— 原著論文 —

ベクトル型計算機によるPIC（Particle-In-Cell）法速度モーメント計算の計算速度比較 ～作業配列法・RETRY法・LISTVEC指示行法～

杉山　徹[1*]

粒子シミュレーション手法の1つであるPIC（Particle-In-Cell）法モデルにおいて，粒子の速度モーメントをベクトル型計算機上で計算するためには，いくつかの工夫が必要である．その中で，広く使われている3つの手法（作業配列法・RETRY法・LISTVEC指示行法）による計算速度を比較した結果を報告する．計測には空間2次元システムを用いた．作業配列法・RETRY法の計算速度は同程度であるが，LISTVEC指示行法はそれらより約1.5倍遅い．一方，使用する主メモリ量は，RETRY法・LISTVEC指示行法は同程度であるが，作業配列法は，それらより数倍多く，扱えるシステムサイズを小さくしてしまう．プログラミングの容易さでは，LISTVEC指示行法は非常に容易であり，次に作業配列法である．RETRY法はプログラム作成までの作業量が他の2つに比べて多い．結果，本報告では，プログラミングは容易ではないが，計算効率を全体的に判断し，RETRY法の使用を推薦する．

キーワード：PIC法，ベクトル計算機，速度モーメント計算

2010年10月1日受領；2010年11月12日受理

1　独立行政法人海洋研究開発機構・地球内部ダイナミクス領域

*代表執筆者：
杉山　徹
独立行政法人海洋研究開発機構・地球内部ダイナミクス領域
〒236-0001　神奈川県横浜市金沢区昭和町3173-25
045-778-5884
tsugi@jamstec.go.jp

— Original Paper —

Performance measurement of WorkArray, LISTVEC compile directive, and RETRY algorithms on SX Super-computer for the PIC (Particle In Cell) simulation model

Tooru Sugiyama[1*]

The calculation performance of three types of moment calculation algorithms is evaluated on SX super computer. Those are WorkArray, LISTVEC compile directive, and RETRY algorithms. The used simulation model consists of two dimensional in space. The calculation time of WorkArray and RETRY methods show almost same cost. That of LISTVEC methods is about 1.5 times larger than those of others. The used main memory size of RETRY and LISTVEC methods is almost same cost. That of WorkArray is a few times larger than those of others. The algorithm of WorkArray and LISTVEC are simple but that of RETRY is complicated. However, in the view points of total merits, we recommend to use RETRY algorithm.

Keywords: PIC simulation model, vector type computer, velocity moment calculation

Received 1 October 2010 ; Accepted 12 November 2010

1 Institute for Research on Earth Evolution (IFREE), Japan Agency for Marine-Earth Science and Technology (JAMSTEC)

*Corresponding author:
Tooru Sugiyama
Institute for Research on Earth Evolution (IFREE), Japan Agency for Marine-Earth Science and Technology (JAMSTEC)
3173-25 Showa-machi, Kanazawa-ku Yokohama 236-0001, JAPAN
Tel. +81-45-778-5884
tsugi@jamstec.go.jp

1. はじめに

PIC（Particle-in-Cell）シミュレーション手法は，プラズマ運動論計算において重要なモデルの1つである（例えば，Birdsall and Langdon, 1991; Fujimoto, 1992; Matsumoto and Omura, 1993）．この手法では，計算領域内において超粒子と呼ばれる仮想粒子の軌道を追う．一方，密度・流量などの流体量や電磁場は格子点上で解かれる．そのため，粒子の速度モーメント値（電荷密度・電流密度など）は，粒子位置に隣接する格子点と関連付けられなければならない．ここで問題となるのが，計算領域内のランダムな位置に配置されている粒子データに対し，どのようにベクトル型計算機でモーメント値を計算するかである．これまでに，作業配列法（Birdsall and Langdon, 1991）・RETRY法（Imamura, 2005）・LISTVEC指示行法（Sugiyama et al., 2004）の3つの方法が提案されており，ここでは，これらの方法の実行速度を測定した結果を報告する．使用したベクトル型計算機は，2009年から稼働している地球シミュレータである．用いたコードは，空間2次元・速度3次元で構成されるシステムを取り扱ったコードである．

2. 粒子モーメントの計算法

粒子法では，位相空間内での粒子の分布関数を滑らかに表現し，運動論を正しく議論するために，格子点当りに百個程度の粒子を分布させる．よって，格子点上での電磁場に関する計算量に比べて，粒子に関する計算量が多い．このことから，計算効率の向上には，粒子計算部を効率良く計算する必要がある．その中で，最も計算コストの高い作業は，システム上にランダムに配置された粒子と，格子点上で定義された場の量とを関連付ける計算（モーメント計算）である．この計算は，格子点位置rにおける電荷密度は，

$$N(r) = \sum_{\#} S(r - r_{\#})$$

電流密度は，

$$F(r) = \sum_{\#} v_{\#} S(r - r_{\#})$$

と定義される．ここで，$r_{\#}$, $v_{\#}$ は粒子の位置と速度であり，全粒子の総和計算を行う．S(r)は，粒子と格子を関係付ける形状関数である．図1に形状関数の概念図を示す．（図の説明では，簡単のため，1次精度1次元モデルで行う）．超粒子は，点電荷としてではなく，粒子座標 Xpを中心とする格子間隔（ΔX）と同じ長さの広がりを持った粒子と

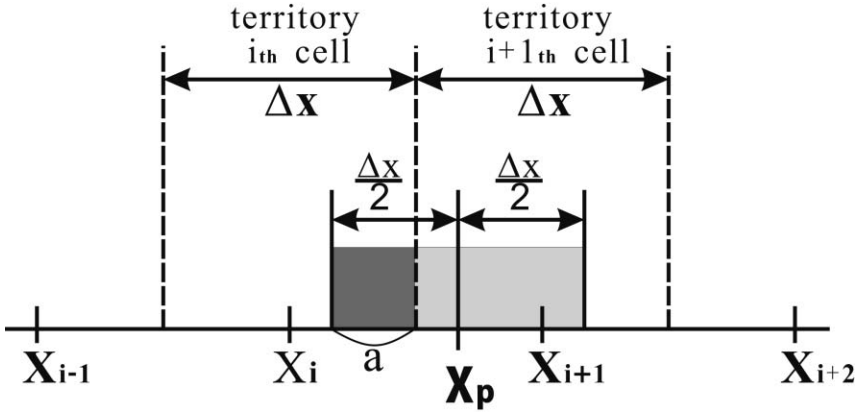

Fig. 1. Schematic illustration of the shape function in one-dimensional case. The particle width (dark gray length of a) which lies in the i^{th} cell contributes to the moment on the grid Xi.

図1. 形状関数の概念図．格子点座標をXi，粒子位置をXpとする．粒子は，格子点幅ΔXと同じ長さの広がりを持ってシステム内に存在する．各格子点は，幅ΔXの領域を担当し，その中に入ってきた粒子の長さ分を，自身の格子点上でのモーメント値に寄与させる．図の例では，長さa分が格子点Xiへの寄与分となる．

して扱われる．また，各格子は，格子点位置（Xi＝i ΔX）を中心とする幅（ΔX）の領域を自身の領域とする．領域内に属した長さ分を，粒子の格子点へのモーメント量の寄与分とする．図の例では，格子X_i とX_{i+1}への寄与分は，それぞれa/ΔX，1- a/ΔXとなる．

プログラム例を図２に示す．この例に見られるように，粒子と格子点の座標を格子間隔ΔXで規格化してからシミュレーションを実行するため，格子点への寄与分を計算する14～15行目に，ΔXは陽には表れない．

この例は，スカラ型計算機向けのコードである．そのため，このままでは，メインループ（12～26行目）において，粒子位置がランダムなため，粒子のID番号NNに対応する格子点位置IMの依存関係が不明となり，ベクトル計算が出来ない．

以下，提案されている3つのベクトル化方法について述べる．

2.1. 作業配列法

電荷密度と電流密度を保存する配列のインデックスを，もう1次元拡張し，その要素数でループ計算を行うことで依存関係を解消する．別名「とまり木」法とも呼ばれる．プログラム例を図3に示す．拡張した配列は，7行目に定義されている．この方法では，拡張したインデックスのサイズが最大ベクトルレジスタ長（以下ベクトル長，

```
 1    !!
 2    !!    PROGRAM  SCALAR
 3    !!
 4    REAL:: PTL(1:NMAX,1:4)           ! Particle
 5    REAL:: MOM(1:IMAX,1:4)           ! Moment
 6
 7    MOM = 0.0
 8
 9    !!
10    !!  MAIN LOOP
11    !!
12    DO NN = 1, NMAX
13
14       IM = FLOOR( PTL(NN,4) );    IP=1  +IM      !.. Grid in X
15       XM = PTL(NN,4)-REAL(IM);    XP=1.0-XM      !.. Shape
16
17       MOM(IM,1) = MOM(IM,1) +  XM * PTL(NN,1)   !.. Flux in X
18       MOM(IP,1) = MOM(IP,1) +  XP * PTL(NN,1)
19       MOM(IM,2) = MOM(IM,2) +  XM * PTL(NN,2)   !.. Flux in Y
20       MOM(IP,2) = MOM(IP,2) +  XP * PTL(NN,2)
21       MOM(IM,3) = MOM(IM,3) +  XM * PTL(NN,3)   !.. Flux in Z
22       MOM(IP,3) = MOM(IP,3) +  XP * PTL(NN,3)
23       MOM(IM,4) = MOM(IM,4) +  XM               !.. Density
24       MOM(IP,4) = MOM(IP,4) +  XP
25
26    ENDDO
```

Fig. 2.　Program for Scalar procedure. NMAX and IMAX mean the total number of particles and cells. PTL(particle ID num, 1~4) means particle velocity with three components and position. The moment values on grid are stored in MOM(grid ID, 1~4) with three components of the flux and the density.

図2.　モーメント計算のプログラム例．スカラ型計算機で実行する場合のプログラミング方法である．各変数は，粒子総数がNMAX．格子点総数がIMAX．粒子の速度3成分がPTL（粒子ID番号，1～3），粒子位置がPTL（粒子ID番号，4），モーメント値のFlux値がMOM（格子番号，1～3），密度がMOM（格子番号，4），である．

```
 1   !!
 2   !!    PROGRAM  WORK VECTOR
 3   !!
 4   INTEGER,PARAMETER:: LVEC=256
 5   REAL::  PTL(1:NMAX,1:4)
 6   REAL::  MOM(1:IMAX,1:4)
 7   REAL::  WRK(1:IMAX,1:LVEC,1:4)
 8
 9   !!
10   !!   INITIALIZATION  LOOP
11   !!
12   MOM = 0.0
13   WRK = 0.0
14
15   !!
16   !!   MAIN  LOOP
17   !!
18   DO NP =  1, NMAX, LVEC
19   DO NN = NP, MIN( NP+LVEC, NMAX)
20
21      NV = NN - NP + 1
22
23      IM = FLOOR( PTL(NN,4) );   IP=1  +IM      !.. Grid in X
24      XM = PTL(NN,4)-REAL(IM);   XP=1.0-XM      !.. Shape
25
26      WRK(IM,NV,1) = WRK(IM,NV,1) + XM * PTL(NN,1)
27      WRK(IP,NV,1) = WRK(IP,NV,1) + XP * PTL(NN,1)
28      WRK(IM,NV,2) = WRK(IM,NV,2) + XM * PTL(NN,2)
29      WRK(IP,NV,2) = WRK(IP,NV,2) + XP * PTL(NN,2)
30      WRK(IM,NV,3) = WRK(IM,NV,3) + XM * PTL(NN,3)
31      WRK(IP,NV,3) = WRK(IP,NV,3) + XP * PTL(NN,3)
32      WRK(IM,NV,4) = WRK(IM,NV,4) + XM
33      WRK(IP,NV,5) = WRK(IP,NV,4) + XP
34
35   ENDDO
36   ENDDO
37
38   !!
39   !!   REDUCTION  LOOP
40   !!
41   DO IX = 1, IMAX
42   DO IC = 1, 4
43   DO NV = 1, LVEC
44      MOM(IX,IC) =  MOM(IX,IC) + WRK(IX,NV,IC)
45   ENDDO
46   ENDDO
47   ENDDO
```

Fig. 3. Program for WORKARRAY algorithm. The extended array is defined as WRK(Grid ID, extended index, components).

図3. 作業配列法によるモーメント計算のプログラム例．拡張した作業配列は，WRK（格子番号，拡張index，成分）と定義されている．

NEC-SXシリーズでは256）の場合（4行目：LVEC=256）に計算速度が最速である．この時のサイズは，超粒子を格子点当り256×4個分布させることと同値であり（×4は，電荷密度と電流密度の4成分用），主メモリ使用量を圧迫してしまう．その使用メモリ量を，粒子用メモリに使用すれば，扱う粒子数が増え，さらに統計的に精密な運動論計算が可能となる．さらに，粒子数に関する総和計算前後に，初期化と総和計算（reduction）が必要である（13行目と41～47行目）．この計算量は，格子点数が多い場合には，無視できない計算コストを生む．後述にあるように，格子点数が約10^7の時は，メインループの7%にも達する．

2.2. LISTVEC指示行法

作業配列の導入による主メモリ使用量の増加を回避する方法の一つが，NEC-SXシリーズ独特のコンパイル指示行LISTVECを用いる方法である（内容の詳細は文献Sugiyama et al., 2004を参照のこと）．概要は，図2におけるメインループ内で，異なる粒子ID番号に対して，同じ格子番号IMが割り当てられる時に（衝突発生と呼ぶ），結果が不整合となるが，衝突が起こらない限り正しい結果を得る．よって，衝突発生を監視し，衝突が発生した時のみベクトル計算からスカラ計算に切り替えれば正しい結果を得る．プログラム例を図4に示す．この指示行（22行目）が記載された直後のループでは，自動的に衝突発生を監視しながら和計算を行う．本指示行を用いると，衝突発生作業配列としての主メモリの圧迫量が，格子点当り2次元計算の場合は4×4個（１次元計算の場合は2×4個）のみの粒子を分布させる量と同値であるため大幅に軽減され，かつ図2にあるスカラ型プログラムからの変更は少ない．（注：補正作業を行うため，ループ内で同じ名前の配列を複数行記載できないため，6～9行目にあるように配列名が増えている）．また，作業配列法に比べ，reduction作業による計算コストは格段に小さい．

2.3. Retry法

この方法は，NEC-SXシリーズに限らず，一般的なベクトル型計算機で使用できる方法である．ここでは，「ベクトル型計算機では，有限長のベクトルレジスタ上のデータがメモリにストアされる時，書き込み順は常に，一定の順序で行われる」というベクトル型計算機のハードウェア特性を利用する（Imamura, 2005）．アルゴリズムの概念図とプログラム例を図5と6に示す．

本アルゴリズムでは，全粒子をRVEC個の束にしてループを考える．図5の例ではRVEC=12である．（注：RVECの大きさは，ベクトル長とは独立であるが，ベクトル長以上でなければ高速計算ができない）．まず，図6の24～28行目では，粒子のID番号NNに対応する格子点位置IMを求め，配列CIG（IM）へ粒子番号を記録する（モーメント値を書き込むのではない）．メモリへの書き込み順が一定であるという特性から，この書き込みでは，格子点位置の衝突が発生した場合は，自動的に，粒子ID番号の大きい方のデータが配列CIGに記録される（図5の上部の灰色網掛け部分）．配列CIGは，格子点当り粒子1個を記録できれば十分であるが，KVEC個を記録できるようインデックスを拡張している．KVEC（=NKの値が取り得る範囲．図5の例では3.） の値を大きくすれば，明らかに，その衝突頻度は下がるが，作業配列法と同じ衝突解消方法であるため，主メモリ使用量が増えてしまう．次に，30～53行目において，モーメント値を計算して，配列TRYmとTRYpに値を代入していく．ここで，上記24～28行目と同じループを回し，衝突発生をチェックする．35行目のIF文において，衝突が発生している場合は，配列CIGの値と粒子ID番号が異なるため判別可能である．衝突が発生している場合は，その粒子ID番号を配列CINへスタックする（49～50行目）．

以降は，このスタックされた粒子群に対して，同じ事を繰り返し，衝突を起こしスタックされる粒子が無くなるまでDO WHILE－ENDDOで反復処理（RETRY）を行う．

最後に，拡張したインデックスの要素に対する総和計算（reduction）を行う．

```
 1   !!
 2   !!    PROGRAM  LISTVEC
 3   !!
 4   REAL::  PTL(1:NMAX,1:4)
 5   REAL::  MOM(1:IMAX,1:4)
 6   REAL::  WXM(1:IMAX), WXP(1:IMAX)
 7   REAL::  WYM(1:IMAX), WYP(1:IMAX)
 8   REAL::  WZM(1:IMAX), WZP(1:IMAX)
 9   REAL::  WNM(1:IMAX), WNP(1:IMAX)
10
11   !!
12   !!   INITIALIZATION  LOOP
13   !!
14   WXM = 0.0;  WXP = 0.0     !..  Flux in X
15   WYM = 0.0;  WYP = 0.0     !..  Flux in Y
16   WZM = 0.0;  WZP = 0.0     !..  Flux in Z
17   WNM = 0.0;  WNP = 0.0     !..  Density
18
19   !!
20   !!   MAIN  LOOP
21   !!
22   !CDIR  LISTVEC
23   DO NN =  1, NMAX
24
25     IM = FLOOR( PTL(NN,4) );    IP=1  +IM     !.. Grid in X
26     XM = PTL(NN,4)-REAL(IM);    XP=1.0-XM     !.. Shape
27
28     WXM(IM) = WXM(IM) + XM * PTL(NN,1)
29     WXP(IP) = WXP(IP) + XP * PTL(NN,1)
30     WYM(IM) = WYM(IM) + XM * PTL(NN,2)
31     WYP(IP) = WYP(IP) + XP * PTL(NN,2)
32     WZM(IM) = WZM(IM) + XM * PTL(NN,3)
33     WZP(IP) = WZP(IP) + XP * PTL(NN,3)
34     WNM(IM) = WNM(IM) + XM
35     WNP(IP) = WNP(IP) + XP
36
37   ENDDO
38
39   !!
40   !!  REDUCTION  LOOP
41   !!
42   DO IX  = 1, IMAX
43      MOM(IX,1) = WXM(IX) + WXP(IX)
44      MOM(IX,2) = WYM(IX) + WYP(IX)
45      MOM(IX,3) = WZM(IX) + WZP(IX)
46      MOM(IX,4) = WNM(IX) + WNP(IX)
47   ENDDO
```

Fig. 4. Program for LISTVEC algorithm.

図4. LISTVEC法によるモーメント計算のプログラム例．メインループの直前にあたる22行目に，指示行が入れられている．

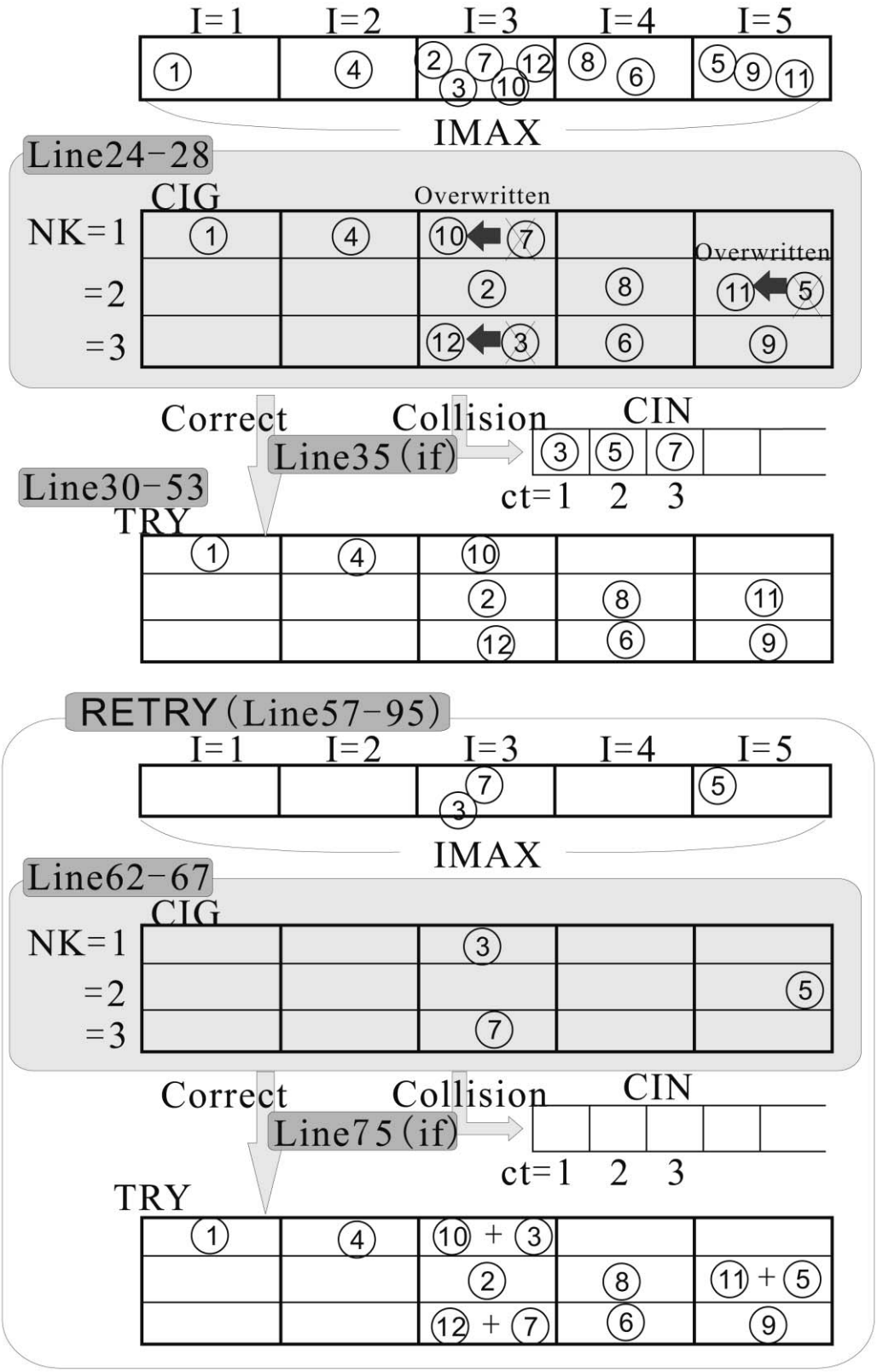

Fig. 5. Conceptual illustration for RETRY algorithm.

図5. RETRY法の概念図.

```
 1 !!
 2 !!    PROGRAM  RETRY
 3 !!
 4 INTEGER,PARAMETER::   RVEC = 2048, KVEC=2
 5 REAL::  PTL(1:NMAX,1:4)
 6 REAL::  MOM(1:IMAX,1:4)
 7 REAL::  TRYm(1:KVEC,1:IMAX,1:4), TRYp(1:KVEC,1:IMAX,1:4)
 8 INTEGER::  CIG(1:IMAX,1:KVEC)
 9 INTEGER::  CIN(1:NMAX)
10 INTEGER::  ct, ca
11
12 !!
13 !!   INITIALIZATION  LOOP
14 !!
15 TRYm = 0.0;  TRYp = 0.0
16 MOM  = 0.0
17
18 !!
19 !!   MAIN  LOOP
20 !!
21 ct = 0
22 DO NP = 1, NMAX, RVEC
23
24    DO NN = NP, MIN( NP+RVEC-1, NMAX)
25       NK = MOD(NN,KVEC) + 1
26       IM = FLOOR( PTL(NN,4) )
27       CIG(IM,NK) = NN
28    ENDDO
29
30 !CDIR NODEP
31    DO NN = NP, MIN( NP+RVEC-1, NMAX)
32       NK = MOD(NN,KVEC) + 1
33
34       IM = FLOOR( PTL(NN,4) )
35       IF( CIG(IM,NK) == NN ) THEN
36           IP = IM + 1
37           XM = PTL(NN,4)-REAL(IM)
38           XP = 1.0 - XM
39
40           TRYm(NK,IM,1) = TRYm(NK,IM,1) + XM * PTL(NN,1)
41           TRYp(NK,IP,1) = TRYp(NK,IP,1) + XP * PTL(NN,1)
42           TRYm(NK,IM,2) = TRYm(NK,IM,2) + XM * PTL(NN,2)
43           TRYp(NK,IP,2) = TRYp(NK,IP,2) + XP * PTL(NN,2)
44           TRYm(NK,IM,3) = TRYm(NK,IM,3) + XM * PTL(NN,3)
45           TRYp(NK,IP,3) = TRYp(NK,IP,3) + XP * PTL(NN,3)
46           TRYm(NK,IM,4) = TRYm(NK,IM,4)
47           TRYp(NK,IP,4) = TRYp(NK,IP,4)
48        ELSE
49           ct = ct + 1
50           CIN(ct) = nn
51       ENDIF
52    ENDDO
53 ENDDO
54 !-----
55
56 ca = ct
57 DO WHILE( 0 < ca )
58
59     ct = 0
60     DO NP = 1, ca, RVEC
61
62        DO NV = NP, MIN( NP+RVEC-1, ca)
63           NN = CIN(NV)
64           NK = MOD(NN,KVEC) + 1
65           IM = FLOOR( PTL(NN,4) )
66           CIG(IM,NK) = NN
67        ENDDO
68
69  !CDIR NODEP
70        DO NV = NP, MIN( NP+RVEC-1, ca)
71           NN = CIN(NV)
72           NK = MOD(NN,KVEC) + 1
73           IM = FLOOR( PTL(NN,4) )
74
75           IF( CIG(IM,NK) == NN ) THEN
76               IP = IM + 1
77               XM = PTL(NN,4)-REAL(IM)
78               XP = 1.0 - XM
79
80               TRYm(NK,IM,1) = TRYm(NK,IM,1) + XM * PTL(NN,1)
81               TRYp(NK,IP,1) = TRYp(NK,IP,1) + XP * PTL(NN,1)
82               TRYm(NK,IM,2) = TRYm(NK,IM,2) + XM * PTL(NN,2)
83               TRYp(NK,IP,2) = TRYp(NK,IP,2) + XP * PTL(NN,2)
84               TRYm(NK,IM,3) = TRYm(NK,IM,3) + XM * PTL(NN,3)
85               TRYp(NK,IP,3) = TRYp(NK,IP,3) + XP * PTL(NN,3)
86               TRYm(NK,IM,4) = TRYm(NK,IM,4)
87               TRYp(NK,IP,4) = TRYp(NK,IP,4)
88             ELSE
89               ct = ct + 1
90               CIN(ct) = nn
91             ENDIF
92        ENDDO
93     ENDDO
94     ca = ct
95 ENDDO
96
97 !!
98 !! REDUCTION  LOOP
99 !!
100 DO IX = 1, IMAX
101 DO IC = 1, 4
102 DO NK = 1, KVEC
103    MOM(IX,IC)= MOM(IX,IC) + TRYm(NK,IX,IC) + TRYp(NK,IX,IC)
104 ENDDO
105 ENDDO
106 ENDDO
```

Fig. 6. Program for RETRY algorithm. The NODEP compile-directive is necessary for vectorized operation.

図6. RETRY法によるモーメント計算のプログラム例．途中NODEP指示行を入れなければ，コンパイラは依存関係があるとしてループがベクトル化されない．

3. 速度比較

作業配列法・LISTVEC法・RETRY法のそれぞれについて，計算速度を測定した．粒子はシステム内にランダムに配置している．測定値は100回測定の平均値である．

3.1. LISTVEC指示行法と作業配列法

作業配列法とLISTVEC指示行法の計算に要した時間（それぞれT_W，T_Lとする）を比べ，格子点数依存を表示したものを図7に示す．粒子は，格子当り平均49個分布させている．上から，全計算時間の比と実時間，メインループ，初期化とreductionに要した時間の和，のそれぞれに関して，LISTVEC指示行法を用いて要した計算時間（赤線）と作業配列法を用いて要した計算時間（黒線）を表示している．結果，計算に要する時間は，常時，作業配列法が短い．しかし，格子点の増加に伴い，LISTVEC指示行法においては衝突発生確率が下がるためと，作業配列法における初期化とreduction作業コストが高くなる事により，格子点の増加とともに，計算時間比（T_L/T_W）が小さくなる事が見てとれ，約1.5である（最上段パネル）．

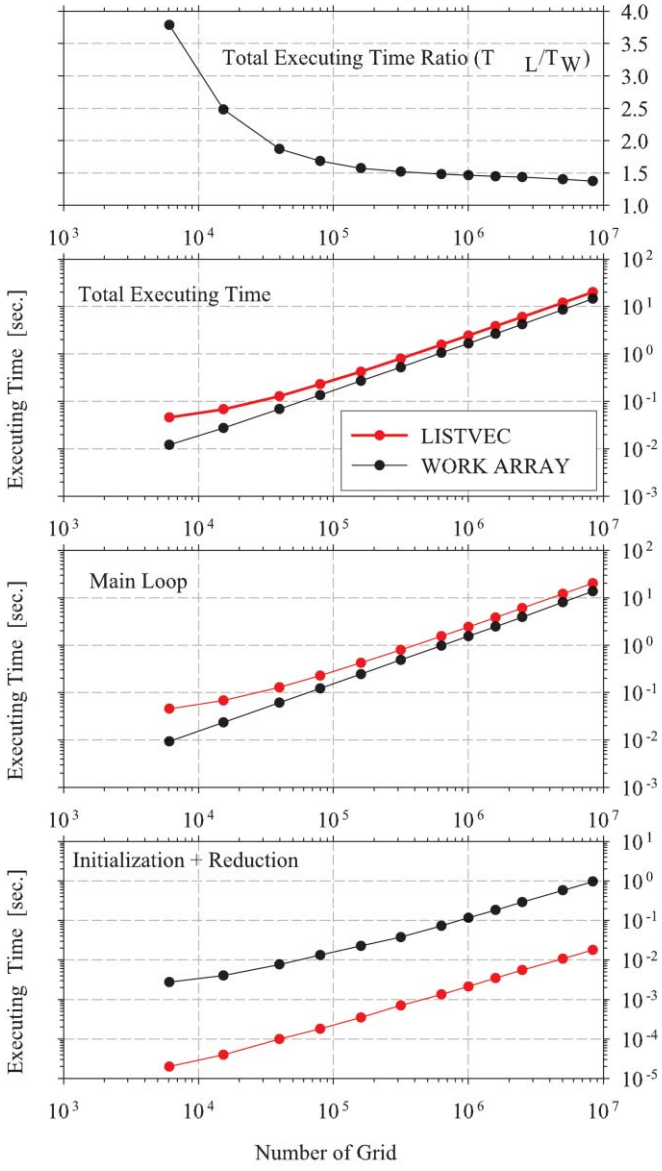

Fig. 7. Executing time for LISTVEC and WORK ARRAY algorithm. Dependence on the total number of grid are plotted. From top to bottom, the executing time of total calculation, main loop calculation and others of initialization plus reduction. The executing time of "others" has large calculation cost relative to total cost which is almost 7 %.

図7. LISTVEC指示行法（赤線）と作業配列法（黒線）を用いた計算に要した時間．上から，全計算時間・メインループ，初期化とreductionの和，に要した時間を格子数に対してプロット．作業配列法においては，格子数が増えると初期化とreduction に要する時間が全体の7%にも達する．

3.2. RETRY法の計算時間特性

RETRY法プログラムである図６において，変数RVECとKVECは，互いに独立であり，かつ，使い方は作業配列法におけるLVECと同じであるが，この変数の大きさは，ベクトル長とは関係が無い．よって，RETRY法の計算時間のRVECとKVECに対する依存性は自明ではない．図8にメインループ計算時間（T_{MR}）に関するKVECとRVECに対する依存性を示す．ここでは，純粋に，粒子ID番号の格子点位置への書き込み衝突頻度に関する依存を見るため，初期化とreductionを除いたメインループに対して計測する．格子数は6×10^5個，粒子は格子当り平均199個分布させている．縦軸の値は，作業配列法の計算時間（T_{MW}）に対する比（T_{MR}/T_{MW}）をプロットしている．上図は，RVEC=256と2,048とに固定した時のKVEC依存である．明らかに，KVECが2の冪上の時が高速に計算されており，中でもKVEC=32の時が最速である．下図は，KVEC=2, 32に固定した時のRVEC依存である．RVEC=256の場合の計算時間が他の場合より遅くなっていることから，RVECがベクトル長と関係ない事が分かる．また，計算時間比（T_{MR}/T_{MW}）の値が1.0～1.1程度であることから，LISTVEC法より高速に計算可能である．

KVECの値を大きくすれば，作業配列法のLVECの設置と同じく主メモリを圧迫するため，可能な限り小さい値が良い．LISTVEC法より使用する主メモリ量を同程度か少なく，かつ高速計算を実現することを考慮して，今後はKVEC=2，RVEC=2,048を採用する．

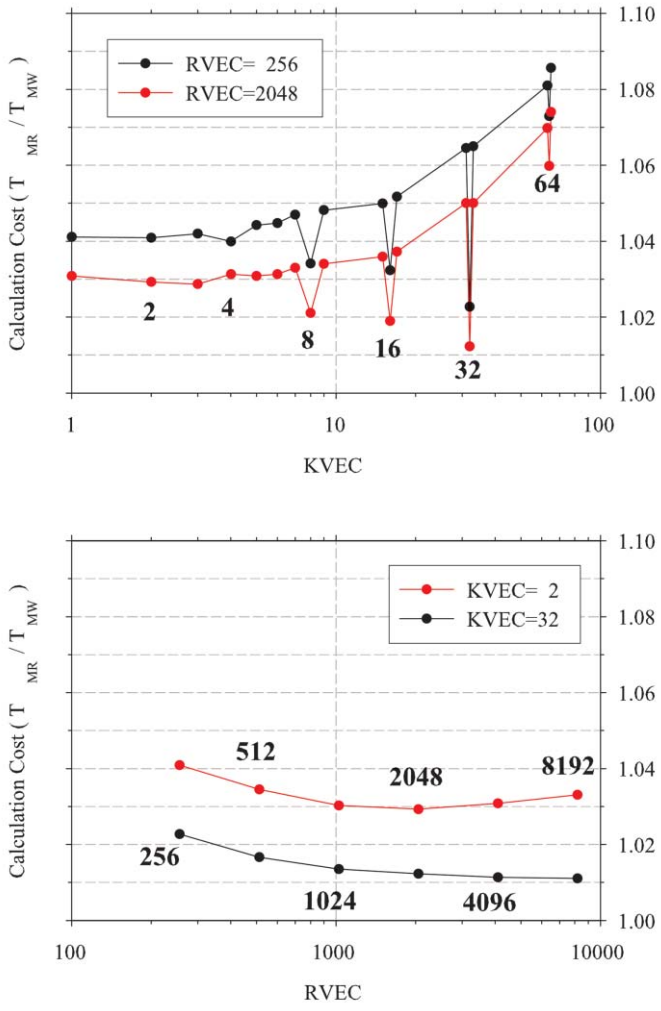

Fig. 8. Executing time for RETRY algorithm. Dependence on KVEC (top) and RVEC (bottom) parameters are plotted. The speed is fastest at KVEC=32. RVEC=256 is not good for RETRY algorithm even though the value corresponds to the length of the vector register.

図8. RETRY法におけるパラメータKVECとRVECに対する計算時間依存性．時間は作業配列法の計算時間との比（T_{MR}/T_{MW}）がプロットされている．（上図）KVECに対しては，2の冪上の値をとる時が高速であり，KVEC=32の時が最速であるが，主メモリ使用量を減らすため，なるべく小さい値を用いる方が良い．（下図）RVECに対しては，最大ベクトル長と同じ長さのRVEC=256の時は，他の場合より遅い事が見てとれる．

3.3. 全計算時間における速度比較

上記3.1節で述べたように，計算速度は格子数や粒子数に依存する．よって，両変数に対する計算速度比の依存性を図9に示す．表示している値は，全計算時間の，作業配列法の計算時間に対する比をプロットしている．左図のLISTVEC法に関しては，右下の領域，すなわち，粒子数が少なく格子点数が多くなる程，計算速度比が向上する．これは，LISTVEC法のメインループ内で衝突確率が減少することと，作業配列法において，格子点数が増加すると初期化とreductionの計算時間が長くなるためである．右図のRETRY法に関しては，粒子数が少ない程，計算速度が速くなるが，全体的に作業配列法と同じ速度で計算出来ていることが最大の特徴である．図10に，計算に使用する主メモリ量に関して，作業配列法との比を図9と同じ書式で示す．KVEC=2を採用したため，RETRY法における使用主メモリ量はLISTVEC法とほぼ同じである．

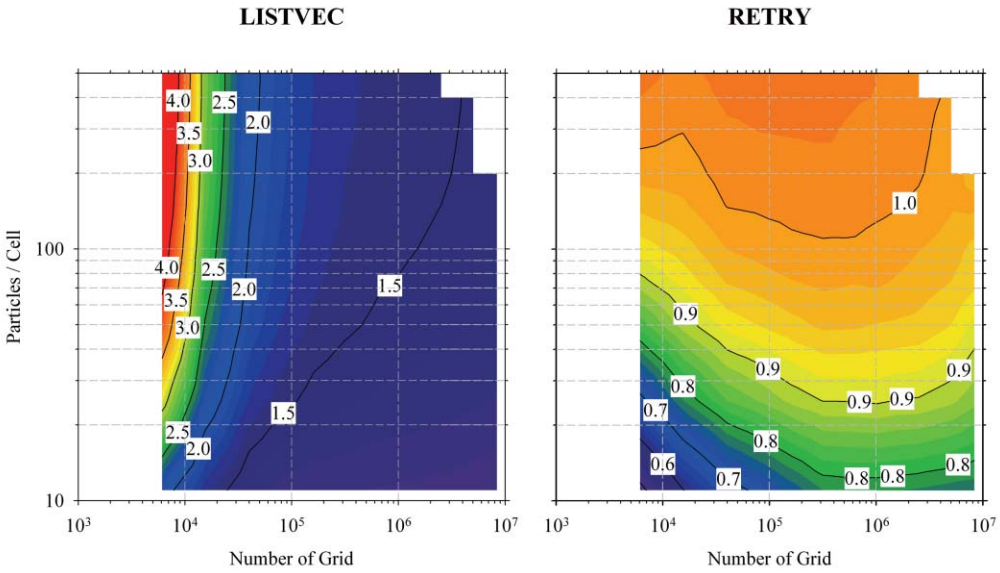

Fig. 9. Contour plot for the executing time ratio relative to WORK ARRAY algorithm in space of number of particle per cell versus total number of grid.

図9. LISTVEC法とRETRY法の計算時間について，作業配列法との比の値に対して等高線表示している．横軸は総粒子数．縦軸は格子点当りの平均粒子分布数．

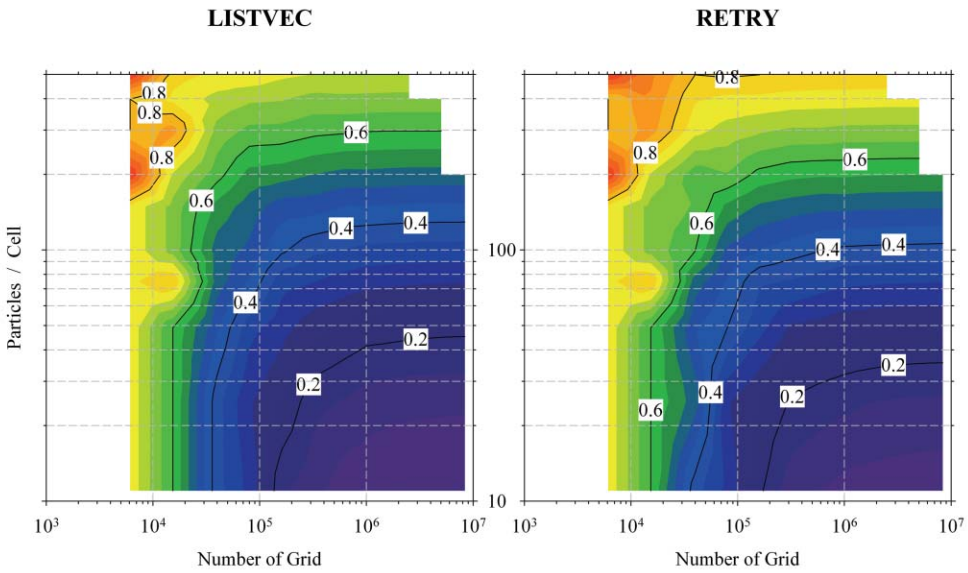

Fig. 10. Contour plot for used main memory in the same format as Fig. 9.

図10. LISTVEC法とRETRY法の使用メモリ量について，作業配列法との比の値に対して等高線表示している．図9とフォーマットは同じ．

4. まとめ

これまで，作業配列法が広くプラズマPICシミュレーションに使われてきた．そこでは，計算速度を上げるために，ベクトル長の256と同じ大きさの作業配列を確保していたため，主メモリを圧迫していた．256要素数の配列を確保することは，その数だけの粒子を格子当りに分布させることと同値である．通常，格子当り100個程度の粒子を使用するPICシミュレーションにおいては，本末転倒である．そこで，主メモリ使用量を抑えたLISTVEC指示行法・RETRY法が提案されている．これらの主メモリ使用量は，典型的なパラメータである粒子数が格子当り100個，格子総数10^6～10^7個では，作業配列法の40%以下である（図10）．今回，これらの手法を用いた場合の計算時間の計測を行った結果，LISTVEC法では計算時間が約1.5倍に，RETRY法では1.0~1.1倍長くなるだけと，ほぼ作業配列法と変わらない計算速度であることが確認された．RETRY法のプログラム作成は，他の２つに比べ煩雑であるが，その作成によって得られる主メモリの節約という効果が大きいため，使用を強く推薦する．

今回は，2次元計算におけるArea-Weighting手法のモーメント値計算に関して，計算速度を測定した．プラズマPIC計算では，1次のモーメント値を得る方法に電荷保存法も存在する．次回は，電荷保存法について検討したい．

参考文献

Birdsall, C. K. and A. B. Langdon, (1991), Plasma Physics via computer simulation, Institute of Physics Publishing, Bristol and Philadelphia.

Fujimoto, M., (1992), Instabilities in the Magnetopause Velocity Shear Layer, PhD thesis, Institute of Space and Astronautical Science, Japan.

Imamura, T., A Group of Retry-type Algorithms on a Vector Computer, (2005), *IPSJ journal, 46, SIG 7 (ACS 10)*, 52-62.

Matsumoto, H., and Y. Omura, (1993), Computer space plasma physics: Simulation techniques and software, Terra Pub.

Sugiyama, T., N. Terada, T. Murata, Y. Omura, H. Usui, and H. Matsumoto, Vectorized Particle Simulation Using "LISTVEC" Compile-directive on SX Super-computer, (2004), *IPSJ journal, 45, SIG 6 (ACS 6)*, 171-175.