陽的差分による3次元電磁流体シミュレーションのスーパコンピューティング†

鵜飼正行†† 津田孝夫†††

3次元電磁流体シミュレーションに対してベクトル計算機向けアルゴリズム設計について定量的に議論する．数値実験でベクトル計算機の実効性能を測定することにより，次の2点が重要な役割を果たすことが示される．まず，ベクトル長がベクトルレジスタの（分割された）各部分領域に入りうるデータ数に一致するとき，ロード／ストアに必要なデータ転送量が最小になり，実効性能は（ベクトル長を大きくした）飽和値よりむしろ高くなる．次に，メモリデータ構造がベクトルレジスタのデータ構造と整合するとき，ベクトルデータ転送速度が格段に上がり，実効性能は（ベクトル計算機 VP-400 を用いたとき）2〜3倍も向上する．

1. まえがき

スーパコンピュータの発展にともない大規模数値シミュレーションが近年盛んになってきており，物理，化学，工学の様々な定量科学の分野で重要な成果を上げてきた[1]．いわゆる計算物理の分野は独自の地位を築きつつあり，スーパコンピュータ応用として科学技術の発展に不可欠になってきている[2]．

数値シミュレーションが対象とする理工学システムは通常非常に複雑であり，膨大な計算量が必要となる．一方，スーパコンピュータの実効性能はプログラムで用いられるデータ構造やアルゴリズムに大きく依存することが知られている．したがって，大規模な数値シミュレーションが実用化されるためにはスーパコンピュータの実効性能を最大限引き出すようなアルゴリズム設計が本質的になってくる．このことはシステムの高速演算性能が増すほど重要になってくる．

本稿では具体的問題として電磁流体シミュレーションをとりあげる．電磁流体方程式はプラズマの巨視的振舞いを一般的に記述するものであり，巨大プロジェクトといわれる宇宙科学や核融合工学において基本的に重要な役割を果たす．プラズマシステムの複雑な挙動やその論理的因果関係を理解する上で数値シミュレーションがきわめて有力な（時として唯一の）手段となることが認識されており，宇宙プラズマ現象の解明や核融合炉解析に不可欠なものになっている[3]．

† Supercomputing for Three Dimensional MHD Simulations by an Explicit Scheme by MASAYUKI UGAI (Department of Computer Science, Faculty of Engineering, Ehime University) and TAKAO TSUDA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 愛媛大学工学部情報工学教室
††† 京都大学工学部情報工学教室

ベクトル計算機の実効性能を上げるための一般的アルゴリズム設計法はよく知られている．しかしながら，実際にプログラム開発を行う場合，具体的問題に対して実効性能を定量的に詳しく調べた議論が重要になってくる．本論文の目的は，3次元電磁流体シミュレーションのベクトル計算機による実効性能がアルゴリズム設計，特にメモリデータ構造，によってどの程度影響されるか定量的に詳しく調べ，具体的なプログラム開発を提案することにある．

2. 数値計算の基本的構造

一般に，s 個の従属変数をもつ偏微分方程式を陽的差分法を用いて数値計算する場合，その計算の構造は次のようになることが知られている．ある格子点における変数値を更新する場合，その格子点のまわりの（x, y, z 方向に対して対称になるように配置される）r 個の格子点における変数値（旧値）を引用して計算される．したがって，各格子点に s 個の変数がのっているから，全部で rs 個の旧値を引用して計算することが必要になる．この形の計算を全格子点にわたって繰返し（ループ）計算を行う場合，ループ中に s 個の式が連立される形になるが，引用データと（演算によって）定義されるデータとは完全に（右辺と左辺に）分離されており，各々の式は並列に処理することが可能である．したがって，この形の計算構造は100% ベクトル化することが可能であり，その意味でベクトル計算機向きであるといえる．

3次元電磁流体方程式は（無次元化した形で）次のような保存形に書ける[4]．

$$\partial \boldsymbol{U}/\partial t+\partial \boldsymbol{F}/\partial x+\partial \boldsymbol{G}/\partial y+\partial \boldsymbol{H}/\partial z=0 \qquad (1)$$

$$\boldsymbol{U}=(\rho, \rho u_x, \rho u_y, \rho u_z, B_x, B_y, B_z, \varepsilon) \qquad (2)$$

$$\boldsymbol{F}=[\rho u_x,\ \rho u_x^2+(P-B_x^2+B_y^2+B_z^2)/2,\ \rho u_x u_y-B_xB_y,\ \rho u_x u_z-B_xB_z,\ 0,\ u_xB_y-u_yB_x-\eta j_z,\ -u_xB_z+u_zB_x-\eta j_y,\ f] \quad (3)$$

$$\boldsymbol{G}=[\rho u_y,\ \rho u_x u_y-B_xB_y,\ \rho u_y^2+(P-B_y^2+B_z^2+B_x^2)/2,\ \rho u_y u_z-B_yB_z,\ -u_xB_y+u_yB_x+\eta j_z,\ 0,\ u_yB_z-u_zB_y-\eta j_x,\ g] \quad (4)$$

$$\boldsymbol{H}=[\rho u_z,\ \rho u_x u_y-B_xB_y,\ \rho u_y u_z-B_yB_z,\ \rho u_z^2+(P-B_z^2+B_x^2+B_y^2)/2,\ u_zB_x-u_xB_z-\eta j_y,\ -u_yB_z+u_zB_y-\eta j_x,\ 0,\ h] \quad (5)$$

ここに,

$$\varepsilon=P/(\gamma-1)+\rho(u_x^2+u_y^2+u_z^2)+B_x^2+B_y^2+B_z^2,$$

$$f=\gamma Pu_x/(\gamma-1)+\rho u_x(u_x^2+u_y^2+u_z^2)+2u_xB_y^2+2u_xB_z^2-2u_xB_xB_x-2u_yB_xB_y+2\eta(j_yB_z-j_zB_y),$$

$$g=\gamma Pu_y/(\gamma-1)+\rho u_y(u_y^2+u_z^2+u_x^2)+2u_yB_z^2+2u_yB_x^2-2u_zB_yB_z-2u_xB_yB_x+2\eta(j_zB_x-j_xB_z),$$

$$h=\gamma Pu_z/(\gamma-1)+\rho u_z(u_z^2+u_x^2+u_y^2)+2u_zB_x^2+2u_zB_y^2-2u_yB_zB_y-2u_xB_zB_x+2\eta(j_xB_y-j_yB_x),$$

であり，電流 $\boldsymbol{j}$ は $\boldsymbol{j}=\nabla\times\boldsymbol{B}$ から求められる（γ,η はそれぞれ比熱比，抵抗率であり，$\rho, P, \boldsymbol{u}, \boldsymbol{B}$ は各々プラズマ密度，圧力，流速，磁場である）．この場合変数ベクトル $\boldsymbol{U}$ は8個の成分からなることがわかる（$s=8$）．

3次元電磁流体方程式（1）を代表的な陽的差分である 2 step Lax-Wendroff 法を用いて数値計算すると，時間 $t=n\Delta t$ における変数値 $\boldsymbol{U}^n$ を用いて，ある格子点 (i,j,k) において，時間ステップ Δt 後の値 $\boldsymbol{U}^{n+1}$，および $2\Delta t$ 後の値 $\boldsymbol{U}^{n+2}$ を計算するのに次の2段階が必要になる[5]．

$$\boldsymbol{U}^{n+1}_{i,j,k}=\boldsymbol{U}^n_{i,j,k}-(\Delta t/2\Delta x)\times(\boldsymbol{F}^n_{i+1,j,k}-\boldsymbol{F}^n_{i-1,j,k})-(\Delta t/2\Delta y)(\boldsymbol{G}^n_{i,j+1,k}-\boldsymbol{G}^n_{i,j-1,k})-(\Delta t/2\Delta z)(\boldsymbol{H}^n_{i,j,k+1}-\boldsymbol{H}^n_{i,j,k-1}) \quad (6)$$

$$\boldsymbol{U}^{n+2}_{i,j,k}=\boldsymbol{U}^n_{i,j,k}-(\Delta t/\Delta x)\times(\boldsymbol{F}^{n+1}_{i+1,j,k}-\boldsymbol{F}^{n+1}_{i-1,j,k})-(\Delta t/\Delta y)(\boldsymbol{G}^{n+1}_{i,j+1,k}-\boldsymbol{G}^{n+1}_{i,j-1,k})-(\Delta t/\Delta z)(\boldsymbol{H}^{n+1}_{i,j,k+1}-\boldsymbol{H}^{n+1}_{i,j,k-1}) \quad (7)$$

ここに，3次元空間は x 方向に N_x 個，y 方向に N_y 個，z 方向に N_z 個の格子点に分割され，Δx, Δy, Δz はそれぞれ x, y, z 方向のメッシュ幅を示す（$1\leqq i\leqq N_x$, $1\leqq j\leqq N_y$, $1\leqq k\leqq N_z$）．また(6)式で $\boldsymbol{U}^n_{i,j,k}=(\boldsymbol{U}^n_{i+1,j,k}+\boldsymbol{U}^n_{i-1,j,k}+\boldsymbol{U}^n_{i,j+1,k}+\boldsymbol{U}^n_{i,j-1,k}+\boldsymbol{U}^n_{i,j,k+1}+\boldsymbol{U}^n_{i,j,k-1})/6$ であり，$\boldsymbol{F}^n_{i,j,k}$, $\boldsymbol{F}^{n+1}_{i,j,k}$ はそれぞれ $\boldsymbol{F}(\boldsymbol{U}^n_{i,j,k})$, $\boldsymbol{F}(\boldsymbol{U}^{n+1}_{i,j,k})$ を示す（$\boldsymbol{G}$, $\boldsymbol{H}$ についても同様）．(6)，(7)式の計算を実行することにより内部格子点領域 $3\leqq i\leqq N_x-2$, $3\leqq j\leqq N_y-2$, $3\leqq k\leqq N_z-2$ で変数値が更新される．

電磁流体方程式の場合(6)および(7)式は各々8個の連立方程式になる．(6)式の計算の構造を**図1**に示す［ある格子点 (i,j,k) における変数値を例えば $\boldsymbol{U}^n(i,j,k)$ の形で表し，$\boldsymbol{F}(\boldsymbol{U}^n)$ の m 番めの成分を $F_m{}^n(i,j,k)$ 等で示している］．この図は引用データ $\boldsymbol{U}^n$ から $\boldsymbol{U}^{n+1}$ の m 番目の成分 $U_m{}^{n+1}$を求める処理過程（変換公式）を階層的に示している（$m=1,\ldots,8$）．この図から明らかなように，ある格子点 (i,j,k) において $\boldsymbol{U}^{n+1}$ を求めるために，その格子点のまわりの6点の格子点における $\boldsymbol{U}^n$ の値を引用しており，各々の格子点には8個の変数値がのっているので結局 48 個のデータを必要とすることがわかる．(7)式の数値計算においても，その計算構造は引用するデータが違うだけで(6)式の場合（図1）とほとんど同じになる．

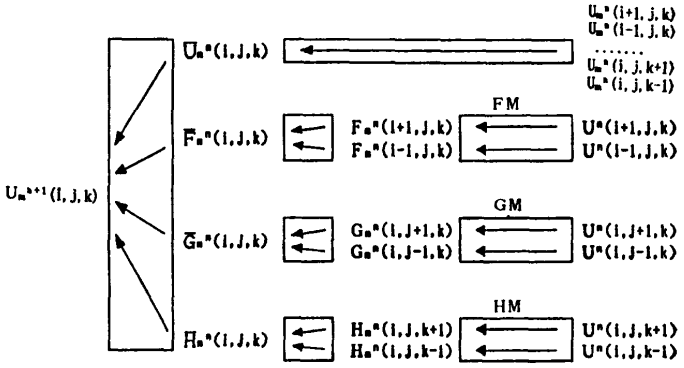


図1 格子点 (i,j,k) において U^{n+1} の m 番目の成分 $U_m{}^{n+1}(i,j,k)$ を求める処理過程を示す数値計算〔(6)式〕の階層的構造，ここで $F_m{}^n(i,j,k)=F_m{}^n(i+1,j,k)-F_m{}^n(i-1,j,k)$ であり $G_m{}^n$, $H_m{}^n$ についても同様（$m=1,2,\cdots,8$）

Fig. 1 Structure of the numerical computations [Eq. (6)] from which $U_m{}^{n+1}$ is obtained, where $F_m{}^n(i,j,k)=F_m{}^n(i+1,j,k)-F_m{}^n(i-1,j,k)$, and so forth ($m=1,2,\cdots 8$).

3. スーパコンピューティングのためのアルゴリズム設計

シミュレーションを実行するために基本的な操作を次のように分類しよう．(a)計算に必要なパラメータ(時間ステップ，格子点数，等)を与える．(b)初期条件を与える．(c)境界条件を与える．(d)(6)式の計算を内部格子点領域で実行する．(e)(7)式の計算を内部格子点領域で実行する．(f)数値計算結果を(図形等で)出力する．(g)数値計算結果の正当性を検討する．このうち(a)，(b)の操作は最初に1回実行されるだけであり，また(f)，(g)の操作は変数値を更新するための処理とは直接関係ない．実行処理ステップの主要部分は(c)，(d)，(e)の操作であり，この部分を1回処理することにより $2\Delta t$ 後の結果が得られる(電磁流体シミュレーションの具体的な処理過程は文献4)に詳しく記述してある)．

シミュレーションからなんらかの有効な結論を導き出すためには，典型的に数千ステップ程度の計算が必要となる．この場合，(d)と(e)の処理に必要な(CPU)時間はほぼ同じであり，また(総格子点数が極端に少なくない限り)(c)の処理時間は(d)もしくは(e)の処理時間の 1/10 以下となることが経験的にわかっている．したがって，以下の議論では(d)の処理過程(図1)を中心に議論するが(e)の処理過程についても同様のことが成立する．

一般に，ベクトル加速性能を引き出すためには次の点が基本的であることが知られている．(i)ベクトル化率を上げる．(ii)ベクトル長を大きくとる．(iii)ベクトルデータの効率良い(ロード／ストア)転送を実現する．(iv)演算パイプラインの並列稼動効率の向上のために(ループ中の)演算ステップ数(演算密度)を十分大きくとる．以上のことを念頭に置いてアルゴリズム設計を試みる．

3.1 数値計算アルゴリズム

(6)式の数値計算を内部格子点領域にわたって実行するためにここでは次のような基本的アルゴリズムを考える．まず，全格子点($N_{all}=N_xN_yN_z$ 個)上の変数値(各々の格子点に8個の変数がのっている)を格納するための配列 U を定義し，$t=n\Delta t$ における変数値 $\boldsymbol{U}^n$ が格納されているものとする．また，(6)式を実行した結果(すなわち変数値 $\boldsymbol{U}^{n+1}$)を格納するための作業領域として配列 V を定義する(ここで定義された配列 U と V はメモリの中の領域名を表しており，前述の物理変数ベクトル $\boldsymbol{U}$ とはおのずから区別される)．図1に示された数値計算構造より，基本的モジュールとして，ある格子点 (i,j,k) において物理変数値 $\boldsymbol{U}$ から $\boldsymbol{F}(\boldsymbol{U})$ を計算するモジュール FM，$\boldsymbol{G}(\boldsymbol{U})$ を計算するモジュール GM，$\boldsymbol{H}(\boldsymbol{U})$ を計算するモジュール HM を各々(3)，(4)，(5)式にもとづいて作成する．このアルゴリズムを実行した段階では内部格子点領域，$2\leqq i\leqq N_x-1$，$2\leqq j\leqq N_y-1$，$2\leqq k\leqq N_z-1$，において $\boldsymbol{U}^{n+1}$ が配列 V に格納される．

(7)式の数値計算も全く同じようにして実行することができる．ただし，この場合はモジュール FM，GM，HM で引用されるデータは $\boldsymbol{U}^n$ の代わりに $\boldsymbol{U}^{n+1}$ を用いる．この段階が実行されると配列 U の内部格子点領域，$3\leqq i\leqq N_x-2$，$3\leqq j\leqq N_y-2$，$3\leqq k\leqq N_z-2$，において新値 $\boldsymbol{U}^{n+2}$ が格納されたことになる．この格子点領域のまわりの格子点における変数値は境界条件によって与えられ，結局最終的値 $t=(n+2)\Delta t$ の変数値が配列 U の全格子点に格納されることになり，再び(6)式の処理が開始される．

以上のアルゴリズムから明らかなように，図1で示された計算が内部格子点領域の各格子点で繰り返し実行されることになり，この計算がベクトル演算の対象になる．また計算の演算密度(図1)は非常に大きく，パイプライン演算器の多重度に応じた効率良いベクトル演算の並列化が期待できる．

3.2 メモリデータの配列構造

数値計算ではメモリに格納された膨大な数値データをベクトルレジスタに順次ロードしながらベクトル演算が実行される．したがって，ベクトルデータのロード／ストアに要する時間がベクトル加速性能に大きく影響を及ぼすことが知られている．ベクトルデータの効率良い転送はベクトルレジスタにおけるデータの割付けやメモリデータの配列構造に基本的に依存する．メモリデータの配列構造はプログラムの中で定義されうるものであり，ここでは(フォートラン形式の)次のような(空間的に3次元の)配列構造を考える．

CASE (A) : $U(8, N_x, N_y, N_z)$

この場合，各格子点 (i,j,k) において，8個の物理変数値が $U(m,i,j,k)=U_m$ $(m=1,2,\cdots,8)$ となるように連続的に配置されている．

CASE (B) : $U(9, N_x, N_y, N_z)$

上述の(A)の配列の場合，配列の最初の数が $8=2^3$ であり，一方メモリはパイプラインによる(並列的

な）高速データアクセスを実現するため領域を 2^n 個のバンクに分割しているので，(A)の配列構造ではいわゆるバンク競合が起きる可能性が増す．したがって，バンク競合を避けるために配列の最初の数を9にしてある．

CASE (C) : $U(N_x, N_y, N_z, 8)$

この場合，8個の物理変数各々に対して全格子点での値が格納されており，物理変数の順序に従って連続的に配置されている．

3.3 3重ループの1重化

3次元的データ配列構造（3.2節）では，格子点全体にわたる数値計算のプログラムは**図2**(a)に示されるように3重ループになり，ベクトル化の対象になるのは最内側のループとなる．したがって，ベクトル長をベクトル演算の繰返し回数と定義するとこの場合 N_x-2 がベクトル長となる．そこで，3重ループを1重ループに変換することによりベクトル長を大きくすることを考える．このために3次元的格子番号 (i, j, k) を1次元的格子番号 nn に変換する公式を次のように定義する．

$$nn(i, j, k)=(k-1)N_xN_y+(j-1)N_x+i \qquad (8)$$
$$(1\leqq i\leqq N_x,\ 1\leqq j\leqq N_y,\ 1\leqq k\leqq N_z)$$

この変換公式により，3次元配列はすべて1次元配列に変換することができ，図2(a)で示された3重ループは図2(b)のような1重ループに簡単に変換することができる．この場合，プログラムの変更に必要な操作は簡単であり，プログラム中で配列要素を指定する添字 (..,*i, j, k*,..) の部分を (..,*n*,..) のように，1次元的格子番号 n で置き換えるだけでよい．

図2(b)のような1重ループに変換することによりそのベクトル長（繰返し回数）は $nn(N_x-1, N_y-1, N_z-1)-nn(2, 2, 2)+1$ となるが通常 N_x, N_y, N_z は1より十分大きくとるので，3重ループにおけるベクトル長 N_x-2 よりはるかに大きくなる．また，$i=1$ もしくは $j=1$ の部分（境界の部分に相当する）においては正しい計算はなされないが，繰返し計算の終了後に改めて境界条件にもとづいて正しい値が入るので問題ない．

```
(a)
DO 10 k=2, Nz-1
DO 10 j=2, Ny-1
DO 10 i=2, Nx-1
U1n+1(i, j, k)=f1(Un)
U2n+1(i, j, k)=f2(Un)
      ......
      ......
U8n+1(i, j, k)=f8(Un)
10 CONTINUE
```

```
(b)
DO 10 n=nn(2, 2, 2), nn(Nx-1, Ny-1, Nz-1)
  U1n+1(n)=f1(Un)
  U2n+1(n)=f2(Un)
     ....
     ....
  U8n+1(n)=f8(Un)
10 CONTINUE
```

図2 (6)式で与えられる数値計算処理のためのフォートラン的 DO LOOP，(a) 3重ループの場合，(b) 1重ループの場合
Fig. 2 Fortran-like DO LOOP for the computations given by (6) for the cases (a) triple nested loop and (b) single loop.

4. 数値実験

前章で述べたアルゴリズムに応じてプログラムを作成し，各々の場合のベクトル演算処理時間（速度）を測定する．ここではベクトル加速性能を評価する基準を次のように定義する．図2で示されたループ計算が実行される格子点の総数を N_p とし，その N_p 個の格子点にわたる繰返し計算にかかる全 CPU 時間を T_p とする．そのとき，格子点1個あたりで（図1に示される）計算に必要な平均的処理時間は $t_0=T_p/N_p$ となる（T_p は図2で DO ループの前後で CPU 時間を測定しその差をとることによって与えられる）．ここでは t_0 の逆数をベクトル演算速度とみなし，ベクトル計算機の実効性能と定義する．

数値実験には，ベクトルレジスタ方式をとるシステムとして代表的な富士通社の VP-400 と VP-200 を使用した．ハードウェア的には，VP-400 は VP-200 に比べてベクトル演算能力とベクトルレジスタ容量が倍になるがロード／ストアのためのデータ転送能力は変わらない．また以下に示すすべての数値計算例でベクトル化率は 99.9% 以上となった．

まず，3重ループの場合［図2(a)の形］に対して調べる．この場合，ベクトル長は N_x-2 になり，計算が実行される全格子点数は $N_p=(N_x-2)(N_y-2)(N_z-2)$ となる．数値計算は次のような場合に対して実行する．(i)計算機システムが VP-200, VP-400 の場合，(ii)全格子点数 $N_{all}=N_xN_yN_z$ が約 10 万，50 万，100 万の場合，(iii)メモリデータ構造が（3.2節で定義された）(A)，(B)，(C)の場合．これらすべての場合の組合せに対して N_x, N_y, N_z を広範囲に変化しながら T_p を測定した．

まず，一般的に次のことを測定結果から確認している．すなわち，計算機システム，メモリデータ構造を一定にしたとき，ベクトル演算処理時間 t_0 はベクトル長 N_x-2 にのみ依存し，N_{all}, N_y, N_z には依存し

ない．さらに，ベクトル長が小さい（10^2 以下）範囲では t_0 は急激に減少するが，ベクトル長がある程度以上になると変動がほとんどなくなり各データ構造に応じて一定値になる（これを飽和値と呼ぼう）．これらの結果は一般的によく知られているベクトル計算機の基本的特性に一致する．

より定量的に調べるために，計算機システムおよびメモリデータ構造［3.2 節で(A)，(B)，(C)の3通り］のとりかたに応じて t_0 がベクトル長 N_x-2 にどのように依存するかを測定した結果を図3に示す．この図で，横軸は対数スケールで示し，縦軸は t_0 を μ 秒（10^{-6} 秒）の単位で示す．この図から直ちに次のことが観測できる．最初に，ほとんど周期的に t_0 が不連続的に変化する．その周期はデータ構造には依存しないが計算機システムによって異なる．例えば VP-200 の場合ベクトル長が 64 から 65 に，128 から 129 に変わるとき，VP-400 の場合ベクトル長が 128 から 129 に，256 から 257 に変わるとき不連続性が顕著に現れる．

次に，メモリデータ構造のとりかたに対して実効性能は大きく影響されることが観測される．VP-200 では(B)のデータ構造の方が(A)の場合より演算速度が速くなり，(C)の場合は(B)の場合とほとんど変わらない．VP-400 の場合，(C)のデータ構造の場合が演算速度が最も速く，(B)，(A)の順に遅くなる．

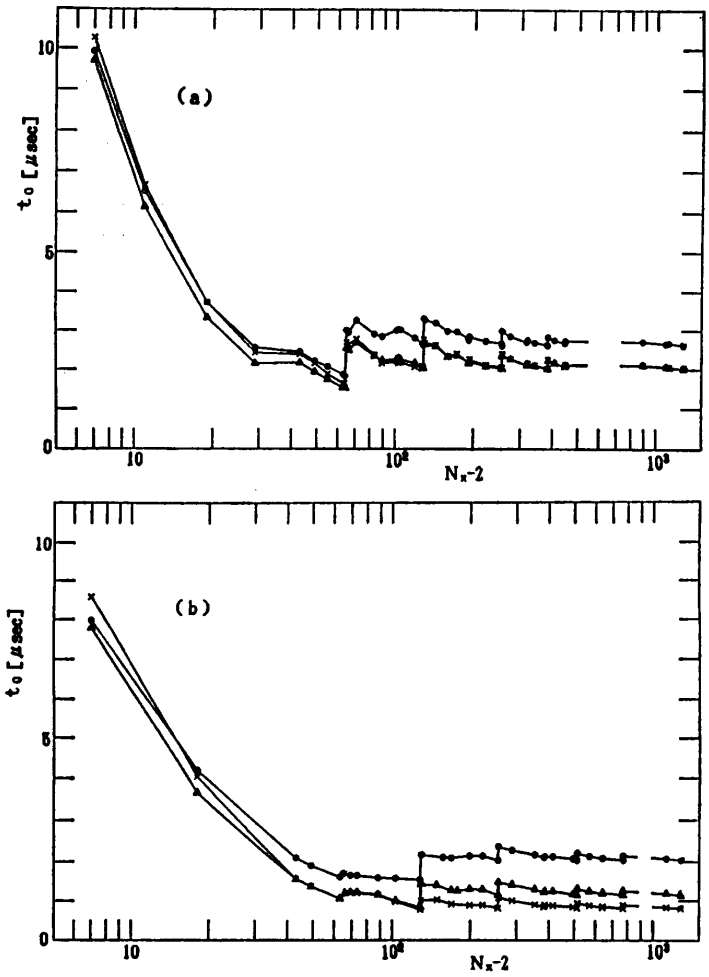


図 3 3重ループによる計算でメモリデータ構造が(A)（○で示す），(B)（△で示す），(C)（×で示す）の各場合においてベクトル長 N_x-2 に対する演算処理時間 t_0(μsec) の依存性，(a)システム VP-200 の場合と(b)VP-400 の場合

Fig. 3 Dependence of the CPU time t_0 on the vector length N_x-2 for the triple nested loop with the memory data structures (A) (denoted by ○), (B) (denoted by △), and (C) (denoted by ×) for the systems (a) VP-200 and (b) VP-400.

さらに注目すべき点は，いずれの場合でも VP-200 ではベクトル長が 64 のとき，VP-400 ではベクトル長が 128 のとき演算速度は最大になり，その（ベクトル長を大きくとった）飽和値より実効性能はむしろ高くなっている．例えば，VP-400 で(C)のデータ構造ではベクトル長が 128 のとき t_0〜0.790 となり飽和値よりやや小さい（図3）（すなわち実効性能は高い）．

1重ループの場合［図2(b)］に対しても上述の(i)，(ii)，(iii)の各々の場合のすべての組合せに対して t_0 を測定した．この場合は N_y はベクトル長［$=nn(N_x-1, N_y-1, N_z-1)-nn(2, 2, 2)+1$］と等しくなり，$N_x$, N_y, N_z が十分大きいならこれは全格子点数にほぼ等しくなる．また，データ構造(A)もしくは(B)では，数値計算構造（図1）から明らかなように，引用データのメモリアドレス間の離れぐあい（データ間距離）の最大値は N_xN_y に比例する［(8)式］．

数値実験結果は次のようにまとめられる．まず，データ構造，システム，N_{all} を一定にした場合，N_x, N_y を $N_x=5$, $N_y=5$ を最小値として広範囲に変化しても演算処理時間 t_0 は変化しなかった．このことは最大引用データ間距離に実効性能が影響されないこと

表1 1重ループによる計算でメモリデータ構造(A), (B), (C)と計算機システム VP-200, VP-400, M 780 の各場合に対して測定された演算処理時間 t_0(μsec)

Table 1 CPU times t_0(μsec) measured for the single loop for different memory data structures (A), (B), (C) and for different computer systems VP-200, VP-400, M 780.

	データ構造 (A)	データ構造 (B)	データ構造 (C)
システム VP-200	2.74	2.06	2.04
システム VP-400	2.09	1.18	0.796
システム M 780		56.0	

を示している．さらに，データ構造，システムが同じなら N_{all} に t_0 は依存しなかった．以上の点を考慮して，計算機システムとデータ構造の6通りの組合せに対して各々の場合における演算処理時間 t_0 の測定結果を表1に示す．この表と図3とを比較すると，各々の結果は対応する3重ループの計算で（N_x を十分大きくしたときの）t_0 の飽和値に等しくなる．さらに表1には富士通の汎用計算機 M 780 を用いてスカラ計算した結果も示してある．この結果から，VP-400 を用いると演算処理速度は最高約70倍に向上しており，システムのスーパベクトル性能を達成している．

5. 数値実験結果の解釈

前述の数値実験結果から実効性能はメモリデータ構造の選びかた，したがってベクトルデータの転送効率に大きく依存することがわかった．ベクトルデータ転送効率はベクトルレジスタにおけるデータ割付け法に依存するがその基本的方式として次の2つが知られている．第一に，数値計算に必要な引用データを一括してレジスタに格納する方式である．この場合は引用データ間の最大（アドレス番号）距離が小さいほど転送効率は良くなる．第二に，引用データ数に応じてベクトルレジスタが分割され，各々の領域に引用される順序でデータが格納される方式である．この数値実験で使用した VP システムでは第二の方式がとられている[6]（実際，メモリデータ間距離によって実効性能は変化しないことが前章で既に確かめられている）．

ここではベクトル計算機システムの基本的動作を簡単化したモデルを設定し，それをもとに前章の結果の解釈を試みよう．まずベクトルレジスタの容量を X バイトとし，ベクトルレジスタは次のように割付けされるものとする[6]．すなわち，数値計算構造が与えられると最大の引用データ数を考慮してベクトルレジスタは $N_v=2^s$ 個の領域に分割される．その結果，分割された各部分のレジスタ領域には $L=X/(8N_v)$ 個のデータが入ることになる（ただし，ベクトルデータは倍精度で8バイト必要とする）．

図1に示す計算構造をもつ演算の実行には既に述べたようにまわりの6個の格子点での値を引用し，各々の格子点に8個の物理変数がのっている．したがって，48個のデータを引用することが必要になり，ベクトルレジスタは $N_v>48$ となるように分割されることになる．さらに中間データの退避や演算結果のデータを保持する場所を考慮して実際のベクトルレジスタの分割が決定されるべきである．

ここでは簡単のため $L(=2^k)$ が与えられているものと仮定してベクトル演算処理時間 t_0 を簡単に推定してみよう．もしベクトル長（ループ長）N_L（3重ループの場合 $N_L=N_x-2$）が L より小さいか等しければベクトル演算の繰返し計算に必要な全ベクトルデータが一度にベクトルレジスタに入ることができ，ベクトル演算の途中でデータをロードする必要がない．一方，$N_L>L$ ならば演算の途中でベクトルデータの入換えが必要になる．$[N_L/L]=m$ for $m<N_L/L\leqq m+1$, と定義すると，ベクトル演算のために m 回のロード／ストアの操作が必要になる．したがって，ベクトルデータをレジスタにロード／ストアする操作に必要な時間を T_a とすると，1つの格子点における平均的演算処理時間 $t_0(N_L)$ は次のようになる．

$$t_0(N_L)=(T_s+N_L\tau+[N_L/L]T_a)/N_L \qquad (9)$$

ここに T_s はベクトル演算が立ち上がるまでに必要とされる時間，τ はベクトル演算が立ち上がった後に1つの格子点あたりの演算に必要な平均的処理時間，を示しており，したがって分子は N_L 個の格子点において変数値を更新するのに必要な時間を示している．

このモデル［(9)式］から一般的に次のことが直ちに帰結される．(i) $t_0(N_L)$ は N_L が L の倍数になるところで不連続的に変化する．例えば，$N_L=(m+1)L$ のとき $t_0=(T_s+mT_a)/(m+1)L+\tau$ となり，$N_L=(m+1)L+1$ のとき $t_0=[T_s+(m+1)T_a]/(m+1)L+\tau$ となるから $N_L=(m+1)L$ において t_0 は $T_a/(m+1)L$ だけ不連続的に増加する．(ii) $N_L\to\infty$ の極限において $t_0\to T_a/L+\tau$ となる．すなわち，ベクトル演算時間（演算速度）はベクトル長が大きくなると（T_a に大きく依存する）飽和値に近付く．

このモデルによって4章で述べられた数値実験結果は次のように解釈される．まず，図3に示されたベクトル演算時間 t_0 の不連続的変化は(9)式で予測される結果と基本的に一致しており，ベクトルデータの転送によるものと解釈することができる．また図3から観測される不連続性から，基本的に計算機システムVP-200では $L=64$, VP-400では $L=128$ となっている．このことは，ベクトルレジスタの容量 X はVP-400の場合はVP-200の場合の倍であり，かつ L はその定義から X に比例するという2つの事実から直ちに理解できる．また1重ループ［図2(b)］の場合，そのベクトル長 $N_L(\sim N_{all})$ は十分大きく（数万以上），3重ループ［図2(a)］の場合の飽和値に等しくなることが理解できる（表1）．

通常，飽和値がベクトル計算機の最高性能を与えるものとされているが，図3はむしろ $N_L=L$ のとき実効性能 (t_0^{-1}) が最高になることを示している．これは次のように解釈される．(9)式より $N_L=L$ のとき $(m=0)$ $t_0=T_s/L+\tau$ となり $N_L=\infty$ のとき $t_0=T_a/L+\tau$ となる．したがって，図3の結果は $T_a>T_s$ であることを示しており，対象とした数値計算構造（図1）に対してはベクトルデータ転送がネックになって実効性能が抑えられていることを示している．

上述の考察より，現在の数値計算構造（図1）ではベクトル計算機の実効性能はデータ転送に要する時間 T_a によって大きく左右されることが理解できる．一般に T_a はデータ転送のためのオーバヘッドとデータ転送時間によって決定されるものである．まずデータ構造(A)の場合と(B)の場合（3.2節）における T_a の違いは既に予測されたようにメモリバンク競合によるものであると考えられる．図3はバンク競合を避けるようにデータ構造をわずかに変えるだけでベクトル演算速度が大きく改善されうることを示している．

さらに，データ構造(B)と(C)による T_a の違い（図3，表1）は次のように解釈できる．ベクトルレジスタへのデータ格納は数値計算構造によって決まるデータ引用構造に整合すべきものである．現在の数値計算の場合，データ引用は各物理量に対して格子点の順序に連続的になされている（図1）．したがって，ベクトルレジスタの（分割された）各領域には，ある特定の物理変数に対して格子点の順序に連続的にデータが格納される構造になる．メモリデータ構造が(C)の場合はこのようなベクトルレジスタのデータ構造と一致しており，連続的なベクトルデータのロード／ストアが可能となり効率的なデータ転送が実現しうる．このことはベクトルレジスタの容量が大きいほど（L が大きいほど）一度に転送されるデータ量が多くなり，より顕著になることが予想される．実際，図3に示された結果より（レジスタの容量が大きい）VP-400において顕著にあらわれており，(C)のデータ構造の場合，(B)の場合に比べてデータ転送効率，したがって実効性能が改善されうることが理解できる．

6. 結　論

本論文では陽的差分を用いた3次元電磁流体シミュレーションに対してベクトル計算機の実効性能を定量的に詳しく調べた．この数値計算構造（図1）の主な特徴は，ベクトル化が容易であること，演算密度が非常に高いこと，演算に必要な引用データ数が多いことである．多くの数値実験を実行した結果，ベクトル計算機の実効性能は結局ロード／ストアにともなうデータ転送によって大きく左右されることが見いだされた．その結果，この数値計算構造に対して実効性能を向上させるためには次の2点がアルゴリズム設計上重要であることを定量的に議論した．

（1） ベクトル長を（分割された）ベクトルレジスタの各部分領域に入りうるデータ数に一致させる．これはベクトル計算に必要なデータ転送量を最小にすることに相当する．実際に，3重ループでベクトル長をこのようにとることにより，ベクトル計算機の実効性能は（1重ループでベクトル長を非常に大きくとった）その飽和値よりむしろ高くなることが示された．

（2） メモリデータ構造をベクトルレジスタに格納されるデータ構造に適合させる．これによって連続的データ転送が可能になり，効率良いロード／ストアが可能になる．実際，ベクトル計算機 VP-400 を用いた場合，メモリデータ構造の取りかたによってその実効性能に約2.6倍の開きが生じることが示された．

本論文でとりあげた電磁流体方程式は特殊であるがその数値計算構造（図1）は陽的差分による数値計算の基本的構造であり，本論文の結果は同じ構造をもつシミュレーションシステムにも適用できるであろう．実際のプログラム開発にはベクトル計算機の実効性能を定量的に詳しく知ることが重要であるが，この問題を総合的に議論するためには，ベクトルレジスタ等システムアーキテクチュアもパラメータとして広く考慮した議論がこれからの課題となろう．

謝辞 本研究で示した数値計算は京都大学大型計算

機センター VP-400, VP-200, M 780, および九州大学大型計算機センター VP-200 によって行われた．記して謝意を表する．

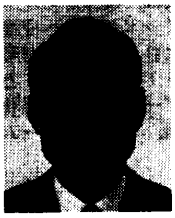
参 考 文 献

1) 唐木幸比古：スーパコンピュータと行列計算，情報処理，Vol. 28, No. 11, pp. 1441-1451 (1987).
2) 日本物理学会編：スーパコンピュータ，培風館，東京 (1985).
3) Ugai, M.: MHD Simulations of Fast Reconnection Spontaneously Developing in a Current Sheet, *Comput. Phys. Commun.*, Vol. 49, No. 1, pp. 185-192 (1988).
4) 鵜飼正行，津田孝夫：電磁流体方程式における高速数値計算法と数値解の評価，情報処理学会論文誌，Vol. 29, No. 10, pp. 907-913 (1988).
5) Richtmyer, R. D. and Morton, K. W.: *Difference Method for Initial-Value Problems*, 2nd ed., pp. 351-368, Interscience Publishers, New York (1987).
6) 磯辺文雄，松延 尚，今村 一，武富 敬，景川耕宇：ベクトルプロセッサ FACOM VP 100 について，九州大学大型計算機センター広報，Vol. 18, No. 5, pp. 397-412 (1985).

(平成元年4月14日受付)
(平成元年9月12日採録)

鵜飼 正行（正会員）

昭和23年生．昭和46年京都大学工学部電子工学科卒業．昭和48年同大学院修士課程修了．同年愛媛大学工学部助手．現在同大学工学部情報工学科助教授．工学博士．計算物理の基礎的研究に従事．現在は数値シミュレーション，特に数値解析，スーパコンピューティング，グラフィックス（アニメーション）等に興味をもつ．電気学会等会員．



津田 孝夫（正会員）

1957年3月，京都大学工学部電気工学科卒業．現在京都大学工学部情報工学科教授，計算機ソフトウェア講座担当．工学博士．自動ベクトル化／並列化コンパイラ，スーパコンピューティング，オペレーティングシステムの研究に従事．「モンテカルロ法とシミュレーション」（培風館），「現代オペレーティングシステムの基礎」（オーム社，共著），「数値処理プログラミング」（岩波書店）などの著書がある．昭和63年度情報処理学会論文賞受賞．本学会関西支部長．ACM SIAM 各会員．