPML.F	PML データコピー
source/ResumeFileIO.F	リスタートファイル処理
source/rk4.bcMetal.F	ルンゲ・クッタ法（超伝導素子上の計算）
source/rk4.CommFieldNodeBoundary.F	ルンゲ・クッタ法（共通領域のデータ送受信）
source/rk4.DifferentiateField.F	ルンゲ・クッタ法（空間微分）
source/rk4.F	ルンゲ・クッタ法（メイン）
source/rk4.PropagateBoundary.F	ルンゲ・クッタ法（反射領域の処理）
source/rk4.SetBoundaryCondition.F	ルンゲ・クッタ法（境界領域）
source/rk4.SetEzBarInphase.F	ルンゲ・クッタ法（電場の処理）
source/SetupProcessInfo.F	プロセス設定
source/SetupWorldParameter.F	座標系設定
source/Solver.F	数値解析（メイン）
source/UpdateJunctionRegion.F	Junction データ更新処理
source/Utility.F	多目的関数
source/zerof.F	初期位相差設定

付録 B 入力ファイル

開発したプログラムの入力ファイルの設定方法を示す。Table B.1 に示すパラメータをファイルに記述し、実行ファイルの引数として与えればよい。例として、全体が(x, y, z) = (128, 128, 20) の領域を持ち、超伝導素子が(x, y, z) = (51, 51, 8)の場合の設定方法を示している。領域長さの設定には、次の式を用いる。

$$\begin{aligned} &\text{全体の長さ} x = 2\times(iNPMx + iNvax) - \mathrm{mod}(iNNx + 1,2) \\ &\text{全体の長さ} y = 2\times(iNPMy + iNvay) - \mathrm{mod}(iNNy + 1,2) \\ &\text{全体の長さ} z = 2\times(iNPMz + iNvaz) - \mathrm{mod}(iNNz,2) \end{aligned} \tag{94}$$

Table B.1 入力パラメーター一覧

パラメータ	内容	例
iread	リスタート指定（0 の場合、リスタートしない）	0
iSuspendCount	リスタート終了時間	100
iSepX	*x* 方向空間分割数	2
iSepY	*y* 方向空間分割数	2
iSepZ	*z* 方向空間分割数	2
iNPMx	*x* 方向 NPM 領域長さ	52
iNPMy	*y* 方向 NPM 領域長さ	52
iNPMz	*z* 方向 NPM 領域長さ	5
iNvax	*x* 方向 PML 領域長さ	12
iNvay	*y* 方向 PML 領域長さ	12
iNvaz	*z* 方向 PML 領域長さ	5
iNNx	*x* 方向 Junction 領域長さ	51
iNNy	*y* 方向 Junction 領域長さ	51
iNNz	*z* 方向 Junction 領域長さ	8
itmax	シミュレーション時間	200
irepeat	繰り返し数（itmax 時間を繰り返す数を指定）	1
ipy	*y* 方向周期条件（0 の場合、反射壁がある）	0
ipz	*z* 方向周期条件（0 の場合、反射壁がある）	0
dt	時間刻み幅	0.4d-3
dx	*x* 方向空間刻み幅	1.0d-2
dy	*y* 方向空間刻み幅	1.0d-2
dz	*z* 方向空間刻み幅	1.0d-2
alpha	超伝導素子物性値	0.05d0
eta	超伝導素子物性値	1.0d4
beta	超伝導素子物性値	0.05d0
betaL	超伝導素子物性値	100.0d0
eps	超伝導素子物性値	10.0d0
epsL	超伝導素子物性値	1.0d0
epsv	超伝導素子物性値	1.0d0

付録 C ジョブファイル

HA8000 へのジョブ投入時に使用するファイルの設定方法[i]を示す。qsubコマンドに対して、ジョブファイル名は任意に指定できる。ただし、ジョブファイルは事前に実行許可を与えておく必要がある（chmodコマンドで与える）。Fig. C.1 にジョブファイルの例を示す。任意のシェルスクリプトを指定できる。1 行目には使用したいシェルのパスを指定する。3 行目～7 行目までは、ジョブのオプションである。3 行目は、ノードの種類（専用ノードで実行するか、デバッグノードで実行するか）を指定する。4 行目は、ノード数を指定する。5 行目は、1 ノードあたりのプロセス数を指定する。6 行目は、実行するプログラムの制限時間を指定する。指定した実行時間を超えた場合、プログラムは強制終了される。7 行目は、確保するメモリの最大容量を指定する。9 行目～10 行目は環境設定である。ユーザのホームディレクトリから、どの位置にあるディレクトリのプログラムを実行するか、また、1 コア上で実行されるスレッド数を何スレッドにするかを指定する。最後の行は、プログラムの実行の指示文となる。MPIを利用する場合、mpirunを指定する。引数には、NUMA制御用スクリプトファイル名と実行ファイル名を入力する（例は、スクリプトファイル名がnumaであり、実行ファイル名がbin/thz.exeである）。NUMA制御用スクリプトの例を Fig. C.2 に示す。1 行目は、前述と同様である。3 行目は、各プロセスのプロセス番号を取得して、nodeという変数に代入する。5 行目は、プロセス番号に応じて、使用するメインメモリを指定する。

```
#!/sbin/bash

#@$-q debug
#@$-N 1
#@$-J T1
#@$-lT 00:05:00
#@$-lm 1GB

cd $PBS_O_WORKDIR
export OMP_NUM_THREADS 1

mpirun ./numa bin/thz.exe
```

Fig. C.1 ジョブファイルの一例

```
#!/sbin/bash

set node = $MXMPI_ID

/usr/bin/numactl --membind $node --cpunodebind $node
```

Fig. C.2　NUMA 制御用スクリプトの一例

[i] 設定方法の詳細は、参考文献 3)に示される。

国際単位系（SI）

表１．SI 基本単位

基本量	SI 基本単位	
	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd

表２．基本単位を用いて表されるSI組立単位の例

組立量	SI 基本単位	
	名称	記号
面積	平方メートル	m^2
体積	立法メートル	m^3
速さ，速度	メートル毎秒	m/s
加速度	メートル毎秒毎秒	m/s^2
波数	毎メートル	m^{-1}
密度，質量密度	キログラム毎立方メートル	kg/m^3
面積密度	キログラム毎平方メートル	kg/m^2
比体積	立方メートル毎キログラム	m^3/kg
電流密度	アンペア毎平方メートル	A/m^2
磁界の強さ	アンペア毎メートル	A/m
量濃度(a)，濃度	モル毎立方メートル	mol/m^3
質量濃度	キログラム毎立法メートル	kg/m^3
輝度	カンデラ毎平方メートル	cd/m^2
屈折率(b)	（数字の） 1	1
比透磁率(b)	（数字の） 1	1

(a) 量濃度（amount concentration）は臨床化学の分野では物質濃度（substance concentration）ともよばれる。
(b) これらは無次元量あるいは次元１をもつ量であるが、そのことを表す単位記号である数字の１は通常は表記しない。

表３．固有の名称と記号で表されるSI組立単位

組立量	SI 組立単位			
	名称	記号	他のSI単位による表し方	SI基本単位による表し方
平面角	ラジアン(b)	rad	1 (b)	m/m
立体角	ステラジアン(b)	sr(c)	1 (b)	m^2/m^2
周波数	ヘルツ(d)	Hz		s^{-1}
力	ニュートン	N		m kg s^{-2}
圧力，応力	パスカル	Pa	N/m^2	m^{-1} kg s^{-2}
エネルギー，仕事，熱量	ジュール	J	N m	m^2 kg s^{-2}
仕事率，工率，放射束	ワット	W	J/s	m^2 kg s^{-3}
電荷，電気量	クーロン	C		s A
電位差（電圧），起電力	ボルト	V	W/A	m^2 kg s^{-3} A^{-1}
静電容量	ファラド	F	C/V	m^{-2} kg^{-1} s^4 A^2
電気抵抗	オーム	Ω	V/A	m^2 kg s^{-3} A^{-2}
コンダクタンス	ジーメンス	S	A/V	m^{-2} kg^{-1} s^3 A^2
磁束	ウエーバ	Wb	Vs	m^2 kg s^{-2} A^{-1}
磁束密度	テスラ	T	Wb/m^2	kg s^{-2} A^{-1}
インダクタンス	ヘンリー	H	Wb/A	m^2 kg s^{-2} A^{-2}
セルシウス温度	セルシウス度(e)	℃		K
光束	ルーメン	lm	cd sr(c)	cd
照度	ルクス	lx	lm/m^2	m^{-2} cd
放射性核種の放射能(f)	ベクレル(d)	Bq		s^{-1}
吸収線量，比エネルギー分与，カーマ	グレイ	Gy	J/kg	m^2 s^{-2}
線量当量，周辺線量当量，方向性線量当量，個人線量当量	シーベルト(g)	Sv	J/kg	m^2 s^{-2}
酸素活性	カタール	kat		s^{-1} mol

(a)SI接頭語は固有の名称と記号を持つ組立単位と組み合わせても使用できる。しかし接頭語を付した単位はもはやコヒーレントではない。
(b)ラジアンとステラジアンは数字の１に対する単位の特別な名称で、量についての情報をつたえるために使われる。実際には、使用する時には記号rad及びsrが用いられるが、習慣として組立単位としての記号である数字の１は明示されない。
(c)測光学ではステラジアンという名称と記号srを単位の表し方の中に、そのまま維持している。
(d)ヘルツは周期現象についてのみ、ベクレルは放射性核種の統計的過程についてのみ使用される。
(e)セルシウス度はケルビンの特別な名称で、セルシウス温度を表すために使用される。セルシウス度とケルビンの単位の大きさは同一である。したがって、温度差や温度間隔を表す数値はどちらの単位で表しても同じである。
(f)放射性核種の放射能（activity referred to a radionuclide）は、しばしば誤った用語で"radioactivity"と記される。
(g)単位シーベルト（PV,2002,70,205）についてはCIPM勧告2（CI-2002）を参照。

表４．単位の中に固有の名称と記号を含むSI組立単位の例

組立量	SI 組立単位		
	名称	記号	SI 基本単位による表し方
粘度	パスカル秒	Pa s	m^{-1} kg s^{-1}
力のモーメント	ニュートンメートル	N m	m^2 kg s^{-2}
表面張力	ニュートン毎メートル	N/m	kg s^{-2}
角速度	ラジアン毎秒	rad/s	m m^{-1} s^{-1}=s^{-1}
角加速度	ラジアン毎秒毎秒	rad/s^2	m m^{-1} s^{-2}=s^{-2}
熱流密度，放射照度	ワット毎平方メートル	W/m^2	kg s^{-3}
熱容量，エントロピー	ジュール毎ケルビン	J/K	m^2 kg s^{-2} K^{-1}
比熱容量，比エントロピー	ジュール毎キログラム毎ケルビン	J/(kg K)	m^2 s^{-2} K^{-1}
比エネルギー	ジュール毎キログラム	J/kg	m^2 s^{-2}
熱伝導率	ワット毎メートル毎ケルビン	W/(m K)	m kg s^{-3} K^{-1}
体積エネルギー	ジュール毎立方メートル	J/m^3	m^{-1} kg s^{-2}
電界の強さ	ボルト毎メートル	V/m	m kg s^{-3} A^{-1}
電荷密度	クーロン毎立方メートル	C/m^3	m^{-3} sA
表面電荷	クーロン毎平方メートル	C/m^2	m^{-2} sA
電束密度，電気変位	クーロン毎平方メートル	C/m^2	m^{-2} sA
誘電率	ファラド毎メートル	F/m	m^{-3} kg^{-1} s^4 A^2
透磁率	ヘンリー毎メートル	H/m	m kg s^{-2} A^{-2}
モルエネルギー	ジュール毎モル	J/mol	m^2 kg s^{-2} mol^{-1}
モルエントロピー，モル熱容量	ジュール毎モル毎ケルビン	J/(mol K)	m^2 kg s^{-2} K^{-1} mol^{-1}
照射線量（X線及びγ線）	クーロン毎キログラム	C/kg	kg^{-1} sA
吸収線量率	グレイ毎秒	Gy/s	m^2 s^{-3}
放射強度	ワット毎ステラジアン	W/sr	m^4 m^{-2} kg s^{-3}=m^2 kg s^{-3}
放射輝度	ワット毎平方メートル毎ステラジアン	$W/(m^2$ sr)	m^2 m^{-2} kg s^{-3}=kg s^{-3}
酵素活性濃度	カタール毎立方メートル	kat/m^3	m^{-3} s^{-1} mol

表５．SI 接頭語

乗数	接頭語	記号	乗数	接頭語	記号
10^{24}	ヨタ	Y	10^{-1}	デシ	d
10^{21}	ゼタ	Z	10^{-2}	センチ	c
10^{18}	エクサ	E	10^{-3}	ミリ	m
10^{15}	ペタ	P	10^{-6}	マイクロ	μ
10^{12}	テラ	T	10^{-9}	ナノ	n
10^9	ギガ	G	10^{-12}	ピコ	p
10^6	メガ	M	10^{-15}	フェムト	f
10^3	キロ	k	10^{-18}	アト	a
10^2	ヘクト	h	10^{-21}	ゼプト	z
10^1	デカ	da	10^{-24}	ヨクト	y

表６．SIに属さないが、SIと併用される単位

名称	記号	SI 単位による値
分	min	1 min=60s
時	h	1h =60 min=3600 s
日	d	1 d=24 h=86 400 s
度	°	1°=(π/180) rad
分	’	1’=(1/60)°=(π/10800) rad
秒	”	1”=(1/60)’=(π/648000) rad
ヘクタール	ha	$1ha=1hm^2=10^4m^2$
リットル	L, l	$1L=11=1dm^3=10^3cm^3=10^{-3}m^3$
トン	t	$1t=10^3$ kg

表７．SIに属さないが、SIと併用される単位で、SI単位で表される数値が実験的に得られるもの

名称	記号	SI 単位で表される数値
電子ボルト	eV	$1eV=1.602\ 176\ 53(14)\times10^{-19}J$
ダルトン	Da	$1Da=1.660\ 538\ 86(28)\times10^{-27}kg$
統一原子質量単位	u	1u=1 Da
天文単位	ua	$1ua=1.495\ 978\ 706\ 91(6)\times10^{11}m$

表８．SIに属さないが、SIと併用されるその他の単位

名称	記号	SI 単位で表される数値
バール	bar	$1\ bar=0.1MPa=100kPa=10^5Pa$
水銀柱ミリメートル	mmHg	1mmHg=133.322Pa
オングストローム	Å	$1\ Å=0.1nm=100pm=10^{-10}m$
海里	M	1 M=1852m
バーン	b	$1\ b=100fm^2=(10^{-12}cm)2=10^{-28}m^2$
ノット	kn	1 kn=(1852/3600)m/s
ネーパ	Np	SI単位との数値的な関係は、対数量の定義に依存。
ベル	B	
デジベル	dB	

表９．固有の名称をもつCGS組立単位

名称	記号	SI 単位で表される数値
エルグ	erg	$1\ erg=10^{-7}$ J
ダイン	dyn	$1\ dyn=10^{-5}$N
ポアズ	P	1 P=1 dyn s cm^{-2}=0.1Pa s
ストークス	St	1 St =$1cm^2$ s^{-1}=$10^{-4}m^2$ s^{-1}
スチルブ	sb	1 sb =1cd cm^{-2}=10^4cd m^{-2}
フォト	ph	1 ph=1cd sr cm^{-2} 10^4lx
ガル	Gal	1 Gal =1cm s^{-2}=$10^{-2}ms^{-2}$
マクスウェル	Mx	1 Mx = 1G cm^2=10^{-8}Wb
ガウス	G	1 G =1Mx cm^{-2} =10^{-4}T
エルステッド(c)	Oe	1 Oe ≙ $(10^3/4\pi)$A m^{-1}

(c) ３元系のCGS単位系とSIでは直接比較できないため、等号「 ≙ 」は対応関係を示すものである。

表10．SIに属さないその他の単位の例

名称	記号	SI 単位で表される数値
キュリー	Ci	$1\ Ci=3.7\times10^{10}Bq$
レントゲン	R	$1\ R=2.58\times10^{-4}C/kg$
ラド	rad	$1\ rad=1cGy=10^{-2}Gy$
レム	rem	$1\ rem=1\ cSv=10^{-2}Sv$
ガンマ	γ	1γ=1 nT=10-9T
フェルミ		1フェルミ=1 fm=10-15m
メートル系カラット		1メートル系カラット = 200 mg = 2×10-4kg
トル	Torr	1 Torr = (101 325/760) Pa
標準大気圧	atm	1 atm = 101 325 Pa
カロリー	cal	1cal=4.1858J（「15℃」カロリー），4.1868J（「IT」カロリー）4.184J（「熱化学」カロリー）
ミクロン	μ	1 μ =1μm=10^{-6}m

（第8版，2006年改訂）

この印刷物は再生紙を使用しています