JAERI-M
83-024

# FACOM230-75APUによる原子力コードのベクトル化

1983年2月

原田　裕夫・樋口　健二・石黒　美佐子
筒井　恒夫・藤井　実

日本原子力研究所
Japan Atomic Energy Research Institute

JAERI-M レポートは，日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは，日本原子力研究所技術情報部情報資料課（〒319－11 茨城県那珂郡東海村）あて，お申しこしください。なお，このほかに財団法人原子力弘済会資料センター（〒319-11 茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布をおこなっております。

JAERI-M reports are issued irregularly.
Inquiries about availability of the reports should be addressed to Information Section, Division of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.



編集兼発行　日本原子力研究所
印　　刷　山田軽印刷所

# FACOM 230-75 APUによる原子力コードのベクトル化

日本原子力研究所東海研究所計算センター
原田　裕夫・樋口　健二・石黒美佐子
筒井　恒夫[+]・藤井　実

（1983年1月28日受理）

将来のスーパーコンピュータの利用に備えて，原研でよく使用されている原子力コードを対象に，FACOM 230－75 APUを使ってベクトル計算処理効率について調査・研究を行った。ここで対象としたコードは，CITATION（3次元中性子拡散），SAP 5（構造解析），CASCMARL（照射損傷シミュレーション），FEM－BABEL（有限要素法による3次元拡散），GMSCOPE（電子顕微鏡像シミュレーション）およびDWBA（原子分子衝突断面積計算）である。またセル濃度計算ルーチンについても調査を行った。

各々のコードに対して，1.8（CASCMARL）から7.5（GMSCOPE）倍の有効な速度向上を得た。

本報告では，これらのコードの実行時における動的挙動解析，ベクトル化で使用された数値解法，ベクトル化のためのプログラム再構成手法，反復解法の数値実験，ベクトル化率，FACOM 230－75 APUによる速度向上率およびベクトル化に対するいくつかの所見について述べる。

＋　原子炉工学部

JAEIR-M 83-024

# Vectorization of Nuclear Codes on FACOM 230-75 APU Computer

Hiroo HARADA, Kenji HIGUCHI, Misako ISHIGURO,
Tsuneo TSUTSUI+ and Minoru FUJII
Computing Center, Tokai Research Establishment, JAERI

(Received January 28, 1983)

To provide for the future usage of supercomputer, we have investigated the vector processing efficiency of nuclear codes which are being used at JAERI. The investigation is performed by using FACOM 230-75 APU computer. The codes are CITATION (3D neutron diffusion), SAP5 (structural analysis), CASCMARL (irradiation damage simulation), FEM-BABEL (3D neutron diffusion by FEM), GMSCOPE (microscope simulation), DWBA (cross section calculation at molecular collisions). A new type of cell density calculation for particle-in-cell method is also investigated.

For each code we have obtained a significant speedup which ranges from 1.8 (CASCMARL) to 7.5 (GMSCOPE), respectively.

We have described in this report the running time dynamic profile analysis of the codes, numerical algorithms used, program restructuring for the vectorization, numerical experiments of the iterative process, vectorized ratios, speedup ratios on the FACOM 230-75 APU computer, and some vectorization views.

Keywords; Vectorization, Supercomputers, Nuclear Codes, CITATION, SAP5, GMSCOPE, CASCMARL, FEMBABEL, DWBA, Neutron Diffusion, Structural Analysis

---

+ Division of Reactor Engineering, Tokai Research Establishment, JAERI

# 目　　次

# CONTENTS

# 1. はじめに

## 1.1 原子力コードのベクトル化概要

前回行った「ベクトル計算処理の大型原子力コードへの適応性[1]」に関する研究の際に，対象となった原子力コードの中からベクトル処理効果が見込まれる3本のコードと，計算機利用者からベクトル化について要望のあった3本のコードと1つのプログラムについて，具体的にFACOM 230−75APU[2](以後F 75 APUと略す）においてベクトル化を行い，その効果を分析した。

ベクトル化を行った6本のコードは，CITATION（3次元多群中性子拡散コード），SAP 5（線型構造解析コード），CASCMARL（照射損傷シミュレーションコード），FEM−BABEL（有限要素法による3次元中性子拡散コード），GMSCOPE（電子顕微鏡シミュレーションコード），DWBA（原子，分子衝突過程の断面積計算コード）である。また，ADPIC（3次元汚染物質移流拡散計算コード）のセル濃度計算部分だけを抜き出したサブルーチンについてもベクトル化を行った。

将来のスーパーコンピュータ利用に備えて，これらの原子力コードのベクトル計算処理に対する適応性とその効果を，ベクトル計算機F 75 APUを使用して分析する。ベクトル化の手順は，まずサンプル入力データを用いて，コード全体の実行時の挙動についてFORTUNEツールによって解析する。このツールは，各サブルーチン，各ステートメント単位に実行回数や計算負荷（近似的には計算時間）およびIF文の真になる割合等を調べるものであるが，この結果は入力データに依存するので，目安として利用する。FORTUNEツールによる解析の結果，計算時間の割合の大きいルーチンを選択し，ベクトル演算用にプログラムを再構成（restructuring）する。そして，この再構成されたプログラムをF 75 APU用のAP−FORTRANで記述し，ベクトル命令を実行させる。

なお，本報告書は，昭和55年度下期から開始された原研と富士通(株)との共同研究「原子力コードの並列計算処理手法に関する研究」の成果の一つとして作成された。

## 1.2 ベクトル処理効果の概要

対象とした6本の原子力コードと1つのプログラムに対するベクトル計算処理効果の概要をTable 1.1に示す。

CITATIONについては，計算時間の主要部分を占めるサブルーチン（SLOR法の計算）において，ベクトル長が23であることが速度向上率（スカラ計算時間に対するベクトル計算時間の比）に影響しているが，このサブルーチンのベクトル処理効率の上がれば，全体の速度向上率も上がることが見込まれる。

SAP 5については，ベクトル長は比較的長く，ベクトル化された部分のみでは約4.1倍の速

度向上率を得ているが，入出力回数が非常に多くコード全体としては，2.0倍の速度向上率である。

CASCMARLについては，ベクトル化率（ベクトル演算が適用可能な部分の計算時間の割合）が約53％であるため，ベクトル長は比較的長いが，速度向上率は，1.8倍である。

FEM－BABELについて，オリジナルのコードに対して，入出力回数削減等の工夫によって，改良されたスカラ計算用コードは約3倍の速度向上を得た。この改良スカラコードに対して，ベクトル化を行った結果，ベクトル化率も99％と高く，4.3倍の速度向上率を得た。

GMSCOPEについては，計算時間が一つのサブルーチンに集中し，ベクトル化率が99％と高く，またDOループの組み換えによって長いベクトル長が得られたことによって，速度向上率は7.5倍となっている。また，像面上の波の振幅計算における指数関数部分を基本外部関数の代りにテーブル化しておくことによっても，大巾な速度向上を得た。

DWBAについて，オリジナルプログラムは入出力回数が非常に多いので，データのメモリ常駐化を行いF75固有の入出力に伴うオーバヘッドを減す改良を行った上でベクトル化した。ベクトル長は比較的長く，速度向上率は2.7倍となっている。

セル内粒子法による濃度計算の新しい方法のベクトル化については，3次元配列を1次元化してループ中のIF文を除去する改良アルゴリズムによって行った。その結果，速度向上率は7.8倍となっている。

Table 1.1 Summary of the vector processing efficiency

| Code name | Contents | Number of source statements | Vectorized ratio (%) | Vector length | Speedup ratio |
|---|---|---|---|---|---|
| CITATION | 3次元多群中性子拡散計算 | 26000 | 90 | 23 (一部 20102) | 2.1 |
| SAP 5 | 構造物の線型構造解析 | 16000 | 87 | 50～200 | 2.0 |
| CASCMARL | 放射線照射固体材質中のはじき出しカスケード シミュレーション | 10000 | 53 | 200～300 | 1.8 |
| FEM-BABEL | 有限要素法による3次元中性子拡散計算 | 6600 | 99 | 200～400 | 4.3 |
| GMSCOPE | 電子顕微鏡像シミュレーション | 240 | 99 | 10201 | 7.5 |
| DWBA | 原子，分子の衝突過程の断面積計算 | 4000 | 79 | 251 | 2.7 |
| セル濃度計算ルーチン | セル内粒子法による濃度計算中のセル濃度配分計算部分 | 100 | 99 | 10000 | 7.8 |

# 2. CITATION（3次元中性子拡散コード）

## 2.1 コードの概要

CITATION[3] は，3次元多群中性子拡散コードで，有限差分法により拡散方程式を解く。反復解法によって固有値（K－effective），固有ベクトル（中性子束の値）を求める。

中性子輸送に対する拡散近似を表わす基礎方程式は次の（2.1）式である。

$$-\nabla^2 D_{r,g}\,\phi_{r,g} + \left(\Sigma_{a,r,g} + h\,\Sigma_{b,r,g} + \sum_n \Sigma_{s,r,g\to n} + D_{r,g}\,B^2_{\perp g}\right)\phi_{r,g}$$

$$= \sum_n \left[\Sigma_{s,r,n\to g} + \frac{\chi_g}{k_e}\left(\nu\Sigma_{f,r,n} + h\,\nu\Sigma_{p,r,n}\right)\right]\phi_{r,n} \qquad (2.1)$$

$\nabla^2$ : The Laplacian geometric operator, $\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}$ in slab geometry

$\phi_{r,g}$ : The neutron flux at location r and in energy group g

$\Sigma_{a,r,g}$ : The macroscopic cross section for absorption, normally weighted over a representative flux energy spectrum

$\Sigma_{s,r,g\to n}$ : The macroscopic cross section for scattering of neutrons from energy group g to energy group n

$D_{r,g}$ : The diffusion coefficient, normally one-third of the recriprocal of the transport cross section

$B^2_{\perp g}$ : The buckling term to account for the effect of the Laplacian operator (leakage) in a dimension not treated explicitly

$\nu\Sigma_{f,r,g}$ : The macroscopic production cross section ( $\nu$ is the number of neutrons produced by a fission and $\Sigma_f$ is the cross section for fission )

$\chi_g$ : The distribution function for source neutrons

$k_e$ : The effective multiplication factor, ratio of rate of production of neutrons to rate of loss of neutrons from all causes, an unknown to be determined

h : The relative nuclide density

$\Sigma_{b,r,n\to g}$ : The absorption cross section associated with the search nuclides

$\nu\Sigma_{p,r,n}$ : The production cross section associated with the search nuclides

（2.1）式を差分化し，次式に示す固有値問題として解く。

$$Ax = \frac{1}{k_e} Fx \qquad (2.2)$$

A : the transport, scattering coupling and loss operator,

F : the fission source operator with distribution,

x : the flux vector,

$k_e$ : the unknown effective multiplication factnr.

数値計算法には，2重反復法（inner outer iteration）が用いられ，内部反復計算では，逐次線過大緩和法（SLOR法）が用いられている。

## 2.2 コードの動的挙動分析

Table 2.1の左側はANALYSISツールによって解析したCITATIONコードの主要ルーチンの木構造である。各ルーチンの主な計算内容は次のとおりである。

IPTM ：入力制御ルーチン

CALR ：全問題の計算制御ルーチン

EIGN ：固有値計算制御ルーチン

KLUX ：3次元形状用の固有値，中性子束計算

KNSD ：中性子束計算制御と境界漏洩計算

KTRI ：3角形シッシュ用逐次線過大緩和法の計算

KINS ：コントロールロッドロスの計算

FORTUNEツールによる動的挙動分析に使用した入力データは，3次元6角形状，6エネルギー群を3角形メッシュ46×23（column, row）×19 plane で解く問題である。Fig. 2.1は，その3角形メッシュを示している。

CITATIONコードの場合，固有値問題の解法では2次元か3次元か，あるいは形状等の問題の種類によって使用されるサブルーチンが異なる。また，取扱う問題のサイズ等によって，ユーザが解法を選択できるようにいくつかのルーチンが用意されている。

今回は前述の問題の入力データの場合について，FORTUNEツールの解析結果（Table 2.1右側）をもとにベクトル化対象ルーチンを選択した。

Table 2.1右側を見ると，中性子束計算の制御と境界漏洩計算を行うサブルーチン KNSDとそれから呼ばれているKTRI（3角形メッシュ用SLOR法の計算）とKINS（ロッドロスの計算）の3つのルーチンで，合計92.9％の計算時間を占めている。従って，ベクトル化対象サブルーチンとして，これら3つのルーチンを選び，次節から各サブルーチンのベクトル化の方法について具体的に述べる。

Table 2.1 Tree structure and FORTUNE output in CITATION code

| Subroutine name (Tree structure) | Executions | Time cost $\times 10^6$ | Ratio(%) |
|---|---|---|---|
| MAIN | 1 | | |
| INPT | 1 | | |
| SETV | 1 | | |
| IPTM | 1 | 1 | |
| KOMP | 2 | | |
| OVER | 1 | 1 | |
| KMOT | 2 | 1 | |
| MACR | 1 | | |
| CALR | 1 | | |
| EIGN | 1 | | |
| BIGS — XSET | 1 | | |
| KNFX | 1 | 2 | 0.1 |
| KLUX | 1 | 2 | 0.1 |
| KOOP | 52 | 52 | 2.1 |
| KNSD | 50 | 367 | 14.8 |
| KTRI | 900 | 1629 | 65.8 |
| KINS | 300 | 305 | 12.3 |
| EXTR | 50 | | |
| KBPR | 50 | 55 | 2.2 |
| KDUE | 1 | 17 | 0.7 |
| ITED | 1 | | |
| KNST | 1 | 43 | 1.7 |
| NMBL | 1 | 1 | |
| OUTC | 1 | | |
| KOUT | 1 | | |
| KDWT | 1 | 1 | |

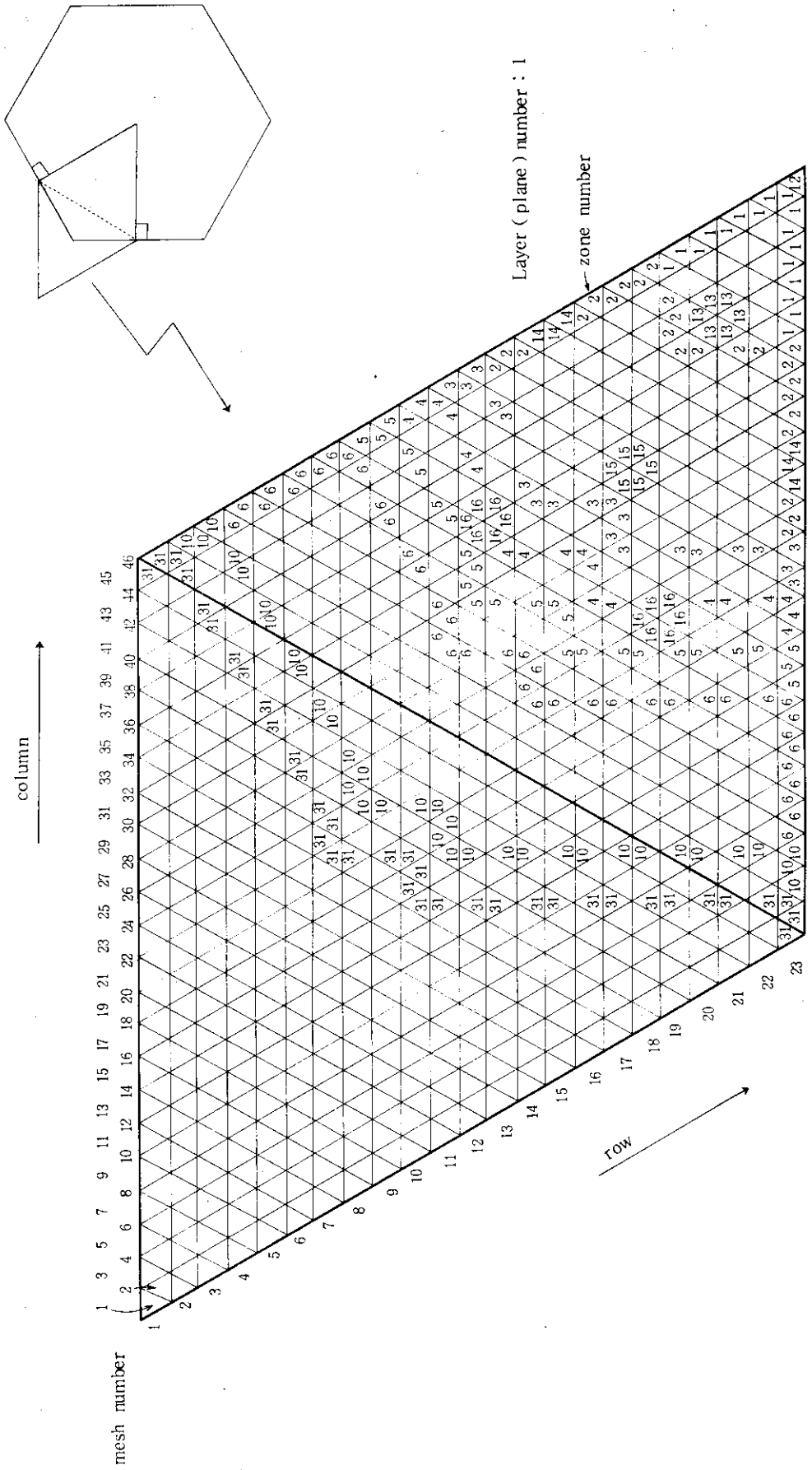



Fig. 2.1 Triangular mesh arrangement

## 2.3 プログラム再構成とベクトル化の方法

### 2.3.1 サブルーチンKNSDのベクトル化

ベクトル化によって大きい速度向上が得られるかどうかの鍵はベクトル長にある。ベクトル長が長ければ，ベクトル演算の準備に要するオーバヘッドも吸収される。従ってベクトル化の工夫の第1点は，できるかぎり長いベクトル長にすることで，サブルーチンKNSDのオリジナルプログラムを見ると，空間メッシュ（column，row，plane）のインデックスに関する3重ループが3箇所，境界漏洩計算など2方向（column，rowなど）メッシュのインデックスに関する2重ループが3箇所ある。これらのループは，参照されているデータに関して再帰的依存関係がないので，最内ループだけに着目せず2重，3重ループについてベクトル化を行う。

Fig. 2.2の上側は空間メッシュ（column，row，plane）に関する3重ループで最大値，最小値を求めるループである。このループでは最内ループ内に除算があり，ゼロ割りによる割込処理を防止するための1F文が含まれている。ベクトル化するためにはMASK関数[4]を使ってIF文を除去する。MASK関数は論理式の真偽によって第2，第3引数のいずれかの値をとる。配列特殊関数FMXVとFMNVはベクトル要素の最大値と最小値を求める関数である。Fig. 2.2の下側が，上の空間メッシュ（column，row，plane）に関する3重ループをベクトル化したAP-FORTRAN[4]によるプログラムである。

Fig. 2.3の上側オリジナルプログラム（inscatter source計算部分）を見ると，空間メッシュのインデックスに関する3重ループの内側に，さらにKKに関するループが2箇所あり，それらは内積計算である。このループのベクトル化の場合，内側のDO 116，118のループは，内積計算用のベクトル命令IPDでベクトル化できるが，FORTUNEツールによる測定では，KKのベクトル長が短いのでベクトル化せず，外側の空間メッシュのインデックスJ，I，KBに関する3重ループをDO 116，118の内側に入れてベクトル化する。そのため一時変数CKSSは，JVX＊IVX＊KBVXの長さの作業用配列WKVに変える。また，column方向メッシュのインデックスに関するループの内側にループ内では変化しない変数IX(24)に関するIF文がある。この場合は，3重ループを分割して，それぞれDO 116とDO 118の中へ入れることによって，IF文はループの外側に出せる。Fig. 2.3の下側が，上の空間メッシュに関する3重ループをベクトル化したAP-FORTRANによるプログラムである。

Fig. 2.4の上側オリジナルプログラムにおいて，3次元配列SIG（巨視的吸収断面積）は間接アドレス用の配列NCOMP（ゾーン番号）によって値を参照している。このループDO 107をベクトル化する場合，AP-FORTRANでは，2次元以上の配列におけるリストベクトル（間接アドレスによって参照される配列）の使用はできないので，作業用の1次元配列（ここではWKS）に置き換えなければならない。また，このDO 107にもループ内で変化しない変数に関するIF文が2つあるが，これもループを分割し，IF文は外に出せる。ベクトル化したプログラムが，Fig. 2.4の下側である。

(Original version)

```
      DO 151 KB=1,KBMAX
      N1 = 0
      DO 150 I=1,IMAX
      DO 149 J=1,JMAX
      N1 = N1 + 1
      TT1 = P1E(N1,KB)
      T2 = P2E(N1,KB,K)
      IF (TT1.EQ.0.0) GO TO 148
      RATO = T2/TT1
      RMX = AMAX1(RMX,RATO)
      RMN = AMIN1(RMN,RATO)
  148 CONTINUE
  149 CONTINUE
  150 CONTINUE
  151 CONTINUE
```

(Vectorized version)

```
      P1E(*,*)=AMASK(P1E(*,*).EQ.0.0 , 1000.0 , P1E(*,*))
      WKV(*)=P2E(*,*,K)/P1E(*,*)
      WKV(*)=AMASK(P1E(*,*).EQ.1000.0, 1.0 , WKV(*))
      RMX=FMXV(WKV(*))
      RMN=FMNV(WKV(*))
```

Fig.2.2 Vectorization of subroutine KNSD in CITATION code

(Original version)

```
      DO 107 L=1,LMAX
      M = NCOMP(L)
      E1(L,K) = XLAMDA*SIG(K,M,5)*PVOL(L)
      IF (IX(24).EQ.0) GO TO 107
      IF ((IX(17).EQ.-2).AND.(IX(71).GT.0)) E1(L,K) =SIG(K,M,5)*PVOL(L)
  107 CONTINUE
```

(Vectorized version)

```
      WKS(*)=SIG(K,*,5)
      E1(*,K)=XLAMDA*WKS(NCOMP(*))*PVOL(*)
      IF(IX(24).EQ.0.OR.IX(17).NE.-2.OR.IX(71).LE.0) GO TO 107
      E1(*,K)=WKS(NCOMP(*))*PVOL(*)
  107 CONTINUE
```

Fig.2.4 Vectorization of subroutine KNSD in CITATION code

(Original version)

```
      DO 122 KB=1,KBVX
      DO 121 I=1,IVX
      NN1 = (I-1)*JVX
      DO 120 J=1,JVX
      N1 = NN1 + J
      L = NRGNE(J,I,KB)
      M = NCOMP(L)
      P1E(N1,KB) = P2E(N1,KB,K)
      IF (IX(24).GT.0) GO TO 117
      CKSS = 0.0
      DO 116 KK=KSCT1,KSCT2
      CKSS = CKSS + SCAC(KK,M,K)*P2E(N1,KB,KK)
  116 CONTINUE
      SCATE(J,I,KB) = CKSS*PVOL(L) + SOURE(J,I,KB)*XII(K)
      GO TO 119
  117 CKSS = SOURE(J,I,KB)*SIG(K,M,4)
      DO 118 KK= KSCT1,KSCT2
      CKSS = CKSS + SCAC(K,M,KK)*P2E(N1,KB,KK)
  118 CONTINUE
      SCATE(J,I,KB) = CKSS*PVOL(L)
  119 CONTINUE
  120 CONTINUE
  121 CONTINUE
  122 CONTINUE
```

(Vectorized version)

```
      MLV(*)=NCOMP(NRGNE(*,*,*))
      P1E(*,*)=P2E(*,*,K)
      IF(IX(24).GT.0) GO TO 117
      WKV(*)=0.0
      DO 116 KK=KSCT1,KSCT2
      WKS(*)=SCAC(KK,*,K)
      WKV(*)=WKV(*)+WKS(MLV(*))*P2E(*,*,KK)
  116 CONTINUE
      SCATE(*,*,*)=WKV(*)*PVOL(NRGNE(*,*,*))+SOURE(*,*,*)*XII(K)
      GO TO 119
CC
  117 WKS(*)=SIG(K,*,4)
      WKV(*)=SOURE(*,*,*)*WKS(MLV(*))
      DO 118 KK=KSCT1,KSCT2
      WKS(*)=SCAC(K,*,KK)
      WKV(*)=WKV(*)+WKS(MLV(*))*P2E(*,*,KK)
  118 CONTINUE
      SCATE(*,*,*)=WKV(*)*PVOL(NRGNE(*,*,*))
  119 CONTINUE
```

Fig.2.3 Vectorization of subroutine KNSD in CITATION code

2.3.2　サブルーチンKINSのベクトル化

Fig. 2.5は，サブルーチンKINSのオリジナルプログラムの主要部分である。サブルーチン全体は，空間メッシュ（column，row，plane）のインデックスに関する3重ループから成り，ループ内には各種の多数のＩＦ文が含まれている。ＩＦ文は，領域の境界判定，問題の形状による判定，中性子束の値（ゼロか否か）による判定などである。このプログラムをベクトル化のために再構成したのが，Fig. 2.6である。Fig. 2.6の上方のａ，ｂの部分は，Fig. 2.5のａｂの部分を再構成したものであり，Fig. 2.6下方のｃ，ｄの部分は，Fig. 2.5のｃｄの部分に対応している。この再構成は，次の2つの方法で行った。一つは，ループ内では変化しない変数（形状判定のNUAC(5)やrotation判定のＩNRB）に関するＩＦ文を3重ループの外に出す方法である。もう一つは，空間メッシュの境界判定によるＩＦ文を除去するため，column，row方向メッシュのインデックスＪ，Ｉ（ＤＯループの制御変数）の初期値1を2に，あるいは終値を（終値－1）に変更することによって，3重ループを分割しベクトル化する方法である。

このようにして再構成され，分割されたそれぞれの3重ループに対して，ベクトル演算用にAP-FORTRANで記述したものが，Fig. 2.7.ａ，Fig. 2.7.ｂ，Fig. 2.7.ｃ，Fig. 2.7.ｄである。Fig. 2.7.ａの上方は，ベクトル演算のために新に必要になった作業用配列の宣言であり，下方はFig. 2.6のａの部分に対応したベクトル化されたプログラムである。Fig. 2.7 ｂ，ｃ，ｄはそれぞれFig. 2.6のｂ，ｃ，ｄの部分に対応したベクトル化されたプログラムである。

Fig. 2.7.ａのベクトル演算用共通準備部分について，作業用1次元配列Ｐ２ＷとＮＲは，中性子束の値（3次元配列Ｐ２Ｅ）やゾーン番号（3次元配列NRGNE）をリストベクトルを使用して間接アドレス参照するために置き換えられたものである。論理型配列ＬＰ０は，中性子束の値に関するＩＦ文を除去するための作業用配列である。中性子束の値（Ｐ２Ｅ）が，ゼロか否かの判定を行うＩＦ文はオリジナルプログラムでは16ケ所もある（例えばFig. 2.5）。Ｐ２Ｅは，サブルーチンKINSの中では参照されるだけで更新されないので，予めゼロか否かの判定を一度行って論理型配列を作っておく。そうすれば，3重ループ内ではインデックスを変えるだけで論理値が参照でき，ＩＦ文が除去される。これら作業用配列の作成も参照もベクトル演算で行える。

効率のよいベクトル処理のためには，参照されるデータは1次元配列に連続的に入っていることが望ましいが，3重ループ内で参照されているtop，bottom diff. equa. constant（3次元配列 DCONBE）やleft，right diff. equa. constant（DCONRE）などを，できるだけ長いベクトル長で1次元化した場合，そのまま作業用1次元配列に代入したのでは，データ参照の仕方によって不規則かつ不連続となってしまう。というのは，ループ内で同時に参照され中性子束の配列Ｐ２Ｅとはサイズがそれぞれ少しずつ異なること，またcolumu，row方向メッシュのインデックスＪ，Ｉに関するＩＦ文除去のため，Ｊ，Ｉの初期値，終値を変更したことによる。配列Ｐ２Ｅ，DCONBE，DCONREのサイズは次のとおりである。

P2E (JVX*IVX, KBVX, KVX)

DCONBE (JVX* (IVX+1), KBVX, KVX)

DCONRE ((JVX+1) * (IVX+1), KBVX, 2*KVX)

3重ループ内で参照される時に，データが1次元配列の中に連続的に入っているようにするため，3次元配列DCONBEまたはDCONREを参照の仕方に従って，一度2次元配列 DCWに代入し，さらにリストベクトル使用のため，1次元配列DWEに代入する。このようにして，中性子束の値と上下，左右の係数などがベクトル演算で同時に参照できる。3次元の空間メッシュ領域の境界における特殊処理がある場合，ベクトル演算に用いられるデータ連続性には注意を要する。

Fig. 2.6の再構成プログラムにおいて，3重ループ内にあった中性子束の値（P2E)に関するIF文を除去して代りに作られたのが，論理型のベクトル演算代入文

LP2(*)=.NOT.LP0(...).AND.LP0(...)

である。（Fig. 2.7.a.b.c参照）この論理型配列LP2からリストベクトルの間接アドレス用ポインタベクトルを作成するのが，配列特殊関数IDXLである。関数IDXLは，論理型の配列から要素値が真であるところのインデックスを取り出す関数である。これによって，リストベクトルで参照されるデータは論理値が真のインデックスに対してのみ演算が実行され，余分な演算が行われないので効率的である。例えば，Fig. 2.1のゾーン番号31から左側の6角形状炉心外側に当る中性子束の値がゼロである部分などの計算が省かれる。また，この論理型配列LP2は制御棒（control rod）の位置を捜す役割をしている。ただし，この論理型配列の値がすべて偽であった場合は，配列特殊関数IONCで調べた上，それ以後の演算をパスするようにした。（Fig. 2.7.a, b, c, d参照）

このように，IF文を含むループのベクトル化においては，そのIF文の真偽の比率によってベクトル化の方法を選択しなければならない。この真偽の比率についてはFORTUNEツールの解析結果が参考になる。

```
      INRB = IX(72) + 1
      N = IX(20)
      DO 146 KB = 1,KBMAX
      DO 145 I = 1,IMAX
      NN1= (I-1)*JVX
      NN2= (I-1)*JVXP1
      DO 144 J = 1,JMAX
      N1= NN1 + J
       . . . . .
      L = NRGNE(J,I,KB)
      M = NCOMP(L)
      IF (P2E(N1,KB,K)) 103,101,103
  101 IF (XMIS(2).GE.0) GO TO 143
       . . . . .
      B2(M,K) = B2(M,K) + TT5*PVOL(L)
      GO TO 143
  103 IF (KB-1) 106,106,104
       . . . . .
  110 IF (I-1) 114,114,111
  111 CONTINUE
      IF (NUAC(5).NE.14) GO TO 112
      IF (P2E(N7,KB,K).NE.0) GO TO 114
      LT = NRGNE(J+1,I-1,KB)
      GO TO 113
  112 CONTINUE
      IF (P2E(N2,KB,K).NE.0) GO TO 114
      LT = NRGNE(J,I-1,KB)
  113 CONTINUE
      MT = NCOMP(LT)
      B2(MT,K) = B2(MT,K)+P2E(N1 ,KB,K)*DCONBE(N1 ,KB,N)
  114 IF (I-IMAX) 117,115,115
       . . . . .
       . . . . .
  122 IF (J.LE.1) GO TO 123
      IF (P2E(N4,KB,K).NE.0.0) GO TO 125
      LL = NRGNE(J-1,I,KB)
      ML = NCOMP(LL)
      GO TO 124
  123 IF (INRB.NE.2) GO TO 125
      N12 = NN1 + JVX
      IF (P2E(N12,KB,K).NE.0.0) GO TO 125
      LL = NRGNE(JVX,I,KB)
      ML = NCOMP(LL)
  124 B2(ML,K) = B2(ML,K) + P2E(N1 ,KB,K)*DCONRE(N8 ,KB,N)
  125 IF (J.GE.JVX) GO TO 126
       . . . . .
  143 CONTINUE
  144 CONTINUE
  145 CONTINUE
  146 CONTINUE
       . . . . .
      RETURN
      END
```

ab

cd

Fig.2.5 Orivinal version of subroutine KINS in CITATION code

```
      . . . . .
      IF(NUAC(5).EQ.14) GO TO 730
      DO 712 KB=1,KBMAX
      DO 712 I=2,IMAX
      NN1=(I-1)*JVX
      N2P=NN1-JVX
      DO 712 J=1,JMAX
      IF(P2E(J+NN1,KB,K).EQ.0.0) GO TO 712
      IF(P2E(J+N2P,KB,K).NE.0.0) GO TO 712
      MT=NCOMP(NRGNE(J,I-1,KB))
      B2(MT,K)=B2(MT,K)+P2E(J+NN1,KB,K)*DCONBE(J+NN1,KB,N)
  712 CONTINUE
      GO TO 750
C
  730 CONTINUE
      DO 732 KB=1,KBMAX
      DO 732 I=2,IMAX
      NN1=(I-1)*JVX
      N7P=NN1-JVX+1
      DO 732 J=1,JMAX
      IF(P2E(J+NN1,KB,K).EQ.0.0) GO TO 732
      IF(P2E(J+N7P,KB,K).NE.0.0) GO TO 732
      MT=NCOMP(NRGNE(J+1,I-1,KB))
      B2(MT,K)=B2(MT,K)+P2E(J+NN1,KB,K)*DCONBE(J+NN1,KB,N)
  732 CONTINUE
  750 CONTINUE
      . . . . .
      . . . . .
      DO 812 KB=1,KBMAX
      DO 812 I=1,IMAX
      NN1=(I-1)*JVX
      NN2=(I-1)*JVXP1
      N4P=NN1-1
      DO 812 J=2,JMAX
      IF(P2E(J+NN1,KB,K).EQ.0.0) GO TO 812
      IF(P2E(J+N4P,KB,K).NE.0.0) GO TO 812
      ML=NCOMP(NRGNE(J-1,I,KB))
      B2(ML,K)=B2(ML,K)+P2E(J+NN1,KB,K)*DCONRE(J+NN2,KB,N)
  812 CONTINUE
C
      IF(INRB.NE.2) GO TO 860
      DO 814 KB=1,KBMAX
      DO 814 I=1,IMAX
      NN1=(I-1)*JVX
      NN2=(I-1)*JVXP1
CC    J=1
      IF(P2E(1+NN1,KB,K).EQ.0.0) GO TO 814
      IF(P2E(JVX+NN1,KB,K).NE.0.0) GO TO 814
      ML=NCOMP(NRGNE(JVX,I,KB))
      B2(ML,K)=B2(ML,K)+P2E(1+NN1,KB,K)*DCONRE(1+NN2,KB,N)
  814 CONTINUE
  860 CONTINUE
      . . . . .
```

a

b

c

d

Fig.2.6 Restructured version of subroutine KINS in CITATION code

```
C
      REAL DCW(JIVX,KBVX),DWE(JIKBZ),P2W(JIKBV),B2W(MVX)
      REAL PDW(NTW)
      INTEGER NR(JIKBV),NNW(NTW),NPW(NTW),N2W(NTW)
      INTEGER NNX(IKBV),NPX(IKBV),N2X(IKBV)
      LOGICAL LP2(LPT),LPO(JIKBV)
C
      INDEX JIV/1,JIVX/,JV/1,JVX/,JN/JNS,JNE/,JX/JXS,JXE,JVX/
       . . . . .
       . . . . .
      JIKBV=JIVX*KBVX
        ALLOCATE P2W,LPO,NR,B2W
      P2W(*)=P2E(*,*,K)
      NR(*)=NRGNE(*,*,*)
      LPO(*)=P2W(*).EQ.0.0
       . . . . .
      JIKBZ=JIKBV
        ALLOCATE DWE,DCW
      DO 710 KB=1,KBVX
      DCW(*,KB)=DCONBE(JIV,KB,N)
  710 CONTINUE
      DWE(*)=DCW(*,*)
      B2W(*)=B2(*,K)
        FREE DCW
C
      IF(NUAC(5).NE.14) GO TO 730
      JNS=2
      JNE=JIKBV-JVX+1
      LPT=JNE-1
        ALLOCATE LP2
      N1P=JVX-1
      LP2(*)=.NOT.LPO(JN+N1P).AND.LPO(JN)
      NTW=IONC(LP2(*))
      IF(NTW.LT.1) GO TO 750
        ALLOCATE NNW,NPW,PDW
      NNW(*)=IDXL(LP2(*))+1
      NPW(*)=NNW(*)+N1P
      NNW(*)=NR(NNW(*))
      NNW(*)=NCOMP(NNW(*))
      PDW(*)=P2W(NPW(*))*DWE(NPW(*))
      DO 718 NW=1,NTW
  718 B2W(NNW(NW))=B2W(NNW(NW))+PDW(NW)
      B2(*,K)=B2W(*)
        FREE PDW,NPW,NNW
      GO TO 750
```

ベクトル演算用
共通準備部分

a

Fig.2.7.a Vectorized version of subroutine KINS in CITATION code

```
  730 CONTINUE
      JNS=1
      JNE=JIKBV-JVX
      LPT=JNE
        ALLOCATE LP2
      LP2(*)=.NOT.LPO(JN+JVX).AND.LPO(JN)
      NTW=IONC(LP2(*))
      IF(NTW.LT.1) GO TO 750
        ALLOCATE NNW,NPW,PDW
      NNW(*)=IDXL(LP2(*))
      NPW(*)=NNW(*)+JVX
      NNW(*)=NR(NNW(*))
      NNW(*)=NCOMP(NNW(*))
      PDW(*)=P2W(NPW(*))*DWE(NPW(*))
      DO 738 NW=1,NTW
  738 B2W(NNW(NW))=B2W(NNW(NW))+PDW(NW)
      B2(*,K)=B2W(*)
        FREE PDW,NPW,NNW
C
  750 CONTINUE
        FREE LP2,DWE
       . . . . .
```

Fig.2.7.b Vectorized version of subroutine KINS in CITATION code

```
      . . . . .
      JIKBZ=JIKBV
        ALLOCATE DWE,DCW
      DO 810 KB=1,KBVX
      DO 810 I=1,IVX
      NN1=(I-1)*JVX
      NN2=(I-1)*JVXP1
      DCW(JV+NN1,KB)=DCONRE(JV+NN2,KB,N)
  810 CONTINUE
      DWE(*)=DCW(*,*)
      B2W(*)=B2(*,K)
      LPT=JIKBV
        FREE DCW
        ALLOCATE LP2
      JNS=2
      JNE=JIKBV
      LP2(JN)=.NOT.LPO(JN).AND.LPO(JN-1)
      JXS=1
      JXE=JIKBV-JVX+1
      LP2(JX)=.FALSE.
      NTW=IONC(LP2(*))
      IF(NTW.LT.1) GO TO 830
        ALLOCATE NPW,NNW,PDW
      NPW(*)=IDXL(LP2(*))
      NNW(*)=NPW(*)-1
      NNW(*)=NR(NNW(*))
      NNW(*)=NCOMP(NNW(*))
      PDW(*)=P2W(NPW(*))*DWE(NPW(*))
      DO 824 NW=1,NTW
  824 B2W(NNW(NW))=B2W(NNW(NW))+PDW(NW)
      B2(*,K)=B2W(*)
      FREE PDW,NNW,NPW
  830 CONTINUE
        FREE LP2,DWE
```

Fig.2.7.c Vectorized version of subroutine KINS in CITATION code

```
      IF(INRB.NE.2) GO TO 860
      IKBV=IVX*KBVX
      LPT=IKBV
      JIKBZ=JP1IXZ*KBVX
        ALLOCATE NPX,NNX,N2X,LP2,DWE
      DWE(*)=DCONRE(*,*,N)
      JN1=0
      DO 842 KB=1,KBVX
      N1P=(KB-1)*JIVX-JVX+1
      NJP=N1P+JVX-1
      N2P=(KB-1)*JP1IXZ-JVXP1+1
      DO 842 I=1,IVX
      JN1=JN1+1
      NPX(JN1)=JVX*I+N1P
      NNX(JN1)=JVX*I+NJP
      N2X(JN1)=JVXP1*I+N2P
  842 CONTINUE
      LP2(*)=P2W(NPX(*)).NE.0.0.AND.P2W(NNX(*)).EQ.0.0
      NTW=IONC(LP2(*))
      IF(NTW.LT.1) GO TO 852
        ALLOCATE NPW,NNW,N2W,PDW
      NNW(*)=IDXL(LP2(*))
      NPW(*)=NPX(NNW(*))
      N2W(*)=N2X(NNW(*))
      NNW(*)=NNX(NNW(*))
      NNW(*)=NR(NNW(*))
      NNW(*)=NCOMP(NNW(*))
      PDW(*)=P2W(NPW(*))*DWE(N2W(*))
      DO 850 NW=1,NTW
  850 B2W(NNW(NW))=B2W(NNW(NW))+PDW(NW)
      B2(*,K)=B2W(*)
        FREE PDW,N2W,NNW,NPW
  852 CONTINUE
        FREE DWE,LP2,N2X,NNX,NPX
  860 CONTINUE
      . . . . .
```

Fig.2.7.d Vectorixed version of subroutine KINS in CITATION code

2.3.3　サブルーチンKTRIのベクトル化

サブルーチンKTRIは，SLOR法（line relaxationとrestrained overrelaxation）を実行するサブルーチンで，オリジナルプログラムを見ると，全体はplane方向とrow 方向メッシュのインデックスKBとIに関する2重ループから成っており，その内側にcolumn 方向メッシュのインデックスJに関して，定数計算，前進消去，後退代入の3つのループがある。（Fig. 2.8，Fig. 2.9，Fig. 2.10参照）

このサブルーチンのベクトル化について，2重ループ内で参照されている中性子束の値が，そのループ内で更新されているため，また，前進消去と後退代入の計算を行うJ（column方向メッシュのインデックス）に関するループで再帰的データ依存関係があるため，プログラム構造の再構成だけではベクトル化できず，反復解法の変更によってベクトル化を行う。

ベクトル化の方針としては，一番外側のplane方向メッシュのインデックスKBに関するループはそのままにし，内側のrow 方向とcolumn方向メッシュのインデックスIとJに関するループをベクトル化するが，column方向インデックスJに関して，再帰的依存関係のある前進消去と後退代入のループについてはrom方向インデックスIについてのみベクトル化する。これは第1段階のベクトル化であり，さらに進んだベクトル化については後述する。

Fig. 2.8は，定数計算部分のrom 方向とcolumn方向メッシュのインデックスIとJに関する2重ループのベクトル化について示している。このループでは再帰的データ依存関係がないので，column方向メッシュのインデックスJについてもベクトル化できる。ただし，オリジナルプログラムにおいてこのループ内で参照されている中性子束の値（配列P2E）は，row方向メッシュのインデックスIに関するループ内の別な箇所で更新されているが，Fig. 2.8下方に示したプログラムでは，ベクトル化によって一部に更新される前の古い値が使用されるため，収束が遅くなり反復回数が増加する。

Fig. 2.8上方のオリジナルプログラムにおいて，DO 108のループには，column方向メッシュのインデックスJが偶数か奇数かによって場合分けを行うIF文が含まれている。これはFig. 2.1を見るとわかるようにcolumn方向のメッシュは偶数のインデックスと奇数のインデックスで3角形メッシュの向きが反対になっているからである。このIF文を含むベクトル化，MASK関数によって行う。

Fig. 2.9上方は，前進消去部分の2重ループのオリジナルプログラムである。ここでは，

$$\beta_{j,i} = \frac{S_j + d_{j-1,i} * \delta_{j-1,i} * \delta_{j-1,i}}{d_{j,i}} \qquad (2.3)$$

$$\delta_{j,i} = \frac{d_{j,i}}{P_{j,i} + e_{j,i} - d_{j-1,i} * \delta_{j-1,i}} \qquad (2.4)$$

の式を計算しているが，これを見ると$\beta$も$\delta$もいずれもcolumn方向メッシュのインデックスJに関して，再帰的データ依存関係がある。従ってrow方向メッシュのインデックスJについてのみベクトル化する。なお，これらの計算は，中性子束の値（P2E）がゼロの時は行わなくてよいので，MASK関数を使ってベクトル化した。ベクトル化したプログラムは，Fig. 2.9下方に示したものである。

Fig. 2.10上方は，後退代入部分の2重ループのオリジナルプログラムである。ここでは，

$$t_{j,i} = \delta_j * (t_{j+1,i} + \beta_j) \quad (2.5)$$

$$\phi_{j,i} = \phi_{j+1,i} + \omega * (t_{j,i} - \phi_{j+1,i}) \quad (2.6)$$

$$0.5 * t_{j,i} < \phi_{j,i} < t_{j,i} + \phi_{j+1,i} \quad (2.7)$$

の式を計算している。このループでは，column方向メッシュのインデックスjが，後退代入のため減少方向に変化するので，tとφはjに関してやはり再帰的データ依存関係となる。

（2.7）式は，反復計算の初期における誤差を抑制するためのもので，Fig.2.10上方のオリジナルプログラムのaの部分に対応しており，この部分をMASK関数を使用してベクトル化したのが，同図下方bの部分である。

以上がサブルーチンKTRIについて今回行ったベクトル化の主な方法であるが，さらにベクトル化を進める場合，次の項目が考えられる。

(1) 1次元のみベクトル化を行う場合，column方向，row方向（またはplane方向）について最もベクトクトル長の長いメッシュ方向を選択し，ベクトル化する。この場合には，配列要素の入れ換えが必要となる。

(2) オリジナルプログラムにおいて，再帰的データ依存関係のあるcolumn方向メッシュのインデックスJに関するループはそのままにし，row方向のループのベクトル化に加えて，plane方向のループのベクトル化についても検討する。この場合，古い値を収束計算に使用することによる反復回数の増加とベクトル計算による計算速度向上とのバランスが問題となる。

(Original version)

```
      DO 136 KB=1,KBVX
      ASSIGN 100 TO IBR1
      IF (KB.LE.1) ASSIGN 101 TO IBR1
      ASSIGN 102 TO IBR2
      IF (KB.GE.KBVX) ASSIGN 103 TO IBR2
      KBM1 = KB - 1
      KBP1 = KB + 1
      DO 135 I=1,IVX
        . . . . .
      DO 108 J=1,JVX
      NOE = J-(J/2)*2
      N1 = N1 + 1
      M2 = M2+1
      M3 = M3+1
      CKSS = SCATE(J,I,KB)
      GO TO IBR1, (100,101)
  100 CKSS = CKSS + P2E(N1 ,KBM1,K)*DCONBK(N1 ,KB,N)
  101 GO TO IBR2, (102,103)
  102 CKSS = CKSS + P2E(N1 ,KBP1,K)*DCONBK(N1 ,KBP1,N)
  103 CONTINUE
      IF (NOE.EQ.0) GO TO 105
      GO TO IBR3, (104,107)
  104 CKSS = CKSS+P2E(M3,KB,K)*DCONBE(N1,KB,N)
      GO TO 107
  105 GO TO IBR4, (106,107)
  106 CKSS = CKSS+P2E(M2,KB,K)*DCONBE(M2,KB,N)
  107 TSOUR(J) = CKSS
  108 CONTINUE
        . . . . .
  135 CONTINUE
  136 CONTINUE
      RETURN
```

(Vectorized version)

```
      REAL TSOU(46,23),P2F(1150),DEW(46,23),BEW(46,23)
      REAL E1W(31),WKM(23),WKP(23),WKT(23),WKI(23),WKX(23)
      INTEGER NWK(23),LL(23)
      LOGICAL LOE(46,23),LPO(23)
C
      DATA P2F/1150*0.0/,LOE/1058*.FALSE./
      INDEX JN/JNS,JNE/,JIV/1,JIVX/,LN/1,LNE/
      INDEX IDJ/IDJS,IDJE,JVX/,IDP/IDPS,IDPE,JVXP1/
      DO 310 JOE=1,JVX,2
  310 LOE(JOE,*)=.TRUE.
      E1W(*)=E1(*,K)
      TSOU(*,*)=SCATE(*,*,KB)
      IF(KB.EQ.1) GO TO 101
      TSOU(*,*)=P2E(*,KB-1,K)*DCONBK(*,KB,N)+TSOU(*,*)
  101 IF(KB.EQ.KBVX) GO TO 103
      TSOU(*,*)=P2E(*,KB+1,K)*DCONBK(*,KB+1,N)+TSOU(*,*)
  103 P2F(JIV+JVM1)=P2E(*,KB,K)
      TSOU(*,*)=TSOU(*,*)+AMASK(LOE(*,*),P2F(JIV)*DCONBE(JIV,KB,N),
     &                          P2F(JIV+J2VM)*DCONBE(JIV+JVM1,KB,N))
```

Fig.2.8 Vectorization of subroutine KTRI in CITATION code

(Original version)

```
      DO 135 I=1, IVX
      DO 112 J=2,JVX
      N1 = N1 + 1
      N5 = N5 + 1
      IF (P2E(N1,KB,K).EQ.0.0) GO TO 111
      L = NRGNE(J,I,KB)
      T = D4*DEL(J-1)
      D4 = DCONRE(N5 ,KB,N)
      BET(J) = ( TSOUR(J) + BET(J-1)*T)/D4
      DEL(J) = D4/(PTSAE(N1 ,KB,N) + E1(L,K) - T)
      IF (INRB.NE.3) GO TO 112
      IF (I.NE.IMAX) GO TO 112
      IF (((J/2)*2).NE.J) GO TO 112
      IF (J.EQ.JVX) GO TO 112
      NN12 = (J/2)*JVX
      MM22 = IVX*JVX+J-1
      BET(J) = BET(J)+P2E(NN12,KB,K)*DCONBE(MM22,KB,N)/D4
      GO TO 112
  111 DEL(J) = 0.0
  112 CONTINUE
  135 CONTINUE
```

(Vectorized version)

```
      DO 112 J=2,JVX
      IDJS=J
      IDJE=JIVX-JVX+J
      IDPS=J+1
      IDPE=JIVX+IVX-JVX+J
      WKP(*)=WKM(*)*DEW(J-1,*)
      WKM(*)=DCONRE(IDP,KB,N)
CC
      LPO(*)=P2E(IDJ,KB,K).EQ.0.0
      LNE=IONC(LPO(*))
      LL(LN)=IDXL(LPO(*))
      WKT(*)=WKM(*)
      WKT(LL(LN))=1.0
      BEW(J,*)=AMASK(LPO(*), 0.0 ,
     &                (BEW(J-1,*)*WKP(*)+TSOU(J,*))/WKT(*) )
      NWK(*)=MASK(NRGNE(J,*,KB).GT.31, 31 ,NRGNE(J,*,KB))
      WKT(*)=PTSAE(IDJ,KB,N)+E1W(NWK(*))-WKP(*)
      WKT(LL(LN))=1.0
      DEW(J,*)=AMASK(LPO(*), 0.0 ,WKM(*)/WKT(*) )
CC
      IF(LPO(IMAX)) GO TO 112
      IF(INRB.NE.3) GO TO 112
      IF(((J/2)*2).NE.J) GO TO 112
      IF(J.EQ.JVX) GO TO 112
      IF(WKM(IMAX).EQ.0.0) WKM(IMAX)=1.0
      BEW(J,IMAX)=P2E((J/2)*JVX,KB,K)*DCONBE(JIVX+J-1,KB,N)/WKM(IMAX)
     &            +BEW(J,IMAX)
  112 CONTINUE
```

Fig.2.9 Vectorization of subroutine KTRI in CITATION code

(Original version)

```
      DO 135 I=1,IVX
  122 DO 129JJ=2,JVX
      J=JVXP1-JJ
      N1= NN1 + J
      T=P2E(N1 ,KB,K)
      TEMP=DEL(J)*(TEMP+BET(J))
      TMF=T+BETTA*(TEMP-T)
      IF (IEP) 123,127,124
  123 P2E(N1 ,KB,K)=TEMP
      GO TO 129
  124 IF (TMF-TEMP) 126,127,125
  125 TMF=AMIN1(TMF,(TEMP+T))
      GO TO 127
  126 TMF=AMAX1(TMF,0.5*TEMP)
  127 CONTINUE
  128 P2E(N1 ,KB,K)=TMF
  129 CONTINUE
  135 CONTINUE
```

a

(Vectorized version)

```
  122 DO 129 JJ=2,JVX
      IDJS=JVXP1-JJ
      IDJE=JIVX-JJ+1
      WKT(*)=P2E(IDJ,KB,K)
      WKP(*)=(WKP(*)+BEW(JVXP1-JJ,*))*DEW(JVXP1-JJ,*)
      WKM(*)=(WKP(*)-WKT(*))*BETTA+WKT(*)
      IF(IEP) 123,127,124
  123 P2E(IDJ,KB,K)=WKP(*)
      GO TO 129
  124 WKI(*)=AMASK(WKM(*).LT.(WKP(*)+WKT(*)),WKM(*),WKP(*)+WKT(*))
      WKX(*)=AMASK(WKM(*).GT.(0.5*WKP(*)),WKM(*),0.5*WKP(*))
      WKT(*)=AMASK(WKM(*).GT.WKP(*), WKI(*) , WKM(*) )
      WKM(*)=AMASK(WKM(*).LT.WKP(*), WKX(*) , WKT(*) )
  127 P2E(IDJ,KB,K)=WKM(*)
  129 CONTINUE
```

b

Fig.2.10 Vectorization of subroutine KTRI in CITATION code

## 2.4 ベクトル処理効果

実際のベクトル処理効果を測定するため，F75APUによって計算時間を測定した。測定に使用した入力データは，FORTUNEツールによる解析時に使用した入力データと同じ，3次元6角形状，6エネルギー群を3角形メッシュ46×23（column，row）×19 plane で解く問題である。

(1) オリジナルプログラムの計算時間

オリジナルプログラムの計算時間をF75CPUとAPUでそれぞれ測定した。F75APUによるオリジナルプログラムの実行について，サブルーチンKNSD，KINS，KTRIの3つのルーチンのみAP−FORTRANコンパイラで翻訳し，他のルーチンはFORTRAN−Hコンパイラで翻訳し結合して実行した。計算時間の測定結果をTable 2.2に示す。

これを見ると，ソースプログラムは全く同じであるが，APUによる実行では外部反復（outer iteration）回数が1回少いにもかかわらず，全体の計算時間はCPUによる実行の場合よりも長くなっている。これはAPUのスカラ演算速度が，CPUのそれに比べてやや遅いことによる。

Table 2.3は，M 200におけるオリジナルプログラムの計算時間の割合を示している。左側は，FORTUNEツールによって測定したタイムコスト（サブルーチン単位に集計した計算負荷）とその比率であり，右側は，基本サブルーチンCLOCKMを使ってミリ秒単位に測定したものを秒単位で示した計算時間とその比率である。

これを見ると，FORTUNEの結果と実測ではやや差があり，また，これらの値は入力データによって使用されるサブルーチンが変ったり，実行回数が変れば変化しうるものである。従って，FORTUNEの結果はベクトル化の目安として利用するのが適当である。

Table 2.3 Ratio of computing time in CITATION code by M 200

| Subroutine name | FORTUNE tool | | Basic subroutine CLOCKM | |
|---|---|---|---|---|
| | Time cost ($\times 10^6$) | Ratio (%) | Computing time (sec) | Ratio (%) |
| KTRI | 1629 | 65.8 | 281 | 61.1 |
| KNSD | 367 | 14.8 | 75 | 16.3 |
| KINS | 305 | 12.3 | 69 | 15.0 |
| Others | 178 | 7.1 | 35 | 7.6 |
| Total | 2476 | 100.0 | 460 | 100.0 |

Table 2.2 Computing time of original version in CITATION by F75 CPU/APU

| Subroutine name | Source program | Compiler* | Run (F75) | F75 CPRUN Outer iteration : 44 | | F75 APRUN Outer iteration : 43 | |
|---|---|---|---|---|---|---|---|
| | | | | Computing time (sec) | Ratio (%) | Computing time (sec) | Ratio (%) |
| KTRI | Original | FRH | CPU | 583 | 57.0 | | |
| | | APF | APU | | | 693 | 61.4 |
| KNSD | Original | FRH | CPU | 129 | 12.6 | | |
| | | APF | APU | | | 193 | 17.1 |
| KINS | Original | FRH | CPU | 223 | 21.8 | | |
| | | APF | APU | | | 155 | 13.7 |
| Others | Original | FRH | CPU | 88 | 8.6 | 88 | 7.8 |
| | | | | 1023 | 100.0 | 1129 | 100.0 |

* { FRH = FORTRAN-H
APF = AP-FORTRAN

(2) サブルーチンKINSのベクトル処理効果

サブルーチンKINSのベクトル処理効果を測定するために，F75APUにおいて，KINSのみベクトル化したプログラムを使用し，KNSD，KTRIについてはオリジナルのプログラムをAP-FORTRANでコンパイルし実行した。この結果，ベクトル化されたKINSのみの計算時間は，34秒であった。従ってサブルーチンKINSのみについてのベクトル処理効果は，

$$\frac{(\text{CPUによるオリジナルプログラムの計算時間})}{(\text{APUによるベクトル化プログラムの計算時間})} = \frac{223}{34} \fallingdotseq 6.56\text{倍}$$

である。

ベクトル化されたKINSにおける主要計算部分のベクトル長は，437～20102（row数×plane数～column数×row数×plane数）である。KINSのように各種のＩＦ文を多数含むプログラムにおいて，より高いベクトル処理効果を得る鍵は，ＩＦ文の真率（そのＩＦ文の全実行回数に対する真になった回数の割合）を調べることであり，それによって，できるだけベクトル長が長くなり，かつ余分な計算が少くなるベクトル化方法を選択しなければならない。例えば，

このKINSでは，主に配列特殊関数IDXLとリストベクトルによる方法を用いたが，サブルーチンKNSDやKTRIではマスク関数などによる方法で，ＩＦ文を含むループのベクトル化を行った。

(3) サブルーチンKNSDのベクトル処理効果

サブルーチンKNSDのベクトル処理効果を測定するために，KINSの時と同様にKNSDのみベクトル化したプログラムを使用し，KINSとKTRIについてはオリジナルプログラムをAP-FORTRANでコンパイルし実行した。その結果，外部反復回数は31回で収束し，KNSDのみの計算時間は，16秒であった。この計算時間短縮の要因は，KNSDのベクトル処理によるものであるが，反復回数がオリジナルプログラムの場合の44回から31回に減少した原因については，境界漏洩計算における内積計算などにおいて，ベクトル演算とスカラ演算で計算順序が異なり，ベクトル演算の方が誤差が少ないことが影響していると考えられるが，定性的な結論には到っていない。この件については，今後の検討課題とし，ここでは，外部反復回数はベクトル化で変化しないとして効果を測定する。従ってベクトル処理効果は，

$$\frac{91}{16} \fallingdotseq 5.69\text{倍}$$

である。

(4) サブルーチンKTRIのベクトル処理効果

サブルーチンKTRIについて，ベクトル演算用に再構成したプログラムに対して，反復解法の変更に伴う外部反復回数の増加を抑えるように内部反復回数と加速係数の値をM 200で実行

して調べた。その結果，内部反復回数4回，加速係数1.10のとき，外部反復回数49回で収束し，再構成プログラムが正しいことが確認された。オリジナルプログラムのM 200による実行では，外部反復回数50回，内部反復回数3回（加速係数は内部で自動計算）で収束した。これらを比べると，反復解法の変更によるベクトル演算の計算量増加は，

$$\frac{49 \times 4}{50 \times 3} \fallingdotseq 1.3\text{倍}$$

である。なお，M 200における再構成プログラムの実行については後で述べる。

ベクトル演算用に再構成したプログラムをF75CPUで，また，AP-FORTRANによるベクトル化したプログラムをF75APUで実行したところ，外部反復回数200回以内には収束しなかった。この原因について，F75計算機によるものか定かではないが，究明できなかったので計算速度の比についてのみ測定し考察する。反復計算1回当りの計算速度をF75 CPUとAPUで測定すると，オリジナルプログラムに比べて，ベクトル化したプログラムは，ベクトル長23（row方向メッシュ数）で，2.19倍であった。

M 200でベクトル演算用の再構成プログラムが正しく収束しているので，反復解法の変更による計算量増加とベクトル化による速度向上から，サブルーチンKTRIのベクトル処理効率は，

$$\frac{2.19}{1.3} \fallingdotseq 1.68\text{倍}$$

である。

M 200におけるサブルーチンKTRIのベクトル演算用に再構成したプログラムの実行結果をTable 2.4に示す。再構成プログラムでは，ベクトル演算用に反復解法を変更し，繰り返しにおいて古い値を参照する形になっているため反復回数が増加する。CITATIONは，入力データによって，初期加速係数（initial overrelaxation factor）と内部反復回数（inner iteration）を指定できるようになっているので，これらを変化させて調べたのが，Table 2.4である。ただし，これらの値を指定しなかった時は，初期加速係数はコード自身で計算した値を用い，内部反復回数は3になる。Table 2.4を見ると，内部反復回数が4回で，初期加速係数は1.10くらいの時が，計算時間が短く，最も短かったのは1.16の時であった。これらの再構成したプログラムにおいては，最適のパラメータを選んでも，オリジナルプログラムの場合の計算量に比べると増加しているが，ベクトル計算機で実行すれば，反復計算1回当り計算時間は短くなる。従って，反復解法のベクトル化については，計算量の増加と計算速度向上率のバランスが問題となる。

Table 2.4 Outer iterations of restructured program on M 200

| Subroutine KTRI program version | Intial overrelaxation factor | Inner iteration | Outer iteration | CPU time by M 200 | K − effective |
|---|---|---|---|---|---|
| Original | default | 3 | 50 | 7 m 43 s | 1.044020 |
| Restructured | default | 3 | 133 | 19 m 43 s | 1.043998 |
| Restructured | 1.20 | 3 | 85 | 12 m 18 s | 1.043990 |
| Restructured | 1.10 | 3 | 84 | 12 m 10 s | 1.043986 |
| Restructured | 1.20 | 5 | 59 | 12 m 6 s | 1.044007 |
| Restructured | 1.10 | 5 | 58 | 11 m 51 s | 1.043963 |
| Restructured | 1.30 | 4 | 68 | 12 m 39 s | 1.044011 |
| Restructured | 1.20 | 4 | 65 | 11 m 14 s | 1.043983 |
| Restructured | 1.16 | 4 | 46 | 8 m 8 s | 1.043981 |
| Restructured | 1.10 | 4 | 49 | 8 m 40 s | 1.043973 |
| Restructured | 1.04 | 4 | 56 | 9 m 57 s | 1.044007 |

(5) CITATION全体のベクトル処理効果

Table 2.5は，CITATIONコード全体のベクトル処理効果について示している。オリジナルの計算時間を100.0としたとき，ベクトル化したプログラムの計算時間は48.0となり，全体としての速度向上率では，2.08倍となる。これは，サブルーチンKTRIのベクトル化によるところが多く，今後，より長いベクトル長で改良を進めて行く予定である。またその他のサブルーチンについても今回はベクトル化を行わなかったが，同様な手順でベクトル化できると考えられるので，今後は別な種類の問題の場合において，計算時間の割合の大きいサブルーチンについてもベクトル化を進め，CITATIONコード全体のベクトル化整備を行っていく予定である。

Table 2.5 Global performance ratio of CITATION code

| Subroutine name | Vector length | (a) Ratio of computing time for F75CPU (%) | (b) Vector processing efficiency on F75 APU | $(\frac{a}{b})$ |
|---|---|---|---|---|
| KTRI | 23 | 57.0 | 1.68 | 33.9 |
| KNSD | 437～20102 | 12.6 | 5.69 | 2.2 |
| KINS | 437～20102 | 21.8 | 6.56 | 3.3 |
| Others | – | 8.6 | 1.00 | 8.6 |
| Total | | 100.0 | – | 48.0 |

Global performance ratio : $\frac{100.0}{48.0} \fallingdotseq 2.08$

# 3. SAP 5（構造解析コード）

## 3.1 解　　法

有限要素法に基づき，構造解析を行うコードである。SAP5によって解析される問題のタイプは，static analysis，frequencies and mode shapes，dynamic analysisの3つであるが，本問題は第2のタイプであり，振動数$\omega^2$と位置及び角度変位を求めることを目的とする。基本となる式は，一般化された固有値問題となる。扱う式は次の形である。

$$K \cdot \phi = \omega^2 \cdot M \cdot \phi \qquad (3.1)$$

ここで，Kは剛性マトリックスと呼ばれる帯行列であり，Mは質量マトリックスと呼ばれる対角行列である。固有値$\omega^2$と固有ベクトル$\phi$を求める。481個の節点に対し最大6種の変位（$\triangle x$，$\triangle y$，$\triangle z$，$\delta\theta_x$，$\delta\theta_y$，$\delta\theta_z$）を求める。変位の一つ一つに方程式が対応することから本問題ではKとMの次数は1800となっている。Kの帯幅は231である。

この問題は，次数が高く帯幅が大きすぎるために，サブスペース反復法が用いられる。この反復法は，Ritz法をくり返し適用したもので1つのステップから得られた固有ベクトルが次の反復に試験基底ベクトルとして用いられる。この解法においては，q個の線型で独立なベクトルに対し，同時に反復計算することにより解が得られる。ここでqは，pを（3.1）式で求める固有値の数とすると，通常$q = \min(2P, P+8)$で選ばれる。この問題では$P = 10$，$q = 18$である。P個の固有値が収束するまで反復計算される。

$V_0$に（Kの次数×q）次のベクトルを初期値ベクトルと置き，ベクトル$\bar{V}_K$を求める。

$$K \cdot \bar{V}_K = M \cdot V_{K-7} \qquad (3.2)$$

部分空間へのKとMの射影を求める。

$$K_K = \bar{V}_K{}^T K \bar{V}_K \qquad (3.3)$$

$$M_K = \bar{V}_K{}^T \cdot K \cdot \bar{V}_K \qquad (3.4)$$

$q \times q$次行列$K_K$と$M_K$に関して次の固有値問題を解く。

$$K_K \cdot Q_K = M_K \cdot Q_K \cdot \Omega_K^2 \qquad (3.5)$$

ここで，$\Omega_K^2$は小さい順にP個の固有値を置いた対角行列である。

こうして，K回目の反復における固有ベクトルを次式より求める。

$$V_K = \bar{V}_K \cdot Q_K \qquad (3.6)$$

このようにして，大次元行列に対する固有値問題（この場合 1800 × 1800 次）がｑ×ｑ次の固有値問題（この場合 18 × 18 次）を反復計算することに帰着される。この問題における反復回数は 7 回で，後に示すように計算時間の約 92 %がこの反復計算に費される。

### 3.2 ＦＯＲＴＵＮＥによる分析

ソフトウエア・ツールFORTUNEによって，サンプルデータによるSAP 5 の実行時の振舞いを分析した結果をTable 3. 1 に示す。このサンプルデータでは金属容器の振動固有モードを節点数 481 点を用いて解析している。今回ベクトル化された 3 つのサブルーチンのうち，DECOMPは（3. 2）式の計算のためのもので，ＥＩＧＳＯＬとREDBAKは（3. 3）～（3. 6）式の計算のためのものである。Table 3. 1 に示されるように，このデータにおいては 3 つのサブルーチン・DECOMPとＥＩＧＳＯＬとREDBAKに計算コストが集中している。いずれのサブルーチンも計算コストが繰り返し回数 50 ～ 100 のＤＯループに集中しており，さらにDOループ内にＩＦ文を含んでいないのでベクトル化が可能である。

### 3.3 ベクトル化の概要

Table 3. 1 に示された 3 つのサブルーチンが，DECOMPに関しては約 88 %，ＥＩＧＳＯＬに関しては約 97 %，REDBAKに関しては 97%がそれぞれベクトル化された。ベクトル長は 50 ～ 200 である。ベクトル化されたＤＯ ループの曲型的なタイプの変換前と変換後の リストをFig. 3. 1 に示す。

### 3.4 ベクトル化効果

このコードのベクトル化前のＦ75ＣＰＵでの実行時間は

449 （秒）

ベクトル化後のＦ75ＡＰＵでの実行時間は

ＣＰＵ時間＋ＡＰＵ時間＝ 125 ＋ 96 ＝ 221 (秒)

であるから，
ベクトル化による速度向上は

449／ 221 ＝ 2. 03 （倍）

である。
コードのベクトル化率は，Table 3. 1 などから

55. 9 × 0. 97 ＋ 31. 1 × 0. 88 ＋ 5. 1 × 0. 97 ＝ 86. 5 （%）

であるから，ベクトル化した場合のコードのCPU時間は，

$$449 \times (1 - 0.865) = 60.6 \text{（秒）}$$

である。ところが，実際にはCPUT時間は125秒と予想の2倍以上の値となっている。これは，SAP5が膨大な量のI/O処理を必要としているためで，F75APUとF75 CPU間のI/Oのやりとりによるの時間の増大と考えられる。また，ベクトル化された部分のみの速度向上は，

$$449 \times 0.865 / 96 = 4.05 \text{（倍）}$$

である。

### 3.5 プログラム規模とサンプルデータによる実行

ソースプログラムは，実行ステップ14604枚

副プログラム数は，185である。

I/O回数は，15,000回程度，メモリ量は1500 KB，使用したファイルは，ワークファイル13ファイル，データが1ファイルである。

Table 3.1 Dynamic profile analysis of SAP5-code by FORTUNE

| Subroutines | Execution times | Cost (%) |
|---|---|---|
| REDBAK | 7 | 55.9 |
| DECOMP | 2 | 31.1 |
| EIGSOL | 7 | 5.1 |
| Other | | 7.9 |
| | | 100.0 |

```
      D=0.00D0
      DO 220 KK=KL,KU,INC
      K=K-1
      C=A(KK)/A(K)
      D=D+C*A(KK)
  220 A(KK)=C
```

ベクトル化前

⇓

```
      DIMENSION EC(1000)
      DIMENSION LV(1000),NUM(1000)
      LOGICAL LTRUE(1000)/1000*.TRUE./
      INDEX K220/1,KU220/
      DATA EC/1000*0.00D0/
      NUM(*)=IDXL(LTRUE(*))
```

⇓

```
      KPARA=KL-INC
      KU220=(KU-KL)/INC+1
      LV(K220)=N-NUM(K220)
      EC(K220)=A(INC*K220+KPARA)/A(LV(K220))
      D=DIPD(EC(K220),A(INC*K220+KPARA))
      A(INC*K220+KPARA)=EC(K220)
```

ベクトル化後

Fig.3.1 Original program and vectorized one of subroutine DECOMP

# 4. CASCMARL（照射損傷シミュレーションコード）

## 4.1 コードの概要

CASCMARL[6]コードは，放射線照射固体材質中のはじき出しカスケードの計算機によるシミュレーションを行うコードで，MARLOWE，CLUSTER，DAIQUIRIの3つのコードから成るコードシステムである。

固体材質が放射線照射を受けると，一般に1次たたき出し原子(Primary Knock-on Atom)ができ，はじき出し原子のカスケードが起こる。MARLOWコードは，このカスケード過程を扱う。

CLUSTERコードは，カスケード過程で生じた欠陥を処理し，欠陥の分布の図形表示も行う。

DAIQUIRIコードは，カスケード過程に続き，原子の熱振動による欠陥の回復を取り扱う。

## 4.2 コードの動的挙動分析

Table 4.1は，CASCMARLコードの主なサブルーチンのtree構造とFORTUNEツールによる各サブルーチンの実行回数と計算時間の割合を示している。これを見ると，はじき出しカスケードを生成するサブルーチンEVENT以下の6個のルーチンとDAIQUIRIコードのメイン制御ルーチンであるDAIQUI以下の10個のルーチンで，全体の計算時間の97.2%を占めている。従って，ベクトル化のため改良の対象となるルーチンは，前述の合計16個の副プログラム（サブルーチンと関数副プログラム）である。

これらの部分のうち，ベクトル長が長く，簡単な手直しでベクトル化できるDOループ部分は，全計算時間の26.5%である。また，ステートメント数が少ないサブルーチンや関数副プログラム（CORL，DSQなど）は，呼び出し元にインライン展開することによって高速化できるが，このインライン展開可能な部分は，全計算時間の32.3%である。

Table 4.1 Tree structure and FORTUNE output in CASCMARL code

| Subroutine name, (f) is function | Executions | Time cost $\times 10^6$ | Ratio (%) |
|---|---|---|---|
| MAIN | 1 - 3 | | |
| MARLOW | | | |
| EVENT | 12 | 51 | 7.6 |
| SEARCH | 32385 | 308 | 46.1 |
| SKATR | 37083 | 37 | 5.5 |
| POTENZ(f) | 23759 | 35 | 5.3 |
| POTENS(f) | 148332 | 19 | 2.9 |
| INELAS | 37083 | 4 | 0.7 |
| MAIN3 | 3 | | |
| MAIN34 (CLUSTER) | | | |
| VCCLF-COMPR | 4792 | 14 | 2.1 |
| PRINT | 3 | 2 | 0.4 |
| INTCLF-COMPR | | | |
| PRINTI | 3 | 1 | |
| DAIQUI | 3 | 7 | 1.1 |
| OCTIN | 3 | 2 | 0.3 |
| FRENKL(f) | 814626 | 5 | 0.7 |
| DSQ | | | |
| UPDATE | 5982 | 11 | 1.6 |
| CONECT(f) | 1336638 | 30 | 4.5 |
| CLOSE | 1021 | 17 | 2.6 |
| CORL(f) | 1119728 | 47 | 7.0 |
| DSQ(f) | 1934656 | 72 | 10.7 |
| JPROB-DSQ | | | |
| JUMP | 3583 | 4 | 0.6 |
| FRENKL-DSQ | | | |
| UPDATE-CONECT | | | |
| CLOSE | | | |
| MAIN4 | 3 | | |
| SITE-SHIFTS | | | |
| SATURN-SHIFTD-SHIFTS | | | |

## 4.3 ベクトル化の方法

### 4.3.1 テーブルサーチDOループのベクトル化

CASCMARLコードの場合，計算時間の割合の大きいDOループは，算術演算よりもテーブルサーチのような配列要素の値の比較や選択を行っているものが多い。

その例として，サブルーチンSEARCHのDO 325やサブルーチンEVENTのDO 50がある。DO 50の方は，最も速度の速いcascade particleを捜すDOループで，そのベクトル化例をFig. 4.1に示す。

Fig. 4.1の下方に示されたベクトル化されたプログラム中で，IDFMXはAP−FORTRANにある配列特殊関数で，ベクトル要素の最大値のインデックスを求める関数である。作業用配列WKAは，AP−FORTRANではリストベクトル（ここではLA（IDN））の使用が1次元配列のみに制限されているので，2次元配列A（＊，1）を1次元配列WKA（＊）に置き換えるのに使われている。

このようなことから，今後は豊富な配列特殊関数や，2次元以上の配列におけるリストベクトルの使用が可能になることが望まれる。

サブルーチンSEARCHのDO 325は，vacancy listをサーチするループである。DOループの中はIF文のみからなり，vacant siteならばDOループから脱出する。このようなDOループの場合，そのままベクトル化すると余分な計算まで行われてしまうので，真率（IF文の条件が真になる割合）に注意しなければならない。（Fig. 4.2参照）

### 4.3.2 インライン展開による高速化とインライン展開後のベクトル化

実行ステップ数が少く，呼び出される場所が少いサブルーチンや関数副プログラムは，呼び出し元にインライン展開する。呼び出される回数が多い場合は，呼び出しに係るオーバーヘッドの時間が節約でき高速化される。また，呼び出し元にDOループがある場合には，新にベクトル化の対象となる。

CASCMARLコードにおいては，Table 4.1に示した関数副プログラムが，インライン展開された。さらに呼び出し元の4つのサブルーチン（UPDATE, JUMP, OCTIN, DAIQUI）には，DOループがあり，全計算時間の26.3％の部分が新にベクトル化の対象となった。（Table 4.2参照）

(Original version)

Select the cascade particle with the highest velocity

```
      INTEGER  KARMA(4000),LA(4000)
      REAL*8   A(4, 4), EK(4000)
      REAL*8   EPS, EP, Q

      EPS=0D0
      EP=0D0
      DO  50  N=1, NCASC
      IF (KARMA(N). LT. 100) GO TO 50
      Q=EK(N)*A(LA(N), 1)
      IF (Q. LE. EPS) GO TO 50
      EPS=Q
      EP=EK(N)
      NP=N
50    CONTINUE
```

(Vectorized version, AP-FORTRAN)

```
      REAL*8 WKA(4)
      INDEX IDN/1, NCASC/

      WKA(*)=A(*, 1)
      NP=IDFMX(EK(IDN)*WKA(LA(IDN)))
      EP=EK(NP)
```

Fig. 4.1 Vectorization of DO 50 in subroutine EVENT

Search vacancy list for target site

```
      DO  325  NV=1, NVAC
      DO  320  I=1, 3
      IF (DABS(RV(NV, I)-RSO(NSIM, I)). GT. DVAC) GO TO 325
320   CONTINUE
      GO TO 340
325   CONTINUE
      ...........
340   SITE(NSIM)=2
      .........
```

Fig. 4.2 Subroutine SEARCH (Original version)

Table 4.2 Spedup by inline expanding in CASCMARL code

| Subroutine, Function | Computing time (msec) Original | Computing time (msec) Inline | (%) Ratio | DO Loop (Vector length) |
|---|---|---|---|---|
| EVENT←INELAS | 1046 | 783 | 74.8 | |
| UPDATE←CONECT | 48511 | 39623 | 81.7 | DO 50 NE (339) |
| 〃 | 85 | 62 | 72.9 | |
| JPROB←DSQ | 14 | 12 | 85.7 | |
| JUMP←FRENKL←DSQ | 19702 | 15341 | 77.9 | DO 20 NE (276) |
| OCTIN←FRENKL←DSQ | 12733 | 9947 | 78.1 | DO 70 NE (360) |
| DAIQUI←CORL←DSQ | 46585 | 35866 | 77.0 | DO 30 NE (293) |
| | 128591 | 101634 | 79.0 | |

| Subroutine name | DO Loop | Vector Length | Time cost |
|---|---|---|---|
| UPDATE | 50 | NE (339) | 40523533 |
| JUMP | 20 | NE (276) | 27188760 |
| OCTIN | 70 | NE (360) | 15931749 |
| DAIQUI | 30 | NE (293) | 92009628 |
| | | Total | 175653670 (26.3%) |

## 4.4 ベクトル処理効果

4.3.2節で述べたインライン展開のみを行った場合，FACOM M200で計算時間を実測したところ，オリジナルコードのCPU時間に比べて，82.7 %に短縮された。さらに，インライン展開後に新にベクトル化の対象となった部分も加えてベクトル化を行った場合，FACOM 230－75 APUに当てはめると，CPU時間は，オリジナルコードに比べて，55.2%になると推定される。(Fig. 4.3参照)

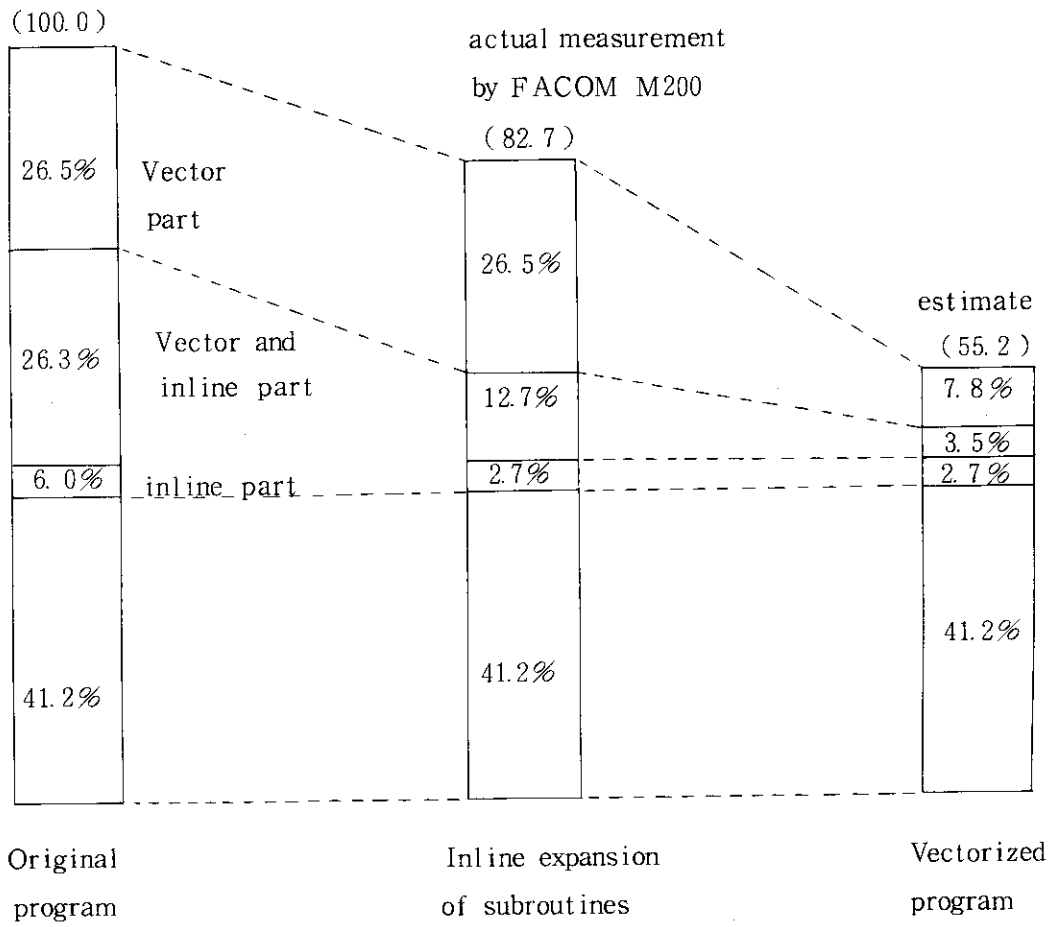



Fig. 4.3 Speedup of CASCMARL code

# 5. FEM-BABEL（有限要素法による3次元中性子拡散コード）

## 5.1 計算内容

(1) 計算内容

原研で開発された有限要素法による3次元拡散コード[7]を高速炉の臨界計算用に改良したもの[8]である。

(2) 計算時間（M 200）（Table 5.1参照）

（問題1） 6角形要素分割 223（x－y）×12（z）×6群 4分12秒

（問題2） 3角形要素分割 369（x－y）×12（z）×6群 11分35秒

(3) 実行ステップ数 メモリ量 約6,600ステップ

（問題1） 1,484 KB

（問題2） 1,772 KB

## 5.2 計算時間のかかるサブルーチンとその計算方法

SOR法－Power methodによる2重反復計算

問題1の場合99％（I/Oに要するCPUを除く）

サブルーチン INNERで約81％，MPYM11で18％

ベクトル化は主に（5.1）式の計算（INNER）と（5.2）式の計算（MPYM11）に対してなされた。（5.1）式のベクトル化については，各サブマトリックス毎の計算に対し，2種の方式が試みられた。

(a) バンド巾の方向にベクトル化

(b) x－y平面上の節点（ここでは総数Nと表示されている）についてベクトル化

(a)はベクトル長20～30と短い，(b)は各節点を並列に計算することによる収束の劣化が起こる。

計算手法の概略は次のとおりである。

内部反復計算は，$S_g$を既知な値として，各gに対して

$$[H]^g \ \phi^{g,(n)} = S^{g,(n-1)} \quad , \quad n\text{は反復数} \tag{5.1}$$

によって計算される。係数行列$[H]^g$は，差分の場合に較べて複雑なものとなっている。

一方，外部反復計算における中性子源の計算は，

$$\sum_{g=1}^{G} (\nu \Sigma_f)_g \ [B]^r \ \phi^g \tag{5.2}$$

により行われる。3次元計算では，$(\nu \Sigma_f)_g$ $[B]^r$はFig. 5.1で示されたものと同形の行列となる。

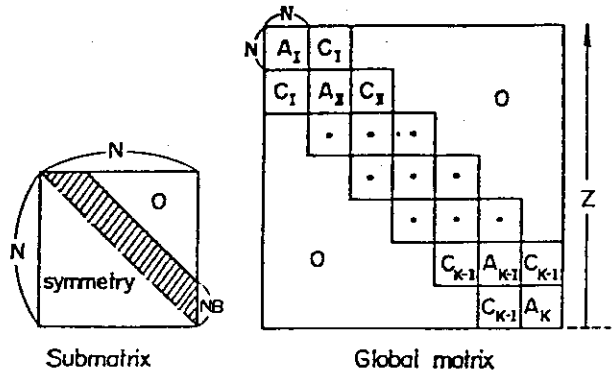
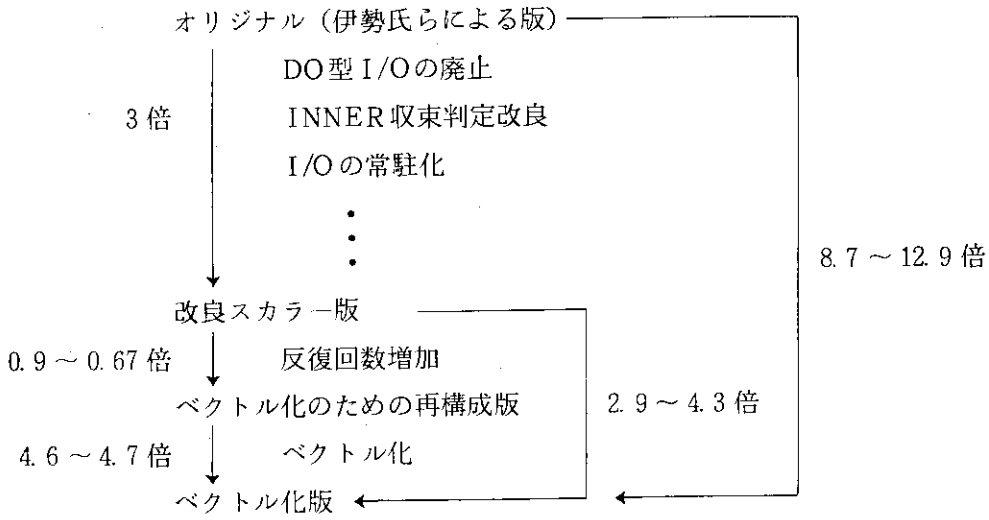



Fig.5.1 Structure of the global matrix, $[H]^g$

## 5.3 ベクトル計算結果（Table 5.1参照）

コード全体の速度向上比




## 5.4 並列計算による反復回数の増加（Table 5.2参照）

中性子束の計算において（サブルーチンINNER），節点方向をベクトル化し，各点を並列に計算すること（方式b）により生じる内側反復回数の増加と計算時間の増加は表5.2で示される。結論としては次のとおりである。

(1) スカラー計算では方式bは1.1～1.5倍の計算時間を要す。

(2) 内側反復の打切回数を方式bの方が30～50％多く取る。

(3) $\omega$の値は方式bの方が小さめに取る。

(4) 初回の外側反復を方式aで実行しても効果がない。

(5) ベクトル化によりINNERルーチンの速度向上は，方式aで約2倍，方式bで8倍。

なお，F 75 APUで計算時間の測定を行う際に，FORTRAN－Hコンパイラの最適化処理に障害があり，浮動小数点オーバフローが頻発し，正常に実行できなかった。そこでM 200の計算時間の3倍をF 75 CPUによる計算時間とみなした。この修正を加えた計算時間に基づく計算時間の比較については，Table 5.3に示される。

## 5.5 ベクトル化による変更点

(1) 変更したソース行数

| | | |
|---|---|---|
| INNER，MPYM11 | 作成直し | 500行 |
| その他の修正 | | 200行 |

(2) 変更した変数，新設した変数，ベクトル長

| | | | |
|---|---|---|---|
| スカラー→ベクトル | 約20変数 | | |
| | ベクトル長 | 200～400 | 方式b |
| | | 20～ 30 | 方式a |

(3) 増加メモリー

| | |
|---|---|
| ベクトル化のため | 約4 KB |
| 内部反復計算用I/Oの常駐化 | 0.7～1.5 MB |

(4) ベクトル化率（Table 5.4参照）

99％（問題1）

APU，CPUの利用分析結果がTable 5.4に示される。計算時間の約10％がI/Oに使用され，残る1～2％がベクトル化不可能な部分である。

(5) 入出力回数

6468～9350

これらのI/Oは外側反復計算時における固有値と中性子源計算（サブルーチンMPYM11内）で起こる。これらを常駐化するためには55～80 MBが必要。

(6) 使用ファイル

| | | |
|---|---|---|
| 作業用 | 10ファイル | BSIZE＝19068，LRECL＝19064 |
| | | 領域合計 80～110 MB |
| 核データファイル | | 1 |
| グラフィックライブラリ | | 1 |

Table 5.1 Comparison of computing times by FEM−BABEL code in the case of gas cooled fast reactor (GCFR) problem

| 比較項目 | | (a) バンド巾の方向にベクトル化 | | (b) x−y平面上の節点に対してベクトル化 | |
|---|---|---|---|---|---|
| | | 6角形要素(*1) | 3角形要素(*2) | 6角形要素 | 3角形要素 |
| M200 | CPU時間 | 255秒(*3) | 693秒(*3) | 389秒(*4) | 758秒(*4) |
| | 主記憶 | 1,484 KB | 1,772 KB | 2,176 KB(*5) | 2,368 KB |
| | I/O回数 | 8,298 | 11,955 | 6,468(*6) | 9,350 |
| F75 H−FORT | CPU時間 | 719秒 | 2,079(推定)(693×3) | 1,167秒(*9) | 2,274秒(*9) |
| | 主記憶 | | | 1,872 KB(*5) | 1,872 KB |
| | I/O回数 | | | 146,975(*10) | 202,809(*10) |
| F75 AP−FORT | APU時間 | 525(内CPU 132(*7)) | | 251(内CPU 129) | 484(内CPU 187) |
| | 主記憶 | | | 1,916 KB(*5) | 1,916 KB |
| | I/O回数 | | | 166,541(*10) | 202,115(*10) |
| 速度比 | APU/CPU | 1.37 | | 4.6(1167/251) | 4.7(2274/484) |
| | APU/CPU(a) | 1.37 | | 2.86 (791/251) | 4.30 (2079/484) |
| 主たるベクトル長 | | 30 | 24 | 223 | 369 |
| 反復計算状況(*8) | 過大緩和因子ω | 1.1 | 1.3 | 1.0 | 1.2 |
| | 内部反復打切数 | 15 | 30 | 25(*4) | 40(*4) |
| | 外部反復回数 | 9 | 10 | 11 | 9 |

＊1　6群3次元6角格子 1/6 形状（60° ratation, symmetry at the midplane）
6角形要素 223（x−y）× 12（z）節点

＊2　同上，3角形要素 369 × 12 節点

＊3　スカラ計算用コードチューニング後の版

＊4　CPU増加の主たる原因は，内部反復回数増による。

＊5　過大な領域設定による主記憶増，ベクトル化による増は，3×（x−y上の点の数）

＊6　内部反復計算用I/Oの常駐化

＊7　主にI/Oのためのタスクスイッチに使用される。これはAPU特有のオーバヘッド

＊8　最短時間を得るパラメータ設定

＊9　F75 FORTH OPT2にエラーが生じ，同等の計算が出来なかったので補正値を利用（Table 5.3）

＊10　APUにブロキング・エラーが生じ，ブロッキングできなかった。

Table 5.2 Comparison of the computing times on overrelaxation factor and computing order in the GCFR problems

| $\omega$ (指定) | element scheme | computing order | inner max (指定) | outer | CPU time on M 200 | $K_{eff}$ |
|---|---|---|---|---|---|---|
| 1.0 | Hexagonal | sequential (*1) | 15 | 9 | 4 M 15 S | 1.03325 |
| 1.0 | 〃 | parallel (*2) | 25 | 9 | 6 M 29 S | 1.03328 |
| 1.0 | 〃 | first iteration sequential second to last parallel | 25 | 9 | 〃 | 〃 |
| 1.0 | 〃 | parallel | 30 | 12 | 8 M 18 S | 1.03336 |
| 1.0 | 〃 | 〃 | 20 | 13 | 7 M 56 S | 1.03348 |
| 1.1 | 〃 | sequential | 15 | 9 | 4 M 15 S | 1.03327 |
| 1.1 | 〃 | parallel | 30 | not | converged for | 10 min |
| 1.1 | 〃 | first iteration sequential second to last parallel | 30 | | 〃 | |
| 1.1 | 〃 | first to second sequential third to last parallel | 30 | | 〃 | |
| 1.1 | 〃 | parallel | 20 | | 〃 | |
| 1.0 | Triangular | parallel | 40 | 9 | 13 M 4 S | 1.03455 |
| 1.1 | 〃 | sequential | 30 | 10 | 11 M 57 S | 1.03471 |
| 1.1 | 〃 | parallel | 40 | 9 | 12 M 38 S | 1.03459 |
| 1.2 | 〃 | sequential | 30 | 10 | 11 M 37 S | 1.03473 |
| 1.2 | 〃 | parallel | 40 | not | converged for | 20 min |
| 1.3 | 〃 | sequential | 30 | 10 | 11 M 35 S | 1.03475 |

＊1 バンド巾の方向をベクトル化する方針でプログラムを構成し，各節点上の中性子束の計算は逐次的に行う。（方式a）

＊2 節点上の中性子束の計算をベクトル化（つまり並列計算）する方針でプログラム構成（方式b）

Table 5.3 Comparison of computing times, where the time on F75 CPU is corrected due to FORTRAN-H optimization error

| CPU \ APU | 節点方向をベクトル化するように再構成したバージョン | | 同左に障害回避のためのオーバヘッドを含めたもの | |
|---|---|---|---|---|
| | 6角形要素 | 3角形要素 | 6角形要素 | 3角形要素 |
| M 200 CPU | 389 (秒) | 758 | 425 | |
| (比) | 0.91 | | 1 | |
| F 75 CPU | 推定 1167 (389 × 3) | 2274 (758 × 3) | 1403 | 3348 |
| F75 CPU/M200 | (3.0) | (3.0) | 3.30 | |
| F 75 APU | 251 (内CPU 129) | 484 (内CPU 187) | | |
| F75 CPU/APU | 4.6 (1167/251) | 4.7 (2274/484) | | |

Table 5.4 Analysis of the use of APU and CPU

| サブルーチン等 | 時間比 FORTUNE | CPU (推定値) | (実績値) |
|---|---|---|---|
| INNER (内側反復計算) | 81 % | 841 (秒) | 100 (APU) |
| MPYM11 (固有値, 中性子源) | 18 % | 186 | 21 (APU) |
| ベクトル化された部分合計 | (99 %) | 1027 | 121 (8.0倍) |
| その他のルーチン | 1 % | 11 | 129 (CPU) |
| I / O (14.6 K回) | $\alpha_1$ | 110 | |
| タスクスイッチ (2500回) | $\alpha_2$ | 8 | |
| | $100+\alpha_1+\alpha_2$ | 1167 | |

時間比 (　　) はI/O時間を含む。

$\alpha_1 = 9.4\%$, $\alpha_2 = 0.7\%$ と推定 (FORTUNEで測定されず)

# 6. GMSCOPE（電子顕微鏡像シミュレーションコード）[9,10,11]

## 6.1 コードの概要

電子顕微鏡における像計算は，試料を通過した電子がレンズによって拡大された像を結ぶ時に行われる。

レンズによる結像の過程をFig. 6.1に示す。物面上の物体の各点から，いろいろな方向へ散乱した波は対物レンズによって後焦点面上に回折像を作るが，ここで絞りによって選ばれた散乱波が，更に干渉し合って像面上で元の物体の拡大像を結ぶ。このように理想的なレンズの場合，後焦点面上に生ずる回折波の振幅は，物体を通過した波の振幅のフーリエ変換で求められ，また像面上での電子波の振幅は，後焦点面上での波の振幅の逆フーリエ変換で求められる。

GMSCOPEの構造は，(1) 結晶モデルを構成する各スライスの構造振幅の計算，(2) 結晶中の電子ビーム散乱の計算，(3) レンズ系による結像計算の3段階にわかれている。理想に近い像を得るための条件を探すために，パラメータをいろいろと変えて，繰り返し計算されるステップが(2)と(3)であり，特に(3)が数多く使用される。今回，ベクトル化の対象とした部分は，(3)のレンズ系による結像計算の部分である。計算内容は，電子波動関数より像面上の強度分布関数を求めるフーリエ変換である。

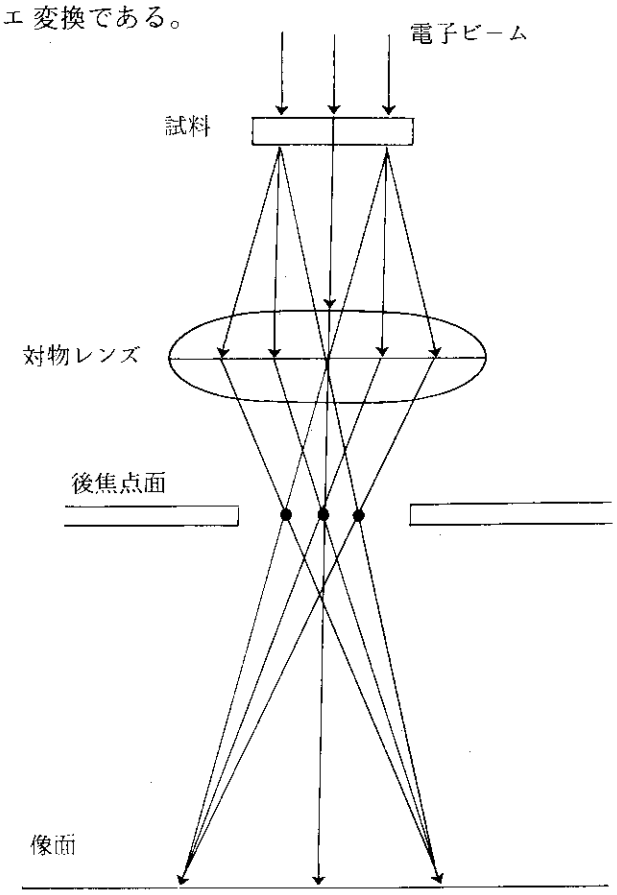



Fig. 6.1 An illustration of the electron microscope

## 6.2 ベクトル化の概要

ソフトウエア・ツールFORTUNEによって，サンプルデータによるGMSCOPE（上述のステップ3）の実行時の振舞いを分析した結果，全体の99%の計算コストが，一つのサブルーチンの4重ループに集中していることがわかった。したがって，このDOループをベクトル化した場合，かなりな速度向上が期待できる。この部分では，像面上の座標（x，y）の各点に対し，電子波動関数により強度分布関数を求めるためのフーリエ変換が行われる。もとのプログラムでは，Fig. 6.2で示すように座標（x，y）に関する2重ループが外側にあり逆格子空間における方向ベクトル（h，k）に関する計算，つまり，（h，k）次のフーリエ変換が内側のループを成していた。そこで（x，y）方向のメッシュ点が101×101と大きいことから，（x，y）方向についてベクトル化することにした。結局，ループ構造をFig. 6.2のように再構成し，その際，x，yに関係のないIF文を（x，y）ループの外側に出すことにした。

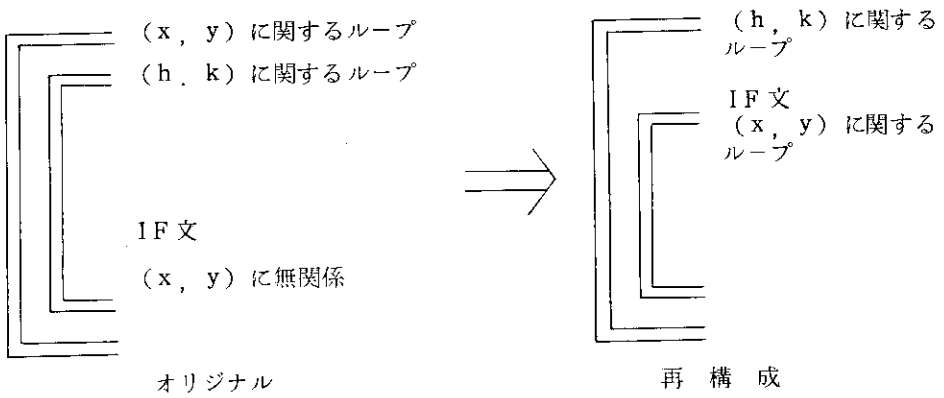



Fig. 6.2 Reconstruction of DO loops

## 6.3 改良した計算方法について

従来は，像面上の波の振幅を計算する際，指数の計算を組み込み関数CEXPを呼んで行なっていたが，このコードにおいては，実数部が常に「0」であることに着目し，$e^z = e^{iy} = \cos y + i \cdot \sin y$と展開した。三角関数については，$2\pi$を20000個に分割したテーブルを用意し，三角関数の値をあらかじめ計算しておき，実際の計算においてはこれを引用するという方法で計算コストを減少させた。テーブル化による誤差は，$10^{-4}$以下で，これは電顕像を得るためには無視してさしつかえない値である。

## 6.4 ベクトル化効果

ベクトル化による，またプログラム変更による速度向上は，以下のとうりである。

Table 6.1において再構成とあるのは，ベクトル化のため，またベクトル化による速度向上のために，Fig. 6.2で述べたようにプログラムを変更したことを指す。これによる速度向上は

約2倍である。((b)/(d)) また，$e^{iy}$ のテーブル化により，同じく約2倍程の速度向上があった。これは，組み込み関数CEXPを約160万回呼んでいたことをやめ，値をテーブルから引き出したことによるコストの差によるものである。結局，コードは，ベクトル化をせずに約6倍の速度向上をみた。最終的な形のコードのベクトル化による速度向上は，(g)/(h)＝7.48 倍ということになる。

## 6.5 プログラム規模とオリジナルのサンプルデータによる実行

ソースプログラムは実行ステップ数238枚，副プログラム数は2である。

サンプルデータ（h×k＝65×65）による実行は，M200でCPU時間4分13秒，メモリ量832 KB，I/O回数200回程度。

通常は，h，kの値，その他のパラメータを変化させ数十回，実行される。CPU時間は，h，kに依存する。

Table 6.1 Computing time of GMSCOPE

| | オリジナル | 再構成 | $e^z$をテーブル化 |
|---|---|---|---|
| M 200 | (a) 4M 13 S | (c) 2M 9 S | (f) 40 S |
| 75 CPU | (b) 10M 47 S | (d) 5M 11 S | (g) 2M 52 S |
| 75 APU | | (e) 56 S | (h) 23 S |

## 6.6 $e^z$ のテーブル化による時間短縮効果

$e^z$ （Zは複素数，ただし実数部は0）の計算を，組み込み関数CEXPを呼んで行なった場合と6.3で述べた改良のように三角関数のテーブルを使用して行なった場合の計算時間をTable 6.2に示す。

4つの方法に対して，8つのケースについてその計算時間を調べた。4つの方法とは，

(1) スカラー計算機で，テーブルを作成せずに組み込み関数CEXPを呼んでそのまま計算した時，

(2) スカラー計算機で，テーブルを作成し計算に利用した時，

(3) ベクトル計算機で，テーブルを作成しないでそのまま計算した時，

(4) ベクトル計算機で，テーブルを作成し計算に利用した時である。

8つのケースとは，TableにおけるM，Nの値をそれぞれ変えて行なったものであるが，Nはベクトル計算機にかけた場合のベクトル長，Mはその外側に安定した計算時間を計るため設けられたくり返し回数である。したがって，スカラー計算機においては，M×N回の指数計算が行なわれたことになる。

また，テーブルを作成するための時間を，Table 6.3に示す。(a)は，2πを200個に分割した時のテーブル作成時間，(b)，(c)はそれぞれ2000分割，20000分割した時のものである。テーブル化による誤差は，(a) $10^{-2}$ 以下，(b) $10^{-3}$ 以下，(c) $10^{-4}$ 以下である。

以上の2つの表より，テーブルを利用し計算時間を短縮することが可能な計算回数xは，例えば計算精度を$10^{-4}$以上必要とする時，次のように求められる。

まず，1回の指数計算に要する時間を求める。これはTable 6.2より，スカラー計算の場合，組み込み関数を呼んだ時が，約0.04（msec），テーブルを利用した時が，約0.021（msec）である。したがって，Table作成時間を考慮すれば，次式が成立しなければならない。

$$0.04 \times x \geqq 0.021 \times x + 666$$

したがって　$x \geqq 35053$

が求まる。

ベクトル計算の場合は，ベクトル長によって計算時間が変わってくるので，それを考慮に入れなければならない。例えば，ベクトル長が10で，$10^{-4}$以上の精度が要求される時は，繰り返し回数Mに対して次式が成り立たなくてはならない。

$$0.9487 \times M \geqq 0.2123 \times M + 166$$

したがって　$M \geqq 225$

が求まる。

Table 6.2　Computing time of $e^z$

| 計算量（N＊M） | | スカラー計算 | | ベクトル計算 | |
|---|---|---|---|---|---|
| N | M | オリジナル | $e^z$ テーブル化 | オリジナル | $e^z$をテーブル化 |
| | | (1) | (2) | (3) | (4) |
| 10 | 10,000 | 4011 | 2130 | 9487 | 2123 |
| 20 | 5,000 | 4013 | 2131 | 5145 | 1238 |
| 100 | 1,000 | 4013 | 2129 | 1726 | 538 |
| 200 | 500 | 4010 | 2131 | 1289 | 453 |
| 1,000 | 100 | 4015 | 2132 | 950 | 382 |
| 2,000 | 50 | 4017 | 2134 | 895 | 375 |
| 10,000 | 10 | 4036 | 2149 | 895 | 357 |
| 20,000 | 5 | 4057 | 2167 | 989 | 345 |

Table 6.3　Time for generating table

| テーブルの大きさ | 計算精度 | CPUでの作成時間 | APUでの作成時間 |
|---|---|---|---|
| (a)　200 | $10^{-2}$以下 | 6 ms | 1 ms |
| (b)　2000 | $10^{-3}$以下 | 66 ms | 7 ms |
| (c)　20000 | $10^{-4}$以下 | 666 ms | 166 ms |

# 7. DWBA（原子分子衝突断面積計算コード）

## 7.1 計算内容

原子，分子衝突過程（イオン＋水素分子）の断面積の計算を行うコード[12]であり，イオン化過程，励起過程，及び荷電交換過程を取扱う。

## 7.2 計算時間のかかるサブルーチン計算方法

ここでの調査は，56年度における富士通の松浦氏によるものである[1]。

物理モデルの近似は，通常のDWBA（Distorted－Wave－Bom－Approximation，歪曲波ボルン近似）ではなく，ユニタリ性を回復させるための工夫を取り入れたUDWA（Unitarized－Distorted－Wave－Approximation）を用いている。

コードの処理の中心は，遷移振巾を数値計算することであり，始状態，後状態の波動関数の重なりを数値積分する。

イオン化過程では，良い精度の水素の束縛状態波動関数が必要であり，毎回計算する手間を省くために，各固有状態の展開係数をディスク・ファイル上にテーブルとして持つことで，計算時間の短縮を計っている。

## 7.3 プログラムの動的挙動分析、及びベクトル化率の推定

プログラムは，入力オプションの指定により，各種の衝突過程，及び近似方法が選択できるようになっている。ここでは，イオン化過程，及び荷電交換過程について分析する。

i）イオン化過程：

この過程に対する遷移振巾は，クーロン波動関数と水素の束縛状態波動関数の重なり積分（多重積分）を数値的に実行して求める。

サブルーチン・コストの分布は，Table 7.1に示されている。重なり積分を求めるサブルーチン，OIF 1のみで76％のコストを占めるが，サブルーチン内の相対ベクトル化率は，約93％と高く，この部分だけで，71％のベクトル化率を持つ。

残りのコストは，主にラゲール，ルジャンドル等の特殊関数の計算に費やされるが，これらは漸化公式を用いているため，全くベクトル化不可能である。

ii）荷重交換過程：

この過程の遷移振幅を与える重なり積分には，初状態と終状態の波動関数を表わす座標原点が一致するものと，異なるものの二種類がある。その計算コストは，それぞれ，2.8％及び11.2％と，余り大きくない。座標原点が異なる波動関数同志の多重積分を行うためには，入射粒子，標的粒子間の座標変換が必要となるが，このための楕円体座標への変換，ルジャ

ンドル関数，球ベッセル関数の値の計算等のコストが，むしろ高い（合計72％）。ルジャンドル関数，球ベッセル関数，ラゲール関数など特殊関数関連で，58％になる。これらは，いずれも漸化公式を用いており，ベクトル化の対象とならない。サブルーチン・コストの高い上位10ルーチンのベクトル化率の合計でも，約23％に過ぎない。

入出力回数を減少させるなどの変更を行った後のプログラムのコストをFORTUNE で測定した結果をTable 7.1 に示す。

## 7.4 イオン化過程のベクトル化

ベクトル化が有効と推定されるイオン化過程のみを対象とした。ここでのF 75 APU によるベクトル化作業は，カナザワコンピュータサービスの室伏氏によりなされた。

ベクトル化を行うために，Table 7.2 に示したインデックスを定義した。

| | |
|---|---|
| ベクトル化率 | 78.7 ％ |
| メモリ量（F 75 APU） | 750 KW（ 3000 KB） |

F 75 APU によるテスト時間（経過時間）を短縮させるため，データのメモリ常駐化を行った。メモリ量には，この分の約 1000 KB が含まれている。

| | |
|---|---|
| 入出力回数（F 75 APU） | 約 6500 回 |

## 7.5 F 75 APU による計算時間

F 75 CPU と APU による計算時間の比較をTable 7.3 に示す。

APU とCPU の間のタスク交信回数を少なくするため，ベクトル化できない部分も APU で実行した。そのため，APU 実行時にスカラ計算部分の実行時間が多くなっている。そこで，スカラ計算がCPU 実行時と同速度と仮定した場合の補正倍率を以下に示す。（Table 7.3参照）

$$\text{補正倍率} = \frac{\text{CPU ③}}{\text{CPU ⑤} + \text{APU ④}} = \frac{3175.930}{1178.127} = 2.70$$

Table 7.1 Output of FORTUNE Tool

| サブルーチン名 | EXECUTIONS | COST(%) | 変更前 COST (%) |
|---|---|---|---|
| OiF 1 | 129177 | 75.9 | 75.9 |
| CT34 | 6975 | 3.9 | 3.4 |
| QEX | 279 | 3.3 | 3.3 |
| LCFC | 8649 | 2.9 | 0.7 |
| XPLM | 7782 | 2.7 | 2.6 |
| RLFP | 129177 | 2.6 | 5.1 |
| LAGD | 279 | 2.2 | 2.2 |

Table 7.2 Vectorized Indexes

| インデックス | 初期値 | 終値 | 増分値 | ベクトル長 |
|---|---|---|---|---|
| iDX001 | 1 | NKi | 1 | 18～52 |
| iDX002 | 1 | NKKi | 1 | 18～86 |
| iDX003 | 1 | iMAX | 1 | 251 ＊ |
| iDX004 | 1 | iMAX | 1 | 31 |
| iDX005 | 1 | NMAX | 1 | 9～43 |
| iDX006 | NFRST | NMAX | 1 | 〃 |
| iDX007 | 1 | LMAX | 1 | 〃 |
| iDX008 | 1 | LMAX | 2 | 〃 |
| iDX009 | 1 | LMAX | 2 | 〃 |
| iDX010 | 1 | NEND | 1 | 〃 |

＊ 計算の大部分はベクトル長251

Table 7.3 Computing Time on F75 CPU, APU

| | | F75－CPU | F75－APU | CPU/APU |
|---|---|---|---|---|
| Core size (Kw) | | 722 | 750 | 0.96 |
| Task switch (回数) | | － | 20977 | |
| CPU time (sec) | ① | 3175.930 | 31.964 | |
| APU time (sec) | ② | － | 1417.008 | |
| ①÷② | ③ | 3175.930 | 1448.972 | 2.19 |
| OiF1以下のルーチン | ④ | | ベクトル計算 703.523 | 3.84 |
| ③－④ | ⑤ | | スカラー計算 745.449 | 0.67 |

# 8. セル内粒子法による濃度計算の新しい方法

## 8.1 ベクトル化の概要

このプログラムは，ADPIC[13]（3次元汚染物質移流拡散計算コード）からセル濃度計算部分を抜き出したもので，計算モデルはセル内粒子法に基づいている。セル濃度計算部分は，2種類（6個）のIF文があり，ベクトル化しても効果があまり期待できないと見られていた。

しかし，セル濃度計算の計算アルゴリズムを改良することによって，2種類のIF文は除去できることがわかった。ここでは，セル濃度計算部分のみに注目し，計算アルゴリズムの改良とその効果について示す。8.2でセル濃度計算の概要を示し，8.3ではセル濃度計算（3次元）を1次元の問題に単純化して，IF文を除去する改良アルゴリズムとその効果を示す。8.4で実際の3次元計算での改良アルゴリズムの効果を示す。

## 8.2 セル濃度計算の概要

大気汚染などにおける汚染濃度の計算は，一般に汚染源から放出された汚染粒子を追跡して行われる。このとき，大気中を3次元メッシュに切り，その1メッシュ空間をセルと呼び，各セルの汚染濃度を計算する。セル内粒子法による濃度計算には，Fig. 8.1のa)，b)の2つの方法がある。

a） 粒子の存在するセルに粒子濃度を加算する方法。

b） 粒子の存在する地点を中心とするセルを想定し，そのセルが元の3次元メッシュで作られたセルと重なり合う部分の体積を計算し，その部分体積に比例してその粒子濃度を隣合ったセルに配分する方法。

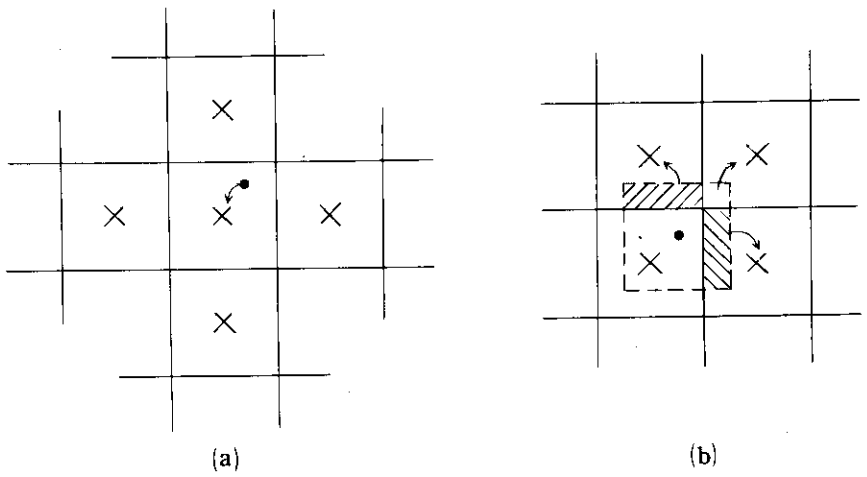



Fig. 8.1 Methods of Assigning the Charge of a Particle on a Difference Mesh for the Solution of Poisson's Equation.[14]

一般に b) の計算方法を使うと，濃度計算結果が平滑化されるという利点があり，ADPICでも b) のセル濃度計算方法が使われている。

## 8.3 セル濃度計算のベクトル化

〔問題の単純化（1次元化）〕

時刻 t における汚染源から放出された粒子（粒子数：Nt，座標：（Xt，Yt，Zt），t＝1～Nt）を3次元メッシュで区切った大気中の各セルに濃度配分する。

ここでは，問題を単純化するために1次元のセル濃度計算（Fig. 8.2）を考える。

放出された粒子数をN，各粒子の現座標を$X'_n$（n＝1～N）とする。セルのX軸方向の幅を$S_x$とすると，$X_n = X'_n / S_n$ が新しく定義される。$X_n$ はセル幅の倍数に基準化された各粒子の現座標である。1粒子で1濃度単位としたとき，各セルの濃度Q(i)，i＝1～M を求める。

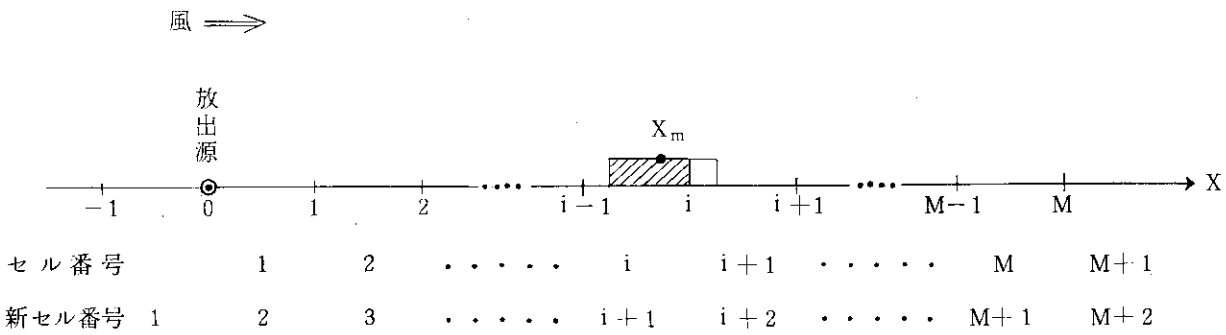



Fig. 8.2 A One-dimensional Calculation Model for Cell Density.

〔元の計算方法と改良した計算方法〕

Table 8.1 に元のセル濃度計算方法と改良したセル濃度計算方法を示す。

(1) 元の計算方法 （ORIGINAL）

① 各粒子がどのセルに属しているか (I) を求める。

②,③ その粒子を中心とした仮想セルは，自分の属しているセルの右側のセルに一部を属しているか左側のセルに一部を属しているかを調べる。（IF文必要）

④,⑤ 自分の属しているセルと重なり合う割合（GSX），隣りのセル（MI）に重なり合う割合（FSX）を計算する。

⑥ 隣りのセル（MI）がX＜0の所にあるならば，セル番号を1にして重なり合う部分を0にする。（IF文必要）

⑦,⑧ 自分の属するセルと隣りのセルの濃度に，重なり合う割合の濃度を加える。

(2) 配列の改良　(DIMENSION IMPROVED)

セル番号をひとつ左にずらすことにより，⑥のIF文は不要となる。

(3) アルゴリズムの改良　(ALGORITHM IMPROVED)

仮想セルは，自分の属しているセルとその右側か左側のどちらかのセルと重なり合っているという考え方をやめて，仮想セルが重なり合っている右側のセルと左側のセルを算出することを考えると，③のIF文は不要となる。

(4) 改良した計算方法　(ALGORITHM, DIMENSION IMPROVED)

上記の2つの改良を行ったものである。

(i)~(iii) 仮想セルの右端が属するセル(IXP)と左端の属するセル(IXM)番号を求める。

(iv)~(vii) それぞれのセルと重なり合う部分を計算し，それぞれのセル濃度を加える。

Table 8.1 Original Calculation Method and Improved Calculation Method.

| 〔元の計算方法〕 | 〔改良した計算方法〕 |
|---|---|
| (入力)　X(n), n=1~N | (入力) X(n), n=1~N |
| (計算)　各X(n)に対して①~⑧を行う | (計算) 各X(n)に対して(1)~(7)を行う |
| ①　I=〔X〕+1 | (1)　XP=X+0.5+2 |
| ②　FX=X−(I−1) | (2)　IXP=〔XP〕+1 |
| ③　FX { ≦0.5 → MI=I−1 | (3)　IXM=IXP−1 |
| ＞0.5 → MI=I+1 | (4)　AP=XP−IXP |
| ④　FSX=\|0.5−FX\| | (5)　AM=1.0−AP |
| ⑤　GSX= 1.0−FSX | (6)　Q(IXP)=Q(IXP)+AP |
| ⑥　MI { ≦0→MI=1, FSX=0 | (7)　Q(IXM)=Q(IXM)+AM |
| ＞0→そのまま | |
| ⑦　Q(I)=Q(I)+GSX | |
| ⑧　Q(MI)=Q(MI)+FSX | |
| (出力)　Q(i), i=1~M | (出力)　Q(i), i=2~M+1 |

〔計算実行時間の比較〕

Table 8.2に簡単化した1次元の粒子濃度配分問題における元の計算方法と改良した計算方法の計算時間を示す。スカラー計算時間はF230−75のCPUでのCPU使用時間，ベクトル計算時間はF230−75のAPUでのAPU使用時間である。

これによると，粒子数N=10,000で見ると，

(イ) スカラー計算機においても，元の計算方法(a)より改良した計算方法(d)が1.5倍速い。

(ロ) 元の計算方法(a)でIF文2つを次のようにMASK関数を使ってベクトル化した場合(a')でもベクトル化によって3.3倍の性能は出る。

Table 8.2 Calculation Times for Cell Density (One－Dimension).

(単位：ms)

| 計算量＝N＊M | | スカラー計算時間 | | | | ベクトル計算時間 | | | | 計算時間短縮 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 粒子数 (ベクトル長) N | 繰返し回数 M | ORIGINAL (a) | DIMENSION IMPROVED (b) | ALGORITHM IMPROVED (c) | ALGORITHM DIMENSION IMPROVED (d) | ORIGINAL (a)' | DIMENSION IMPROVED (b)' | ALGORITHM IMPROVED (c)' | ALGORITHM DIMENSION IMPROVED (d)' | (a)/(d) | (a)/(d)' | (a)'/(d)' | (a)/(a)' | (d)/(d)' |
| 10 | 10,000 | 900 | 783 | 696 | 614 | 1,506 | 1,122 | 1,081 | 679 | 1.5 | 1.3 | 2.2 | 0.6 | 0.9 |
| 20 | 5,000 | 889 | 775 | 688 | 607 | 910 | 673 | 677 | 443 | 1.5 | 2.0 | 2.1 | 1.0 | 1.4 |
| 50 | 2,000 | 885 | 768 | 681 | 598 | 520 | 381 | 409 | 259 | 1.5 | 3.4 | 2.0 | 1.7 | 2.3 |
| 100 | 1,000 | 884 | 767 | 679 | 597 | 403 | 287 | 319 | 202 | 1.5 | 4.4 | 2.0 | 2.2 | 3.0 |
| 200 | 500 | 884 | 767 | 679 | 597 | 339 | 240 | 275 | 172 | 1.5 | 5.1 | 2.0 | 2.6 | 3.5 |
| 500 | 200 | 886 | 768 | 680 | 598 | 302 | 213 | 247 | 157 | 1.5 | 5.6 | 1.9 | 2.9 | 3.8 |
| 1,000 | 100 | 889 | 770 | 683 | 599 | 291 | 203 | 238 | 151 | 1.5 | 5.9 | 1.9 | 3.1 | 4.0 |
| 2,000 | 50 | 897 | 771 | 684 | 602 | 283 | 198 | 233 | 148 | 1.5 | 6.1 | 1.9 | 3.2 | 4.1 |
| 5,000 | 20 | 904 | 786 | 699 | 620 | 273 | 188 | 225 | 144 | 1.5 | 6.3 | 1.9 | 3.3 | 4.3 |
| 10,000 | 10 | 902 | 785 | 700 | 620 | 277 | 185 | 225 | 137 | 1.5 | 6.6 | 2.0 | 3.3 | 4.5 |

〔表1の⑥の場合〕

```
                                YES   NO
                                 ↓    ↓
MI(*)=MASK(MI(*).GT.0, MI(*), 1)
FSX(*)=AMASK(MI(*).GT.0, FSX(*), 0.0)
```

(ハ) ベクトル計算において，改良した計算方法(d') は元の計算方法(a') より2倍速い。

(ニ) スカラーの元の計算方法(a)とベクトルの改良した計算方法(d')では6.6倍速くなっている。

## 8.4 実際の3次元のセル濃度計算における改良した計算方法の効果と問題点

8.3では問題をわかりやすいように単純化したが，実際の計算コードADPICへの適用においては，3次元問題なので以下の対応が必要である。

(i) 粒子の地表面沈着（Z軸方向の境界条件）を考えねばならない。→ これもセル番号をひとつ下にずらして，最後に地表面（2次元データ）の含まれているセルより1つ下のセルの濃度を地表面のセルに加えることで解決される。

(ii) F 230－75　AP－FORTRANでは，3次元のリストベクトルは処理されない。→ 1度，1次元のダミー・リストベクトルを作らねばならない。（計算時間がかかる）。

(iii) 主記憶容量が多く必要である。→ スカラーの元の計算方法に比べて約7倍必要である。

(iv) セル番号を1つずらすと，計算コード全体の見直しが必要である。→ この計算部分の先頭でで調整してもよい（計算時間，主記憶量が増える）。

Table 8.3に地表面が0mのときの3次元濃度計算におけるベクトル効果を示す。

Table 8.3　Calculation Times for Cell Density (Three－Dimension)

（単位：ms）

| 計算量＝N＊M | | スカラー計算 | | ベクトル計算 | | 計算時間短縮 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 粒子数（ベクトル長）N | 繰返し回数 M | 元の計算方法 (a) | 改良した計算方法 (d) | 元の計算方法 (a') | 改良した計算方法 (d') | (a)/(d) | (a)/(d') | (a')/(d') | (a)/(a') | (d)/(d') |
| 10 | 10,000 | 4,149 | 9,967 | 7,935 | 4,392 | 0.4 | 0.9 | 1.8 | 0.5 | 2.3 |
| 100 | 1,000 | 4,296 | 4,464 | 1,888 | 1,005 | 1.0 | 4.3 | 1.9 | 2.3 | 4.4 |
| 1,000 | 100 | 4,496 | 3,944 | 1,250 | 631 | 1.1 | 7.1 | 2.0 | 3.6 | 6.3 |
| 10,000 | 10 | 4,491 | 3,871 | 1,133 | 574 | 1.2 | 7.8 | 2.0 | 4.0 | 6.7 |

スカラー計算において，ベクトル長が100以下の場合，改良した計算方法が元の計算方法より計算時間がかかっているのは，粒子の地表沈着計算をxy平面（22×22＝484セル）について余分に行っているためである。

ベクトル計算において，ベクトル長が10のときにスカラー計算より計算時間がかかっている

のは，F230－75APUのベクトル計算の立上がりが遅いことと，3次元のリストベクトル処理のために，ダミー計算を入れていることに主な原因がある。

Table 8.3より，ベクトル長10,000において，改良した計算方法のベクトル処理（d'）は，元のスカラー計算（a）に比べて約8倍，元のベクトル計算に比べて約2倍の速さとなっている。

現在ADPICの実行ケースでは，粒子数（N）は0から始まり，最終的には2～5万，粒子放出サイクル（M）は50回程度と少ないが，高速大容量な計算機であれば，Nは数十万，Mは1,000以上でも実行できると思われる。

なお，ADPICコード全体の計算時間に占めるセル濃度計算部分の割合は小さく，また，ここでのベクトル化手法をコードに採用するには大巾な書き換えを必要とする。ADPICコード全体のベクトル化については，文献(15)に述べられている。

# 9. おわりに

6本の原子力コードと1つのプログラムに対して，具体的にF 75 APUにおいてベクトル化を行い，そのベクトル処理効率を調べてきた。このベクトル化の段階において，本質的に問題となる点は，次の2点である。第1点はループ内の演算において再帰的データ依存関係がないこと，第2点はできるだけ長いベクトル長でベクトル化することである。

ベクトル化によってコード全体としてのより高い効率を得るために重要なことは，全計算時間に占めるベクトル化可能な部分の割合，すなわちベクトル化率を増すことであり，第1点で述べた再帰的データ依存関係さえなければ，ほとんどすべてのループが何らかの工夫によってベクトル化できる。その具体的方法のいくつかは，前述の原子力コードの何箇所かで述べてきた。

次に，ベクトル化可能な部分において，より高いベクトル処理効率を得るためには，より長いベクトル長であることが望ましい。従って，単に最内ループのみにとどまらず，2重，3重のループやサブルーチンの外側のループ構造にも注目することが必要である。また，その長ベクトル化においても，ベクトル演算用に再構成されたプログラムは，できるかぎり単純な制御構造であることが，オーバヘッドを少くする意味からも大切になってくる。

最後にAP－FORTRANの使用経験から，改善が望まれる点について述べ，今後スーパーコンピュータなどの言語仕様の参考になるものをあげる。

(1) 減少方向のインデックスの使用，または，減少方向と増加方向の両方のインデックスが一つの式の中で使用できる。(Fig. 9.1 Example 1参照)

(2) 2次元以上の配列での間接アドレス（リストベクトル）の使用を可能にする。(Fig. 9.1 Example 2参照)

(3) インデックス変数の整数型変数としての使用を可能にする。(Fig. 9.1 Example 3 参照)

(4) 行列と行列の積を1関数（ベクトル命令）で行えるようにする。(Fig. 9.1 Example 4参照)

(5) ベクトル要素積を行う関数（ベクトル命令）を用意する。

$$\prod_{i=1}^{n} A(i) = A(1) * A(2) * \cdots * A(n)$$

(Fig. 9.1 Example 5参照)

(6) マスク関数で算術IF文に対応する3値（負，0，正）を引数にもてる関数を用意する。(Fig. 9.1 Example 6参照)

(7) 2種類のインデックス，例えば配列断面子[4]とインデックス変数が一つの配列の中で使用可能にする。ただし，インデックスの変化の順序は，次元の低い方からとする。(Fig. 9.1 Example 7参照)

```
( Example 1 )
    INDEX INV/N,1,-1/,IND/1,N,1/
    A(IND)=B(INV)*B(IND)

( Example 2 )
    A( I, L(*), K)

( Example 3 )
    INDEX IND/1,N/
    A(IND)=B(IND)+2*IND-1

( Example 4 )
    A(*,*)=MMP(B(*,*),C(*,*))

( Example 5 )
    X=VMLT(A(*))

( Example 6 )
    A(*)=AMSK(S(*), B(*), C(*), D(*))

( Example 7 )
    INDEX IND/1,NEND,2/
    A(*)=B( IND, *, K)+C( IND, *)
```

Fig.9.1 Some examples of desired vector function

# 謝　　辞

原子，分子衝突過程の断面積計算コードDWBAについて，F 75 APUにより具体的にベクトル化を行っていただいたカナザワコンピュータサービス㈱の室伏昭氏に感謝致します。照射損傷シミュレーションコードCASCMARL，電子顕微鏡像シミュレーションコードGMSCOPEのベクトル化に当りコードの提供を受け，ベクトル化への考察に対し御協力いただいた原子炉システム研究室の田次邑吉氏，西田雄彦氏に感謝致します。AP－FORTRANの使用，およびベクトル化について数々の助言をいただきました富士通㈱　科学システム開発部第二原子力システム課の松浦俊彦氏，田中幸夫氏に感謝致します。この仕事の全般に渡り御指導いただきました計算センター室長代理浅井清氏に深謝致します。

また，2年余りの期間に渡るFACOM 230－75 APU/CPUの利用に当り，数々のお世話をいただいた富士通関係の方々に深く感謝致します。

# 参 考 文 献

(1) 石黒美佐子，松浦俊彦，奥田基，原田裕夫，太田文雄，梅谷真；ベクトル計算処理の大型原子力コードへの適応性，JAERI－M 82－018（1982）

(2) 三輪修，久米宣明，内田啓一郎，鈴木滋，棚倉由行，磯辺文雄；FACOM 230－75アレイプロセッサシステム，FUJITSU，Vol. 29，No. 1，p. 93，（1978）

(3) Fowler，T. B.，Vondy，D. R.，Cunningham，G. W.；Nuclear Reactor Core Analysis Code：CITATION，ORNL－TM－2496，（1969，Revision 2：1971）

(4) FACOM 230－75 M－Ⅶ AP－FORTRAN 文法書，75 SP 03701，（1977），及び FACOM 230－75 M－Ⅶ AP－FORTRAN 使用手引書，75 SP 05001，（1980）富士通㈱

(5) Bathe，K. J.，et al.，A Structural Analysis Program for Static and Dynamic Response of Linear Systems，EERC 73－11（1973）

(6) 朝岡卓見，田次邑吉，筒井恒夫，中川正幸，西田雄彦，中原康明；放射線照射固体材質中のはじき出しカスケードの計算機シミュレーション・コード，JAERI－M 8178（1979）

(7) Ise，T.，et al.；A Computer Program for Solving Three－Dimensional Neutron Diffusion Equation by Finite Element Method，JAERI 1256（1978）

(8) 石黒美佐子，樋口健二；有限要素法における6角形要素の高速炉3次元拡散問題への応用，JAERI－M 82－071（1982）

(9) Nishida，T.；Electron Optical Conditions for the Formation of Structure Images of Silicon Oriented in (110)，Jpn. J. Appl. Phys.，19，No,5 (1980) 799

(10) Izui，K.，Furuno，S.，Nishida，T.，Otsu，H.，High Resolusion Structure Images and their Application to Defect Studies，Chemica，Scripta 14（1978－1979）99

(11) Nishida，T.，Izui，K.；Identification of Small Point Defect Clusters by

High Resolution Electron Microscopy, Proc. 5th Yamada Inter. Conf. (1981) Kyoto

(12) Ryufuku, H., Watanabe, T.; Charge Transfer in Collisions of Atomic Hydrogen with $O^{8+}$, $He^{2+}$, and $H^{+}$, Phys. Rev. A, 18 (1978) 2005

(13) Lange, R.: ADPIC — A Three-Dimensional Particle in Cell Model for the Dispersal of Atomospheric Pollutants and its Comparison to Regional Tracer Studies, Journal of Applied Meteorology, Vol. 17, No. 3 (March 1978)

(14) Hockney, R. W.: Measurements of Collision and Heating Times in a Two Dimensional Thermal Computer Plasma, J. Comp. Phys., Vol. 8, No. 1. (1971)

(15) 浅井清，篠沢尚久，石川裕彦，茅野政道，林隆；放射性物質大気中移行・拡散・外部被曝線量計算コードのベクトル計算処理，JAERI-M 82-218 (1982)