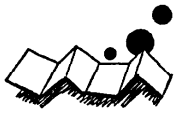
解　説



ベクトル計算機における粒子コードの高速化手法†

折　居　茂　夫††

1. はじめに

使いやすいスーパコンピュータの普及によって，だれもが容易に高速計算を行うことができるようになり，さまざまな分野で大規模なシミュレーションが行われつつある[1]．しかし計算の高速化の度合は，シミュレーションによってまちまちである．なぜなら，現在のスーパコンピュータのほとんどに用いられているパイプライン方式というアーキテクチャでは，すべての計算処理を一律に高速化できないからである．このような計算処理の主なものの一つとして回帰計算をあげることができる．回帰計算には大別して次の二つのタイプがある；

$$F(I)=F(I-1)+\cdots \qquad (I=2, I\max)$$

タイプ−1

$$F(IX(N))=F(IX(N))+\cdots \qquad (N=1, N\max)$$

タイプ−2

計算の高速化のためには，これらを除去することが必要となる．

高速計算を必要とする有限差分法，有限要素法を用いたシミュレーションコードでは，タイプ−1 の回帰計算が高速化の妨げになる場合がある．このような計算は，F(3)の計算を行うために直前に計算した F(2)を必要とするため回帰計算となり，パイプライン方式のスーパコンピュータ（以後，ベクトル計算機と呼ぶ）では，高速計算することができない．

高速計算を必要とする粒子シミュレーションでは，タイプ−2 の回帰計算が高速化の妨げになる場合がある．ベクトル計算機では，$IX(N)$ がすべて異なった値を持つ場合は，高速計算できる．しかし粒子シミュレーションコードでは，$IX(N)$ のいくつかが同じ値を持つ．たとえば，$IX(1)=IX(2)=1$ のとき，$N=1$ の値 F(1)が $N=2$ の F(1)によって更新されるため回帰計算となり，高速計算することができない．

しかしながら，多くのシミュレーションコードにおいて，これらの回帰計算を回避することが可能である．タイプ−1 の回帰計算に対しては，パイプライン方式で高速化できかつ同じ計算結果となるアルゴリズムはないが，もう少し大きな処理単位で見ると，連立1次方程式を解くときに現れる回帰計算である場合が多い．そこで有限差分法や有限要素法に現れる特有な行列のパターンを利用して，タイプ−1 の回帰計算を回避する方法が考えられている．また回帰計算がない，他の解法と置き換えることによってこれを除去し，高速化が達成されている[2]．

タイプ−2 の回帰計算に対しては，パイプライン方式で高速化できかつ同じ計算結果となる，二つのアルゴリズムが知られている[3),4)]．

本解説では，2 章において，ベクトル計算機による粒子シミュレーションの高速化における問題点を示す．3 章において，この問題点を解決する，タイプ−2 の回帰計算に対する高速化アルゴリズムを示す．4 章において，アルゴリズムの一つ[3)]を採り上げ，利用技法を示す．

2. 粒子シミュレーションの高速化における問題点

粒子シミュレーションにおいて，タイプ−2 の回帰計算が問題となるのは PM (Particle-Mesh) 法[5)]と PIC (Particle in Cell) 法のように，粒子とメッシュを使ってモデル化した場合である．

本章では，これらのシミュレーションのベクトル計算機を用いた高速化における問題点を示す．次に PM 法と PIC 法の例を示し，高速化における問題点とおのおののシミュレーションの計算手順を結びつける．

2.1 高速化における問題点

ベクトル計算機で PM 法と PIC 法の計算を行う場

† An Accelerating Technique for Particle Model Codes on Vector Computers by Shigeo ORII (FUJITSU LIMITED).

†† 富士通(株)

合，これらの計算を計算機の記憶域からデータをアクセスする立場から見ると，次の4つのカテゴリに分類することができる．

① メッシュまたはセル上での計算

例 $F(I)=F(I)+A(I)$ $(I=1,2,3,\cdots,I\max)$,
$(I=1,3,5,\cdots,I\max)$.

② メッシュまたはセル上のデータを使って粒子上のデータを計算

例 $P(N)=F(IX(N))+B(N)$
$(N=1,2,3,\cdots,N\max)$.

③ 粒子上のデータを使ってメッシュまたはセル上のデータを計算

例 $F(IX(N))=F(IX(N))+B(N)$
$(N=1,2,3,\cdots,N\max)$.

④ 粒子上での計算

例 $P(N)=P(N)+B(N)$ $(N=1,2,3,\cdots,N\max)$.

ここに，$I\max$：メッシュまたはセルの数，$N\max$：粒子数．$F(I)$ と $A(I)$ はメッシュまたはセル上で定義されたデータ，$P(N)$ と $B(N)$ は粒子上で定義されたデータである．また $IX(N)$ は，メッシュまたはセル上の粒子の位置である．

カテゴリ①，④の計算は，連続アクセスデータあるいは一定間隔アクセスデータを用いた計算であり，従来より十分に高速計算することができた．

カテゴリ②は間接アクセスデータを取り扱うため，従来のベクトル計算機の中には，この処理を高速化できないものがあった．そのためマルチプロセッサの適用が検討されたこともあった[6]．しかし現在の商用のベクトル計算機では，間接アクセスデータ用のハードウェアを持つため，ほとんどのものでこの処理の高速化を期待できる．

しかしながらカテゴリ③の例に示された計算は，1章で述べたように回帰計算の可能性がある．実際問題としてPM法，PIC 法は，メッシュまたはセル数より粒子の数の方が多い．したがって平均すれば，一つのメッシュまたセルに数個の粒子が存在することになる．これはいくつかの $IX(N)$ が同じ値を持つことを意味し，カテゴリ③が回帰計算となることを意味する．PM法，PIC 法ではこの回帰計算のため，十分な高速化を達成することができなかった[6),7)]．

ベクトル計算機における粒子シミュレーションの高速化のポイントは，カテゴリ②が高速化を期待できる現在，カテゴリ③の回帰計算をいかにしてベクトル処理するかにある，といえる．

2.2 粒子シミュレーションの計算手順と高速化の問題点

ここでは，PM法と PIC 法の例を用いて各モデルの計算手順を示し，2.1 節で述べた4つのカテゴリと対応を取ることによって，おのおののモデルの計算手順とベクトル計算機における高速化の問題点を結びつける．

2.2.1 PM 法

PM法では，現象を有限個の粒子でモデル化し，各粒子に対して運動方程式を解く．粒子間に働く力は，空間に固定したメッシュをとおして伝達される．

「磁場の変化がないプラズマの挙動を解析するコード」にPM法を適用した例を示し，その計算手順を示す．

解くべき方程式は，以下の三つである；

$$\Delta\Phi=e/\varepsilon_0(n_e-n_i) \tag{2-1}$$

$$d\vec{V}_i/dt=q_i m_i\vec{E} \tag{2-2a}$$

$$d\vec{V}_e/dt=q_e/m_e\vec{E} \tag{2-2b}$$

ここに，$\vec{E}=-\mathrm{grad}\,\Phi$ である．また，n：粒子密度，$\vec{V}$：粒子の速度，q：電荷，m：粒子の質量，ε_0：誘電率，そして添字 i, e はおのおのイオン粒子，電子粒子である．

電場 $\vec{E}$ のポテンシャル Φ は，空間に固定したメッシュ上で計算される．時間に対しては蛙飛び法を，Φ の計算には FFT を利用することが多い．

PM法では，これらの計算を次のような手順で解く．なお，手順ごとに各カテゴリの番号を示す．

(i) 各粒子の電荷 q を粒子の分布状態に従ってメッシュ上に内挿し，メッシュの各点における電荷 $e/\varepsilon_0(n_e-n_i)$ を計算する． ⇒ ③

(ii) ポアソン方程式(2-1)を解く． ⇒ ①

(iii) ポアソン方程式より求めたポテンシャルより電場を求め，その電場を各粒子の位置に内挿し，運動方程式(2-2)を計算する． ⇒ ②

(iv) 移動した粒子の位置を調べ，境界条件に従った処理を行う． ⇒ ④

PM法の計算は，基本的には，以上の4ステージよりなる．

カテゴリ③の回帰計算は，PM法ではステージ(i)において現れる．

2.2.2 PIC 法

PIC 法では，質量を持った流体粒子を追跡しながら，空間に固定したメッシュ上において離散化された方程式を解く．“圧縮性の流体”に PIC 法を適用した

例を示し，その計算手順を示す．

解くべき方程式は，以下の4つである；

$$\partial\rho/\partial t+\mathrm{div}(\rho\vec{V})=0 \qquad (2\text{-}3)$$

$$\rho(\partial/\partial t+\vec{V}\cdot\mathrm{grad})\vec{V}=-\mathrm{grad}\,P \qquad (2\text{-}4)$$

$$\rho(\partial/\partial t+\vec{V}\cdot\mathrm{grad})\cdot(I+1/2\vec{V}\cdot\vec{V})=\mathrm{div}\,(P\vec{V}) \qquad (2\text{-}5)$$

$$P=f(\rho,\ I) \qquad (2\text{-}6)$$

ここに，ρ：流体の密度，$\vec{V}$：流体の速度，P：流体の圧力，I：流体の内部エネルギである．

PIC 法では，これらの計算を次のような手順で解く．なお，手順ごとに各カテゴリの番号を示す．

（i） 各セル上で，状態方程式(2-6)より圧力 P を求める． ⇒ ①

（ii） (2-4)，(2-5)式の移流項 $(\vec{V}\cdot\mathrm{grad})\Rightarrow 0$ とした式より，各セル上での速度の中間値 V'，内部エネルギの中間値 I' を計算する． ⇒ ①

（iii） 粒子の位置するセルとその近傍のセルの $\vec{V}'$ を用い，各粒子を移動する． ⇒ ②

（iv） 移動した粒子の位置を調べ，境界条件に従った処理を行う． ⇒ ④

（v） 各セルに含まれる粒子の運動量，エネルギを足し合わせ，Δt 秒後の各セルの速度，エネルギを求める． ⇒ ③

PIC 法の計算手順は，基本的には，以上の5ステージよりなる．

カテゴリ③の回帰計算は，PIC 法ではステージ(v)において現れる．

3. 高速化のためのアルゴリズム

本章では，PM法，PIC 法を用いたコードの高速化を図るためのアルゴリズムを示す．1，2章で説明したように，図-1 の FORTRAN コーディングは，回帰参照を起こす場合，高速化することができない．

現在，このアルゴリズムと同じ計算結果となり，かつベクトル計算機で高速化できるアルゴリズム（本解説では以後このアルゴリズムを“ベクトル化アルゴリズム”と呼ぶ）は，次の二つが知られている．

（1） 配列 F(I) を多数個複製した作業配列 F'(I, N) を設け，図-1 の F($IX(N)$) と置き換えて計算する．$IX(N)$ が同じ値を持つ場合でも F'($IX(N)$, N) では右の添字 N が異なっているため回帰計算を回避できる．次に F'(I, N) のおのおのの I に対して N で総和をとることによって，元の計算と同じ計算結果を得る[3]．

```
      DO 1 N=1, Nmax
        F(IX(N))=F(IX(N))+A(N)
    1 CONTINUE
```

図-1 従来のスカラアルゴリズム

（2） たとえば $IX(1)$, $IX(2)$, $IX(3)$ が同じ値を持つ場合，IX が同じにならない N のグループを作る．今，IG：グループ番号，M：グループ中のデータ番号としたとき，この例では(IG=1, M=(1, Nmax−2), N=(1, 4, 5, ⋯, Nmax)), (IG=2, M=1, N=2), (IG=3, M=1, N=3) という三つのグループに分けることができる．おのおののグループでは，IX が同じ値を持たないため，回帰計算を回避することができる．なおこのアルゴリズムはグループ分けをスカラ計算によって行うため，回帰計算を回避するために追加したグループ分けの計算よりも時間のかかる計算が図-1 のループ中で行われる必要がある[4]．

4. ベクトル化アルゴリズムの利用技法

3 章で紹介した二つのベクトル化アルゴリズムは，おのおの独自の個性を持ち，異なった利用上の留意点がある．そこで本章では，この中から比較的 $IX(N)$ (図-1 参照) の値の持ちかたに対して性能変化の少ないベクトル化アルゴリズム(1)[3]を採り上げ，ベクトル計算機上で有効に利用できるように，子細に説明する．

このベクトル化アルゴリズムの基本的な考え方は，メッシュ上にデータを割り当てること（以後“ASSIGNMENT”と呼ぶ）と，割り当てられたデータ同士を加えるという，二つの処理を別々に行うということである．このため第一にメッシュを N 枚複製し，粒子のデータを別々の複製メッシュ上に割り当てる（図-2，ステージ 1)．第二に割り当てられたデータを1枚のメッシュ上に足しこむ（図-2，ステージ 2)．こ

```
      DO 10 N=1, Nmax
      DO 10  I=1, Imax
          W (I, N)=0.                          ステージ 0
   10 CONTINUE
      DO 1 N=1, Nmax
          W(IX(N), N)=W(IX(N), N)+A(N)         ステージ 1
    1 CONTINUE
      DO 2 N=1, Nmax
      DO 2  I=1, Imax
          F(I)=F(I)+W(I, N)                    ステージ 2
    2 CONTINUE
```

図-2 回帰計算を回避したアルゴリズム（加算のオーバヘッドが大きく，実用的でない）

れら二つの処理は回帰計算を含まないので高速化が期待されるが，残念ながらこの方法では，複製した N 枚のメッシュ上のデータを1枚のメッシュに足しこむために $N\max\times I\max$ 回の加算が必要となる．この加算のオーバヘッドにより，このままでは高速化を期待できない．また N 枚のメッシュを複製するために必要なメモリも $N\max\times I\max$ 語となる．たとえば $N\max=10,000$, $I\max=50\times50$ を考えると 200メガ ($=10,000\times50\times50\times8$) バイトのメモリを必要とし，現在のスーパコンピュータでさえ限界に近いぐらい大きなメモリが必要となる．

この加算のオーバヘッドと大容量のメモリの要請は，全粒子を一度に ASSIGNMENT するために生じる．そこで Lr 個ずつの粒子を，$N\max/Lr$ 回に分けて ASSIGNMENT を行うことによって，加算のオーバヘッド及び必要なメモリを $Lr\times I\max$ に減らすことができる．もちろん，$N\max>Lr$ である．現在のベクトル計算機では，一回のベクトル命令で扱うデータ数（以後“ベクトル長”と呼ぶ）が数百で十分高速化が達成できると考えられるから，Lr は数百で足りる．**図-3** にこのベクトル化アルゴリズムの FORTRAN コーディングを示す．まずステージ0において，複製された N 枚のメッシュに初期値をセットする．ステージ1の内側のループ“DO 1”において，Lr 個の粒子を Lr 枚の複製メッシュに ASSIGNMENT する．ステージ1の外側のループ“DO 10”において，内側のループの処理を $N\max/Lr$ 回繰り返す．ステージ2において，Lr 枚のメッシュに ASSIGNMENT されたデータを1枚のメッシュ上に足しこむ．このようにパラメータ Lr を導入することによって，実用的なベクトル化アルゴリズムが得られる．

```
      DO 10 K=1, Lr
      DO 10  I=1, Imax
          W(I, K)=0.                          ステージ 0
   10 CONTINUE

      IE=0
      DO 100 KK=0, Nmax/Lr
        IS  =IE+1
        IS 1=IS-1
        IE  =IS 1+Lr
        IF(IF. GT. Nmax)  IE=Nmax             ステージ 1
      DO 1 N=IS, IE
          K=N-IS 1
          W(IX(N), K)=W(IX(N), K)+A(N)
    1 CONTINUE
  100 CONTINUE

      DO 2 K=1, Lr
      DO 2  I=1, Imax
          F(I)=F(I)+W(I, K)                   ステージ 2
    2 CONTINUE
```

図-3 ベクトル化アルゴリズム（加算のオーバヘッドを，軽減した）

ここで，Lr を増加してベクトル長を長くすることを考える．個々の計算が速くなる反面，加算のオーバヘッドも増加する．したがってこのアルゴリズムの性能はこの二つの陽陰ファクタの競合によって決まる．

4.1 ベクトル化アルゴリズムの性能解析

今まで述べてきたように，アルゴリズムのベクトル性能は，ベクトル長 Lr に依存する．本節では，ベクトル化アルゴリズムを Hockney のパラメータ[8]を用いて記述し，計算性能と Lr の関係を明らかにする．Hockney のパラメータ表示によれば，図-1 に示す従来のアルゴリズムのスカラ計算機あるいはベクトル計算機のスカラ・モードによる実行時間は，

$$ts=N\max/(rs_\infty) \tag{4-1}$$

で表すことができる．ここに rs_∞ は，最大フロプス値である．また図-3 に示すステージに対し，次のように式をたてることができる．

$$\begin{aligned}
tv_0&=(n_{1/2}+Lr\cdot I\max)/(rv_\infty), && \text{(ステージ 0)}\\
tv_1&=N\max/Lr\ (n'_{1/2}+Lr)/(rv_\infty), && \text{(ステージ 1)}\\
tv_2&=Lr(n_{1/2}+I\max)/(rv_\infty), && \text{(ステージ 2)}
\end{aligned} \tag{4-2}$$

ここに rv_∞ は，ベクトル計算機の最大フロプス値である．$n_{1/2}$ は，加算に対する rv_∞ の半値性能のときのベクトル長．$n'_{1/2}$ は，間接データアクセスを含んだ加算に対する $n_{1/2}$ である．簡単化のため**図-3** のスカラ計算は，考慮しない．またステージ0とステージ2の $n_{1/2}$ は，同じものとする．

従来のアルゴリズムのスカラ計算による実行時間 ts と，ベクトル化アルゴリズムのベクトル計算による実行時間 $tv_0+tv_1+tv_2$ の比 R は式(4-1)，(4-2)より，次のようになる；

$$\begin{aligned}
R&=\frac{ts}{tv_0+tv_1+tv_2}\\
&=\frac{rv_\infty/rs_\infty}{1+n'_{1/2}/Lr+Lr[2+(1+1/Lr)n_{1/2}/I\max]/N\mathrm{mesh}}
\end{aligned} \tag{4-3}$$

ここに，$N\mathrm{mesh}=N\max/I\max$ である．

上式より R が Lr にたいして極値を持つことがわかる．そこで $R/Lr=0$ とし，そのときの Lr の値を算出すると，以下のようになる；

$$Lr=\sqrt{\frac{n'_{1/2}N\,\mathrm{mesh}}{2+n_{1/2}/I\,\mathrm{max}}}\Bigg|_{R=R\,\mathrm{max}} \tag{4-4}$$

これを(4-3)式に代入し，Rmax を求めると，次の式を得る；

$$R\,\mathrm{max}=\frac{rv_\infty/rs_\infty}{1+2\sqrt{[(2+n_{1/2}/I\mathrm{max})n'_{1/2}/N\,\mathrm{mesh}]}+n_{1/2}/N\mathrm{max}} \tag{4-5}$$

式(4-4)，(4-5)より次のことがわかる．

（1） Rmax のときのベクトル長は，$n'_{1/2}$ と Nmesh の平方根に比例して増加する．したがって $n'_{1/2}$ と Nmesh が大きい場合は，最大性能をだすため，多くのメモリを必要とする．

（2） $n_{1/2}/I$max〜0，$n_{1/2}$ と $n'_{1/2}$ を一定としたとき，Rmax は Nmesh のみに依存する．したがって1メッシュ当たりに平均何個の粒子が割り当てられるかによって，このベクトル化アルゴリズムの性能が決まることがわかる．

次に1粒子当たりの計算時間を示す；

$$\tau=\frac{tv_0+tv_1+tv_2}{N\,\mathrm{max}}=(1+2\sqrt{[(2+n_{1/2}/I\,\mathrm{max})n'_{1/2}/N\,\mathrm{mesh}]}+n_{1/2}/N\,\mathrm{max})/rv_\infty \tag{4-6}$$

$n_{1/2}/I$max〜0 のとき，τ は $1/N$mesh の平方根に比例して増加する．

4.2 ベクトル化アルゴリズムの VP-200 上での性能

ここでは，VP-200 上でのベクトル化アルゴリズムの性能を例として示す．また，使用上の留意点を示す．

測定は，2次元 CIC (Cloud in Cell) 法[5]の回帰計算の部分のみを用いて行った（**図-4**）．これは，回帰計算でない重みづけの計算を別に計算したほうが，計算全体の性能が上がるからである．ASSIGNMENT される粒子の位置は，レーマーの合同法による一様乱数によって，決定した．

図-5 に Lr-R の関係を Nmesh をパラメータとして示した．この図より，R が最大値を持つことが確認できる．またこの図より VP-200 上においてベクトル化アルゴリズムを利用する際の，三つの留意点を示すことができる；

```
DO 1 N=1, Nmax
  F(IX(N), IY(N))=F(IX(N), IY(N))+A(N, 1)
  F(IX(N)+1, IY(N))=F(IX(N)+1, IY(N))+A(N, 2)
  F(IX(N), IY(N)+1)=F(IX(N), IY(N)+1)+A(N, 3)
  F(IX(N)+1, IY(N)+1)=F(IX(N)+1, IY(N)+1)
                      +A(N, 4)
1 CONTINUE
```

図-4 性能測定に用いた2次元 CIC 法の回帰計算

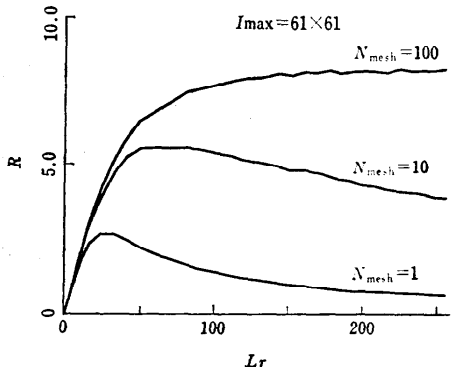


図-5 ベクトル化アルゴリズムの VP-200 における相対性能 R をベクトル長 Lr に対して示した．

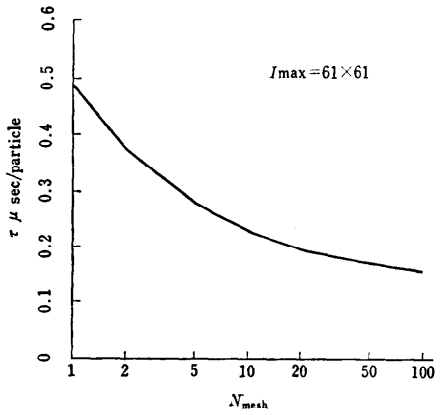


図-6 ベクトル化アルゴリズムの VP-200 における性能を，1粒子当たりの計算時間として示した．なお Nmesh=100 のとき，Rmax=8.2 である．

（1） Nmesh が大きくなるに従って，Rmax が大きくなる．

（2） Nmesh=100 のとき，カーブは Rmax に達するかなり前から飽和する．したがって Nmesh が大きいとき，Rmax に近い性能を比較的低い Lr で得ることが可能である．

（3） Nmesh=1 で，Lr>150 のとき，スカラアルゴリズムより遅くなる領域がある．したがって Nmesh が小さい計算に対しては，注意が必要である．

また**図-6** に τ-Nmesh の関係を示した．ここに，Rmax の点の値を τ とした．この図より最大性能が 0.15 μsec/個に漸近していることがわかる．

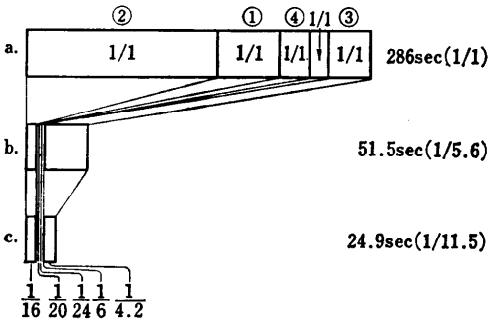


図-7 2次元流体 PIC コードの VP-200 による実行時間（a；オリジナルコードのスカラー・モード計算時間，b；オリジナルコードのベクトル・モード計算時間，c；ベクトル化アルゴリズムを適用したコードのベクトル・モード計算時間）ここで①はセル上での計算，②はセル上から粒子上への計算，③は粒子上からセル上への計算，④は粒子上の計算である．残りは，初期値計算などである．

4.3 ベクトル化アルゴリズムの実コードへの適用効果

ここでは，ベクトル化アルゴリズムを実コード“2次元流体 PIC コード[9]”に適応した例を示し，その効果を示す．計算した問題は，半径 22 cm の鉄球がアルミニウムの板に 1.5 cm/μsec の速度で衝突するときの様子をシミュレートするもので，これを 800 タイムステップの間計算して適用効果を調べた．ここにメッシュは，41×81．粒子数は，14,266 個である．アルゴリズムの適用効果を知るため，オリジナルコードを VP-200 のスカラ・モードとベクトル・モードで計算し，ベクトル化アルゴリズムを適用したものをベクトル・モードで計算し，この三つの結果を比較した(図-7)．図-7 は，2章で述べたカテゴリに分けて計算時間を示した．図-7(a)，(b)は，おのおのベクトル化アルゴリズムを適用する前のスカラ・モードの計算とベクトル・モードの計算の実行時間である．カテゴリ②の間接アクセスデータの処理を含んだ計算は，16倍と，十分高速化される．またカテゴリ①においては，本コードの場合，状態方程式の計算が簡単なため，20 倍と，十分高速化される．ベクトル化アルゴリズムを適用したときのベクトル計算の実行時間を図-7(c)に示す．Lr の最大値を実験的に調べた結果，カテゴリ③の計算は，4.2 倍 ($Lr=32$) となった．このときの全実行時間は，ベクトル化アルゴリズムを適用する前のオリジナルコードの実行時間と比較して，11.5 倍と十分高速化される．

5. おわりに

ベクトル計算機を用いて，PM法，PIC 法を用いた粒子シミュレーションコードの計算を高速化する場合の問題点を整理し，高速化のポイントとなるベクトル化アルゴリズム(1)（3 章参照）に対して利用技法を示した．

ベクトル化アルゴリズム(1)は，タイプ−2 の回帰計算の計算結果と同じ結果を得ることができるため，分野を問わずこの回帰計算を除去するために使用することが可能である．そこで問題となるのは，この問題が大容量のメモリを必要とすることである．たとえば 4.3 節の流体シミュレーションコードの例では，このアルゴリズムの適用によってコード全体の計算速度が適用以前と比べて約2倍となる．しかしこの2倍のスピードアップのために，1.06×10^5 語のメモリを必要とする．したがってこのアルゴリズムの適用に際しては，スピードアップとメモリ量が利用者にとって引き合うものかどうかを，事前に検討することが必要である．

プラズマ・核融合の研究では，ベクトル化アルゴリズム(1)を利用した粒子シミュレーションコードによる研究成果がすでにあり[10)~12)]，利用されている．これは，一回当たりの計算時間が長いため，利用者にとって引き合うものになるためと考えられる．

このベクトル化アルゴリズム(1)をスカラ計算したときの実行時間は，オリジナルのそれよりも遅くなる．これは回帰計算を回避するためループを分割した際，計算処理が増えたためである．ベクトル化アルゴリズムの中には，このように「肉を切らせて骨を断つ」タイプのアルゴリズムが存在する．

最後に，プラズマ・核融合の粒子シミュレーションの高速化手法には，計算スキームに陰解法を用いて積分時間ステップを大きく取って高速化を図る方法[13)]があることを指摘する．スーパコンピュータの普及にともなって，従来より行われてきた計算スキームを工夫する高速化方法に加えて，スーパコンピュータを有効に利用する高速化手法（ベクトル化アルゴリズムを用いる方法）が加わりつつある．

これら二つの高速化手法の相乗効果によって，計算機シミュレーションによる研究開発の一層の発展を期待したい．

プラズマ・核融合の研究におけるベクトル化アルゴリズムの利用状況について，名古屋大学プラズマ研究

所の阿部芳彦助教授より貴重なご意見をいただきました．深謝いたします．また VP-200 におけるデータ取得について，(株)富士通徳島システムエンジニアリングの瀧山匡彦氏に協力いただきました．感謝いたします．

参 考 文 献

1) 例えば，日本物理学会編：スーパーコンピュータ，培風館 (1985)．富士通(株)：FACOM ジャーナル 特集 FACOM VP シリーズ，Vol. 11, No. 8 (1985)．
2) 例えば，徳永康男，原田裕夫，石黒美佐子：原子力コードにおける数値解法とそのベクトル化，JAERI-M 85-143 (1985)．
3) Orii, S.: Fully Vectorizable Particle-Mesh Model Codes on the FACOM VP-100/200, Proc. ANS/ENS/JAES Int. Topical Meeting, Knoxville, Tennessee, April 9-11, Vol. 1, p. 22(1985). Nishiguchi, A., Orii, S. and Yabe, T.: Vector Calculation of Particle Code, J. Comp. Phys., Vol. 61, No. 3, p. 519 (1985).
4) Horowitz. E. J.: Vectorizing the Interpolation Routines of Particle-in-Cell Codes, UCRL-93673 (1985).
5) Hockney, C. W. and Eastwood, J. W.: Computer Simulation Using Particles, McGraw-Hill (1981).
6) Buzbee, B. L.: Plasma Simulation and Fusion Calculation, NATO ASI Series, Vol. F7, p. 417 (1984).
7) 阿部芳彦：プラズマ・核融合研究分野におけるスーパーコンピュータの性能測定，週間コンピューターワールド，Vol. 25, pp. 18 (1984)．
8) Hockney, C. W. and Jesshope, C. R.: 並列計算機，共立出版 (1984)．
9) Amsden, A. A.: The Particle-in-Cell Method for the Calculation of the Dynamics of Compressible Fluids, Los Alamos Scientific Laboratory, Report No. LA-3466 (1966).
10) Nishiguchi, A., Yabe, T. and Haines, M. G.: Nernst Effect in Laser-Produced Plasmas, Phys. Fluids, Vol. 28, p. 3683 (1985).
11) Tanaka, Motohiko. and Sato, T.: Particle Simulation of Relativistic Electron Beam Injection into a Magnetized Plasma Channel, Phys. Fluids, Vol. 29 (1986).
12) Abe, H., Sakairi, N., Itatani, R. and Okuda, H.: High-Order Spline Interpolations in the Particle Simulation, J. Comp. Phys., Vol. 63, p. 247 (1986).
13) Barnes, D. C., Kamimura, T., Leboeuf, J. N. and Tajima, T.: Implicit Particle Simulation of Magnetized Plasmas, J. Comp. Phys., Vol. 52, p. 480 (1983).

(昭和 61 年 7 月 3 日受付)