JAERI-M
89-030

原子力コードのベクトル化 88-1
—SRAC, CITATION, TWOTRAN-II, COREBN, CITATION-FBR—

1989年3月

野々宮 厳*・原田 裕夫

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 Division, Department 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　日立高速印刷株式会社

原子力コードのベクトル化 88－1
── SRAC, CITATION, TWOTRAN－II COREBN, CITATION－FBR ──

日本原子力研究所東海研究所計算センター
野々宮 厳*・原田 裕夫

（1989年2月10日受理）

本報告は，熱中性子炉体系標準核設計コードシステムＳＲＡＣ，３次元中性子拡散コードＣＩＴＡＴＩＯＮ，２次元Sn輸送コードTWOTRAN－II，多次元炉心燃焼計算コードCOREBN，２・３次元中性子拡散コードＣＩＴＡＴＩＯＮ－ＦＢＲのベクトル化作業について述べている。ＳＲＡＣシステムについては，その中のＣＩＴＡＴＩＯＮ，TWOTRAN，衝突確率法計算の各コードを対象としている。

ベクトル化版のオリジナル版に対する性能向上は，ＳＲＡＣシステムで1.8～8.6倍，ＣＩＴＡＴＩＯＮコードで2.3～10.1倍，TWOTRAN－IIコードで4.2倍，COREBNコードで2.9～10.4倍，ＣＩＴＡＴＩＯＮ－ＦＢＲコードで，3.9～13.3倍である。

本報告では，ベクトル化に際して使用した入力データの概要，ベクトル化の方法及びベクトル化の結果と評価について述べる。

東海研究所：〒319-11 茨城県那珂郡東海村白方字白根２－４
＊ 外来研究員，富士通株式会社

Vectorization of Nuclear Codes 88-1
- SRAC, CITATION, TWOTRAN-II, COREBN, CITATION-FBR -

Iwao NONOMIYA* and Hiroo HARADA

Computing Center
Tokai Research Establishment
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received February 10, 1989)

In this report, we describe the vectorization of thermal reactor standard neutronics code system SRAC, three dimensional neutron diffusion code CITATION, two dimensional discrete ordinates transport code TWOTRAN-II, multi-dimensional core burn-up calculation code COREBN, two and three dimensional neutron diffusion code CITATION-FBR. CITATION code, TWOTRAN code, collision probability method code PIJ in SRAC system are also vectorized.

The performance ratio of the vectorized version to the original one is from 1.8 to 8.6 for SRAC, from 2.3 to 10.1 for CITATION, 4.2 for TWOTRAN-II, from 2.9 to 10.4 for COREBN, from 3.9 to 13.3 for CITATION-FBR. In this report, we describe sample input data, summary of the codes, vectorization techniques and performance evaluation of the vectorized codes.

Keywords : Vectorization, Nuclear Codes, Reactor Physics, SRAC, CITATION, TWOTRAN-II, COREBN, CITATION-FBR, FACOM VP-100, Supercomputer

---

* On leave from FUJITSU, LTD.

目　　次

Contents

1. はじめに

1.1 原子力コードのベクトル化の経緯

原研計算センタでは，ベクトル計算機FACOM VP−100の導入の前後から，これまで約50本の原子力コードのベクトルを行い，原子力コードの利用効率の改善とベクトル計算機の有効利用を図っている。しかしながら，ベクトル化されたコードは，単体コードが中心で，各研究室の個別のバージョンに対してのベクトル化整備は十分ではなかった。

昭和６２年１１月に大型計算機の利用将来計画を検討するために，原研東海研究所に「大型計算機利用将来計画アドホック会議」が設置された。

その中で，スーパーコンピュータの高度利用のための方策についても検討され，その結果，スーパーコンピュータの有効利用のために組織的，体系的なベクトル化を推進することが必要であること，原子炉工学部，動力炉・安全性研究管理部，核融合炉設計，NUCEF建設室，研究炉管理部，材料試験炉部などの協力を得て，まずはじめに，炉物理関連コードのベクトル化に努力を傾注することが報告された。

この報告を受け計算センタを中心に，これら炉物理関連コード及び他の研究分野のコードについても有効性の高いものについて，ベクトル化を推進することになった。

本レポートでは，この炉物理関連コードのベクトル化について述べる。炉物理関連コードについては，これまで，CITATION, TWOTRAN, DOT, ANISNなどカーネルとなるコードについては，一部ベクトル化が進められてきたが，各研究室で使用されている個別の，あるいは，一部修正，改良されて使用されているバージョンまでは整備されていなかった。また，コードシステムとして炉物理分野で広く使用されているSRACにもベクトル化されたコードの組み込み整備は十分ではなかったので，今回ソースコードを整備すると共に，ベクトル版の計算結果についても十分な評価，検証を行う。

ベクトル化効果の概要については，次節で述べ，以下２章以降に各原子力コードのベクトル化について具体的に述べる。

1.2 ベクトル化効果の概要

今回の作業でベクトル化，高速化されたコードのＣＰＵ時間，倍率，ベクトル化率，メモリ，Ｉ／Ｏ回数等をTable 1.1に示す。Table 1.1において各数値の上段はオリジナル版，下段はベクトル化版を意味する。ここでオリジナル版とはオリジナルプログラムをVP−100でスカラ計算した（すべてのサブルーチンをFORTRAN77でコンパイルしたものを実行した）ものであり，ベクトル化版とは，ベクトル化したプログラムをVP−100でベクトル計算した（すべてまたは一部のサブルーチンをFORTRAN77−VPでコンパイルしたものを実行した）ものである。倍率は，オリジナル版のＣＰＵ時間をベクトル化版のＣＰＵ時間で割ったもので

ある。

ベクトル化率についてはいくつかの定義があり，ベクトル化版のＣＰＵ時間とＶＵ時間だけから求めるものやベクトル化されたライン数から定義しているものがあるが，本レポートでは以下の定義で求めた値をベクトル化率として用いる。

$$\text{ベクトル化率}\ (\%) = \frac{\text{オリジナル版CPU時間} - (\text{ベクトル化版CPU時間} - \text{ベクトル化版VU時間})}{\text{オリジナル版CPU時間}} \times 100$$

また，コメントの欄に記してあるＶＩＯ／Ｆ使用とは，Ｉ／Ｏ回数及び経過時間を減らすために入出力ファイルのいくつかについてベクトル化版でＶＩＯ／Ｆ機能を用いたことを意味している。

Table 1.1 Vectorization effect

上段：オリジナル版スカラ計算
下段：ベクトル化版ベクトル計算

コード名		入力データの概要	CPU時間(秒)	VU時間(秒)	倍率	ベクトル化率(%)	メモリ	I/O回数(KB)	コメント
CITATION		25群 2次元円筒形状 メッシュ51×49	54.02 15.58	0.0 12.00	3.47	93.4	2224 2379	102 81	
		70群 2次元円筒形状 メッシュ46×32	120.87 25.43	0.0 18.14	4.75	94.0	3008 3114	145 146	
		24群 2次元三角形状 メッシュ86×43	62.56 26.86	0.0 16.93	2.33	84.1	2364 2587	149 151	
		3群 3次元x-y-z形状 メッシュ51×56×24	241.76 23.87	0.0 19.23	10.13	98.1	3380 5695	37615 107	VIO/F 3MB使用
		24群 3次元三角形状 メッシュ32×16×15	160.39 35.31	0.0 30.90	4.54	97.3	4250 4787	262 462	
TWOTRAN		70群 円筒形状 メッシュ61×40, P_0, S_8	4453 1057	0 725	4.21	92.5	1972 5126	2412 685	
SRAC	CITATION部分	26群 2次元円筒形状 メッシュ50×35	99.28 47.94	0.0 35.32	2.07	87.3	5956 6064	1244 1669	
		4群 3次元x-y-z形状 メッシュ37×37×35	392.70 45.74	0.0 21.07	8.59	93.7	5908 6216	2291 1833	
	TWOTRAN部分	10群 円筒形状 メッシュ35×35, P_0, S_4	768.73 425.06	0.0 85.29	1.81	55.8	5900 6264	1238 1321	
	PIJ2部分	x−y形状 55群	215.74 159.14	0.0 36.04	1.36	42.9	6040 6092	3319 3317	
		x−y形状 68群	874.79 467.05	0.0 135.29	1.87	62.1	6000 6052	9183 9182	
COREBN		7群 2次元三角形状 メッシュ34×17	49.70 17.38	0.0 5.34	2.86	75.8	1372 1422	2266 2323	
		8群 3次元三角形状 メッシュ34×17×32	510.91 49.23	0.0 25.71	10.38	95.4	3692 4016	16061 1909	VIO/F 15MB使用
		8群 3次元三角形状 メッシュ34×17×32	3330.61 332.15	0.0 195.61	10.03	95.9	3632 4016	103703 10373	VIO/F 15MB使用
CITATION−FBR		70群 2次元円筒形状 メッシュ35×25	71.23 18.50	0.0 13.52	3.85	93.0	1411 1477	1362 167	VIO/F 1MB使用
		25群 3次元x-y-z形状 メッシュ32×32×21	977.31 73.53	0.0 51.50	13.29	97.7	4200 5000	28645 702	VIO/F 9MB使用

2. CITATIONコード

3次元多群中性子拡散計算コードCITATION[1]は，1972年にオークリッジ国立研究所で開発されたコードである。CITATIONコードのベクトル化は，最初，3次元のX−Y−Z形状及びQ−R−Z形状用のルーチンを中心に行われ，その有効性が示された[2]。

今回は，まだベクトル化されていなかった3次元の三角形状用ルーチンを始め，2次元関係のルーチンのベクトル化整備，3次元関係ルーチンのチューニング等を行い，CITATIONコードのオリジナル版に対するベクトル化整備を行い，計算結果についても評価を行う。

また，4章，5章，6章にも述べるように，CITATIONをコードシステムに組み込んだもの，別なコードとの組み合せ，研究室での独自の改良等を含んだバージョンについてもベクトル化整備を行う。

2.1 コードの概要

CITATIONコードは，中性子拡散方程式を有限差分法により空間メッシュとエネルギー群について離散化し，反復法により解く。実効増倍率Keを固有値，中性子束Φを固有ベクトルとした（2.1）式のような固有値問題として解かれる。

$$A\Phi = \frac{1}{Ke} F\Phi, \quad \Phi = \begin{pmatrix} \varphi_1 \\ \varphi_3 \\ \vdots \\ \varphi_G \end{pmatrix} \qquad (2.1)$$

メッシュ数をN，エネルギー群数をGとしたとき，A，FはG・N元の行列となる。Aは中性子の輸送，散乱，吸収に係る係数で，Fは中性子源に係る係数である。

（2.1）式は反復計算法を用いて，

$$\Phi^{(n)} = M^{(n)} \Phi^{(n-1)}, \quad n：外側反復回数 \qquad (2.2)$$

ここで，

$$M^{(n)} = \frac{1}{k_e^{(n)}} A^{-1} F \qquad (2.3)$$

（2.2），（2.3）式によって，ΦやKeを求める過程は，内側・外側の2重反復計算によって行われる（Fig. 2.1）。

実効増倍係数k_eは次式で求められる。

$$k_e = \frac{P}{A + L} \qquad (2.4)$$

ここで，P：中性子発生率，A：吸収率

L：表面からの漏れ

外側反復計算は，ベキ乗法を使って固有値問題として解かれる。内側反復計算は，エネルギー群$\ell = 1, 2, \cdots$，Gの順にN元連立方程式を解く問題となり，オリジナル版ではSLOR法を採

用している。

差分法による反復計算では，あるメッシュ点（x_i，y_j，z_k）における中性子束の値は，同一群のその点に隣接する点を使って計算される。3次元問題では，次の7点階差式となる。

$$\varphi^{(m)}_{i,j,k} = a + b\varphi^{(m)}_{i-1,j,k} + c\varphi^{(m)}_{i,j-1,k} + d\varphi^{(m)}_{i,j,k-1} + e\varphi^{(m-1)}_{i+1,j,k} + f\varphi^{(m-1)}_{i,j+1,k} + g\varphi^{(m-1)}_{i,j,k+1} \qquad (2.5)$$

ここで，a～g：（i，j，k）に依存した定数

m：内側反復回数

CITATIONコードでは，内側反復の打切り回数とSLOR法などで用いられる初期加速係数の値は入力データで与えられるようになっている。

2.2 コードの動的挙動分析

Table 2.1に主なサブルーチンの木構造とその計算内容を示す。

CITATIONコードは，次の形状に対する計算が可能である。

(1) One－dimensional slab (X)
(2) 〃 cylinder (R)
(3) 〃 sphere (S)
(4) Two－dimensional slab (X，Y)
(5) 〃 cylinder (R，Z)
(6) 〃 circle (θ，R)
(7) 〃 hexagonal (H)
(8) 〃 triangular (T)
(9) Three－dimensional slab (X，Y，Z)
(10) 〃 cylinder (θ，R，Z)
(11) 〃 hexagonal (H，Z)
(12) 〃 triangular (T，Z)

各形状に対応してサブルーチンが用意されている。したがって形状毎に動的挙動分析を実施し，その結果に対応して，ベクトル化対象サブルーチンを決める必要がある。

上記形状のうち(1)～(3)の1次元形状については，2次元形状と同じサブルーチンを使用していることから2次元に含めることにした。さらに(7)，(11)の六角形状データについては，使用頻度が少なく，入力データも入手できなかったことからベクトル化対象からはずした。また，(9)(10)の三次元形状用サブルーチンはすでにベクトル化済[2]である。

以上から，今回のベクトル化対象形状は，(4)，(5)，(6)，(8)，(12)とする。以下では，2次元と3次元に分けて解析する。

2.2.1 2次元形状

計算時間分布状況を調べるためにFORTUNEツール[3]で動的挙動分析を行う。そのとき使用した入力データをTable 2.2に示す。

さらに分析結果をそれぞれTable 2.3～Table 2.5に示す。

この結果によると円筒形状のケース1，ケース2ではFXRD，FWRD，DNSDの3ルーチンでそれぞれ97.2%，96.7%を占めており，三角形状のケース3では，DNSD，FTRI，FINSの3ルーチンで96.5%を占めている。

よって，以降はこれらのルーチンを中心にベクトル化作業について述べる。

2.2.2　3次元形状

(1)　三角形状データ

3次元形状のうちX－Y－Z形状及びθ－R－Z形状については，以前に計算時間分布を調べ，ベクトル化している[2]ので，三角形状についてのみFORTUNEツールによる動的挙動分析を行う。動的挙動分析に使用した入力データをTable 2.6に示す。参考のためX－Y－Z形状の入力データも同時に示す。

さらに分析結果をTable 2.7，2.8に示す。この結果によるとケース2ではKTRI，KNSD，KINS，KBPR，KOOPの5ルーチンで96.2%を占めている。よって，以降は，これらのルーチンを中心にベクトル化作業を実施する。参考のため，ケース1の分析結果をTable 2.7に示す。

(2)　black absorberによるロス計算を含むデータ

Table 2.9に3次元三角形状でblack absorberによるロスの計算を含むデータによるCPU時間を示す。ここで，KNSD及びKTRIルーチンはベクトル化済のものである。TABLE 2.9の計算結果は，COREBNコードのものであるが，COREBNコードの計算の中心である中性子束計算はCITATIONを組み込んで用いている。

Table 2.9によると全CPU時間の約7割をKINSルーチンで占めており，KINSルーチンの高速化が必要なことが解る。

2.3　ベクトル化方法

ここでも2次元形状と3次元形状に分けてベクトル化方法を解説する。

2.3.1　2次元形状

2次元円筒，平板，球形状に対応するサブルーチンについては富士通㈱能村博人氏によるベクトル化コードを一部参考にした。

(1)　2重ループの1重化によるベクトル性能向上

CITATIONのオリジナルコードの2次元問題計算部分では，内側反復計算を行うDNSDや中性子源計算を行うLOOPなどのサブルーチンにおいて，3重ループがある。つまりFig. 2.2のオリジナル版に示すように，2次元空間x－y形状の各方向メッシュのインデックスi，jに関する2重ループと，その内側にエネルギー群のインデックスℓに関するループがある。最内ループは内積計算なのでベクトル化は可能であるがエネルギー群に関するループのベクトル長は2次元メッシュのサイズに比べて一般に短い。したがって，このℓに関するループを外側に出し，i，jに関する2重ループを内側に入れ，かつ1重化してベクトル性能を向上させ

た。

この場合，サブルーチン内で2次元空間の各点は独立な計算となるため，2次元配列の宣言部分を1次元配列に宣言を変え，作業用変数も配列化した（Fig. 2.2のベクトル化版参照）。

現在のFORTRAN77／VPコンパイラの自動ベクトル化機能では，簡単な2重ループは1重化できるが，Fig. 2.2に示すようなループ構造の組み換えについては難しく，人手で行うしかない。

(2) odd－even SOR法によるベクトル化

2次元計算の場合，最も計算時間のかかる中性子束を計算しているサブルーチンは，体系の幾何形状により，大きく分けて3種類のサブルーチンFWRD（FXRD），FTRI，HWRD（HXRD）がある。

カッコでくくったサブルーチンFXRD，HXRDは，SLOR法の収束を早めるためのものであり，FWRD，HWRDとの違いは，計算方向が行方向と列方向という点のみである。

FWRDは，平板，円筒，球形状用であり，FTRIは三角形状用であり，HWRDは六角形状用である。このうちHWRDは使用頻度が少ないということから，今回の作業では除いている。

① FWRDルーチン

FWRDルーチンは，オリジナルではSLOR法であったが，3次元計算部分のベクトル化を実施した時の経験（ベクトル計算ではSLOR法より，odd－even SOR法が早い。）により，odd－even SOR法を採用した。ベクトル化版をFig. 2.3に，odd－even法をFig. 2.4に示す。

2次元平板にodd－even法を適用する場合，y方向のメッシュ数（CITATIONのソースプログラムではJVXまたはJMAX）が偶数か奇数かによってFig. 2.5に示すようなダミー領域を設けることによって，形状の境界における特殊処理を除去することができる。Fig.2.5の偶数と奇数の判別はプログラム内で自動的に行われ，ダミー領域も設定される。また，偶数と奇数でダミー領域のとり方が違うのは，2次元メッシュを1次元化して使用すると同じ行がすべて奇数（または偶数）になる場合があり（Fig. 2.6(b)参照），同じになると回帰的参照となる。それを防ぐダミー領域の設定方法がFig. 2.6(a)，(c)である。ダミー領域を設けると中性子束$\varphi_{i,j}$を計算するときに，前後左右の値を使用するが，$\varphi_{i,j}$が形状の境界にきたとき，前後左右の値が形状の外側になることがある。オリジナルコードでは，IF文を用いてこの部分の特殊処理を行っていた。これに対してベクトル化版では，形状の外側に当る部分のφ及び係数に0.0を入れたダミー領域を設定することによって，境界を意識しない計算となり，1つ置きに計算を進めればよいようにできる。

これによってデータ参照方式は，一定間隔（1つ置き）直接番地方式になる。

この方法による場合，ベクトル化していない部分との整合性をとるために，ダミー領域等を含めた配列の置き直しを内側反復計算の外で行うことが必要になる。だが，これによるオーバーヘッドはあるものの，直接番地方式を採用することによって，データ参照処理速度は向上する。

② FTRIルーチン

三角平面における各中性子束の対応を考えるとFig. 2.7の様になる。

したがって，FTRIルーチンにodd－even SOR法を適用したベクトル版はFig. 2.8のよ

うになる。FTRIルーチンにodd−even法を適用する場合，Fig. 2.9に示すようなダミー領域を設けることによって，形状の境界における特殊処理を除去することができる。なお，三角形状の場合，常にJVXは偶数であるので，その場合だけ考えている。

(3) 形状に対称性を有する場合の境界の特殊処理

2次元のX−Y，R−Z形状用のFWRD，三角形状用のFTRI及び3次元のX−Y−Z，θ−R−Z形状用のKWRD，三角形状用のKTRIルーチンについて，形状の境界において，対称性または周期性を有する場合のオプションが用意されている。

例えば，X−Y−Z形状で$\frac{1}{4}$炉心を解く場合，または，三角形状で$\frac{1}{6}$炉心を解く場合などである。（Fig. 2.10参照）

CITATIONコードのオリジナル版においては，これらのオプションの処理をＩＦ文を用いて毎回判定しながら，この境界部分だけを特別に処理している。これに対して，ベクトル化版では，Fig. 2.10に示すように，ダミー領域に予め，対応する特殊な部分の値を入れておく。これによって，計算は境界部分を特に意識せず，連続して計算可能となり，ベクトル化効率も上がる。

2.3.2　3次元形状

(1) 3次元三角形状用の中性子束計算ルーチンKTRIのベクトル化方法

ベクトル化の方法としては，3次元の他の形状，X−Y−Z形状やθ−R−Z形状用のルーチン（KWRD）と同様に[2]，オリジナル版で用いられていたSLOR法をOdd−even SOR法に変更して行う。

ただし，その場合に，z方向のインデックスについては，真上と真下の値を参照するので，他の形状のルーチンと同様でよいが，y方向のインデックスについては参照の仕方に注意がいる。2次元三角形状用のルーチンFTRIでも述べたように，Fig. 2.7(a)に示す三角形メッシュのy方向（インデックスｉ）の参照する点は，求める点が偶数番目のときはｉ＋１を，奇数番目のときはｉ−１の点を参照する。これが，メモリ上の2次元配列ではFig. 2.7(b)に示すようにｊ方向に＋1，−1それぞれずれた点を参照する形になるようにインデックスを指定する。

また，この場合にx方向（インデックスｊ）のメッシュサイズは，三角形メッシュの作り方から，常に偶数になるので，ダミー領域の設け方も一通りになり，かつ，求める点と参照される点は，メモリの2次元配列を1次元化しても偶数番目と奇数番目に分かれるので，Odd−even SOR法が適用できる。

(2) KBPRルーチン

KBPRルーチンは，3次元の中性子の吸収，生成を計算するサブルーチンである。

KBPRルーチンでは，各メッシュの中性子束をゾーン毎に分類する操作があり，このままでは回帰的参照となりベクトル化できない（Fig. 2.11参照）。そこでこの部分をベクトル化するために作業用配列を作り（Fig. 2.12のWORK1，WORK2），Fig. 2.12のようにベクトル化した。なお，Fig. 2.12のWORK1とWORK2は同じ番地をさしている。

(3) KINSルーチン

KINSルーチンは，三次元形状でblack absorberによるロスの計算（NUAC17＞0）の

場合に呼ばれるサブルーチンであり，代表的な例は，Fig. 2.13のような1／6炉心を三角形状で近似するものである。この場合斜線の部分が炉心の外になり中性子束密度はゼロになる。このような無限吸収領域とそうでない領域の境界をx，y，z方向について検索し，その点での中性子束を計算するのがKINSルーチンである。

KINSルーチンは，外側反復及び群のループの中にあり，内側反復の外にある（Fig. 2.14参照）。したがって，内側反復回数が多い場合は影響は小さいが，試計算例のように内側反復回数が少ない（4回）場合は影響が大きい。

KINSルーチンでは，呼ばれるたびに無限吸収領域とそうでない領域の境界を探索している。しかし一般に，この探索は，群及び外側反復では変化しないので1度だけ行えばよい。そこで，群と外側反復の外側で1度探索をし，そのデータを保存しておき，その境界値についてのみ中性子束の計算を行えば，計算時間は飛躍的に短くなる。なお，群に依存して変わる場合もあるのでその場合は群毎に計算をする。

また，KINSルーチンでは，KBPRルーチンと同様に，各メッシュの中性子束をゾーンに分類する操作があり，このままでは回帰的参照となりベクトル化できない。そこで，KBPRルーチンと同じ手法を用いて，その部分をベクトル化した。

2.4 計算結果の評価

ベクトル化したコードの計算結果を評価するために，2次元形状について3ケース，3次元形状について2ケースの試計算を実施した。

2.4.1 2次元形状

Table 2.10に示す4ケースについて，試計算を実施した。試計算結果をTable 2.11に示す。

各ケースの値を比較すると，全項目について10^{-4}以下の誤差である。収束判定条件を10^{-4}と設定していることから考えて妥当な値とみなすことができる。

2.4.2 3次元形状

Table 2.12に示す2ケースについて試計算を実施した。これは，各々のもっとも計算時間の短い内側反復回数と初期加速係数である。

試計算結果をTable 2.13に示す。これによると，全項目について3×10^{-5}以下の誤差である。収束条件を10^{-4}と設定していることから考えて妥当な値とみなすことができる。

2.5 ベクトル化効果

2.5.1 2次元形状

ここで計算の対象とした問題は2.2.1で示した3ケースである。

プログラムについては，CITATIONコードのオリジナル版，ベクトル化版の2種類を対象

とする。

計算時間は，内側反復回数と初期加速係数の値に依存するので，これに関する考察は2.6.1で述べるが，各ケースの計算時間は，パラメータサーベイの結果，最適な内側反復回数と初期加速係数を指定したときの値である。

(1) ケース1

ケース1の計算時間，外側反復回数等をTable 2.14に示す。

これによると，ベクトル化版ベクトル計算はオリジナル版の3.47倍，ベクトル化版スカラ計算の6.08倍に高速化された。ベクトル化版スカラ計算がオリジナル版に比べて遅くなっているのはこのデータの場合odd－even SOR法がSLOR法に比べて収束のための繰返し回数が多いためと思われる。

ベクトル化率は，93.4％である。

(2) ケース2

ケース2の計算時間，外側反復回数等をTable 2.15に示す。

これによると，ベクトル化版ベクトル計算はオリジナル版の4.75倍，ベクトル化版スカラ計算の4.67倍に高速化された。

ベクトル化率は94.0％である。

(3) ケース3

ケース3の計算時間，外側反復回数等をTable 2.16に示す。

これによると，ベクトル化版ベクトル計算は，オリジナル版の2.33倍，ベクトル版スカラ計算の4.41倍に高速化された。オリジナル版に対してあまり高速化されていないのは，外側反復回数のループの中にあり，内側反復回数のループの外にあるFINSルーチン（制御棒の効果を考慮するサブルーチンがベクトル化されていないためと，本ケースではオリジナル版と比較して反復回数（外側反復回数と内側反復回数の積）が多いためと考えられる。

ベクトル化率は84.1％である。

2.5.2 3次元形状

ここで計算の対象とした問題は，2.2.2で示した2ケースである。

(1) ケース1

ケース1は3次元X−Y−Z形状であり，すでに主要なサブルーチンはベクトル化済であった。今回の作業では，この主要なサブルーチンをより高速化し，さらにKBPRのベクトル化をはかった。

ケース1の計算時間，外側反復回数等をTable 2.17に示す。

これによるとベクトル化版ベクトル計算は，オリジナル版の10.13倍に高速化された。ベクトル化率は98.1％である。

(2) ケース2

ケース2は，3次元三角形状であり，今回の作業ではじめてベクトル化がなされた。

ケース2の計算時間，外側反復回数等をTable 2.18に示す。これによると，ベクトル化版ベクトル計算は，オリジナル版の4.54倍，ベクトル化版スカラ計算の6.05倍に高速化された。

ベクトル化版（ベクトル計算）のベクトル化率は97.3％である。

(3) KBPR

Table 2.19にKBPRルーチン高速化の効果を示す。これによるとKBPRルーチンは2.7倍，全体で1.1倍となった。

(4) KINSルーチン

Table 2.20にKINSルーチン高速化の効果を示す。これによるとKINSルーチンは，今回のデータでは全メッシュサイズに比べてblack absorberの境界数が少なかったこともあり，170倍以上，全体でも3倍以上の効果となった。

2.6 議　論

2.6.1 2次元形状の内側反復回数及び初期加速係数

2.5.1で述べたように，計算時間は入力データで指定した内側反復回数と初期加速係数の値に依存して変化する。CITATIONコードでは入力データで指定しなかったとき，内側反復回数は1回になる。

各ケースの最適な内側反復回数，初期加速係数の一覧をTable 2.21に示す。

これによると次のことが言える。

(1) オリジナル版

① 内側反復回数は，3～5回であり，省略値を用いても大きな違いはない。

② 初期加速係数は，1.6～1.8でかなりまとまっている。3ケースの平均である1.7前後の値がよいと思われる。

(2) ベクトル化版

① 内側反復回数は，3次元形状では3～4回が最適な例が多かったが，2次元形状では23，19，15回であり，20回の前後に集まっている。しかも20回との差も小さいので，20回程度がよいと思われる。

② 初期加速係数は，1.85，1.85，1.5であり，メッシュサイズにもよるが，1.8程度がよいと思われる。

ケース毎，オリジナル版，ベクトル化版毎に，内側反復回数と初期加速係数を変化させたときの計算時間の比較結果を下で示すような表としてまとめた。

ケース1	オリジナル版	…… Table 2.22
	ベクトル化版	…… Table 2.23
ケース2	オリジナル版	…… Table 2.24
	ベクトル化版	…… Table 2.25
ケース3	オリジナル版	…… Table 2.26
	ベクトル化版	…… Table 2.27

2.6.2 領域の増加

ベクトル化版では，プログラムをベクトル化するために作業用配列をいくつかに作ったので

領域が増加している。各計算条件での増加領域をTable 2.28に示す。

2.7 まとめ

今回のベクトル化作業により，2次元円筒，平板，三角形状及び3次元三角形状のベクトル化がなされた。さらに3次元形状では無限吸収帯のあるデータの計算をするサブルーチンもベクトル化された。

現在原研でベクトル化されていないサブルーチンとしては，2次元及び3次元六角形状，2次形状の無限吸収帯のあるデータの計算をするサブルーチンがある。したがって，今後さらにベクトル化を進めるならば，これらのサブルーチンを中心に行うのがよいと考える。

Table 2.1 Tree structure and contents of subroutines

SUBROUTINE NAME	CONTENTS
MAIN	Allocates total core storage
└INPT	Initializes variables
├IPTM	Controls the calculations for the entire problem
└CALR	Input control routine
└EIGN	Eigenvalue calculation control
├FLUX	Eigenvalue-flux calculation for 1D and 2D geometry
├DNSD	Controls 1D and 2D diffusion theory flux-eigenvalue problem and calculates boundary leakage
├FWRD	Line relaxation along rows for flux calculation for 2D geometry
├FXRD	Line relaxation on columns for flux calculation for 2D geometry
├HWRD	Line relaxation along rows for flux calculation for 2D hexagonal geometry
├HXRD	Line relaxation on columns for 2D hexagonal geometry
├DPER	Line relaxation for 2D periodic boundary condition problems
├FTRI	Line relaxation for flux calculation for 2D
└FINS	Calculates rod losses for 1D and 2D dimensional geometry
└ABPR	Calculates absorptions and productions for 1D and 2D geometry
└KLUX	Eigenvalue-flux calculation for 3D geometry
├KNSD	Controls 3D diffusion theory flux-eigenvalue problem and calculates boundary leakage
├KWRD	Line relaxation along rows for flux calculation for 3D geometry
├KXRD	Line relaxation along columns for 3D
├KZRD	Line relaxation fore and aft for 3D
├MWRD	Line relaxation along rows for flux calculation for 3D hexagonal geometry
├KPER	Line relaxation for 3D periodic boundary condition problems
├KTRI	Line relaxation for flux calculation for 3D
└KINS	Calculates rod losses for 3D geometry
└KBPR	Calculates absorptions and productions for 3D geometry

Table 2.2 CITATION 2-D input data used for dynamic profile analysis

入力項目	ケース1	ケース2	ケース3
幾何形状	円筒	円筒	三角形状
X方向メッシュ数	51	46	86
Y方向メッシュ数	49	32	43
群数	25	70	24
初期加速係数	1.8	1.6	1.4
インナーループ数	8	3	3

Table 2.3 Time distribution for CITATION 2-D Case-1 probrem

サブルーチン名	実行回数	時間比率(%)
MAIN	1	–
INPT	1	–
IPTM	1	0.1
CALP	1	–
EIGN	1	–
CNST	1	0.8
INFX	1	–
FLUX	1	–
BEGN	1	–
LOOP	29	0.7
DNSD	28	8.4
FXRD	5600	44.1
FWRD	5600	44.2
ABPR	28	0.7
EXTR	28	–
RDUE	1	0.9
ITED	28	–
NMBL	1	–
OUTC	1	–

– 0.1%未満

Table 2.4 Time distribution for CITATION 2-D Case-2 probrem

サブルーチン名	実行回数	時間比率(%)
MAIN	1	—
INPT	1	—
IPTM	1	0.1
CALR	1	—
EIGN	1	—
CNST	1	0.6
INFX	1	—
FLUX	1	—
BEGN	1	—
LOOP	16	0.3
DNSD	15	6.3
FXRD	21000	45.5
FWRD	21000	45.4
ABPR	15	0.3
EXTR	15	—
RDUE	1	1.4
ITED	5	—
NMBL	1	—
OUTC	1	—

— 0.1%未満

Table 2.5 Time distribution for CITATION 2-D Case-3 probrem

サブルーチン名	実行回数	時間比率(%)
MAIN	1	−
INPT	1	−
IPTM	1	0.1
CMOT	2	−
OVER	1	−
MACR	1	−
CALR	1	−
EIGN	1	−
BIGS	1	−
XSET	1	−
CNST	1	0.6
INFX	1	−
FLUX	1	0.1
BEGN	1	−
LOOP	57	1.2
DNSD	53	74.8
FTRI	3816	18.1
FINS	1272	3.6
ABPR	53	1.2
EXTR	53	−
RDUE	1	0.3
ITED	53	−
NMBL	1	−
OUTC	1	−
POUT	2	−

− 0.1%未満

Table 2.6 CITATION 3-D input data used for dynamic profile analysis

入力項目	ケース1	ケース2
次元	3次元	3次元
形状	XYZ形状	三角形状
メッシュ数	51×56×24	32×16×15
群数	3	24
初期加速係数	1.8	無指定
内側反復回数	2	1

Table 2.7 Time distribution for CITATION 3-D Case-1 probrem

Subroutine name	実行回数	時間比率(%)
MAIN	1	–
INPT	1	–
IPTM	1	–
CALR	1	–
EIGN	1	–
KNST	1	1.5
KNFX	1	0.1
KLUX	1	0.1
KEGN	1	–
KOOP	69	4.3
KNSD	67	20.3
KWRD	402	69.0
KBPR	67	3.5
EXTR	67	–
KDUE	1	1.1
RQED	1	–
ITED	67	0.1
NMBL	1	–
OUTC	1	–

– 0.1%未満

Table 2.8 Time distribution for CITATION 3-D Case-2 probrem

Subroutine name	実行回数	時間比率(%)
NAIN	1	–
INPT	1	–
IPTM	1	–
CALR	1	–
EIGN	1	–
KNST	1	2.1
KNFX	1	0.1
KLUX	1	0.2
KEGN	1	–
KOOP	40	2.0
KNSD	37	29.0
KTRI	1776	49.0
KINS	888	13.6
KBPR	37	2.6
EXTR	37	–
KDUE	1	1.3
RQED	44	–
ITED	37	–
NMBL	1	–
OUTC	1	–

– 0.1%未満

Table 2.9 Time distribution for case of black absorber

SUBROUTINE	CPU Time (sec)	VU Time (sec)	CPU比率 (%)
KNSD	28.10	23.41	10.47
KTRI	18.61	16.16	6.94
KINS	183.80	5.01	68.51
その他	37.78	5.56	14.08
合計	268.29	50.14	100.00

Table 2.10 CITATION 2-D input data

入力項目	ケース1	ケース2	ケース3	ケース1′
幾何形状	円筒	円筒	三角	円筒
X方向メッシュ数	51	46	86	51
Y方向メッシュ数	49	32	43	49
群数	25	70	24	25
オリジナル版内側反復回数	3	4	5	23
オリジナル版初期加速係数	1.73	1.8	1.6	1.85
ベクトル化版内側反復回数	23	19	15	23
ベクトル化版初期加速係数	1.85	1.85	1.5	1.85

Table 2.11 Evaluation of calculation results for CITATION 2-D

ケース	項目	オリジナル版 (A)	ベクトル化版 スカラー計算(B)	ベクトル化版 ベクトル計算(C)	相対誤差 $\frac{(B)-(A)}{(A)}$
1	Leakage	9.41585×10^{-2}	9.41550×10^{-2}	9.41550×10^{-2}	-3.7×10^{-5}
	Total loss	1.01111	1.01111	1.01111	0.0
	K_{eff}	0.9890583	0.9890564	0.9890564	-1.9×10^{-6}
2	Leakage	5.72579×10^{-2}	5.72592×10^{-2}	5.72592×10^{-2}	2.3×10^{-5}
	Total loss	1.00222	1.00221	1.00221	-1.0×10^{-5}
	K_{eff}	0.9978433	0.9978454	0.9978455	2.2×10^{-6}
3	Leakage	0.0	0.0	0.0	0.0
	Total loss	7.61275×10^{16}	7.61355×10^{16}	7.61355×10^{16}	1.0×10^{-4}
	K_{eff}	1.0226526	1.0225811	1.0225811	-7.0×10^{-5}
1′	Leakage	9.41586×10^{-2}	9.41550×10^{-2}	9.41550×10^{-2}	-3.8×10^{-5}
	Total loss	1.01111	1.01111	1.01111	0.0
	K_{eff}	0.9890585	0.9890564	0.9890564	-2.1×10^{-6}

Table 2.12 CITATION 3-D input data

入力項目	ケース 1	ケース 2
次元	3 次元	3 次元
形状	x－y－z形状	三角形状
メッシュ数	51×56×24	32×16×15
群数	3	24
オリジナル版 内側反復回数	2	2
オリジナル版 初期加速係数	1.8	1.5
ベクトル化版 内側反復回数	4	2
ベクトル化版 初期加速係数	1.8	1.3

Table 2.13 Evaluation of calculation results for CITATION 3-D

ケース	項　目	オリジナル版 (A)	ベクトル化版 (B)	相対誤差 $\frac{(B)-(A)}{(A)}$
1	Leakage	3.13173×10^{13}	3.13182×10^{13}	2.9×10^{-5}
	Total loss	7.46212×10^{16}	7.46190×10^{16}	-2.9×10^{-5}
	K_{eff}	1.0497437	1.0497656	2.1×10^{-5}
2	Leakage	6.60696×10^{15}	6.60686×10^{15}	1.5×10^{-5}
	Total loss	7.77649×10^{16}	7.77647×10^{16}	2.6×10^{-6}
	K_{eff}	1.0015917	1.0015984	6.7×10^{-6}

Table 2.14 Vectorization effect for CITATION 2-D Case-1

	オリジナル版 スカラ計算	ベクトル化版 スカラ計算	ベクトル化版 ベクトル計算
内側反復回数	3	23	23
初期加速係数	1.73	1.85	1.85
外側反復回数	25	20	20
CPU時間　(秒)	54.02	94.74	15.58
VU時間　(秒)	0.0	0.0	12.00
倍　率*	1.00	0.57	3.47
I／O回数	102	64	81
メモリ使用量(KB)	2224	2379	2379

* 倍率：オリジナル版スカラ計算のCPU時間を1とした時の倍率

Table 2.15 Vectorization effect for CITATION 2-D Case-2

	オリジナル版 スカラ計算	ベクトル化版 スカラ計算	ベクトル化版 ベクトル計算
内側反復回数	4	19	19
初期加速係数	1.80	1.85	1.85
外側反復回数	26	17	17
CPU時間 (秒)	120.87	118.66	25.43
VU時間 (秒)	0.0	0.0	18.14
倍 率	1.00	1.02	4.75
I/O回数	145	129	146
メモリ使用量(KB)	3008	3114	3114

Table 2.16 Vectorization effect for CITATION 2-D Case-3

	オリジナル版 スカラ計算	ベクトル化版 スカラ計算	ベクトル化版 ベクトル計算
インナーループ数	5	15	15
初期加速係数	1.6	1.5	1.5
アウターループ数	29	30	30
CPU時間 (秒)	62.56	118.36	26.86
VU時間 (秒)	0.0	0.0	16.93
倍 率	1.00	0.53	2.33
I/O回数	149	149	151
メモリ使用量(KB)	2364	2587	2587

Table 2.17 Vectorization effect for CITATION 3-D Case-1

	オリジナル版 スカラ計算	ベクトル化版 スカラ計算	ベクトル化版 ベクトル計算
内側反復回数	2	4	4
初期加速係数	1.8	1.8	1.8
外側反復回数	67	53	46
CPU時間 (秒)	241.76	215.77	23.87
VU時間 (秒)	0.0	0.0	19.23
倍率	1.00	1.12	10.13
I/O回数	37615	105	107
メモリ使用量(KB)	3380	5695	5695

Table 2.18 Vectorization effect for CITATION 3-D Case-2

	オリジナル版 スカラ計算	ベクトル化版 スカラ計算	ベクトル化版 ベクトル計算
内側反復回数	2	2	2
初期加速係数	1.5	1.3	1.3
外側反復回数	37	61	47
CPU時間 (秒)	160.39	213.57	35.31
VU時間 (秒)	0.0	0.0	30.90
倍率	1.00	0.75	4.54
I/O回数	262	493	462
メモリ使用量(KB)	3380	4787	4787

Table 2.19 Vectorization effect of subroutine KBPR on CITATION

	オリジナル版 ＣＰＵ時間（秒）	ベクトル化版 ＣＰＵ時間（秒）	倍　率 （倍）
KBPR	57.30	21.15	2.71
そ の 他	313.25	313.25	1.00
全　体	370.55	334.40	1.11

Table 2.20 Vectorization effect of subroutine KINS on CITATION

サブルーチン	オリジナル版 ＣＰＵ時間（秒）	ベクトル化版 ＣＰＵ時間（秒）	倍　率 （倍）
KINS	183.80	1.04	176.73
そ の 他	84.49	84.49	1.00
全　体	268.29	85.53	3.14

Table 2.21 The optimal number of inner iterations and the initial relexation factor

		内側反復回数	初期加速係数
オリジナル版	ケース 1	3	1.73
	2	4	1.80
	3	5	1.60
	ケース 1	23	1.85
	2	19	1.85
	3	15	1.50

Table 2.22 The number of outer iterations and CPU time for CITATION original version 2-D Case-1

ω \ inner	1	2	3	4
無指定			25 54.20	
1.73	63 83.10	24 76.88	25 54.02	24 77.09
1.85			34 72.69	
1.90			43 89.89	

inner : 内側反復回数
ω : 初期加速係数

上段 : 外側反復回数
下段 : CPU時間（秒）

Table 2.23 The number of outer iterations and CPU time for CITATION vector version 2-D Case-1

ω \ inner	10	15	20	22	23	24
1.6					48 33.89	
1.7					34 24.48	
1.8				26 18.76	26 19.26	24 18.47
1.85	54 23.71	29 16.80	24 16.67	25 18.15	20 15.58	20 15.91
1.9				26 19.08	23 17.71	24 18.75

Table 2.24 The number of outer iterations and CPU time for CITATION original version 2-D Case-2

ω \ inner	2	3	4	5	10
1.6		48 181.75			
1.7		42 159.48			
1.8	64 193.16	39 148.29	26 120.87	23 124.20	16 147.18
1.85		43 163.11			
1.9		43 163.10			

Table 2.25 The number of outer iterations and CPU time for CITATION vector version 2-D Case-2

ω \ inner	10	18	19	20	21	25
1.6				38 50.98		
1.7				28 39.09		
1.8		21 29.36	21 30.04	20 29.58	19 28.99	
1.85	33 34.47	18 25.	17 25.43	17 26.01	17 26.55	18 30.04
1.9		25 33.91	32.34	23 33.12	33.87	

Table 2.26 The number of outer iterations and CPU time for CITATION original version 2-D Case-3

ω \ inner	1	2	3	4	5	6	7
1.2					39 83.55		
1.3					36 77.78		
1.4	106 134.08	61 90.88	53 90.34	45 85.89	31 66.67	30 70.57	29 74.40
1.5				41 78.69	29 62.61	33 77.22	
1.6				37 71.80	29 62.56	31 72.75	
1.7				36 70.18	33 71.09	33 77.64	
1.8					37 79.28		

Table 2.27 The number of outer iterations and CPU time for CITATION vector version 2-D Case-3

ω \ inner	5	10	13	14	15	16	20
1.2					39 34.43		
1.3				39 33.43	39 34.21	33 30.14	
1.4	56 37.53	41 31.92	38 31.83	31 27.08	30 26.86	30 27.43	29 28.88
1.5				38 33.05	43 37.03	44 39.29	
1.6					40 35.64		
1.7					49 43.01		

Table 2.28 Increased region for CITATION vector version

計　算　条　件	増　加　領　域
2次元　三角形状	((JVX+2)*(IVX+2)*6 +JVX*IVX+4*MVX*KVX)*2+1
2次元　その他の形状	$((\frac{JVX}{2}*2+3)*(IVX+2)*6$ +JVX*IVX+4*MVX*KVX)*2+1
3次元　三角形状　無限吸収帯あり	(JVX+2)*(IVX+1)*(KBVX*6+9)+JVX* IVX*KBVX+(JVX+5)*KVX*24+8
3次元　三角形状　無限吸収帯なし	(JVX+2)*(IVX+1)*(KBV*6+9)+JVX* IVX*KBVX
3次元　その他の形状　無限吸収帯あり	$(\frac{JVX}{2}*2+3)*(IVX+1)*(KBVX*6+9)+$ JVX*IVX*KBV+(JVX+5)*KVX*24+8
3次元　その他の形状　無限吸収帯なし	$(\frac{JVX}{2}*2+3)*(IVX+1)*(KBVX*6+9)+$ JVX*IVX*KBVX

JVX　：J方向メッシュサイズ
IVX　：I方向メッシュサイズ
MVX　：ゾーン数
KVX　：群数
KBVX　：K方向メッシュサイズ

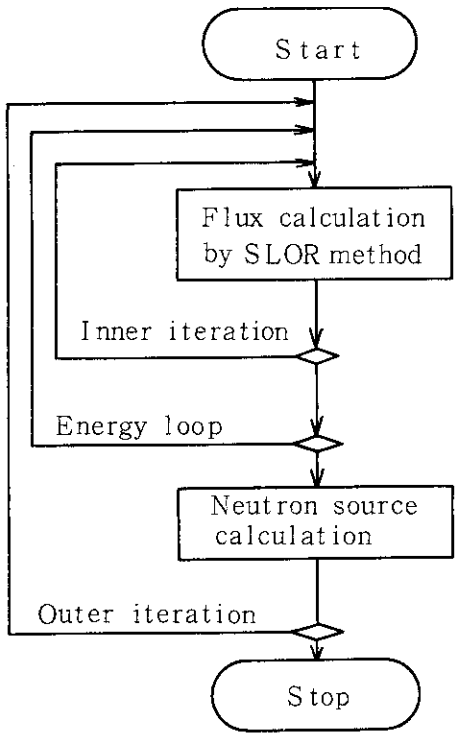


Fig. 2.1 Inner-outer iteration in CITATION code

〔オリジナル版〕

```
REAL  A( imax, jmax )
for  j=1, jmax
for  j=1, imax
     C=0.0
     for  ℓ= ℓ1, ℓ2
          C=C+S(i,j,ℓ)×φ(i,j,ℓ)
          A(i,j) = C×V(i,j) + R(i,j)×E
```

〔ベクトル化版〕

```
REAL  A (imax×jmax), C (imax×jmax)
for  n=1, imax×jmax
     Cn = 0.0
for  ℓ=ℓ1, ℓ2
     for  n=1, imax×jmax
          Cn=Cn+Sn,ℓ×φn,ℓ
for  n=1, imax×jmax
     An = Cn×Vn + Rn×E
```

Fig. 2.2 Vectorization of double loop in 2-D geometry calculation for CITATION code

〔m : iteration〕

for ij = 1, imax × jmax, 2 〔odd index〕

$$\tilde{\varphi}_{ij}^{(m)} = \{ A \times \varphi_{ij-1}^{(m-1)} + B \times \varphi_{ij-id}^{(m-1)} + A' \times \varphi_{ij+1}^{(m-1)} + B' \times \varphi_{ij+id}^{(m-1)} + S \} \div D$$

for ij = 1, imax × jmax, 2

$$\varphi_{ij}^{(m)} = \varphi_{ij}^{(m-1)} + \omega \times \{ \tilde{\varphi}_{ij}^{(m)} - \varphi_{ij}^{(m-1)} \}$$

for ij = 2, imax × jmax, 2 〔even index〕

$$\tilde{\varphi}_{ij}^{(m)} = \{ A \times \varphi_{ij-1}^{(m)} + B \times \varphi_{ij-id}^{(m)} + A' \times \varphi_{ij+1}^{(m)} + B' \times \varphi_{ij+id}^{(m)} + S \} \div D$$

for ij = 2, imax × jmax, 2

$$\varphi_{ij}^{(m)} = \varphi_{ij}^{(m-1)} + \omega \times \{ \tilde{\varphi}_{ij}^{(m)} - \varphi_{ij}^{(m-1)} \}$$

Fig. 2.3 Odd-even SOR vectorized version in 2-D slab geometry calculation for CITATION code

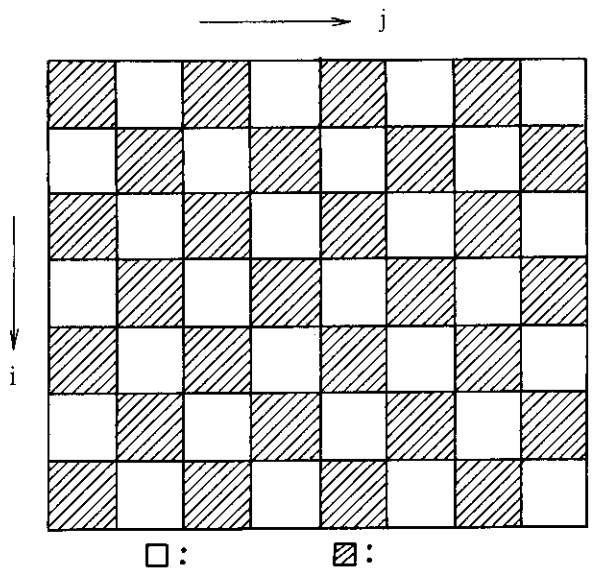
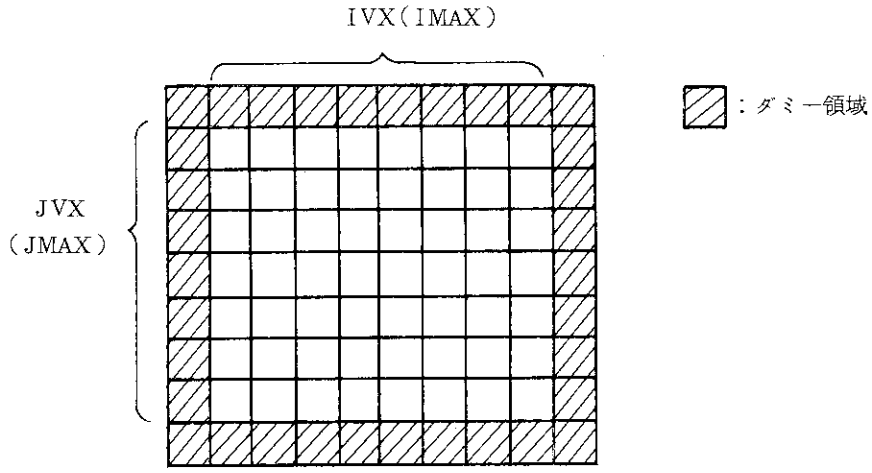
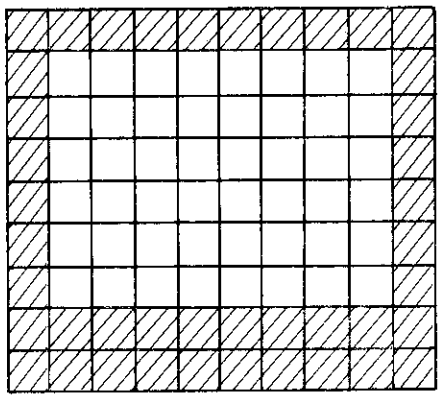


Fig. 2.4 Odd-even ordering in 2-D slab geometry calculation for CITATION code

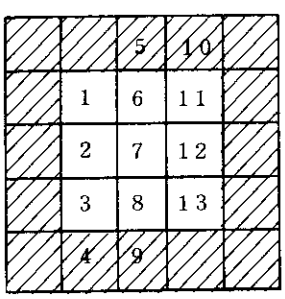


(1) ＪＶＸが奇数の場合のダミー領域のとり方

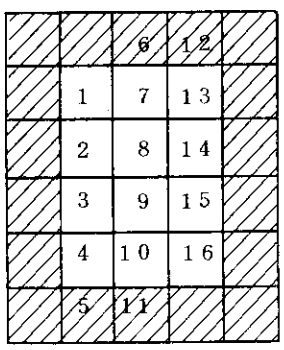


(2) ＪＶＸが偶数の場合のダミー領域のとり方

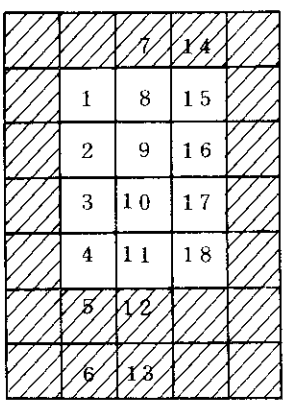
Fig. 2.5 Dummy area in 2-D slab geometry calculation for CITATION code



(a) 奇数の例



(b) 偶数で奇数と同じようにダミー領域を設定した場合



(c) 偶数の例

Fig. 2.6 Example of dummy area in 2-D slab geometry calculation for CITATION code

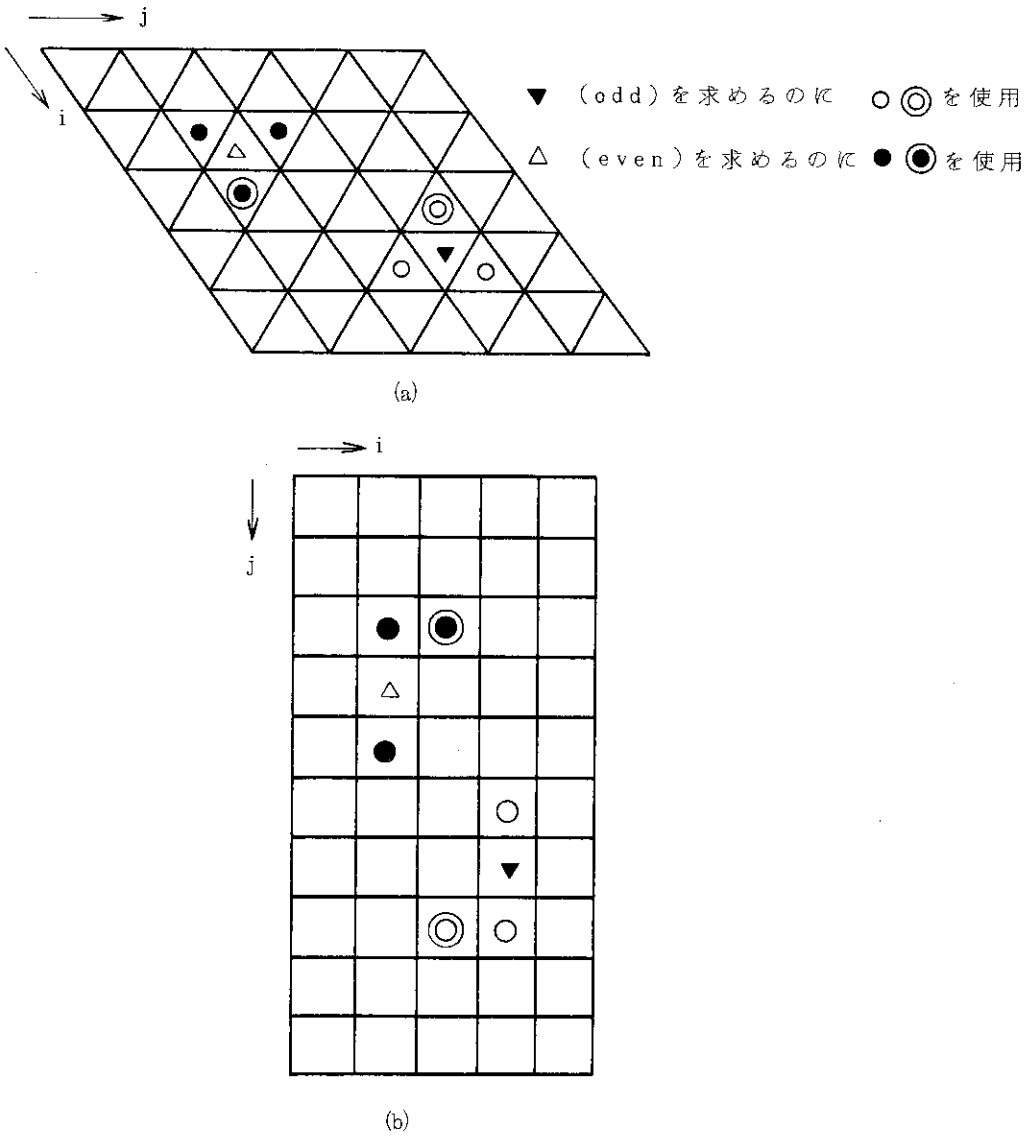


Fig. 2.7 Odd-even ordering in 2-D triangular geometry calculation for CITATION code

〔m : iteration〕

for ij = 1, imax × jmax, 2 〔odd index〕

$$\tilde{\varphi}_{ij}^{(m)} = \{ A \times \varphi_{ij-1}^{(m-1)} + B \times \varphi_{ij-id+1}^{(m-1)} + A' \times \varphi_{ij+1}^{(m-1)} + S \} / D$$

for ij = 1, imax × jmax, 2

$$\varphi_{ij}^{(m)} = \varphi_{ij}^{(m-1)} + \omega \times \{ \tilde{\varphi}_{ij}^{(m)} - \varphi_{ij}^{(m-1)} \}$$

for ij = 2, imax × jmax, 2 〔even index〕

$$\tilde{\varphi}_{ij}^{(m)} = \{ A \times \varphi_{ij-1}^{(m)} + A' \times \varphi_{ij+1}^{(m)} + B' \times \varphi_{ij+id-1}^{(m)} + S \} / D$$

for ij = 2, imax × jmax, 2

$$\varphi_{ij}^{(m)} = \varphi_{ij}^{(m-1)} + \omega \times \{ \tilde{\varphi}_{ij}^{(m)} - \varphi_{ij}^{(m-1)} \}$$

Fig. 2.8 Odd-even SOR vectorized version in 2-D triangular geometry calculation for CITATION code

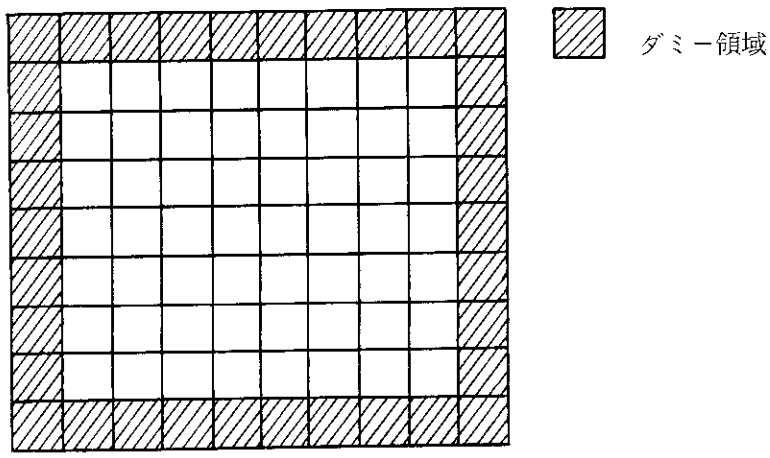
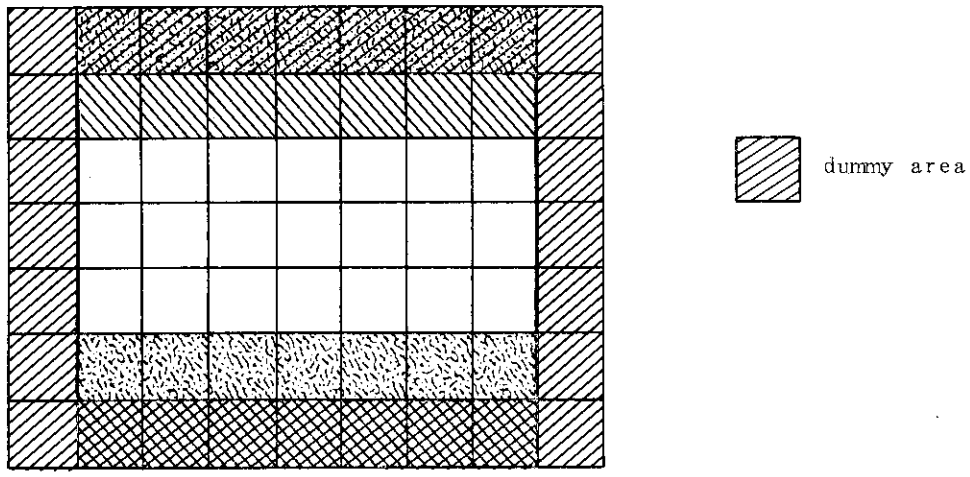
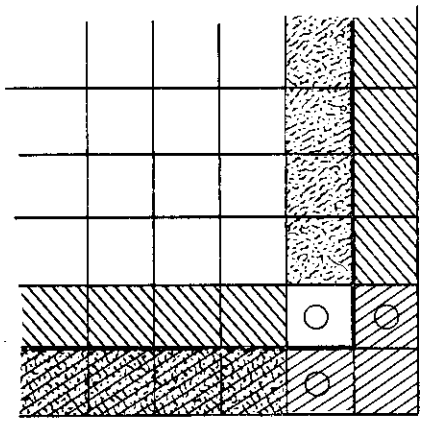


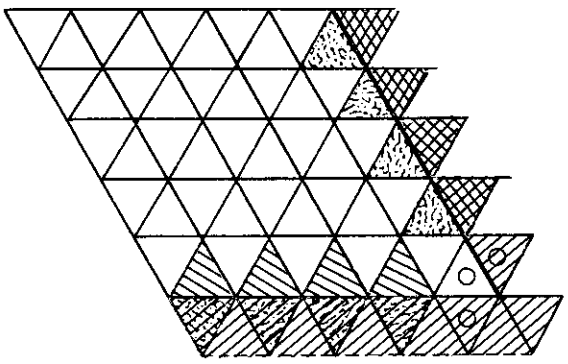
Fig. 2.9 Dummy area in 2-D triangular geometry calculation for CITATION code



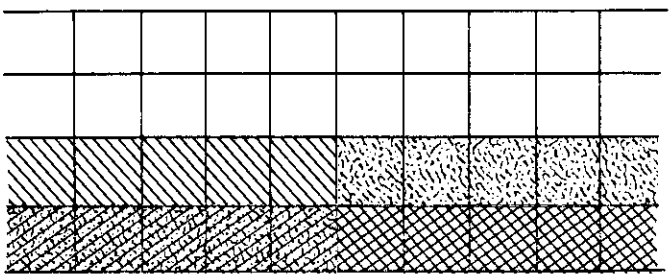
(a) periodic



(b) 90° ratation



(c) 60° rotation



(d) 180° ratation

Fig. 2.10 Rotation option for CITATION

```
          DO 105 K=1,KVX
          DO 104 KB=1,KBVX
          N1 = 0
          DO 103 I=1,IVX
          DO 102 J=1,JVX
          N1 = N1 + 1
          L = NRGNE(J,I,KB)
          M = NCOMP(L)
          B1(M,K) = B1(M,K) + P2E(N1,KB,K)*PVOL(L)
      102 CONTINUE
      103 CONTINUE
      104 CONTINUE
      105 CONTINUE
```

Fig. 2.11 Recurrence of subroutine KBPR for CITATION

```
REAL       WORK1(MVX,32),WORK2(MVX*32)
                 ⎧
                 ⎩
      JIKBXX=JIVX*KBVX
CNN
      I32 = 32
      IZN  = MVX * I32
CNN
        DO 1901 K=1,KVX
         DO 1900 IZ=1,IZN
          WORK2(IZ) = 0.0
 1900    CONTINUE
C
        IE = 0
        DO 1902 II=0,(JIKBXX-1)/I32
          IS = IE + 1
          IS1 = IS - 1
          IE = IS1 + I32
          IF(IE.GT.JIKBXX) IE=JIKBXX
*VOCL LOOP,REPEAT(32)
*VOCL LOOP,NOVREC
          DO 1903 JI=IS,IE
            JA = JI - IS1
            ML = NCOMP(NRGNE(JI))
            WORK1(ML,JA) = WORK1(ML,JA) +
     2        P2E(JI,K) * PVOL(NRGNE(JI))
 1903     CONTINUE
 1902   CONTINUE
C
        DO 1904 NN=1,I32,4
          DO 1905 I=1,MVX
            B1(I,K) = B1(I,K) + WORK1(I,NN) + WORK1(I,NN+1) +
     1                WORK1(I,NN+2) + WORK1(I,NN+3)
 1905     CONTINUE
 1904   CONTINUE
 1901   CONTINUE
```

Fig. 2.12 Vectorization of subroutine KBPR for CITATION

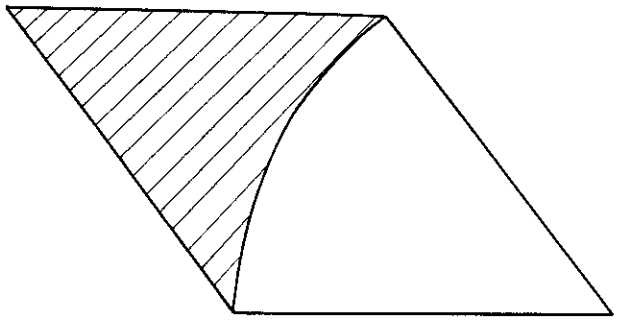


Fig. 2.13 Approximation of triangular geometry for 1/6 core

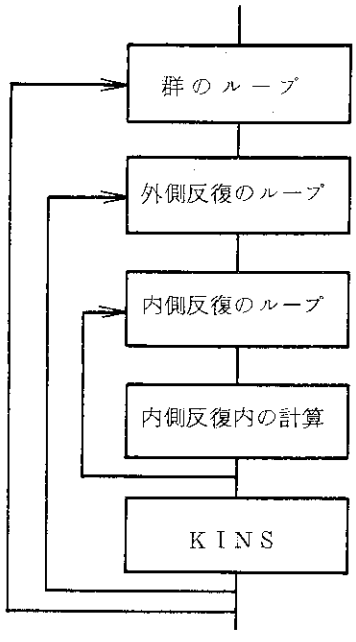


Fig. 2.14 Position of subroutine KINS for CITATION

3. TWOTRAN－Ⅱコード

TWOTRAN－Ⅱコードは，既に動力炉・核燃料開発事業団で斜めスイープ法を用いてベクトル化した版[4]がある。今回の作業ではこのTWOTRAN－Ⅱコードベクトル化版を基に，TWOTRAN－Ⅱコードベクトル化版を原研のデータでさらにテスト，整備し，さらに機能の向上，使い易さの向上をはかった。

3.1 コードの概要

TWOTRAN－Ⅱコード[5]は，ロスアラモス国立研究所で開発された2次元多群ディスクリート・オーディネーツ輸送コードである。

本コードは，多群の中性子輸送方程式をディスクリート・オーディネーツ法という差分法により離散化して解いている。（x－y）体系，（r－z）体系，（r－θ）体系の3種類の2次元体系を扱うことが可能である。また，中性子輸送計算の特徴として，中性子の散乱，中性子源の角度依存性を取り扱うことができる。

中性子輸送に対する基礎方程式は次の（3.1）式である。

$$\nabla\cdot(\underline{\Omega}\psi)+\sigma_t\psi(\underline{r},E,\underline{\Omega})=$$
$$\iiint dE'd\Omega'\psi(\underline{r},E',\underline{\Omega}')\sigma_s(E'\to E,\underline{\Omega}'\cdot\underline{\Omega})$$
$$+\chi(E)\iiint dE'd\Omega'\psi(\underline{r},E',\Omega')\nu\sigma_f(E')/4\pi+Q(\underline{r},E,\underline{\Omega}) \qquad (3.1)$$

ψ : particle flux

$\psi dVdEd\Omega$: the flux of particles in the volume element dV about $\underline{r}$

$QdVdEd\Omega$: the number of particles in the same element of phase space emitted by sources independent of ψ

dV : the volume element

$d\Omega$: the element of solid angle

dE : the energy range

σ_t : the macroscopic total interaction cross section

σ_s : the macroscopic scattering transfer probability (from energy E′ to E through an angle with cosine $\underline{\Omega}'\cdot\underline{\Omega}$)

σ_f : macroscopic fission cross section

ν : the number of particles emitted isotropically ($1/4\pi$) per fission

$\chi(E)$: the fraction of these liberated in the range dE about E

本コードの概略フローダイアグラムをFig. 3.1に示す。

Fig. 3.1 では明確ではないが，本コードは計算上のループとして2つのくり返しループを持っている。各々のループを次に示す。

① 分裂項があるための反復計算（outer iteration ループ）

② 自群散乱項があるための反復計算（inner iteration ループ）

このように各々の反復計算で分裂項，散乱項の効果を取り入れるようになっている。この①，②のループの内側には，空間メッシュ（i，j）と角度メッシュ（m）に関するDOループがあり，分裂項，散乱項を考慮した後の主要な計算を行っている。

この主要な計算部は連立一次方程式の解法部であり，サブルーチン名で言うとIN及びOUTルーチンに相当する。

本コードは，i，jの空間メッシュとmの角度メッシュを持つので差分式を7点差分とし，Fig. 3.2のような関係で中性子束を求める。

3.2 コードの動的挙動分析

Fig. 3.3は，ANALYSISツールによって解析したTWOTRAN－II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。

OUTER	Calculates source to group, calls INNER, generates coarse－mesh balance (over－all group) data and group sum information, calls GSUMS, calculates new fission rate.
INNER	Calculates total source for group, perfoms space－angle mesh sweeps by calling SETBC, IN and OUT, saves and stores angular fluxes, if required, by calling SAVEAF and STORAF, applies rebalance factors calculated by REBAL and calculates inner iteration error and controls convergence.
IN	Makes inward space－angle traverse for all I and M on level J. Calculates partial flows and calls FIXUP.
OUT	Makes outward space－angle traverse for all I and M on level J. Calculates partial flows and calls FIXUP.
FIXUP	Prevents negative fluxes unless sources are negative.

FORTUNEツールによる動的挙動分析に使用した入力データは，Table 3.1に示すように，2次元r－z形状，70エネルギー群28×32メッシュで解く問題である。

TWOTRAN－IIコードの場合，固有値問題の解法では，形状等の種類によって使用される主要サブルーチンは異ならない。したがって，Table 3.1で示したサンプルデータのFORTUNEツールによる解析結果を分析することで十分と考えられる。

上記入力データのFORTUNE解析結果をTable 3.2に示す。

Table 3.2を見ると内側反復計算内の2つのサブルーチン（IN，OUT）で，93％以上の計算時間を占めている。しかもこれらのサブルーチンのステップ数を合計しても600ステップ程度と少ない。したがってこの部分のみを集中的に解析すればよい。

3.3 ベクトル化方法

TWOTRAN－Ⅱコードのチューニングで用いられたベクトル化技法と高速化技法について述べる。

ベクトル化技法としては，「斜めスイープ法」を主に用い，高速化技法としては，「ダミー領域の使用によるＩＦ文の除去」，「ループの分割によるＩＦ文のループ外への移動」，「ループの分割，統合によるベクトル長の拡大」を用いた。

高速化技法で上記手法を用いた理由として，ＶＰコンパイラは，ＩＦ文を含むＤＯループもベクトル化できるが，処理効率の面からみると，なるべく少ない方がよく，ベクトル長も長い方がよいためである。

以下で各手法について具体的に述べる。

(1) 斜めスイープ法[5]

ＩＮ及びOUTルーチンのループ構造はFig. 3.2で示したように最内にｍループ，その上にｉループ，さらにその上にｊループとなっている。また，ｍ方向，ｉ方向，ｊ方向とも，隣のメッシュ点を参照している。具体的にはFig. 3.4のようなコーディングとなっている。

ここでＣ(ｉ)に注目するとFig. 3.5のようになる。

Ｍループ中はＣ(ｉ)は変化しないのでただの変数とみるとFig. 3.5右のように書換えられる。Fig. 3.5右のようなループはＭループ中のｍ番目に定義されたＣの値をｍ＋１番目の計算で使用する。この関係を回帰的参照があるといい，他の配列についてもみてみると次のような回帰的参照がある。

Ａ（ｍ，ｊ）⟶ ｊループについて回帰的参照がある

Ｂ（ｍ，ｊ）⟶ ｉループについて回帰的参照がある

Ｃ（ｉ）⟶ ｍ，ｊ ループについて回帰的参照がある

従ってこの３重ループは回帰的参照があり，このままではベクトル化できないことになる。

Fig. 3.4のコーディングはそのまま前進代入のロジックになっている。通常，前進（後退）代入処理は，ベクトル化可能なので，Fig. 3.4のコーディングも多少の変更でベクトル化することができる。

しかし，そのようなコーディングの変更でベクトル化の効果があがるのは大型の密行列の場合であり，大型でも疎行列の場合にはベクトル化の効果はあがらない。

従って他の方法でベクトル化を考える必要がでてくる。

現在のコーディングではベクトル化の妨げとなる回帰的参照があることがわかった。本節では回帰的参照を避けベクトル化可能となる方法について検討する。

尚，ここで述べる方法は，本コードのような疎行列の前進（後退）代入処理一般をベクトル化するのに役立つ方法である。

今ここで簡単のため，ｉ方向メッシュと角度メッシュのみの２方向を考える。

オリジナルの計算方法はFig. 3.6のようになっている。

Fig. 3.6(a)のような順序で計算すると，同一ループ内でFig. 3.6(b)のような回帰的参照が発生する。これをFig. 3.7(a)のような計算順序をとることにするとFig. 3.7(b)のような参照関係

となる。

Fig. 3.7(b)を見ると同一ループ内の点の参照が無いので回帰的参照はなくなる。従って，同一ループ内はベクトル化可能となる。このような方法を「斜めスウィープ法」とよぶ。ここでは簡単のため平面内で考えたが，実際には，i－j－mの3次元内で考えるのでFig. 3.8のような平面内スウィープでの回帰的参照をなくすことができる。

(2) その他のベクトル化技法[4]

オリジナルコーディングでは，Aの参照に重なりがあるため(I＝1～5，I＝6，10がそれぞれ重なっている）ベクトル化できない。それをFig. 3.9のように変更してやることにより，参照に重なりがなくなりベクトル化できる。なおかつ，ベクトル長も増大する。

(3) ダミー領域の使用によるＩＦ文の除去[4]

オリジナルソースプログラムは，計算点の場所により計算式を使い分けるためのＩＦ文が入っている。このようなＩＦ文を除去するためには，ダミー領域をつくり，そのダミー領域には「０」をつめておく方法が有効である。

Fig. 3.10(a)のようにオリジナルでは，●の点ではM とM－1 の点のfluxを参照し，○の点ではM の点のみ参照している。

Fig. 3.10(b)のようなコーディングの変更後は，式－hを使い○点を計算する。

この時φM－1は，ダミーとして付け加えた◎点を参照することになる。

◎点のfluxには，0が入っているので，Fig. 3.10(a)のコーディングと等しくなる。

(4) ループの分割によるＩＦ文のループ外への移動[4]

オリジナルプログラムソースは，メモリ節約のため，全メッシュスイープ後に計算すればよい値をループ中で計算している。この時，計算するメッシュ点かどうかを判定するＩＦ文が必要となる。そのようなＩＦ文を追い出すためには，全メッシュスイープのためのループと計算のためのループに分割し，ＩＦ文を最内ループの外へ移動する方法が有効である。Fig. 3.11に例を示す。

(5) ループの分割，統合によるベクトル長の拡大[4]

最内ループのベクトル長が短い場合，ベクトル化の効率が悪い場合が多い。このような場合，ループの分割，統合の方法が有効である。Fig. 3.12に例を示す。

3.4 計算結果の評価

ベクトル化したコードの計算結果を評価するために25ケースの試計算を実施した。

25ケースの試計算入力データをTable 3.3に，試計算結果をTable 3.4に示す。

各ケースの結果を比較すると固有値（K_{eff}）はすべて10^{-4}以下の相対誤差であり，中性子束は10^{-3}以下の誤差である。これは，収束条件が10^{-4}という点から考えると十分妥当な値と言える。

3.5 ベクトル化効果

Table 3.5, 3.6 に示す4ケースについてベクトル化効果を調べた。これによるとＣＰＵ時間の比較的長いケース１～ケース３は 3.5～4.2 倍の速度向上となりCPU時間の短いケース４も 2.8 倍の速度向上となった。

3.6 ベクトル化版の修正及び機能拡張

動燃で作成したTWOTRAN－Ⅱベクトル化版は，作成時の試計算が１ケースと少なかったため，そのケースにしか対応できないような制約があった。今回，原研で行う多彩なプロダクションランに対応するためプログラムを修正し，制限されていた機能を復活させ，入出力もTWOTRAN－Ⅱオリジナルコードと同じフォーマットにする作業を行った。

今回の作業で修正を行ったサブルーチンは，次の11ルーチンである。

① メイン
② VINOUT
③ INNER
④ SNCON
⑤ INPT11
⑥ SWEEP1
⑦ SWEEP2
⑧ SWEEP3
⑨ SWEEP4
⑩ REBAL
⑪ DUMPRD

原研で行った作業内容を以下に示す。

(1) VINOUT ルーチンの配列インデックスの修正

・改良方法

① BT22のインデックスを次のように修正した。

```
C********** 4 BLOCK START ***************
24250 DO 31650 M=1,MM
        DO 31650 I=1,IT
          K=1
          L=MM-M+I
C         FEDGJ(LSIGJ(L)+MM*(K-LSTAJ1(L))+M)=BT22(I,2)  } 修正部分
          FEDGJ(LSIGJ(L)+MM*(K-LSTAJ1(L))+M)=BT22(I,M)  }
31650 CONTINUE
```

② JCONV＝2の部分について，ほぼ全面的に書き直した。

・修正サブルーチン

VINOUT

(2) オーバーレイを使用した時の異常終了の調査

・改良方法

各メッシュが境界か内側か外側かを判定するフラグの配列を初期設定している部分に誤りがあった。

ＤＯループのくり返し回数を本来より少なくしていた。そのため，その配列の一部しか初期設定されていなかった。動燃でテストした時は，オーバーレイを用いなかったためゼロがあらかじめ入っており，正常に動作した。しかし，今回のテストでは，オーバーレイを使用したため前に使用したエリアが割り当てられ，オーバーフローが発生した。

ＤＯループ(550)のくり返し回数を（ＩＴ＋Ｍ－１）から，（ＩＴ＋ＪＴ＋ＭＭ－２）に変更した。

・修正サブルーチン

ＶＩＮＯＵＴ

(3) パラメータ文の廃止

・改良方法

ＶＩＮＯＵＴルーチンのＰＡＲＡＭＥＴＥＲ文を廃止し，メインルーチンで定義している共通配列（変数名Ａ）の一部としてベクトル化用作業配列を組み込んだ。これに伴う変更として，ＱＴ１Ｄ，ＱＴ２Ｄ，ＱＴ３Ｄ，ＱＴ４ＤのＥＱＵＩＶＡＬＥＮＣＥ文を廃止し，共通配列の同じ位置にポインターを設定した。さらに，ＩＮＰＴ１１ルーチンでは，共通配列で定義した大きさが実際に使用する大きさより大きいことを確認するために，実際に使用する大きさを計算し，その値を出力している。

・修正サブルーチン

ＩＮＮＥＲ

ＶＩＮＯＵＴ

ＩＮＰＴ１１

メイン

(4) ルジャンドル展開次数の高次化

・現　状

ルジャンドル展開次数の項は考えていない。つまり，ルジャンドル展開次数はゼロ次で固定である。

・改良方法

ルジャンドル展開次数に関連する変数の引数に，ルジャンドル展開次数（実際は，その計算式）のインデックスを追加する。また，その数をサブルーチンの引数に加える。

・修正サブルーチン

ＩＮＮＥＲ

ＶＩＮＯＵＴ

(5) 変数ＬＡＦ１，ＬＡＦ２の配列の大きさの誤り

・改良方法

次のように修正する。

① INNERルーチン

```
CNN    N33 = N32 + JK * KM
CNN    N34 = N33 + JK * KM
       N33 = N32 + IK * KM
       N34 = N33 + IK * KM
```

② VINOUTルーチン

```
CNN    DIMENSION LAF1(JK*KM),LAF2(JK*KM)
       DIMENSION LAF1(IK*KM),LAF2(IK*KM)
```

(6) オリジナル版とベクトル化版の入力フォーマットの相違の調査

・現　状

ベクトル化版では，入出力機番，計算時間等のデータを1レコード追加している。

・改良方法

ベクトル化版をオリジナル版と同じ入力フォーマットにする。つまり，ベクトル化版で追加した1レコードを削除し，定数で与える。

上記レコードの出力リストも出力しないようにする。

・修正サブルーチン

メイン

(7) 空間メッシュが小さいケースで，領域侵害が発生した原因の調査

・改良方法

境界，内側，外側の判定を保存する配列の値を設定している部分で，空間メッシュ数を越えて値をセットしていた。

空間メッシュのインデックスをたしあげする所すべてに判定文を挿入し，空間メッシュ数を越えないようにした。

・修正サブルーチン

VINOUT

(8) XY形状でのS_n定数（ETA）相違の調査

・修正方法

XY形状でのS_n定数（出力リストでの名称：ETA）が空間メッシュをインデックスとしてベクトル化版とオリジナル版で順序が逆であったので，オリジナル版と同じ順になるよう修正した。

・修正サブルーチン

SNCON

(9) iメッシュ数がjメッシュ数より大きいケースで，フラックスが正しく計算されない。

・改良方法

チェック出力を入れて調査したところ，形状関数の1つの関数（変数名：A4）の宣言に誤りがあった。すなわち，本来，IT+1で宣言すべきところをJTで定義していた。

宣言文のJTをIT+1に変更し，さらに，同じ意味で新しく定義した変数A4DもIT+1に変更した。

また，値を移しているループのくり返し回数もJTからIT+1に変更した。

・修正サブルーチン

VINOUT

(10) Zone thickness search（IEVT＝4）問題

・改良方法

固有値タイプがzone thickness searchの場合，正しく動作せず，原因も調査したが不明である。そこでこの場合，メッセージを出力してプログラムを終了させるようにした。

・修正サブルーチン

INPT11

(11) オリジナル版とベクトル化版のソースレベルの比較

次の3ケ所の違いがあったので，オリジナル版にあわせた。

① フラックスが負かどうかの判定の誤り

ベクトル化版は，INNERルーチンの一部をSWEEPルーチンとして書き直しているが，その書き直しの際，IF文（フラックスの負判定）を誤って書いた。

・修正サブルーチン

SWEEP1

SWEEP2

SWEEP3

SWEEP4

② フラックスの合計値を計算する中間変数の相違

オリジナル版では，一時的な変数（TT）を用いて，下で使う変数（T）の値をこわしていない。ベクトル化版では，下で使う変数（T）に値を入れており，Tの値をこわしている。

・修正サブルーチン

REBAL

③ DUMPRDルーチンの未定義変数

ベクトル化版では，DUMPRDルーチンの作業領域を計算する部分のIF文中の変数が未定義である。

・修正サブルーチン

DUMPRD

(12) I2の値がこわされる原因の調査

VINOUTルーチンで新しく定義した変数の中にI2という変数があり，中性子束の出力を制御する変数I2の値がこわされていた。次のように修正する。

```
      COMMON /FWBGN1/ IDUSE(18),LAST,LASTEC,IGCDMP,IPSO,LTSO,IPFL,LTFL,
CNN  1IPFX,LTFX,LXFX,IPXS,IPXSCT,LTXS,LTOXS,LTAXS,IPQS,LTQS,IEREC,I2,I4,
CNN  2I6,ISPANQ,IPHAF,IPVAF,LTHAF,LTVAF,IFO
     1IPFX,LTFX,LXFX,IPXS,IPXSCT,LTXS,LTOXS,LTAXS,IPQS,LTQS,IEREC,I2WK,
     2I4,I6,ISPANQ,IPHAF,IPVAF,LTHAF,LTVAF,IFO
```

・修正サブルーチン

VINOUT

3.7 議　論

3.7.1 領域の増加

ベクトル化版では，プログラムをベクトル化するために作業用配列をいくつか作ったので領域が増加している。増加領域の計算式を式（3.2）に示す。

$$
\begin{aligned}
\text{増加領域} = {} & 3\times IK\times JK\times KM+KIM+KJM+(2\times IK+6\times JK)\times KM+ \\
& 2\times KM\times JK\times(IK+JK+KM-2)+6\times(IK+JK+KM-2)+ \\
& 8\times(IK+JK+KM-1)+4\times((JK+1)\times KM+100)+ \\
& (1+4\times KM)\times IK\times JK\times KM+(IK\times JK)+IK\times KM+ \\
& 3\times(IK+KM-1)\times JK\times KM+(IK+KM)\times JK\times KM+6\times KM+ \\
& 4\times NM\times IK\times JK\times KM+2\times JK+4\times IK+1)\times 2+IK+JK+1 \qquad (3.2)
\end{aligned}
$$

ここで，

IK　：total number of radial fine-mesh intervals

JK　：total number of axial fine-mesh intervals

KM　：number of directions per octant

KIM　：number of radial coarse-mesh intervals

KJM　：number of axial coarse-mesh intervals

NM　：number of anisotropic components of flux

MM = ((ISN*(ISN+2))/8

NM = ((ISCT+1)*(ISCT+2))/2

ISN　：order of S_n

ISCT：scattering order

上式は計算が煩雑なので，通常は次に示す簡便式を用いると良い。

簡便式：$2\times JK\times KM\times(8\times IK+6\times NM\times IK+6\times KM+JK+10)$

Table 3.1 Input data used for dynamic profile analysis factor on TWOTRAN-II

Geometry	r − z
Number of spatial mesh	28 × 32
Number of angular ordinates (S_n)	10 (S_8)
Number of flux moments (P_l)	1 (P_0)
Number of energy groups	70

Table 3.2 Time distribution for TWOTRAN-II Case-1

サブルーチン名	時間比率 (%)
OUT	48.5
IN	44.9
INNER	4.7
OUTER	0.9
ECRD	0.4
SETBC	0.4
FIXUP	0.1
その他	0.1

Table 3.3 Input data of test run for evaluation of calculation results on TWOTRAN-II

ケース No.	計算項目	幾何形状	x方向 メッシュサイズ	y方向 メッシュサイズ	S_n 次 数	群 数
1	K_{eff}	x－y	41	46	8	9
2	〃	r－z	20	20	8	6
3	〃	r－θ	7	7	4	3
4	〃	〃	7	7	4	3
5	〃	r－z	7	7	4	3
6	〃	〃	7	7	4	3
7	〃	〃	7	7	4	3
8	〃	x－y	7	7	4	3
9	〃	〃	7	7	4	3
10	〃	〃	7	7	4	3
11	〃	r－z	7	14	4	3
12	〃	x－y	41	47	4	3
13	〃	r－θ	7	7	4	3
14	*Q	x－y	12	10	4	1
15	〃	〃	12	10	4	1
16	〃	〃	12	10	4	1
17	〃	r－z	12	10	4	1
18	〃	〃	12	10	4	1
19	〃	〃	12	10	4	1
20	〃	〃	12	10	4	1
21	〃	x－y	10	12	4	1
22	〃	〃	10	12	4	1
23	〃	r－θ	7	7	4	3
24	〃	r－z	42	10	2	9
25	〃	〃	20	20	8	1

* inhomogeneous source

Table 3.4 Evaluation of calculation results for TWOTRAN-II

ケース No.	固有値 (K_{eff}) オリジナル版 (A)	ベクトル化版 (B)	誤差 $\frac{(B)-(A)}{(A)}$	代表的な中性子束* オリジナル版 (A)	ベクトル化版 (B)	誤差 $\frac{(B)-(A)}{(A)}$
1	1.020668	1.020668	0.0	2.15037×10^{-5}	2.15029×10^{-5}	-3.72×10^{-5}
2	1.00000×10^{-2}	1.00000×10^{-2}	0.0	1.57768	1.57753	-9.51×10^{-5}
3	0.999441	0.999411	-3.00×10^{-5}	0.250986	0.250954	-1.27×10^{-4}
4	1.005757	1.005751	-5.97×10^{-6}	0.253427	0.253428	3.95×10^{-6}
5	0.948529	0.948529	0.0	6.00229×10^{-4}	6.00228×10^{-4}	1.67×10^{-6}
6	1.085562	1.085562	0.0	3.51214×10^{-3}	3.51213×10^{-3}	2.85×10^{-6}
7	1.115761	1.115762	8.96×10^{-7}	7.88291×10^{-4}	7.88293×10^{-4}	2.54×10^{-6}
8	0.997546	0.997537	-9.02×10^{-6}	1.29454×10^{-2}	1.29507×10^{-2}	4.09×10^{-4}
9	0.867611	0.867609	-2.31×10^{-6}	8.97283×10^{-3}	8.97282×10^{-3}	-1.11×10^{-6}
10	1.028896	1.028897	9.72×10^{-7}	1.24287×10^{-2}	1.24288×10^{-2}	8.05×10^{-6}
11	0.948539	0.948539	0.0	6.00234×10^{-4}	6.00240×10^{-4}	1.00×10^{-5}
12	1.032483	1.032395	-8.52×10^{-5}	0.121053	0.121051	-1.65×10^{-5}
13	0.999441	0.999411	-3.00×10^{-5}	0.373786	0.373416	-9.90×10^{-4}
14	0.0	0.0	0.0	0.693894	0.693892	-2.88×10^{-6}
15	0.0	0.0	0.0	0.710123	0.710120	-4.22×10^{-6}
16	0.0	0.0	0.0	0.710575	0.710573	-2.81×10^{-6}
17	0.0	0.0	0.0	6.46258×10^{-2}	6.46258×10^{-2}	0.0
18	0.0	0.0	0.0	6.46258×10^{-2}	6.46258×10^{-2}	0.0
19	0.0	0.0	0.0	6.46258×10^{-2}	6.46258×10^{-2}	0.0
20	0.0	0.0	0.0	0.250226	0.250227	4.00×10^{-6}
21	0.0	0.0	0.0	0.195900	0.195899	-5.10×10^{-6}
22	0.0	0.0	0.0	0.693898	0.693893	-7.10×10^{-6}
23	0.0	0.0	0.0	1.18645×10^{-4}	1.18646×10^{-4}	8.43×10^{-6}
24	0.0	0.0	0.0	0.121053	0.121051	-1.65×10^{-5}
25	0.0	0.0	0.0	6.73507×10^{-4}	6.73508×10^{-4}	1.48×10^{-6}

* 1群の1番大きな値

Table 3.5 Input data of test run for evaluation of vectorization effect on TWOTRAN-II

	ケース1	ケース2	ケース3	ケース4
形　状	r − z	r − z	r − z	r − z
メッシュサイズ	61×40	31×31	28×32	20×20
計算タイプ	P_0, S_8	P_0, S_8	P_0, S_8	P_0, S_8
群　数	70	70	70	6
固有値収束条件	10^{-3}	10^{-4}	10^{-4}	10^{-4}

Table 3.6 Vectorization effect for TWOTRAN-II

	ケース1		ケース2		ケース3		クース4	
	オリジナル版	ベクトル化版	オリジナル版	ベクトル化版	オリジナル版	ベクトル化版	オリジナル版	ベクトル化版
CPU 時間	74分13秒	17分37秒	69分27秒	18分52秒	57分28秒	16分20秒	57秒	20秒
VU 時間	0秒	12分05秒	0秒	13分34秒	0秒	11分38秒	0秒	13秒
倍　率	1.0	4.21	1.0	3.68	1.0	3.52	1.0	2.85
領域 (KB)	1972	5126	948	4874	869	2241	525	1725
I/O回数	2412	685	852	285	629	235	1252	1540

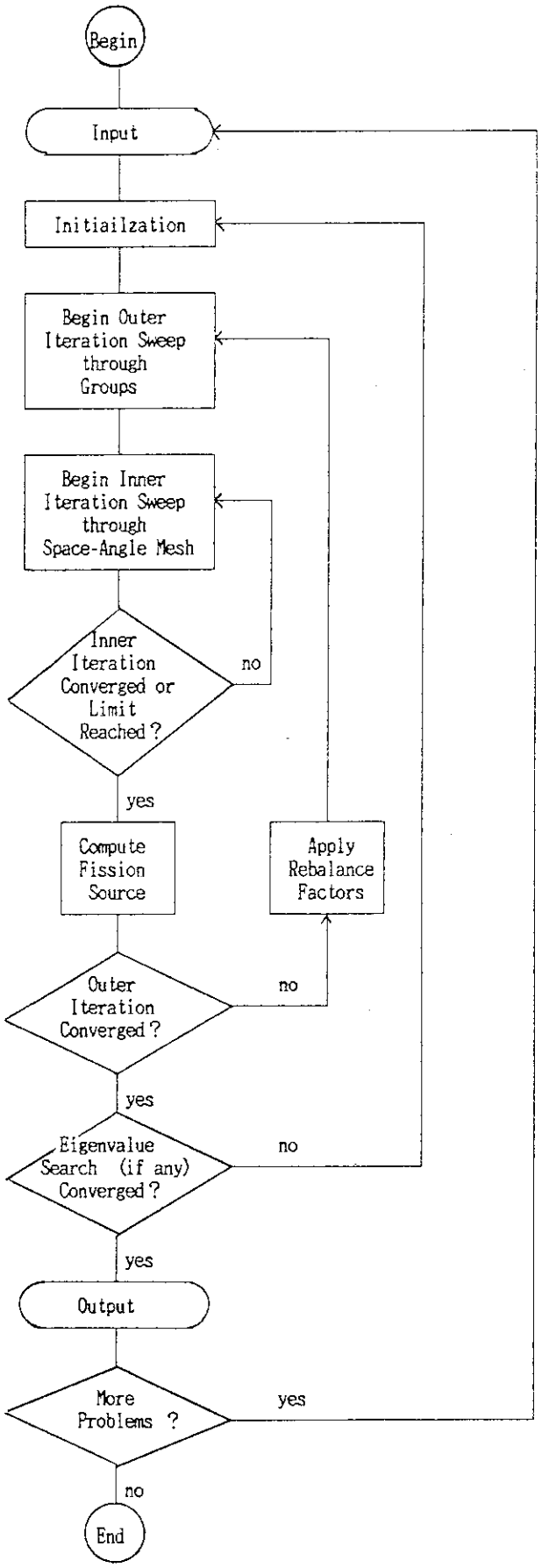


Fig. 3.1 Simplified logical flow diagram for TWOTRAN-II

for j = 1, jmax

for i = 1, imax

for m = mmax, 1, −1

$$\Phi_{i,j,m} = \alpha + \beta\Phi_{i-\frac{1}{2},j,m} + \gamma\Phi_{i,j-\frac{1}{2},m} + \delta\Phi_{i,j,m+\frac{1}{2}}$$

$$\Phi_{i,j,m-\frac{1}{2}} = 2\Phi_{i,j,m} - \Phi_{i,j,m+\frac{1}{2}}$$

$$\Phi_{i+\frac{1}{2},j,m} = 2\Phi_{i,j,m} - \Phi_{i-\frac{1}{2},j,m}$$

$$\Phi_{i,j+\frac{1}{2},m} = 2\Phi_{i,j,m} - \Phi_{i,j-\frac{1}{2},m}$$

where m : angular ordinates, i, j : spatial mesh,

Φ : perticle flux,

and $\alpha, \beta, \gamma, \delta$: some constant coefficients.

Fig. 3.2 The typewriter scanning method for the forward substitution of the inner iteration for TWOTRAN-II

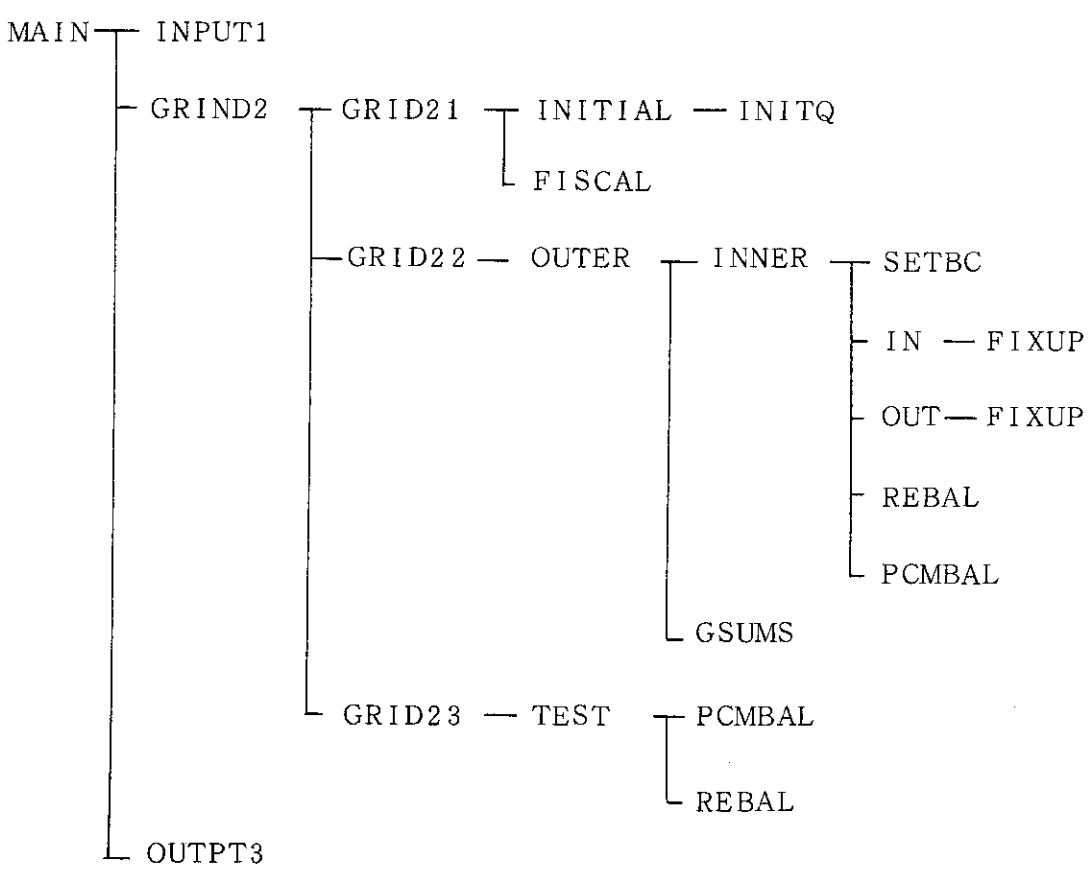


Fig. 3.3 Main tree structure in TWOTRAN-II

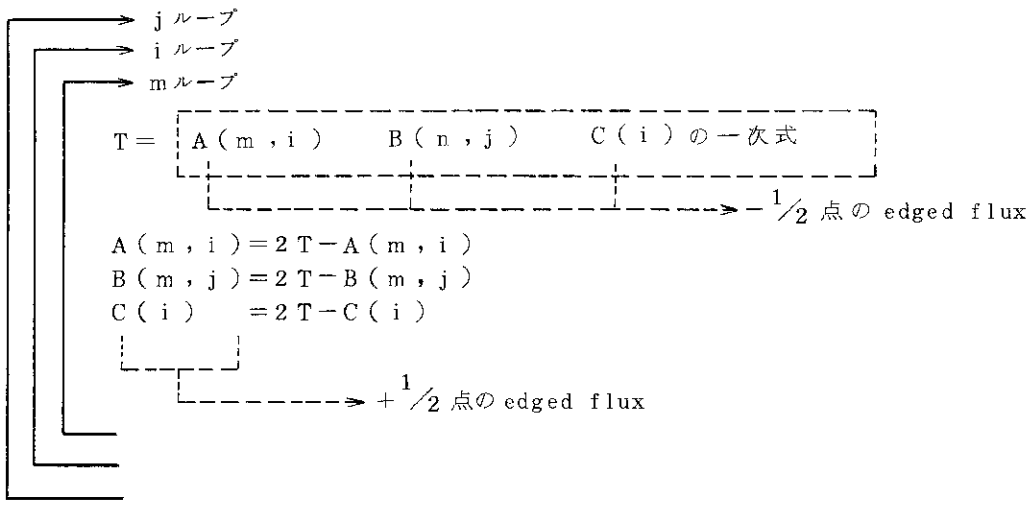


Fig. 3.4 Loop structure of the inner iteration for TWOTRAN-II

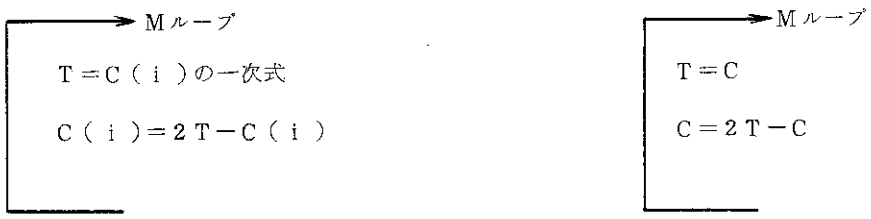


Fig. 3.5 Recurrence of variable C for TWOTRAN-II

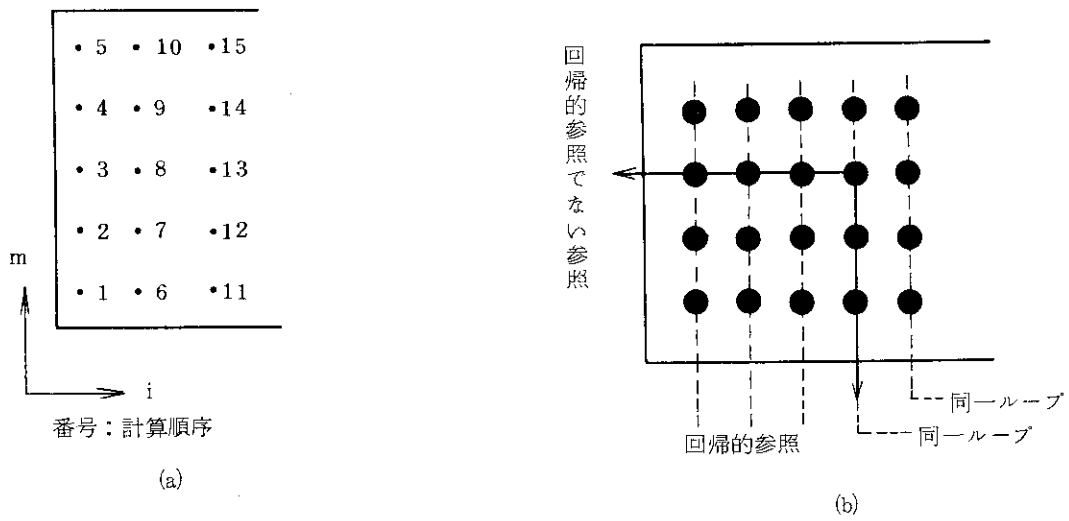


Fig. 3.6 Recurrence of X-Y plane for TWOTRAN-II

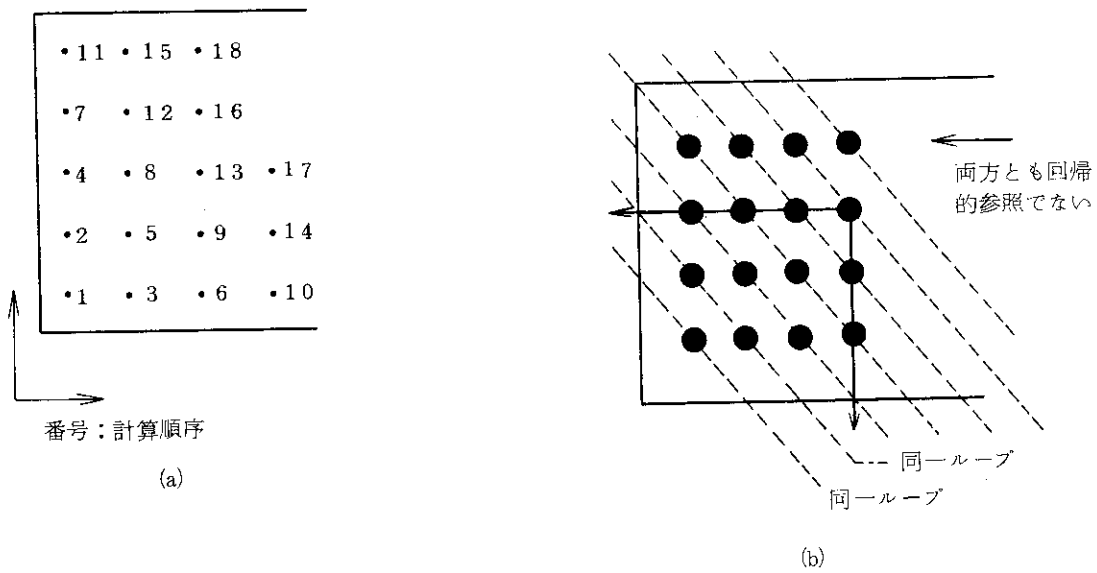


Fig. 3.7 Hyper-plane method of X-Y plane for TWOTRAN-II

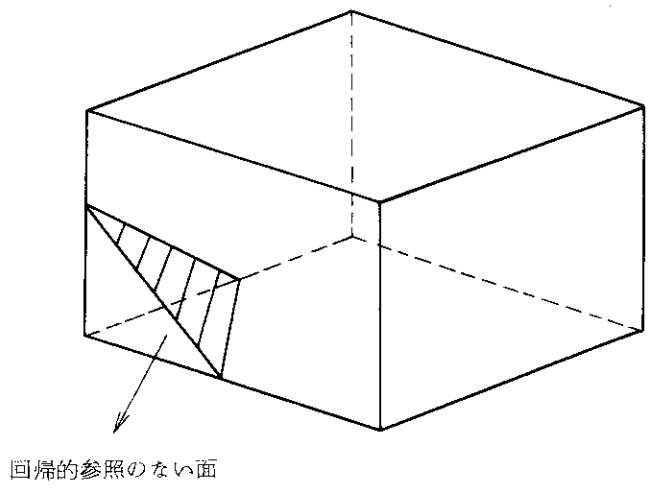


Fig. 3.8 Hyper-plane in (i,j,m) plane

Lの内容

I	1	2	3	4	5	6	7	8	9	10
L (I)	1	1	1	1	1	2	2	2	2	2

```
I = 1 , 10
  M = 1 , 10
  A ( L (I) ) = A ( L (I) ) + B ( M , I ) * CONST
```

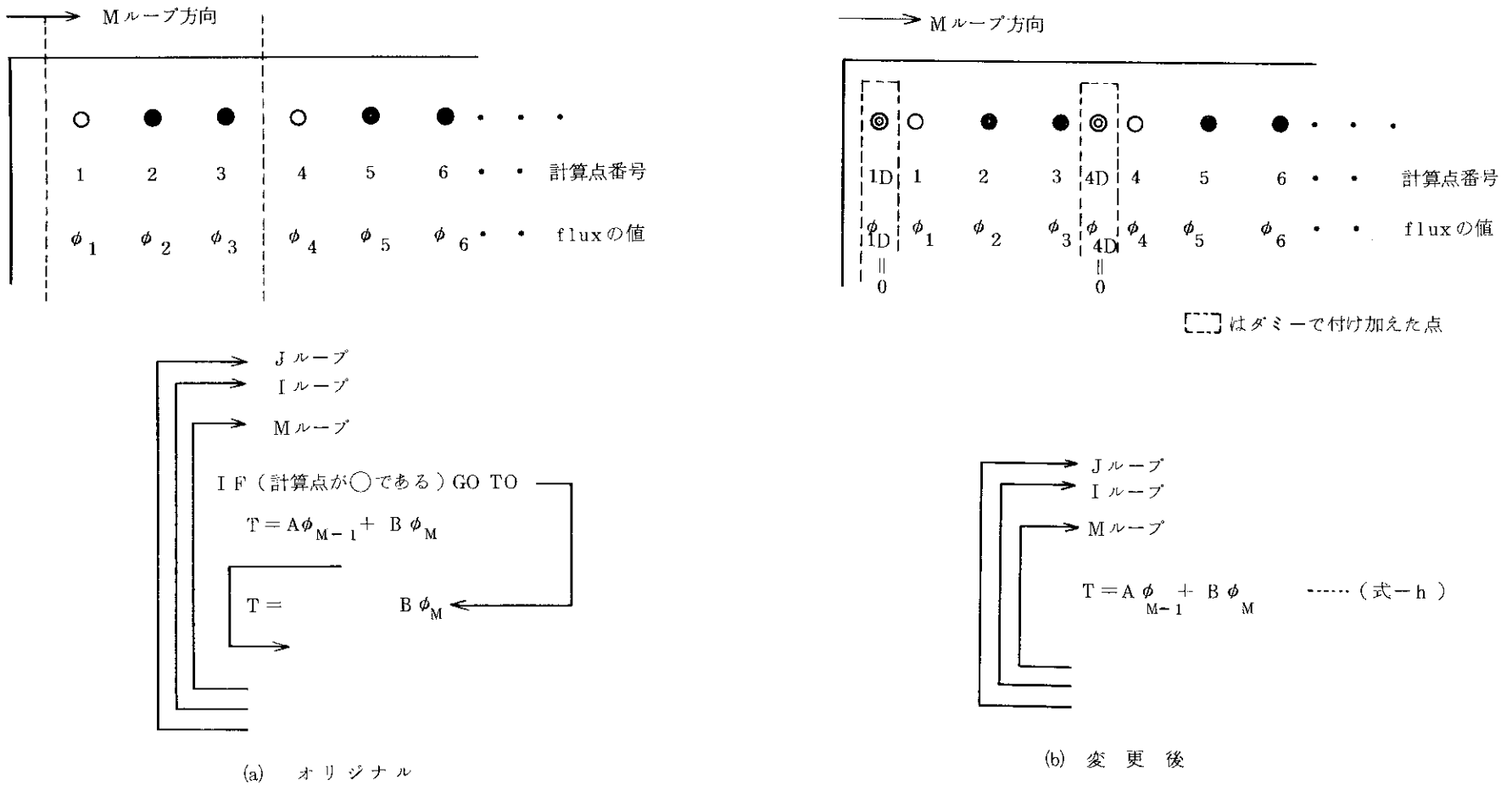
オリジナル
コーディング

⇩

```
EQUIVALENCE(BEQ(1) , B ( 1 ,1 ) )
I = 1 , 2
  IM = 1 , 50
  A(I) = A(I) + BEQ ( ( I-1 ) * 50+IM) * CONST
```

チューニング後
コーディング

Fig. 3.9 Vectorization method for TWOTRAN-II



(a) オリジナル

(b) 変更後

Fig. 3.10 Removal of IF-sentence using dummy area for TWOTRAN-II

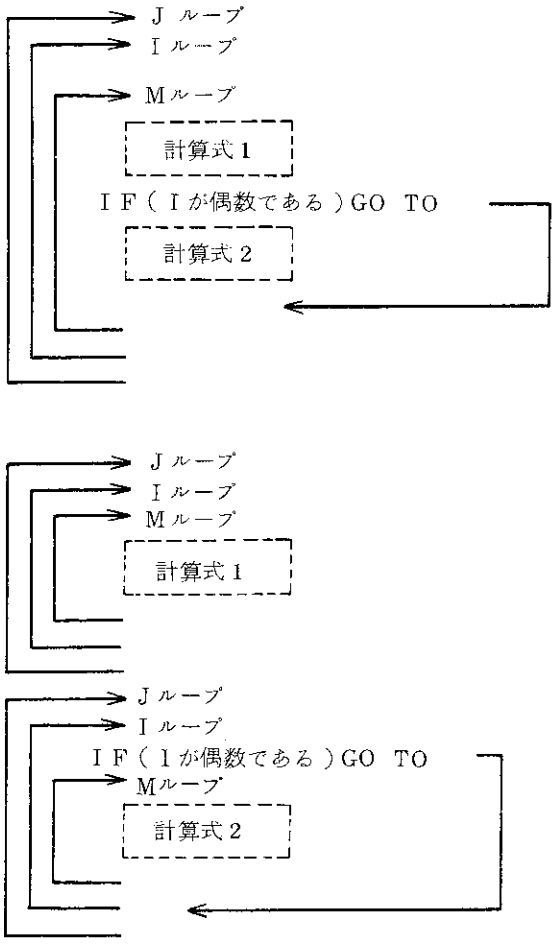


Fig. 3.11 Move of IF-sentence using DO-loop division for TWOTRAN-II

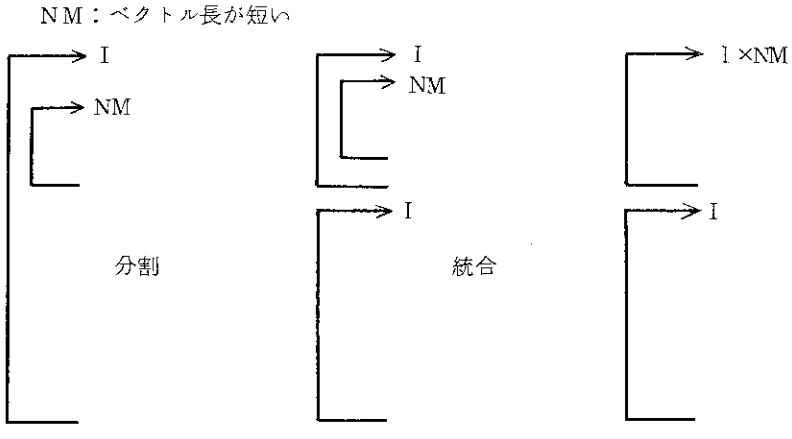


Fig. 3.12 Extension of vector length using DO-loop division and unification for TWOTRAN-II

4. SRACシステム

4.1 システムの概要

SRAC[7]（熱中性子炉体系標準核設計コード）システムは，熱中性子炉体系の設計計算コードシステムの標準化をめざして，原研で作成，整備されてきたコードシステムである。

本システムは，Table 4.1に示すサブコードより成る。主なサブコードとしては，種々の幾何形状に対する衝突確率計算コード群，S_n輸送計算コード，拡散コード，反応率計算コード，格子燃焼計算コード，超詳細共鳴吸収計算コード，等がある。

SRACシステムのフローダイアグラムをFig. 4.1に示す。

4.2 ベクトル化指針

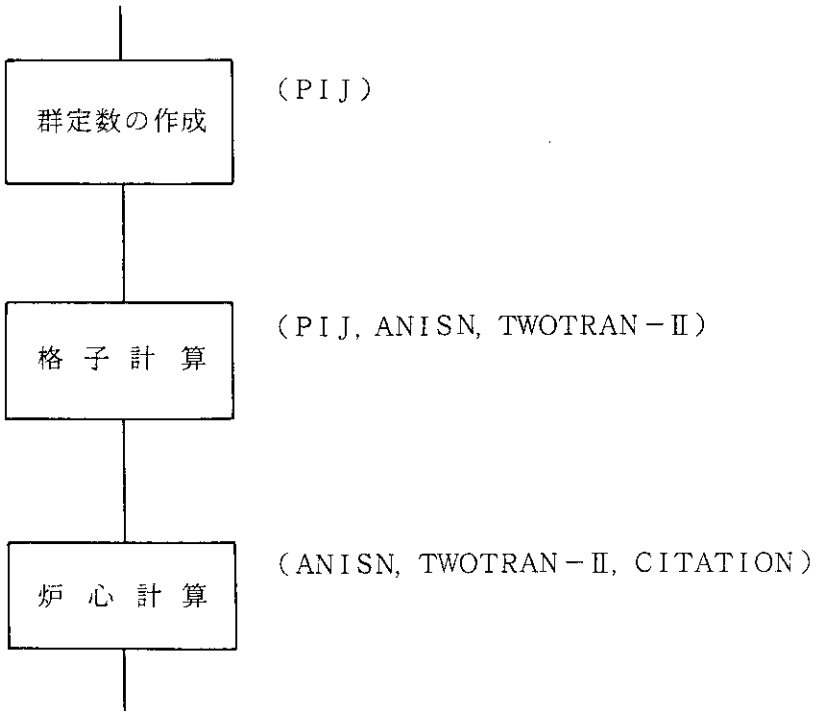
SRACシステムは，多くのコードの集まりであるので実行ケースにより通るコードが異なる。それらの中で比較的実行時間が多くかかるコードを選びベクトル化した。ベクトル化を実施したコードは次の3コードである。

(1) CITATIONコード部分（Table 4.1のCIT2，Fig. 4.1のDIFFUSION）

(2) TWOTRANコード部分（Table 4.1のTWTRN2，Fig. 4.1のSN）

(3) 衝突確率計算部分（Table 4.1，Fig. 4.1のPIJ2）

SRACシステム中の上記コードの位置づけを次に示す。



4.3 CITATIONコード部分のベクトル化

4.3.1 コードの動的挙動分析

CITATIONコード部分を炉心計算として使用するデータ（Table 4.2参照）を用いてCPU時間を測定した。これによると各データ共CPU時間の90%以上をCITATION部分が占めている。そこで，この種のデータの場合，CITATION部分をベクトル化，高速化することにより，全体の高速化がはかれる。

4.3.2 ベクトル化方法

CITATIONコードのベクトル化技法と高速化技法をSRACシステムのCITATION部分に適用した。具体的な方法は，2.3節を参照のこと。

4.3.3 計算結果の評価

ベクトル化したコードの計算結果を評価するために，2次元形状について1ケース，3次元形状について1ケースの試計算を実施した。各ケースの入力条件をTable 4.3に示す。

計算結果をTable 4.4に示す。これによるとケース2のLeakageを除いて10^{-4}以下の相対誤差であり，ケース2のLeakageについても1.6×10^{-4}の相対誤差にとどまっている。したがって中性子束及びK_{eff}の収束判定条件が両ケース共10^{-4}と定義していることから考えて妥当な値とみなすことができる。

4.3.4 ベクトル化効果

ここで計算の対象とした入力条件は，4.3.3節で示した2ケースである。

計算時間は，CITATIONコードと同様に内側反復回数と初期加速係数に依存するので，ここでは，最も計算時間の短い内側反復回数と初期加速係数を指定した時の結果を比較している。

Table 4.5に計算時間，外側反復回数等を示す。これによるとベクトル化版は，オリジナル版と比較してケース1で2.1倍，ケース2で8.6倍に高速化された。また，ベクトル化率は，ケース1で87.3%，ケース2で93.7%である。

4.3.5 議　論

(1) 内側反復回数及び初期加速係数

計算時間は，CITATIONコードと同様に，入力データで指定する内側反復回数と初期加速係数の値に依存して変化する。SRACシステムでは，入力データで内側反復回数を指定しなかった場合1になり，初期加速係数を指定しなかった場合システムが自動的に決定する。ベクトル化版を用いて内側反復回数及び初期加速係数をパラメータサーベイした結果をTable 4.6に示す。これによると2次元データでは，内側反復回数5回，初期加速係数1.4を用いた場合，計算時間が最も短くなった。

(2) 領域の増加

SRACシステムにおいてもCITATIONコードと同様にベクトル化版では領域が増加する。増加する領域はCITATIONコードと同じである（Table 2.28 参照）。

4.4 TWOTRAN部分のベクトル化

4.4.1 コードの動的挙動分析

TWOTRANコード部分を炉心計算に使用するデータ（Table 4.7 参照）を用いてCPU時間を測定した。

これによるとCPU時間の74％をTWOTRAN部分が占めている。そこで，TWOTRAN部分をベクトル化，高速化することにより高速化がはかれる。

4.4.2 ベクトル化方法

TWOTRANコードのベクトル化技法と高速化技法をSRACシステムのTWORAN部分に適用した。具体的な方法は，3.3節を参照のこと。

4.4.3 計算結果の評価

ベクトル化したコードの計算結果を評価するために試計算を実施した。入力条件をTable 4.8に示す。

計算結果をTable 4.9に示す。これによると中性子バランスは10^{-4}のオーダーであり，固有値の誤差は10^{-5}のオーダーである。収束条件が，中性子バランス，固有値とも10^{-3}であることから，十分な精度で一致していると判断できる。

4.4.4 ベクトル化効果

ここで計算の対象とした問題は，4.4.3節で示した1ケースである。

Table 4.10に計算時間，倍率，外側反復回数を示す。Table 4.10によるとベクトル化版はオリジナル版の1.8倍に高速化された。また，TWOTRAN部分のみを見ると2.5倍に高速化された。ベクトル化率は55.8％である。

4.5 衝突確率法計算部分のベクトル化

衝突確率法計算部分は，衝突確率法を数値積分で計算するステップと，これを用いて連立一次方程式を解いて中性子束分布を求めるステップに分かれる。しかし，計算時間のほとんどを数値積分計算部分が占める。したがって，数値積分を行うサブルーチンDELTについてベクトル化を実施するのが良い。このベクトル化については，原子炉システム研究室の筒井恒夫氏が既に行っていた。このベクトル化版ではベクトル長の大小によりベクトル化ルーチンとスカラルーチンを切り換えて使用している。今回の作業では，そのベクトル化版を一部整備し，テストした。

4.5.1 コードの動的挙動分析

Table 4.11 は,衝突確率法を計算するデータの実行時間をタイマーを挿入して計測した結果を示している。これを見てわかる様にサブルーチンDELTが全体の計算時間の93.7%を占めている。従って，サブルーチンDELTをベクトル化，高速化するのが全体を高速化することにつながる。

4.5.2 ベクトル化の方法

DELTルーチンのベクトル化及び高速化手法を以下に記す。

(1) ベクトル化手法

① DOループを分割する。

② 分割したDOループを判定文の中に入れる。

③ FUNCTION文を展開する。

④ DOループからの飛び出しを抑止し，フラグを用いて判定する。

(2) 高速化手法

① ベクトル長に応じて，ベクトル処理とスカラー処理を切り換える。ベクトル化版では，5で切り換えを行っている。

Fig. 4.2 にサブルーチンDELTのオリジナル版のフローを，Fig. 4.3にベクトル化版のフローを示す。

4.5.3 計算結果の評価

ベクトル化したコードの正統性を検証するために，2ケースの試計算を実施した。

2ケースの試計算データをTable 4.12 に，試計算結果をTable 4.13 及びTable 4.14 に示す。

各ケースのFinal Fission Source Normalizationを比較すると，各ケース共最後の桁が1違うだけであり，十分な精度で一致していると判断できる。

4.5.4 ベクトル化効果

試計算を実施したデータ2ケースについてのベクトル化効果をTable 4.15 及びTable 4.16 に示す。

4.5.5 議　論

今回のテストデータでは主要ループのベクトル長は51であるが，速度向上は2.3倍にとどまっている。この理由としてPの計算が回帰計算であるためスカラー計算している点がある。

これは，2.3(2)と同様の手法を用いてベクトル化することが可能である。しかし，その際Pの配列が2次元から3次元になるので領域が増える。したがって，今後この部分をベクトル化する場合，領域の増加に注意してベクトル化を進める必要がある。

Table 4.1 Sub-codes table for SRAC system

INPUT1	: Read the control data and energy structure infomation
USER. FASTLIB	: Compose the user fast neutron library on FASTU
USER. THERMAL. LIB	: Compose the user fast neutron library on THERMALU
PIJ. INPUT	: Read the input for the collision probability method and compose the trace table on FT81, 82, 83
PLOT. GEOMETRY	: Figure out the cell geometry and region number map
SN-INPUT	: Read the input for ANISN and/or TWOTRAN
DIFFUSION-INPUT	: Read the input for TUD and/or CITATION
REACT. IN	: Read in the input for reaction rate calculation
INPUT2	: Read in the material specification
BURN. IN	: Read in the input for burn-up calculation and compose the case-dependent nuclide and chain tables
MACRO-FAST	: Compose macroscopic cross section sets with self shielding factor in fast neutron energy range
SHIELD	: Calc. self shielding factor by table look-up on NRA
PIJ2	: Calculate collision probabilities
MACRO. THERMAL	: Compose macroscopic cross section sets with self shielding factor in thermal neutron energy range
GAM-P1B1	: Modify transport cross sections and diffusion coefficients in the macroscopic cross section sets prepared by MACRO. FAST and MACRO. THERMAL by P1 or B1 approximation for homogeneous material and for component materials in a cell. In the latter case the neutron flux and current are calculated in a homogenized material.
IR-METHOD	: Modify capture cross section of fertile material by table-look-up on IRA
MCROSS	: Compose the user's resonance neutron cross section file
PEACO	: Calculate ultra-fine neutron spectrum in multiregion cell in Resonance II energy range by collision probability method, and modify the absorption and fission cross sections of resonant material
MIX. X-SECTION	: Homogenize the macroscopic cross section by X-region
PIJ3	: Solve linear equations by S. O. R for collision probability method
ANISN2	: Solve one-dimensional Sn equations
TWTRN2	: Solve two-dimensional Sn equations
TUD2	: Solve one-dimensional diffusion theory equations
CIT2	: Solve multi-dimensional diffusion theory equations

Table 4.1 (Continued)

HOMOSP	:	Solve a bare reactor equation by P1 or B1 approximation
CONDENSE	:	Collapse the energy structure of the macroscopic cross sections to get few group cross sections in whole energy range
CONCAT	:	Concatenate multi-group constants separately stored for fast and thermal ranges into whole energy range ca;c.
BURNUP	:	Calculate the change of nuclide concentrations during burn-up in a cell
CVMACT	:	Convert the format of the macroscopic cross sections into the original CITATION format
REACT	:	Calculate the reaction rate for the specified detector with or without the filter, the spectrum parameters δ_{25}, ρ_{28}, δ_{28}, C^*, and the convertion rate.

Table 4.2 CPU time ratio of CITATION in SRAC system

	ケース1	ケース2
次　　元	2次元	3次元
形　　状	r－z	x－y－z
メッシュ数	50×35	37×37×35
群　　数	26	4
全CPU時間	96.47秒	392.70秒
CITATION部分CPU時間	87.68秒	374.17秒
CITATION部分の割合（%）	90.9 %	95.3 %

Table 4.3 Input data of test run for evaluation of calculation results on CITATION part of SRAC

	ケース1	ケース2
次　元	2次元	3次元
形　状	r－z	x－y－z
群　数	26	4
メッシュ数	50×35	37×37×35
オリジナル版　初期加速係数	無指定	無指定
ベクトル化版　初期加速係数	1.4	1.8
オリジナル版　内側反復回数	1	1
ベクトル化版　内側反復回数	5	4

Table 4.4 Evaluation of calculation results for CITATION part of SRAC

ケース	項　目	オリジナル版 (A)	ベクトル化版 (B)	相対誤差 $\frac{(B)-(A)}{(A)}$
1	Leakage	2.57723×10^{14}	2.57741×10^{14}	7.0×10^{-5}
	Total Losses	7.77045×10^{16}	7.77045×10^{16}	0.0
	Total Productions	7.80602×10^{16}	7.80604×10^{16}	2.6×10^{-6}
	K_{eff}	1.0045748	1.0045748	0.0
2	Leakage	5.03899×10^{13}	5.03979×10^{13}	1.6×10^{-4}
	Total Losses	7.67524×10^{16}	7.67515×10^{16}	-1.2×10^{-5}
	Total Productions	7.80516×10^{16}	7.80516×10^{16}	0.0
	K_{eff}	1.0169096	1.0169163	6.6×10^{-6}

Table 4.5 Vectorization effect for CITATION part of SRAC

	ケース 1		ケース 2	
	オリジナル版 スカラ計算	ベクトル化版 ベクトル計算	オリジナル版 スカラ計算	ベクトル化版 ベクトル計算
内側反復回数	1	5	1	4
初期加速係数	無指定	1.4	無指定	1.8
外側反復回数	127	155	150	47
CPU時間　(秒)	99.28	47.94	392.70	45.74
VU時間　(秒)	0.0	35.32	0.0	21.07
倍　率*	1.00	2.07	1.00	8.59
I/O回数	1244	1669	2291	1833
メモリ使用量 (KB)	5956	6064	5908	6216

* 倍率＝オリジナル版スカラ計算CPU時間／ベクトル化版ベクトル計算CPU時間

Table 4.6 The number of outer iterations and CPU time for SRAC vector version

ω \ INNER	3	4	5	6	7
1.2			163 49.38		
1.3		190 52.88	190 56.01	168 53.22	
1.4		172 49.25	155 47.94	166 53.00	
1.5	187 50.15	194 54.43	158 48.58	152 49.67	182 59.21
1.6			200 58.47		
1.7					
1.8		198 55.04			

上段：外側反復回数　　ω：初期加速係数

下段：CPU時間（秒）　　INNER：内側反復回数

Table 4.7 CPU time ratio of TWOTRAN in SRAC system

形　状	r − z
細メッシュ数	35×35
群　数	10
全CPU時間	768.73秒
TWOTRAN部分CPU時間	570.40秒
TWOTRAN部分の割合	74.2 %

Table 4.8 Input data of test run for evaluation of calculation results on TWOTRAN part of SRAC

形　状	r − z
群　数	10
粗メッシュ数	5×5
細メッシュ数	35×35
S_n 次　数	4

Table 4.9 Evaluation of calculation results for TWOTRAN part of SRAC

項　目	オリジナル版 (A)	ベクトル化版 (B)	相対誤差 $\frac{(B)-(A)}{(A)}$
中性子バランス	3.09646130×10^{-4}	3.16262245×10^{-4}	—
固　有　値	1.05245686	1.05240440	-4.98×10^{-5}

Table 4.10 Vectorization effect for TWOTRAN part of SRAC

	オリジナル版	ベクトル化版
外側反復回数	37	37
全CPU時間 (秒)	768.73	425.06
TWOTRAN部分CPU時間 (秒)	570.40	226.73
VU時間 (秒)	0.0	85.29
倍　率*	1.0	1.81
I／O回数	1238	1321
メモリ使用量 (KB)	5900	6264

* 倍率＝オリジナル版CPU時間／ベクトル化版CPU時間

Table 4.11 CPU time ratio of subroutine DELT in SRAC system

	CPU Time (sec)
DELT	785.38
そ の 他	52.59
全　体	837.97

Table 4.12 Input data of test run on PIJ-2 part of SRAC

	ケース 1	ケース 2
Geometry Type	Rectangular pillar divided by X−Y coordinates with pin rods on grid points	
Number of fast energy group	33	22
Number of thermal energy group	37	31

Table 4.13 Evaluation of calculation results for PIJ-2 Case-1 of SRAC

VERSION	Final fission source normalization
オリジナル（スカラー）	0.13522
VP版　（スカラー）	0.13523
VP版　（ベクター）	0.13523

Table 4.14 Evaluation of calculation results for PIJ-2 Case-2 of SRAC

VERSION	Final fission source normalization
オリジナルスカラ計算	1.7717
ベクトル化版スカラ計算	1.7718
ベクトル化版ベクトル計算	1.7718

Table 4.15 Vectorization effect for PIJ-2 Case-1 of SRAC

VERSION	スカラ計算	ベクトル計算		速度向上比	I/O	REGION
	CPU時間 (sec)	CPU時間 (sec)	VU時間 (sec)	(倍)	(回)	(KB)
オリジナル版	215.74	—	—	1.00	3319	6040
ベクトル化版	214.31	—	—	1.01	3317	6088
ベクトル化版	—	159.14	36.04	1.36	3317	6092

Table 4.16 Vectorization effect for PIJ-2 Case-2 of SRAC

VERSION	スカラ計算	ベクトル計算		速度向上比	I/O	REGION
	CPU時間 (sec)	CPU時間 (sec)	VU時間 (sec)	(倍)	(回)	(KB)
オリジナル版	874.79	—	—	1.00	9183	6000
ベクトル化版	815.62	—	—	1.07	9184	6048
ベクトル化版	—	467.05	135.29	1.87	9182	6052

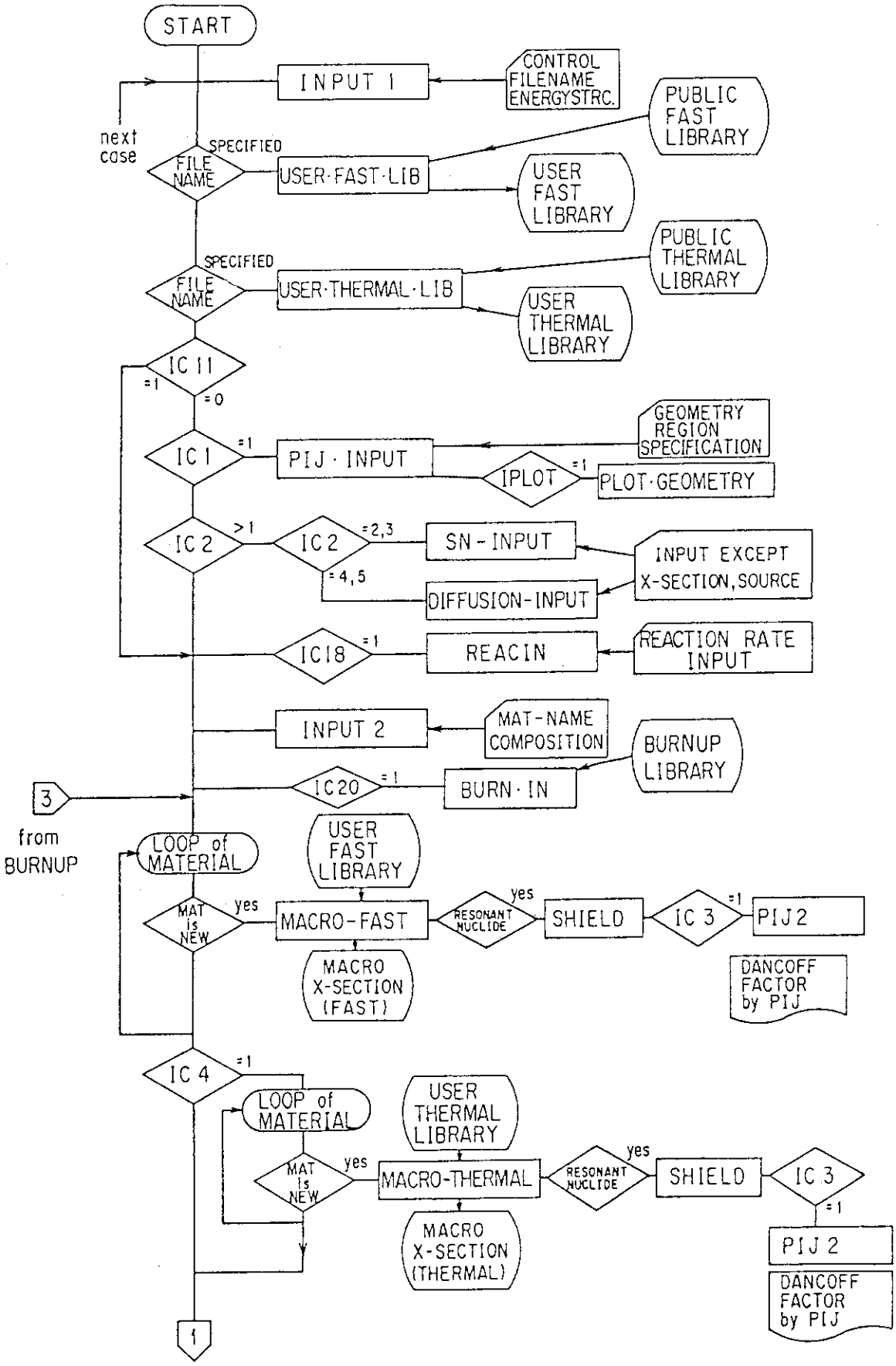


Fig. 4.1 Flow diagram of SRAC

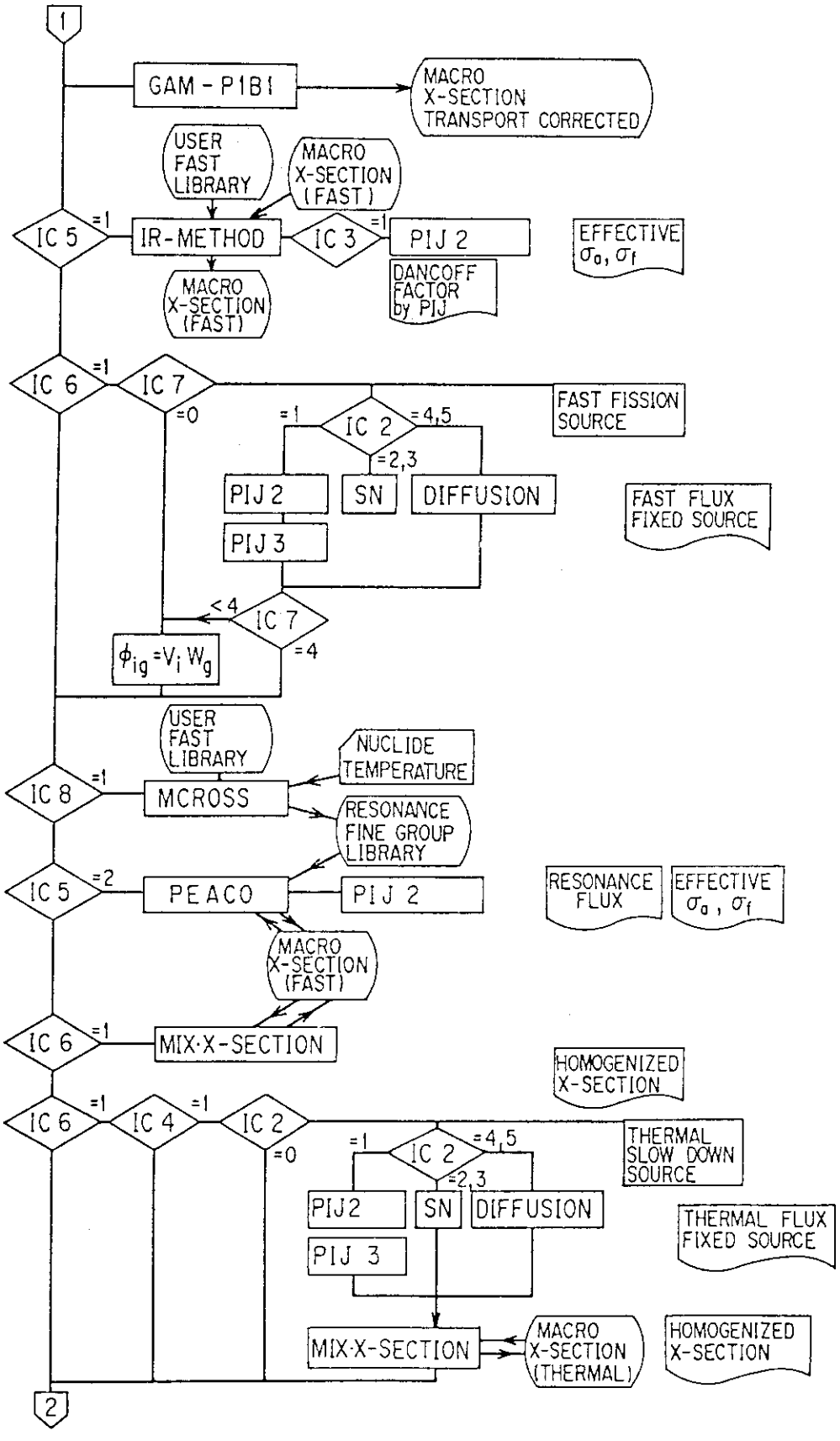


Fig. 4.1 (Continued)

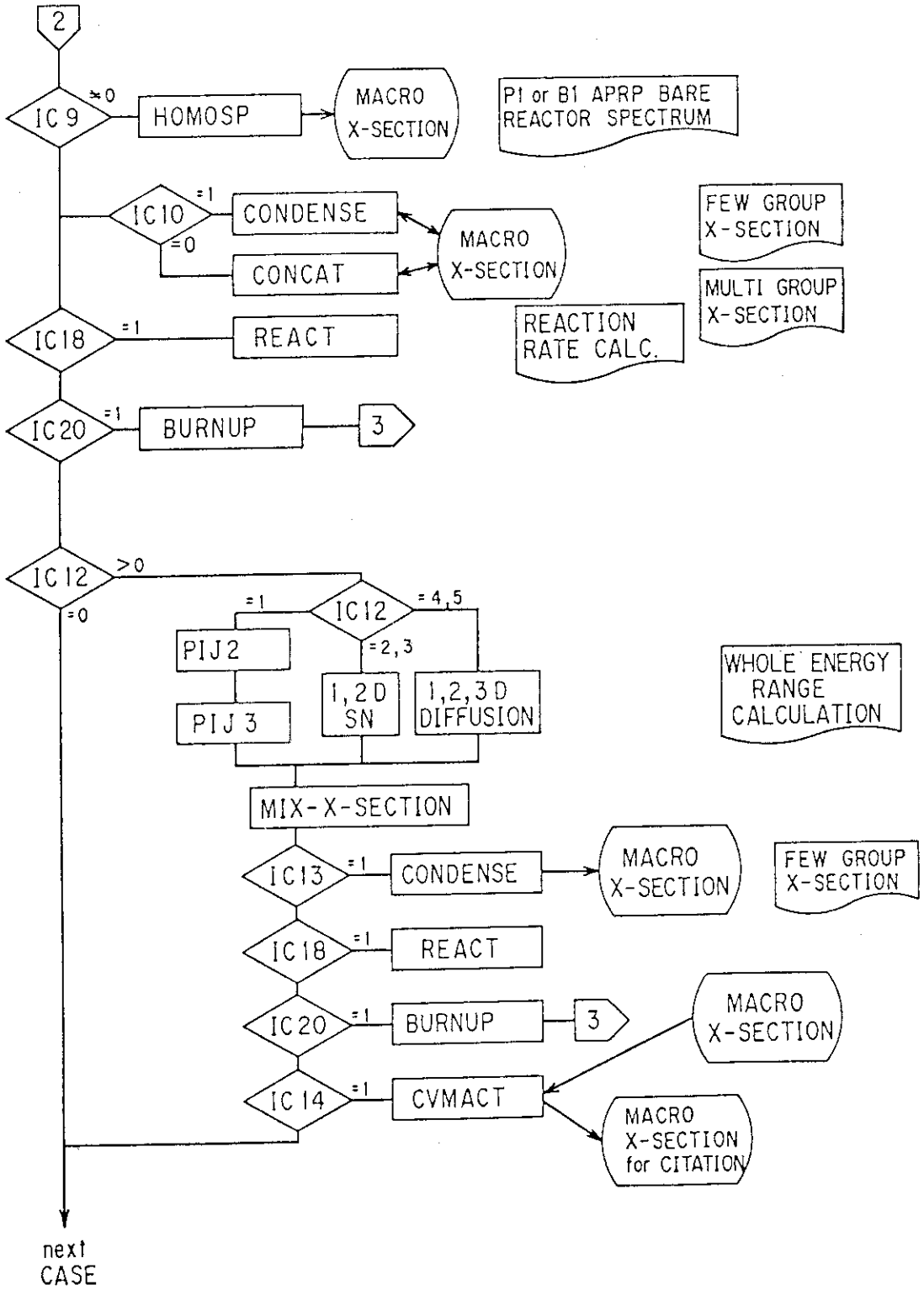


Fig. 4.1 (Continued)

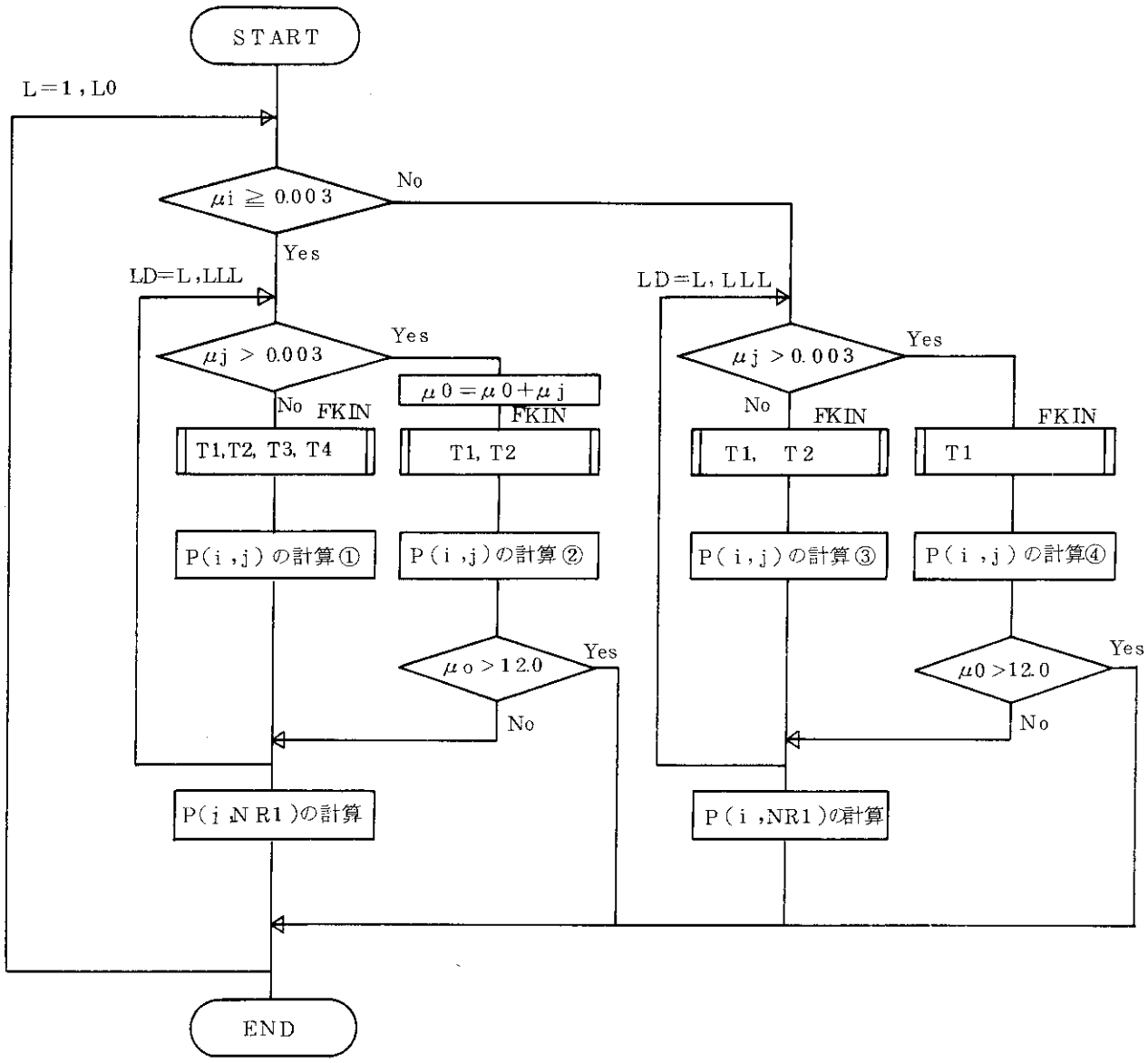


Fig. 4.2 Flow diagram of SRAC PIJ-2 original version

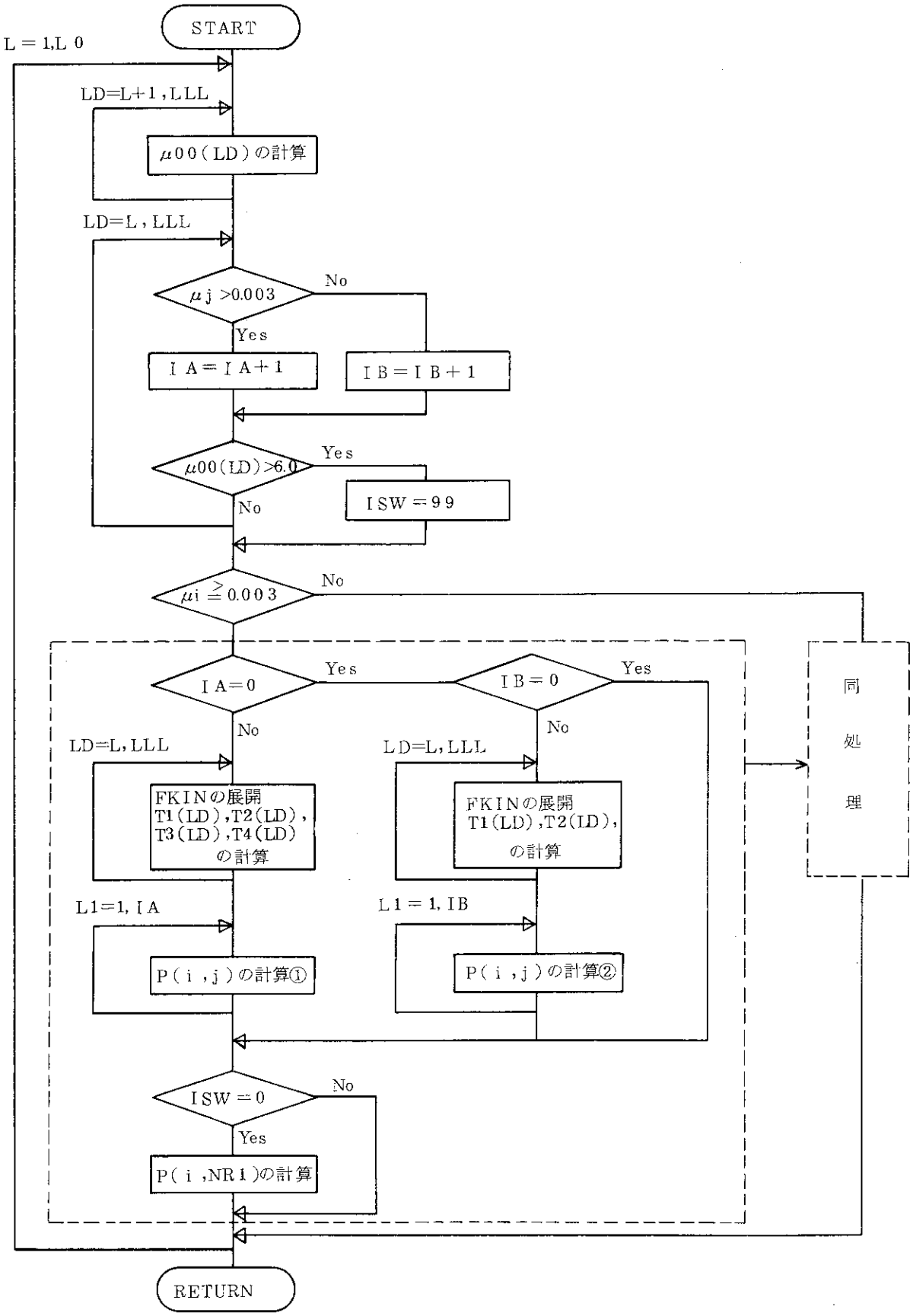


Fig. 4.3 Flow diagram of SRAC PIJ-2 vector version

5. COREBNコード

5.1 コードの概要

COREBNコード[7]は，1次元，2次元及び3次元の炉心燃焼計算と燃料管理を行うコードである。COREBNコードは，SRACシステムの補助プログラムとして原研で開発された。SRACシステムとCOREBNコードの関係をFig. 5.1に示す。

5.2 コードの動的挙動分析

COREBNコードの計算部分でもっとも計算時間が多くかかると予想されるのは，2次元及び3次元の拡散理論による炉心計算である。この部分は，CITATION部分がもっとも計算時間がかかっており，さらにCITATION部分の中の各サブルーチンの計算時間分布がオリジナルCITATIONコードと同じ傾向であることを確認するためにFORTUNEツール[3]による動的挙動分析を実施した。動的挙動分析に使用した入力データをTable 5.1に示す。

分析結果をTable 5.2，Table 5.3に示す。この結果によるとケース1でCITATION部分が90.2％であり，その中でもDNSD，FTRI，FINS，ABPRの時間比率が高い。これは，CITATIONコードの2次元三角形状問題と同じ傾向の時間比率である。ケース2ではCITATION部分が97.1％であり，その中でもKTRI，KNSD，KINS，KBPRの時間比率が高い。これは，CITATIONコードの3次元三角形状問題と同じ傾向の時間比率ということができる。したがって，COREBNコードを高速化するには，CITATIONコードに用いた手法が有効であると判断できる。

5.3 ベクトル化方法

CITATIONコードのベクトル化技法と高速化技法をCOREBNコードに適用した。具体的な手法は，2.3節を参照のこと。

5.4 計算結果の評価

ベクトル化したコードの計算結果を評価するために，2次元形状について1ケース，3次元形状について2ケースの試計算を実施した。各ケースの入力条件をTable 5.4に示す。

Table 5.5に示すように各ケースの結果を比較すると，もっとも誤差が蓄積すると考えられる燃焼末期において，固有値はすべて10^{-5}以下の相対誤差であり，収束条件(10^{-4})以下である。出力密度については，最大で1.3×10^{-3}であり，これも許容範囲内と考えられる。燃焼度，組成，転換比等の燃焼データは，いずれも各燃焼ステップにおけるゾーン平均（燃焼ノード）

出力密度を基に算出される。したがって，出力密度の一致は，必然的にその他の燃焼データの一致をも保証する。

5.5 ベクトル化効果

ここで計算の対象とした問題は，5.4節で示した3ケースである。Table 5.6にベクトル化効果を示す。これによると2次元のデータであるケース1で2.9倍，3次元のデータであるケース2，ケース3で10倍に高速化された。ケース1，ケース2，ケース3のベクトル化率はそれぞれ，75.8%，95.4%，95.9%である。

Table 5.1 Input data of test run on COREBN

	ケース 1	ケース 2
次　元	2	3
形　状	三 角	三 角
メッシュ数	34×17	34×17×32
群　数	7	8
ゾーン数	58	786
燃焼期間 (h)	1200, 1200, 2400, 3240	3600
初期加速係数	無指定	無指定
内側反復回数	1	1

Table 5.2 Time distribution for COREBN Case-1

サブルーチン	実行回数	時間比率 (%)
MAIN	1	—
CRBN	1	—
ADRSET	1	0.9
CRBN4	5	1.3
IPTM	5	0.4
MACR	5	1.2
CALR	5	—
EIGN*	5	—
CNST*	5	1.0
FLUX*	5	0.2
LOOP*	629	4.6
DNSD*	613	32.0
FTRI*	4291	27.7
FINS*	4291	18.2
ABPR*	613	5.9
RDUE*	5	0.6
OUTC	5	0.2
CNVCA1	5	4.1
CRBN6	1	0.1
CRBN61	101	0.6

* CITAION 部分

— 0.1 %未満

Table 5.3 Time distribution for COREBN Case-2

サブルーチン名	実行回数	時間比率(%)
MAIN	1	—
CRBN	1	—
ADRSET	1	0.2
CRBN4	1	0.2
IPTM	1	—
MACR	1	0.2
CALR	1	—
EIGN*	1	—
KLUX*	1	0.1
KOOP*	190	3.5
KNSD*	187	24.9
KTRI*	1496	39.6
KINS*	1496	23.3
KBPR*	187	4.6
KDUE*	1	0.4
KNST*	1	0.7
CNVCA1	1	1.9
CRBN6	1	—
CRBN61	101	0.3

* CITATION 部分

— 0.1%未満

Table 5.4 Input data of test run for evaluation of vectorization effect on COREBN

項目＼ケース	ケース 1	ケース 2	ケース 3
次元	2	3	3
形状	三角	三角	三角
群数	7	8	8
メッシュ数	34×17	34×17×32	34×17×32
ゾーン数	58	786	786
燃焼期間	1200, 1200, 2400, 3240 h	3600h×1step	600h×6 step

Table 5.5 Evaluation of calculation results for COREBN

ケース No.	燃焼末期の固有値			燃焼末期のゾーン平均出力密度*		
	オリジナル版 (A)	ベクトル化版 (B)	誤差 $\frac{(B)-(A)}{(A)}$	オリジナル版 (A)	ベクトル化版 (B)	誤差 $\frac{(B)-(A)}{(A)}$
1	1.0001802	1.0001783	1.90×10^{-6}	16.8267	16.8268	5.94×10^{-6}
2	1.0398102	1.0398092	9.62×10^{-7}	265.026	265.375	1.32×10^{-3}
3	1.0095606	1.0095606	0.0	234.718	234.772	2.30×10^{-4}

* 全領域中最大の値

Table 5.6 Vectorization effect for COREBN

	ケース1		ケース 2		ケース 3	
	オリジナル版	ベクトル化版	オリジナル版	ベクトル化版	オリジナル版	ベクトル化版
内側反復回数	1	5	1	4	1	4
初期加速係数	無指定	1.8	無指定	1.8	無指定	1.8
外側反復回数	125	99	187	81	225	92
CPU 時間 (秒)	49.70	17.38	510.91	49.23	3330.61	332.15
VU時間 (秒)	0.0	5.34	0.0	25.71	0.0	195.61
倍 率**	1.0	2.86	1.0	10.38	1.0	10.03
I/O回数	2266	2323	16061	*1909	103703	*10373
メモリ使用量 (KB)	1372	1422	3692	4016	3632	4016
拡張メモリ使用量(MB)	0	0	0	15	0	15

* VIO/F使用

** 倍率＝オリジナル版CPU時間／ベクトル化版CPU時間

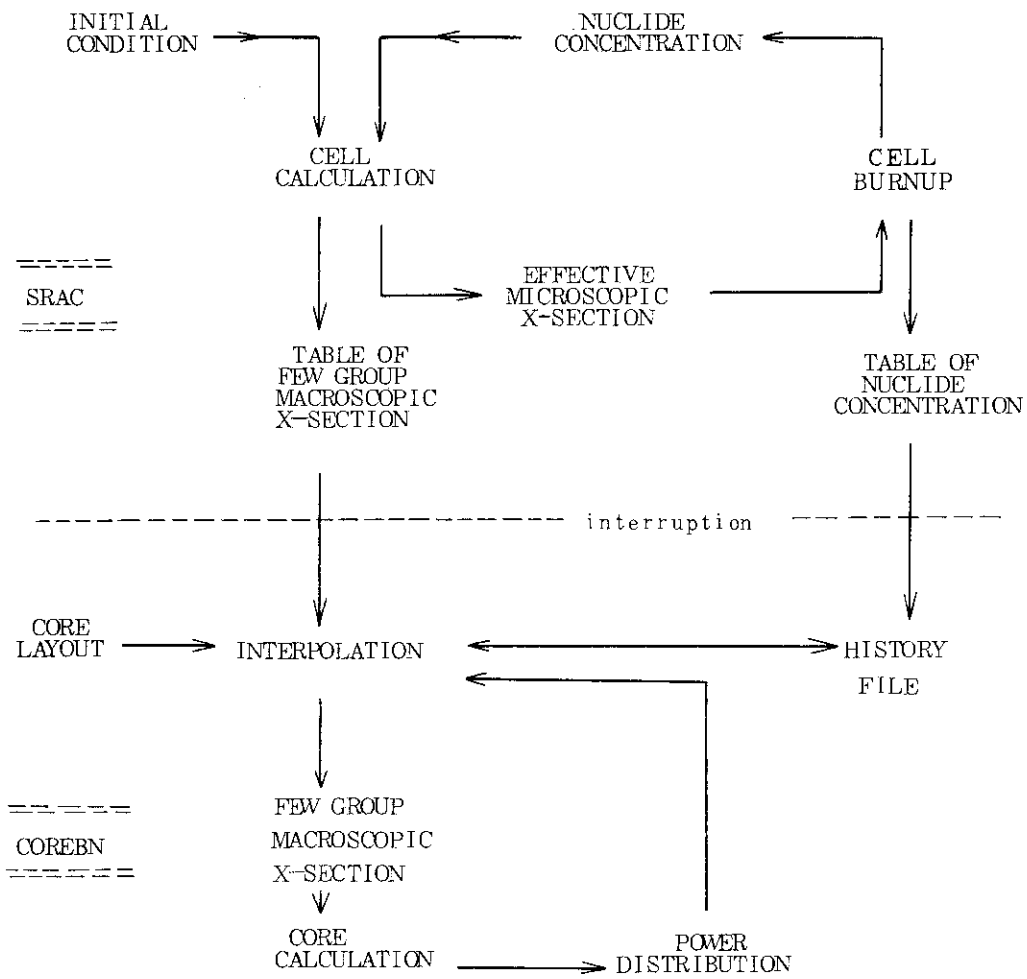


Fig. 5.1 Flow diagram of burn-up calculation

6. CITATION－FBRコード

6.1 コードの概要

CITATION－FBR[8]は高速炉体系の核設計計算コードシステム"EXPARAM"の中核となる計算コードであり，拡散理論に基づき，2次元，3次元形状での体系計算を行い，実効増倍率，中性子束，随伴中性子束を計算する。本コードは(a)計算コードCITATIONの拡散計算の解法ルーチンと，(b)高速炉体系での核設計計算における計算精度の向上を目的として新たに開発したルーチンから構成される。

新たに開発した項目は，以下のとおりである。

(1) 異方性拡散係数を用い中性子漏洩の方向依存性を考慮した計算

(2) 軸方向位置とエネルギーに依存したバックリングの計算（2次元R－Z体系）

(3) 体系内各領域ごとの中性子バランスの計算

また，システム内の計算コード相互のデータのやり取りを容易にするため，入出力部分に以下のような改良を加えた。

(1) 巨視的断面積の入力はJOINTコードを通じてPDSファイルから直接行う。

(2) 計算体系や，計算結果など他のコードとの連結を考慮した出力ファイルを作成する。

6.2 コードの動的挙動分析

CITATION－FBRコードの主要部分は，CITATIONコードのルーチンを利用しているので，時間分布は同様の傾向である。

6.3 ベクトル化方法

CITATIONコードのベクトル化技法と高速化技法をCITATION－FBRコードに適用した。具体的な手法は2.3節を参照のこと。

6.4 計算結果の評価

ベクトル化したコードの計算結果を評価するために2ケースの試計算を実施した。

2ケースの試計算データをTable 6.1に示す。

2ケースの試計算結果をTable 6.2に示す。各ケース及び各項目の値を比較するとK_{eff}（実効増倍率）は，2.0×10^{-5}以下，その他の項目も10^{-4}以下である。K_{eff}の収束判定条件が10^{-4}であることから，十分な精度で一致していると判断できる。

6.5 ベクトル化効果

試計算を実施した2ケースについてベクトル化の効果を調べた（Table 6.3）。これによるとCPU時間は，2次元のデータで3.9倍，3次元のデータで13.3倍となっている。

また，I／O回数を比較すると3次元のケースで極端にベクトル化版が少なくなっている。これは，equation constantsをVIO／Fで処理したためである。

6.6 議　論

(1) AE版について

CITATION－FBRベクトル化版計算の際，equation constantsをインコアで処理できず，ディスクに読み書きする場合がある。ディスクI／Oにすると経過時間が長くなり，I／O回数も多くなる。そこで，拡張領域を使えるバージョン(AE版)を用意した。AE版は，メインで定義する共通配列を大きくとり（今回の試計算で実施した3次元用データでは350万），各ソースをAEオプションでコンパイルしたものを用いればよい。一般に共通配列の大きさ（equation constantsも含めた大きさ）が180万以下の場合，通常のベクトル化版を用い，180万を越える場合ベクトル化AE版を用いるのがよい。

(2) 初期加速係数と内側反復回数の設定方法

CITATION－FBRスカラー版では，省略値として初期加速係数は無指定，内側反復回数は3回がとられる。VP版では省略値を用いると，その性能が出ない場合があるので，次の値を指定するとよい。

- 2次元
  - 初期加速係数 ： 1.8
  - 内側反復回数 ： 10 ～ 15
- 3次元
  - 初期加速係数 ： 1.8
  - 内側反復回数 ： 3 ～ 4

Table 6.1 Input data of test run on CITATION-FBR

ケースNo.	次元	形 状	メッシュサイズ	群 数
1	2次元	r－z	35×25	70
2	3次元	x-y-z	32×32×21	25

Table 6.2 Input data of test run for evaluation of vectorization effect on CITATION-FBR

ケース No.	項 目	オリジナル版 (A)	ベクトル化版 (B)	誤差 $\frac{(B)-(A)}{(A)}$
1	K_{eff}	0.9925365	0.9925512	1.48×10^{-5}
	LEAKAGE	5.49958×10^{-2}	5.49983×10^{-2}	4.55×10^{-5}
	TOTAL LOSSES	2.93217	2.93219	6.82×10^{-6}
	TOTAL PRODUCTIONS	2.91036	2.91037	3.44×10^{-6}
2	K_{eff}	1.0307283	1.0307388	1.02×10^{-5}
	LEAKAGE	8.03138×10^{1}	8.03095×10^{1}	-5.35×10^{-5}
	TOTAL LOSSES	5.24022×10^{2}	5.24018×10^{2}	-7.63×10^{-6}
	TOTAL PRODUCTIONS	5.40123×10^{2}	5.40120×10^{2}	-5.55×10^{-6}

Table 6.3 Vectorization effect for CITATION-FBR

	ケース 1		ケース 2	
	オリジナル版	ベクトル化版	オリジナル版	ベクトル化版
CPU 時間 (秒)	71.23	18.50	977.31	73.53
VU時間 (秒)	0.0	13.52	0.0	51.50
倍 率*	1.0	3.85	1.0	13.29
ベクトル化率 (%)	0.0	93.0	0.0	97.7
I／O回数	1362	167	28645	702
メモリ使用量 (KB)	1411	1477	4200	5000
拡張メモリ使用量 (MB)	0	1	0	9

* 倍率＝オリジナル版CPU時間／ベクトル化版CPU時間

謝　　辞

今回のベクトル化作業全般に亘り貴重な助言をいただきました，原研計算センター石黒美佐子氏，富士通㈱折居茂夫氏に感謝します。

TWOTRANコードベクトル化版を提供していただきました，動力炉・核燃料開発事業団佐藤一夫氏に感謝します。また，TWOTRANコードベクトル化版を整備するにあたり貴重な助言をいただきました，ファコム・ハイタック㈱南一生氏に感謝します。TWOTRANコードの入力データを提供していただきました，原研新型炉検討特別チーム高野秀機氏，並びに日本情報サービス㈱金子邦男氏に感謝します。

SRACシステムのベクトル化作業の際に，コード及びデータの提供と貴重な助言をいただきました，原研原子炉システム研究室土橋敬一郎氏に感謝します。衝突確率法ベクトル化部分をまとめるにあたって貴重な助言をいただきました，原研原子炉システム研究室筒井恒夫氏に感謝します。衝突確率法ベクトル化部分の整備作業を行っていただきました，原研計算センター外来研究員染谷千浩氏に感謝します。

COREBNコードのベクトル化作業の際に，コード及びデータの提供と貴重な助言をいただきました，原研システム研究室奥村啓介氏に感謝します。

CITATION－FBRコードのベクトル化作業の際に，コード及びデータの提供と貴重な助言をいただきました，原研高速炉物理研究室飯島進氏に感謝します。

本レポートを書く機会を与えていただきました，原研計算センター室長浅井清氏に感謝の意を表します。

参　考　文　献

(1) T. B. Fowler, D. R. Vondy, and G. W. Cunningham ; Nuclear Reactor Core Analysis Code : CITATION, ORNL−TM−2496, (Rev. 2, 1971)

(2) 原田裕夫，石黒美佐子；3次元中性子拡散コードCITATIONのベクトル化，日本原子力学会誌　Vol. 27, No.11 (1985)

(3) FACOM ソフトウエア説明書 OSⅣ/F4 MSP FORTUNE V10 (1985), 富士通㈱

(4) TWOTRAN−Ⅱコードチューニング報告書 (1985), 動力炉・核燃料開発事業団

(5) Toshihiko MATSUURA, Shin−ichi ICHIKAWA, and Kazuo MINAMI ; A FULLY VECTORIZABLE METHOD FOR THE DISCRETE ORDINATES NEUTRON TRANSPORT CODES, Proceedings of JSST Conference on Recent Advances in Simulation of Complex Systems, Tokyo, Japan July 15−17, 1986

(6) K. D. Lathrop and F. W. Brinkley ; TWO DIMENSIONAL MULTIGROUP DISCRETES ORDINATES TRANSPORT CODE : TWOTRAN Ⅱ, LA−4848−MS (1973)

(7) Keichiro TSUCHIHASHI, Yukio ISHIGURO, Kunio KANEKO, and Masaru IDO ; Revised SRAC Code System, JAERI 1302 (1986)

(8) 飯島　進；CITATION−FBR, 私信 (1988)

付録A　CITATIONコードベクトル化版使用手引

(1)　JCL

今回のベクトル化作業で作成されたロードモジュールは，J0002. CITVP. LOADである。これを使用するJCL 2例を以下に示す。ただし，ロードモジュールのメンバー名はCITVPである。以下に示す2例のうち最初のJCL(a)はロードモジュールを使用した実行ステップのみのJCLであり，2番目のJCL(b)はプログラムを一部（本例ではメインルーチン）を修正してロードモジュールを一時的に置き換えて実行するJCLである。

(a)　実行JCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.6 C.8 I.4 W.4
  OPTP PASSWORD=?,CLASS=0
//CITO  EXEC LMGO,LM='J0002.CITVP',
//  PNM=CITVP
//FT05F001 DD DSN=J9202.CIT.DATA(VHE5V6),DISP=SHR,LABEL=(,,,IN)
//FT31F001 DD DSN=J3472.CITORD.DATA(VHE31),DISP=SHR,LABEL=(,,,IN)
//FT51F001 DD SYSOUT=*
//FT15F001  DD UNIT=WK10,SPACE=(CYL,(15,3)),DISP=(NEW,DELETE)
//FT01F001 DD DSN=&&FT01,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT02F001 DD DSN=&&FT02,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT03F001 DD DSN=&&FT03,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT09F001 DD DSN=&&FT09,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT10F001 DD DSN=&&FT10,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT11F001 DD DSN=&&FT11,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT14F001 DD DSN=&&FT14,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT16F001 DD DSN=&&FT16,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT18F001 DD DSN=&&FT18,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT19F001 DD DSN=&&FT19,UNIT=WK10,SPACE=(TRK,(200,50)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT20F001 DD DSN=&&FT20,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
```

```
//FT21F001 DD DSN=&&FT21,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT22F001 DD DSN=&&FT22,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT23F001 DD DSN=&&FT23,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT24F001 DD DSN=&&FT24,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT25F001 DD DSN=&&FT25,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT26F001 DD DSN=&&FT26,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT27F001 DD DSN=&&FT27,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT28F001 DD DSN=&&FT28,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT29F001 DD DSN=&&FT29,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT30F001 DD DSN=&&FT30,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT32F001 DD DSN=&&FT32,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT33F001 DD DSN=&&FT33,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT34F001 DD DSN=&&FT34,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
```

(b) メインの修正＆実行ＪＣＬ

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.6 C.8 I.4 W.4 E.15
 OPTP PASSWORD=?,CLASS=3
//   EXEC FORT77,B='AE'
//SYSIN DD *
      COMMON /ARRAY / A(3000000)
       MEMORY = 3000000
       CALL INPT(A,MEMORY)
       STOP
       END
 /*
 //*
 //   EXEC LKEDIT77,LM=J0002.CITVP,CNTL=NO,SECTION=31
 //SYSIN DD *
      INCLUDE OLDLM(CITVP)
      ENTRY MAIN
      NAME  CITVP
 /*
 //CITO  EXEC GO,PNM=CITVP
 //FT05F001 DD DSN=J3472.CITORD.DATA(VHE5V),DISP=SHR,LABEL=(,,,IN)
 //FT31F001 DD DSN=J3472.CITORD.DATA(VHE31),DISP=SHR,LABEL=(,,,IN)
```

```
//FT51F001 DD SYSOUT=*
//FT15F001 DD UNIT=WK10,SPACE=(CYL,(15,3)),DISP=(NEW,DELETE)
//FT01F001 DD DSN=&&FT01,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT02F001 DD DSN=&&FT02,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT03F001 DD DSN=&&FT03,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT09F001 DD DSN=&&FT09,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT10F001 DD DSN=&&FT10,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT11F001 DD DSN=&&FT11,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT14F001 DD DSN=&&FT14,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT16F001 DD DSN=&&FT16,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT18F001 DD DSN=&&FT18,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT19F001 DD DSN=&&FT19,UNIT=WK10,SPACE=(TRK,(200,50)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT20F001 DD DSN=&&FT20,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT21F001 DD DSN=&&FT21,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT22F001 DD DSN=&&FT22,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT23F001 DD DSN=&&FT23,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT24F001 DD DSN=&&FT24,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT25F001 DD DSN=&&FT25,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT26F001 DD DSN=&&FT26,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT27F001 DD DSN=&&FT27,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT28F001 DD DSN=&&FT28,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT29F001 DD DSN=&&FT29,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT30F001 DD DSN=&&FT30,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT32F001 DD DSN=&&FT32,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT33F001 DD DSN=&&FT33,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
//FT34F001 DD DSN=&&FT34,UNIT=WK10,SPACE=(TRK,(30,10)),
//  DISP=(NEW,DELETE),DCB=(BLKSIZE=23000,LRECL=X,RECFM=VBS)
++
//
```

(2) メモリとI/O

CITATIONコードベクトル化版は，オリジナル版に比べてメモリが増加しており（メモリ増加量の計算は本文2.6.2節を参照），その分もメインルーチンで定義している共通配列（配列名：A）に組み入れた。

現ロードモジュールの共通配列はA（1700000）（8メガバイドで入るほぼ最大のサイズ）で定義しており，特別に大きなメッシュサイズのデータでない限り十分に入る大きさである。今回のテストデータにおける入力条件とメモリの大きさの関係をTable A.1に示す。これによると最大で130万であり通常の使用には十分たえられる。

また，CITATIONコードは1/O回数が多いというのが一般ユーザの考えである。I/Oが多くかかるのはEquation constantsをインコアで処理できずdisk I/Oで行うのが原因である。今回の計算でもdisk I/Oをインコア処理に変えることにより37600回から100

回にI／O回数が減少している。メインで大きく共通配列をとることによりEquation constantsをインコアで処理できるようになり，I／O回数を減らすことができる。今回のデータにおけるEquation constantsの大きさをTable A. 1に示す。これによると1つのデータ以外170万でおさまっている。Equation constantsがインコアで処理されたかdisk I／Oになったか，インコアにするためにはどれだけ必要か，の情報は出力リストに表示される。出力例をFig. A. 1に示す。これより，出力リストのこの部分を見れば次の計算のとき，どれだけ共通配列が必要かわかる。

共通配列の大きさが170万を越える場合，メインルーチンを一部修正して実行しなければならない（CATATIONの修正＆実行JCL(b)参照)。修正箇所をFig. A. 2に示す。

(3) ベクトル化版の内側反復回数と初期加速係数の推奨値

オリジナル版とベクトル化版では収束計算の手法が異なるので，最適な内側反復回数，初期加速係数も異なる。最適な値は，手法だけでなくデータにも依存するので必ずしもここで示す値が最適となるとは限らないが，今回テストに使用したデータを感度解析して得られた最適な値を推奨値としてTable A. 2に示す。

Table A.1 Sample length of common array for vectorized CITATION code

形　状	メッシュサイズ	群数	Disk I/Oモード			インコアモード	
			オリジナル版で必要な共通配列の大きさ (A)	ベクトル化版で必要な共通配列の大きさ (B)	増加分 (B)−(A)	equation constantsをインコアで処理する為に必要な大きさ (C)	合計 (B)+(C)
2次元円筒形状	51×49	25	157855	196690	38835	182328	379018
2次元円筒形状	46×32	70	266296	292761	26465	310086	602847
2次元三角形状	86×43	24	216885	272570	55685	258129	530699
3次元XYZ形状	51×56×24	3	852111	1302635	450524	559200	1861835
3次元三角形状	32×16×15	24	254686	340908	86222	734896	1075804

Table A.2 The recommended number of inner iterations and initial relaxation factor for vectorized CITATION code

形状	2 次 元	3 次 元
内側反復回数	15～20	2～4
初期加速係数	1.5～1.8	1.5～1.8

```
CORE STORAGE DIFFERENCE (WORDS) EQUATION CONSTANTS I/O ~ 559200

EQUATION CONSTANTS WILL BE STORED ON I/O LOGICAL 15
NUMBER OF---COLUMNS, ROWS, PLANES, GROUPS, UPSCAT, DOWNSCAT
MEMORY LOCATIONS RESERVED FOR DATA STORAGE---1700000
MEMORY LOCATIONS USED FOR THIS PROBLEM-------1302635
MEMORY LOCATIONS NOT USED-------------------- 397365
```

Fig. A.1 Sample output list of used area for CITATION code

```
COMMON /ARRAY / A(1800000)  <---> この２ケ所を修正する
MEMORY = 1800000
CALL INPT(A,MEMORY)
STOP
END
```

Fig. A.2 Modification of common array size for CITATION code

付録B　TWOTRAN－Ⅱコードベクトル化版使用手引

(1)　JCL

今回のベクトル化作業で作成されたロードモジュールは，J0002. TWOVP. LOADである。これを使用するJCL 2例を以下に示す。ただし，ロードモジュールのメンバー名はTWOVPである。以下に示す2例のうち最初のＪＣＬ(a)はロードモジュールを使用した実行ステップのみのJCLであり，2番目のJCL(b)はプログラムの一部を修正してロードモジュールを一時的に書き換えて実行するＪＣＬである。

(a)　実行JCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.5 C.5 W.4 I.4
  OPTP PASSWORD=?,CLASS=O
//***************  TWOTRAN-II FORTVP-GO    ***************
//  EXEC LMGO,LM='J0002.TWOVP',PNM=TWOVP
//*********************************************************
//FT05F001 DD DSN=J9202.TWOTRAN.DATA(#AOORGNL),DISP=SHR
//FT06F001 DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=137)
//FT01F001 DD UNIT=WK10,SPACE=(TRK,(2000,200)),DISP=NEW
//FT03F001 DD UNIT=WK10,SPACE=(TRK,(500,100)),DISP=NEW
//*********** RESTART DUMP FILE (FT08 & FT09) **********
//FT08F001 DD UNIT=WK10,SPACE=(TRK,(100,30)),DISP=NEW
//FT09F001 DD UNIT=WK10,SPACE=(TRK,(100,30)),DISP=NEW
//*********************************************************
//FT10F001 DD DUMMY
//FT11F001 DD DUMMY
//FT17F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT18F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW
//FT30F001 DD UNIT=WK10,SPACE=(TRK,(500,100)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT31F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT32F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT33F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT34F001 DD DUMMY
//FT40F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT66F001 DD SYSOUT=*
++
//
```

(b) サブルーチンの修正&実行JCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.6 C.8 I.4 W.4 GRP
 OPTP PASSWORD=?,CLASS=O
//    EXEC FORT77,B='AE'
//SYSIN DD *
```

修正するサブルーチンを入れる

```
/*
//*
//    EXEC LKEDIT77,LM=J0002.TWOVP,CNTL=NO,SECTION=31
//SYSIN DD *
      INCLUDE OLDLM(TWOVP)
      ENTRY MAIN
      NAME  TWOVP
/*
//*
//TWOVP  EXEC GO,PNM=TWOVP
//*******************************************************
//FT05F001 DD DSN=J9202.TWOTRAN.DATA(#AOORGNL),DISP=SHR
//FT06F001 DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=137)
//FT01F001 DD UNIT=WK10,SPACE=(TRK,(2000,200)),DISP=NEW
//FT03F001 DD UNIT=WK10,SPACE=(TRK,(500,100)),DISP=NEW
//*********** MACRO SET FROM SNGC-III ******************
//FT07F001 DD DISP=SHR,DSN=J9202.TWOTRAN.MACRO,
//      LABEL=(,,,IN)
//*********** RESTART DUMP FILE (FT08 & FT09) **********
//FT08F001 DD UNIT=WK10,SPACE=(TRK,(100,30)),DISP=NEW
//FT09F001 DD UNIT=WK10,SPACE=(TRK,(100,30)),DISP=NEW
//*******************************************************
//FT10F001 DD DUMMY
//FT11F001 DD DUMMY
//FT17F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT18F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW
//FT30F001 DD UNIT=WK10,SPACE=(TRK,(500,100)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT31F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT32F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT33F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT34F001 DD DUMMY
//FT40F001 DD UNIT=WK10,SPACE=(TRK,(500,50)),DISP=NEW,
//      DCB=(LRECL=876,BLKSIZE=6136,RECFM=VBS)
//FT66F001 DD SYSOUT=*
++
//
```

(2) メモリ

TWOTRAN－II コードベクトル化版は，オリジナル版に比べてメモリが増加しており（メモリ増加量の計算は本文 3.6.1 節を参照），その分もメインルーチンで定義している共通配列（配列名：A）に組み入れた。

現ロードモジュールの共通配列はA（1700000）（8メガバイトで入るほぼ最大のサイズ）で定義しており，特別に大きなメッシュサイズのデータでない限り十分に入る大きさである。今回のテストデータにおける入力条件とメモリの大きさの関係をTable B.1 に示す。これによると最大で 117万であり通常の使用には十分たえられる。必要なメモリ量は，リストに出力される。出力例をFig. B.1 に示す。これより，出力リストのこの部分を見れば次の計算のとき，どれだけ共通配列が必要かわかる。

共通配列の大きさが 170万を越える場合，メインルーチンを一部修正して実行しなければならない（TWOTRAN－II の修正＆実行 JCL(b)参照）。修正箇所をFig. B.2 に示す。

Table B.1 Sample length of common array for vectorized TWOTRAN-II code

形　　状	細メッシュサイズ	群数	計算タイプ	オリジナル版で必要な共通配列の大きさ (A)	ベクトル化版で必要な共通配列の大きさ (B)	増加分 (B) － (A)
2次元円筒形状	61×40	70	PO, S8	417485	1169426	751941
2次元円筒形状	31×31	70	PO, S8	161255	483385	322130
2次元円筒形状	28×32	70	PO, S8	141541	448134	306593

```
STORAGE     REQUIRED  ALLOWED
SMALL CORE     11022  1700000
LARGE CORE    150233  1688978
V P   CORE    322130  1538745
```

Fig. B.1 Sample output list of used area for TWOTRAN-II code

```
C
C     MAIN PROGRAM OF TWOTRAN-II, THE EXPORT VERSION OF TWOTRAN FOR
C     XY, RZ, AND RT GEOMETRIES
C
C
C
      COMMON /FWBGN1/ IDUSE(18),LAST,LASTEC,IGCDMP,IPSO,LTSO,IPFL,LTFL,
     1IPFX,LTFX,LXFX,IPXS,IPXSCT,LTXS,LTOXS,LTAXS,IPQS,LTQS,IEREC,I2,I4,
     2I6,ISPANQ,IPHAF,IPVAF,LTHAF,LTVAF,IFO
      COMMON /FWBGN2/ TIMBDP,TIMSLD,TIMOFF,MAXLEN,MAXECS,LENMCB,LENCIA,
     1IFNOVY,IRCOVY,I1,I3,I5
      COMMON /LOCAL/ NERROR,ITLIM,ISNT,MCR,MTP,MTPS,NISOXS,LMTP,IEDOPS,
     1NEXTER,JFISC,IEDIT(2),LIMIT,LENCLR
      COMMON /SWEEP/ BA,BC,J,J1,J2
      COMMON /UNITS/ NINP,NOUT,NAFLUX,NDUMP1,NDUMP2,NEXTRA,NEDIT,IAFLUX,
     1ITFLUX,ISNCON,IFIXSR,ISOTXS
      COMMON /ICONS/ ICON(20)
      REAL*4 TIN,TIMACC
C
      EQUIVALENCE (IA(197),TIN),(IA(247),TIMACC)
C* ADD                          75/7/31  BY M.SASAKI
      REAL*8   E9(3)
      DATA    E9  / 8HNO MORE , 8HPROBLEMS /
C/ END
C*----ADD                                                 78.08.08  M.SASAKI
      REAL   SEC
      EQUIVALENCE .( IA(244) , SEC )
C/----END
C
      COMMON  / MNEMO1 /  IA(250)
      COMMON  / MNEMO2 /  A(850000)
C
      MAXLEN        =        850000
```

この２ケ所を修正する

Fig. B.2 Modification of common array size for TWOTRAN-II code

付録C　SRACシステムベクトル化版使用手引

(1)　JCL

今回のベクトル化作業で作成されたロードモジュールは，J0002. SRACV. LOAD である。これを使用するＪＣＬ２例を以下に示す。ただし，ロードモジュールのメンバー名はSRACVPである。以下に示す２例のうち最初のＪＣＬ(a)はロードモジュールを使用した実行ステップのみのＪＣＬであり，２番目のＪＣＬ(b)はプログラムを一部（本例ではメインルーチン）を修正してロードモジュールを一時的に置き換えて実行するＪＣＬである。

(a)　実行JCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.6 C.8 I.4 W.4 E.12 GRP
 OPTP PASSWORD=     ,CLASS=3
//SRAC  EXEC LMGO,LM='J0002.SRACV',PNM=SRACVP
//SYSIN      DD DSN=J9202.SRAC.DATA(PORX04A),DISP=SHR,
//    LABEL=(,,,IN)
// EXPAND GRNLP,SYSOUT=M
//FT81F001  DD DSN=&&WRK81,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
//FT82F001  DD DSN=&&WRK82,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
//FT83F001  DD DSN=&&WRK83,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
//FT84F001  DD DSN=&&WRK84,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
//FT91F001  DD DSN=&&WRK91,SPACE=(TRK,(5,2)),UNIT=WK10,
//           DCB=(RECFM=FB,BLKSIZE=6400,LRECL=80)
//FT92F001  DD DSN=&&WRK92,SPACE=(TRK,(5,2)),UNIT=WK10, REACTION RATE
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160)
//FT31F001  DD DSN=&&WRK31,SPACE=(TRK,(5,2)),
//          UNIT=WK10,DISP=(NEW,PASS),
//         DCB=(RECFM=FB,BLKSIZE=6400,LRECL=80)
//FT01F001 DD DSN=&&WRK01,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT02F001  DD DSN=&&WRK02,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT03F001 DD DSN=&&WRK03,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT04F001 DD DSN=&&WRK04,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT08F001  DD DSN=&&WRK08,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT09F001  DD DSN=&&WRK09,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT10F001  DD DSN=&&WRK10,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT11F001  DD DSN=&&WRK11,SPACE=(TRK,(30,10)),UNIT=WK10,
//           DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
```

```
//FT12F001  DD DSN=&&WRK12,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT13F001  DD DSN=&&WRK13,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT14F001  DD DSN=&&WRK14,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT15F001  DD SUBSYS=(VPCS,'SPACE=10M')
//*FT15F001  DD SPACE=(CYL,(15,3)),UNIT=WK10,DISP=(NEW,DELETE)
//FT16F001  DD DSN=&&WRK16,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT18F001  DD DSN=&&WRK18,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT19F001  DD DSN=&&WRK19,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT21F001  DD DSN=&&WRK21,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT22F001  DD DSN=&&WRK22,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT26F001  DD DSN=&&WRK26,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT28F001  DD DSN=&&WRK28,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT32F001  DD DSN=&&WRK32,SPACE=(TRK,(20,5)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT33F001  DD DSN=&&WRK33,SPACE=(TRK,(20,5)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FASTP     DD DSN=J0002.FASTLBB4.DATA,DISP=SHR,LABEL=(,,,IN)
//THERMALP  DD DSN=J0002.THERMLB4.DATA,DISP=SHR,LABEL=(,,,IN)
//MCROSS    DD DSN=J0002.MCROSSB4.DATA,DISP=SHR,LABEL=(,,,IN)
//FASTU     DD DSN=&&WRKFTU,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//          DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//THERMALU   DD DSN=&&WRKTHU,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//          DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//MACROWRK   DD DSN=&&WRKMCR,SPACE=(TRK,(100,5,20)),UNIT=WK10,
//          DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//MACRO     DD DSN=&&WRKMCO,SPACE=(TRK,(50,5,10)),UNIT=WK10,
//         DCB=(RECFM=U,BLKSIZE=19069),DISP=(NEW,PASS)
//MICREF    DD DSN=&&WRKMIC,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//          DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//FLUX      DD DSN=&&WRKFLX,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//        DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//FT50F001  DD DSN=J1480.BURN2.DATA(IIJIMA2),DISP=SHR,LABEL=(,,,IN)
//FT51F001  DD DSN=&&WRK51,SPACE=(TRK,(5,5)),UNIT=WK10,
//          DCB=(RECFM=VS,BLKSIZE=3200,BUFNO=2)
//FT52F001  DD DSN=&&WRK52,SPACE=(TRK,(1,1)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT99F001  DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=19043)
++
//
```

(b) メイン修正＆実行ＪＣＬ

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.6 C.8 I.4 W.4 GRP
 OPTP PASSWORD=     ,CLASS=0
//   EXEC FORT77
//SYSIN DD *
C*********** SRAC MAIN PROGRAM ********************************
      COMMON /MAINC/ IOPT(20),JNFSTL(2),FNFSTL(2),JNTHEL(2),FNTHEL(2)
     1    ,JNEFST(2),FNEFST(2),JNETHE(2),FNETHE(2),JNMACR(2),FNMACR(2)
     2    ,JNMCRS(2),FNMCRS(2),JNEMIC(2),FNEMIC(2),JNFLUX(2),FNFLUX(2)
     3   ,NEFL      ,NETL      ,NEF       ,NET       ,NERF      ,NERT
     4   ,NMAT      ,NETL1     ,BSQ       ,NIN1      ,NIN2      ,NOUT1
     5   ,NOUT2     ,ITO       ,NEFL1     ,NEFL2     ,NEFL3     ,NEF1
     6   ,NEF2      ,NEF3      ,ISTEP     ,NSOUC     ,NFIN      ,NFOUT
     7   ,ITYPE     ,IMCEF     ,I79       ,I80
     8   ,LCNEGF    ,LCNEGT    ,LCNECF    ,LCNECT    ,LCMTNM    ,LCNISO
     9   ,LCTEMP    ,LCXL      ,LCXCDC    ,LCLISO    ,LCIDNT    ,LCDN
     A   ,LCIRES    ,LCIXMC    ,NFTOT     ,MEMORY    ,IOPEN     ,IRANG
     B   ,ICF       ,INITL
     C   ,CASEID(2),TITLE(18)
     D   ,II(1880)
C    D   ,II(880)
C**********AND SET THE VARIABLE 'MEMORY' TO THIS VALUE
C
C
      COMMON /WORK/ A(1600000)
      COMMON /TW1C/ CC(1),LIM1,IAA(4000)
      COMMON /SN1C/ BB(1),LIM2,IBB(1000)
      COMMON /CIT1C/ DD(8),LIM3,IDD(4000)
      LIM1=3998
      LIM2=1000
      LIM3=4000
      CALL DTLIST
      CALL ERRSET(202,256,10,2,1)
C     CALL ICLEA(IOPT,1000,0)
      CALL ICLEA(IOPT,2000,0)
      MEMORY = 1600000
      CALL SRAC
      STOP
      END
/*
//*
//   EXEC LKEDIT77,LM=J0002.SRACV,CNTL=NO,PARM='OVLY,LIST,LET'
//SYSIN DD DSN=J0002.SRACOVLY.DATA(SRACVP),DISP=SHR,LABEL=(,,,IN)
//SYSPRINT DD DUMMY
//*
//SRAC  EXEC GO,PNM=SRACVP
//SYSIN     DD DSN=J9202.SRAC.DATA(POBR26A),DISP=SHR,
//   LABEL=(,,,IN)
// EXPAND GRNLP,SYSOUT=M
//FT81F001  DD DSN=&&WRK81,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
//FT82F001  DD DSN=&&WRK82,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
//FT83F001  DD DSN=&&WRK83,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
```

```
//FT84F001  DD DSN=&&WRK84,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=16324,LRECL=16320,BUFNO=2)
//FT91F001  DD DSN=&&WRK91,SPACE=(TRK,(5,2)),UNIT=WK10,
//          DCB=(RECFM=FB,BLKSIZE=6400,LRECL=80)
//FT92F001  DD DSN=&&WRK92,SPACE=(TRK,(5,2)),UNIT=WK10, REACTION RATE
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160)
//FT31F001  DD DSN=&&WRK31,SPACE=(TRK,(5,2)),
//         UNIT=WK10,DISP=(NEW,PASS),
//        DCB=(RECFM=FB,BLKSIZE=6400,LRECL=80)
//FT01F001 DD DSN=&&WRK01,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT02F001  DD DSN=&&WRK02,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT03F001 DD DSN=&&WRK03,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT04F001 DD DSN=&&WRK04,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT08F001  DD DSN=&&WRK08,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT09F001  DD DSN=&&WRK09,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT10F001  DD DSN=&&WRK10,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT11F001  DD DSN=&&WRK11,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT12F001  DD DSN=&&WRK12,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT13F001  DD DSN=&&WRK13,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT14F001  DD DSN=&&WRK14,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT15F001  DD SPACE=(CYL,(15,3)),UNIT=WK10,DISP=(NEW,DELETE)
//FT16F001  DD DSN=&&WRK16,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT18F001  DD DSN=&&WRK18,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT19F001  DD DSN=&&WRK19,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT21F001  DD DSN=&&WRK21,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT22F001  DD DSN=&&WRK22,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT26F001  DD DSN=&&WRK26,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT28F001  DD DSN=&&WRK28,SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT32F001  DD DSN=&&WRK32,SPACE=(TRK,(20,5)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT33F001  DD DSN=&&WRK33,SPACE=(TRK,(20,5)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FASTP     DD DSN=J0002.FASTLBB4.DATA,DISP=SHR,LABEL=(,,,IN)
//THERMALP  DD DSN=J0002.THERMLB4.DATA,DISP=SHR,LABEL=(,,,IN)
//MCROSS    DD DSN=J0002.MCROSSB4.DATA,DISP=SHR,LABEL=(,,,IN)
//FASTU     DD DSN=&&WRKFTU,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//          DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//THERMALU   DD DSN=&&WRKTHU,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//          DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//MACROWRK   DD DSN=&&WRKMCR,SPACE=(TRK,(100,5,20)),UNIT=WK10,
```

```
//            DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//MACRO       DD DSN=&&WRKMCO,SPACE=(TRK,(50,5,10)),UNIT=WK10,
//           DCB=(RECFM=U,BLKSIZE=19069),DISP=(NEW,PASS)
//MICREF      DD DSN=&&WRKMIC,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//            DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//FLUX        DD DSN=&&WRKFLX,SPACE=(TRK,(50,5,50)),UNIT=WK10,
//          DCB=(RECFM=U,BLKSIZE=19069),DISP=(,PASS)
//FT50F001    DD DSN=J1480.BURN2.DATA(IIJIMA2),DISP=SHR,LABEL=(,,,IN)
//FT51F001    DD DSN=&&WRK51,SPACE=(TRK,(5,5)),UNIT=WK10,
//            DCB=(RECFM=VS,BLKSIZE=3200,BUFNO=2)
//FT52F001    DD DSN=&&WRK52,SPACE=(TRK,(1,1)),UNIT=WK10,
//            DCB=(RECFM=VBS,BLKSIZE=8164,LRECL=8160,BUFNO=2)
//FT99F001    DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=19043)
++
//
```

(2) メモリとI／O

SRACシステムベクトル化版は，オリジナル版に比べてメモリが増加しており（メモリ増加量の計算は本文4.3.5(2)，3.6.1節を参照），その分もメインルーチンで定義している共通配列（配列名：A）に組み入れた。

現ロードモジュールの共通配列はA（1500000）（8メガバイトで入るほぼ最大のサイズ）で定義しており，特別に大きなメッシュサイズのデータでない限り十分に入る大きさである。CITATION部分のテストデータにおける入力条件とメモリの大きさの関係をTable C.1に示し，TWOTRAN部分のテストデータにおける入力条件とメモリの大きさの関係をTable C.2に示す。これによると最大で95万であり通常の使用には十分たえられる。

また，SRACシステムの中のCITATION部分はI／O回数が多いというのが一般ユーザの考えである。I／Oが多くかかるのはEquation constantsをインコアで処理できずdisk I／Oで行うのが原因である。メインで大きく共通配列をとることによりEquation constantsをインコアで処理できるようになり，I／O回数を減らすことができる。今回のデータにおけるEquation constantsの大きさをTable C.1に示す。これによると1つのデータ以外150万でおさまっている。Equation constantsがインコアで処理されたかdisk I／Oになったか，インコアにするためにはどれだけ必要か，の情報は出力リストに表示される。出力例をFig.C.1に示す。これより，出力リストのこの部分を見れば次の計算のとき，どれだけ共通配列が必要かわかる。

共通配列の大きさが150万を越える場合，メインルーチンを一部修正して実行しなければならない（SRACの修正＆実行JCL(b)参照）。修正箇所をFig. C. 2に示す。

(3) ベクトル化版の内側反復回数と初期加速係数の推奨値

オリジナル版とベクトル化版では収束計算の手法が異なるので最適な内側反復回数，初期加速係数も異なる。最適な値は，手法だけでなくデータにも依存するので必ずしもここで示す値が最適となるとは限らないが，今回テストに使用したデータを感度解析して得られた最適な値を推奨値としてTable C. 3に示す。

Table C.1 Sample length of common array for vectorized SRAC system (CITATION code)

形　状	メッシュサイズ	群数	Disk I/Oモード			インコアモード	
			オリジナル版で必要な共通配列の大きさ (A)	ベクトル化版で必要な共通配列の大きさ (B)	増加分 (B)−(A)	equation constansをインコアで処理する為に必要な大きさ (C)	合計 (B)+(C)
2次元円筒形状	50×35	26	112800	139805	27005	133375	273180
3次元XYZ形状	37×37×35	4	581183	953656	372473	586857	1540513

Table C.2 Sample length of common array for vectorized SRAC system (TWOTRAN-II)

形　状	細メッシュサイズ	群数	計算タイプ	オリジナル版で必要な共通配列の大きさ (A)	ベクトル化版で必要な共通配列の大きさ (B)	増加分 (B)−(A)
2次元円筒形状	35×35	10	P1, S4	69897	276902	207005

Table C.3 The recommended number of inner iterations and initial relaxation factor for vectorized SRAC system

形状	2 次 元	3 次 元
内側反復回数	5〜10	3〜5
初期加速係数	1.4〜1.8	1.5〜1.8

```
                    ⎧
CORE STORAGE DIFFERENCE (WORDS) EQUATION CONSTANTS I/O ~ 734896
                    ⎩
EQUATION CONSTANTS WILL BE STORED IN CORE
NUMBER OF---COLUMNS, ROWS, PLANES, GROUPS, UPSCAT, DOWNSCAT ~
MEMORY LOCATIONS RESERVED FOR DATA STORAGE---1800000
MEMORY LOCATIONS USED FOR THIS PROBLEM-------1075804
MEMORY LOCATIONS NOT USED-------------------- 724196
                    ⎩
```

(a) CITATION部分

STORAGE	REQUIRED	ALLOWED
SMALL CORE	11022	1700000
LARGE CORE	150233	1688978
V P CORE	322130	1538745

(b) TWOTRAN部分

Fig. C.1 Sample output list of used area for SRAC system

```
C*********** SRAC MAIN PROGRAM ********************************
      COMMON /MAINC/ IOPT(20),JNFSTL(2),FNFSTL(2),JNTHEL(2),FNTHEL(2)
     1    ,JNEFST(2),FNEFST(2),JNETHE(2),FNETHE(2),JNMACR(2),FNMACR(2)
     2    ,JNMCRS(2),FNMCRS(2),JNEMIC(2),FNEMIC(2),JNFLUX(2),FNFLUX(2)
     3   ,NEFL      ,NETL      ,NEF       ,NET       ,NERF      ,NERT
     4   ,NMAT      ,NETL1     ,BSQ       ,NIN1      ,NIN2      ,NOUT1
     5   ,NOUT2     ,ITO       ,NEFL1     ,NEFL2     ,NEFL3     ,NEF1
     6   ,NEF2      ,NEF3      ,ISTEP     ,NSOUC     ,NFIN      ,NFOUT
     7   ,ITYPE     ,IMCEF     ,I79       ,I80
     8   ,LCNEGF    ,LCNEGT    ,LCNECF    ,LCNECT    ,LCMTNM    ,LCNISO
     9   ,LCTEMP    ,LCXL      ,LCXCDC    ,LCLISO    ,LCIDNT    ,LCDN
     A   ,LCIRES    ,LCIXMC    ,NFTOT     ,MEMORY    ,IOPEN     ,IRANG
     B   ,ICF       ,INITL
     C   ,CASEID(2),TITLE(18)
     D   ,II(1880)
C    D   ,II(880)
C**********AND SET THE VARIABLE 'MEMORY' TO THIS VALUE
C
C
      COMMON /WORK/ A(1600000)
      COMMON /TW1C/ CC(1),LIM1,IAA(4000)
      COMMON /SN1C/ BB(1),LIM2,IBB(1000)
      COMMON /CIT1C/ DD(8),LIM3,IDD(4000)
      LIM1=3998
      LIM2=1000
      LIM3=4000
      CALL DTLIST
      CALL ERRSET(202,256,10,2,1)
C     CALL ICLEA(IOPT,1000,0)
      CALL ICLEA(IOPT,2000,0)
      MEMORY = 1600000
      CALL SRAC
      STOP
      END
```

この２ケ所を修正する

Fig. C.2 Modification of common array size for SRAC system

付録D　COREBNコードベクトル化版使用手引

(1)　JCL

今回のベクトル化作業で作成されたロードモジュールは，J0002. COREBNV. LOAD である。これを使用するＪＣＬ２例を以下に示す。ただし，ロードモジュールのメンバー名はCRBNVPである。以下に示す２例のうち最初のＪCL(a)はロードモジュールを使用した実行ステップのみのＪCLであり，２番目のJCL(b)はプログラムを一部（本例ではメインルーチン）を修正してロードモジュールを一時的に置き換えて実行するJCLである。

(a)　実行ＪCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.6 C.8 I.5 W.4 E.50
 OPTP PASSWORD=     ,CLASS=4
//HIST2 EXEC LMGO,PNM=HIST,LM=J0002.COREBNV
//FT11F001 DD DSN=&&PSLIB,UNIT=WK10,DISP=(,PASS),SPACE=(TRK,(10,10))
//USERPDS  DD DSN=J9331.J4244.KFKMACB2.DATA,DISP=SHR,LABEL=(,,,IN)
//SYSIN DD *
  99 0 11  / PDS TO PS
/*
//COREBN2 EXEC LMGO,PNM=CRBNVP,LM=J0002.COREBNV
//FT06F001 DD SYSOUT=* DCB=(RECFM=FBA,LRECL=137,BLKSIZE=32743)
//FT01F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT02F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT03F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT09F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//*FT09F001 DD DSN=J7125.KFKF1I.DATA,UNIT=D0430,DISP=OLD,
//*          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT10F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT11F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//*FT13F001 DD SPACE=(TRK,(30,10)),UNIT=WK10
//FT14F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT15F001  DD SUBSYS=(VPCS,'SPACE=10M')
//*FT15F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//*          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT16F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT19F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT20F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT26F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT31F001 DD SPACE=(TRK,(10,10)),UNIT=WK10,
```

```
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT32F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,    === POWER FILE ===
//          DCB=(RECFM=VBS,LRECL=X,BLKSIZE=32760)
//*FT32F001 DD DSN=J7125.KFKP1I.DATA,UNIT=TDS,DISP=OLD, = POWER FILE
//*         DCB=(RECFM=VBS,LRECL=X,BLKSIZE=32760)
//FT89F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT90F001 DD DSN=&&PSLIB,UNIT=WK10,DISP=(OLD,DELETE)    === MACRO ===
//FT91F001 DD SPACE=(TRK,(2,1)),UNIT=WK10,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT92F001 DD DSN=J9331.J4244.HISTI.DATA,DISP=SHR,LABEL=(,,,IN)
//FT93F001 DD DUMMY
//FT94F001 DD SPACE=(TRK,(2,1)),UNIT=WK10,
//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT95F001 DD SPACE=(TRK,(2,1)),UNIT=WK10,

//              DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT96F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT96F001  DD SUBSYS=(VPCS,'SPACE=40M')
//*FT97F001 DD SPACE=(TRK,(90,10)),UNIT=WK10,
//*          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT99F001 DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=13700)
//*FT99F001 DD DUMMY,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=13700)
//SYSIN DD *
```

入力データ

```
/*
++
//
```

(b) メイン修正＆実行 JCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.6 C.8 I.4 W.4 E.20
 OPTP PASSWORD=     ,CLASS=3
//COMP  EXEC FORT77
C CORE BURN-UP WITH ENLARGED CORE STRAGE
C CHANGE THE ARRAY LENGTH OF BLANK COMMON
      COMMON/ /ARAY(1400000)
      MEMORY=1400000
      CALL CRBN (ARAY,MEMORY,1)
      STOP
      END
//LINK EXEC LKEDIT,CNTL=NO,MODS='200,10,1',WORKS='200,10',
//        LM=J0002.COREBNV,PARM='OVLY,LET,LIST'
//SYSIN DD DSN=J0002.CRBNOVLY.DATA,DISP=SHR,LABEL=(,,,IN)
//HIST2 EXEC LMGO,PNM=HIST,LM=J0002.COREBNV
//FT11F001 DD DSN=&&PSLIB,UNIT=WK10,DISP=(,PASS),SPACE=(TRK,(10,10))
//USERPDS  DD DSN=J9331.J4244.KFKMACB2.DATA,DISP=SHR,LABEL=(,,,IN)
//SYSIN DD *
  99 0 11  / PDS TO PS
/*
```

```
//COREBN2 EXEC GO,PNM=CRBNVP
//FT06F001 DD SYSOUT=* DCB=(RECFM=FBA,LRECL=137,BLKSIZE=32743)
//FT01F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT02F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT03F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT09F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//*FT09F001 DD DSN=J7125.KFKF1I.DATA,UNIT=D0430,DISP=OLD,
//*          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT10F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT11F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//*FT13F001 DD SPACE=(TRK,(30,10)),UNIT=WK10
//FT14F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT15F001 DD SUBSYS=(VPCS,'SPACE=10M')
//*FT15F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//*          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT16F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT19F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT20F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT26F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT31F001 DD SPACE=(TRK,(10,10)),UNIT=WK10,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT32F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,    === POWER FILE ===
//          DCB=(RECFM=VBS,LRECL=X,BLKSIZE=32760)
//*FT32F001 DD DSN=J7125.KFKP1I.DATA,UNIT=TDS,DISP=OLD, = POWER FILE =
//*         DCB=(RECFM=VBS,LRECL=X,BLKSIZE=32760)
//FT89F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT90F001 DD DSN=&&PSLIB,UNIT=WK10,DISP=(OLD,DELETE)    === MACRO ===
//FT91F001 DD SPACE=(TRK,(2,1)),UNIT=WK10,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT92F001 DD DSN=J9331.J4244.HISTI.DATA,DISP=SHR,LABEL=(,,,IN)
//FT93F001 DD DUMMY
//FT94F001 DD SPACE=(TRK,(2,1)),UNIT=WK10,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT95F001 DD SPACE=(TRK,(2,1)),UNIT=WK10,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//FT96F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//FT97F001 DD SPACE=(TRK,(30,10)),UNIT=WK10,
//          DCB=(RECFM=VBS,BLKSIZE=32760,LRECL=X,BUFNO=1)
//*FT99F001 DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=13700)
//FT99F001 DD DUMMY,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=13700)
//SYSIN DD *
```

入力データ

```
/*
++
//
```

(2) メモリとI／O

COREBNコードベクトル化版は，オリジナル版に比べてメモリが増加しており(メモリ増加量の計算は本文2.6.2節を参照)，その分もメインルーチンで定義している共通配列（配列名：ARAY）に組み入れた。

現ロードモジュールの共通配列はARAY（1400000）（8メガバイトで入るほぼ最大のサイズ）で定義しており，特別に大きなメッシュサイズのデータでない限り十分に入る大きさである。今回のテストデータにおける入力条件とメモリの大きさの関係をTable A.1に示す。これによると最大で74万であり通常の使用には十分たえられる。

また，COREBNコードはI／O回数が多いというのが一般ユーザの考えである。I／Oが多くかかるのはEquation constantsをインコアで処理できずdisk I／Oで行うのが原因である。今回の計算でもdisk I／Oをインコア処理に変えることにより10万回から1万回にI／O回数が減少している。メインで大きく共通配列をとることによりEquation constantsをインコアで処理できるようになり，I／O回数を減らすことができる。今回のデータにおけるEquation constantsの大きさをTable D.1に示す。これによると1つのデータ以外140万でおさまっている。Equation constantsがインコアで処理されたかdisk I／Oになったか，インコアにするためにはどれだけ必要か，の情報は出力リストに表示される。出力例をFig.D.1に示す。これより，出力リストのこの部分を見れば次の計算のとき，どれだけ共通配列が必要かわかる。

共通配列の大きさが140万を越える場合，メインルーチンを一部修正して実行しなければならない（COREBNの修正＆実行JCL(b)参照)。修正箇所をFig. D.2に示す。

(3) ベクトル化版の内側反復回数と初期加速係数の推奨値

オリジナル版とベクトル化版では収束計算の手法が異なるので，最適な内側反復回数，初期加速係数も異なる。最適な値は，手法だけでなくデータにも依存するので必ずしもここで示す値が最適となるとは限らないが，今回テストに使用したデータを感度解析して得られた最適な値を推奨値としてTable D.2に示す。

Table D.1 Sample length of common array for vectorized COREBN code

形　状	メッシュサイズ	群数	Disk I/Oモード				インコアモード	
			ゾーン数	オリジナル版で必要な共通配列の大きさ (A)	ベクトル化版で必要な共通配列の大きさ (B)	増加分 (B)−(A)	equation constantsをインコアで処理する為に必要な大きさ (C)	合　計 (B)+(C)
2次元三角形状	34×17	7	58	14334	26957	12613	28472	55429
3次元三角形状	34×17×32	8	786	591427	740171	148744	1523117	2263288

Table D.2 The recommended number of inner iterations and initial relaxation factor for vectorized COREBN code

形状	2 次 元	3 次 元
内側反復回数	5	4
初期加速係数	1.8	1.8

```
EQUATION CONSTANTS WILL BE STORED ON I/O LOGICAL 15

NUMBER OF---COLUMNS, ROWS, PLANES, GROUPS, UPSCAT, DO1 ~

MEMORY LOCATIONS RESERVED FOR DATA STORAGE--- 741000
MEMORY LOCATIONS USED FOR THIS PROBLEM------- 740171
MEMORY LOCATIONS NOT USED-------------------    829
```

Fig. D.1 Sample output list of used area for COREBN code

```
C CORE BURN-UP WITH ENLARGED CORE STRAGE
C CHANGE THE ARRAY LENGTH OF BLANK COMMON
      COMMON/ /ARAY(1400000)
      MEMORY=1400000
      CALL CRBN (ARAY,MEMORY,1)
      STOP
      END
```

この２ケ所を修正する。

Fig. D.2 Modification of common array size for COREBN code

付録E　CITATION－FBRコードベクトル化版使用手引

(1)　JCL

CITATION－FBRコードベクトル化版は，ソースプログラムのみ保存してある（J0002. CITFBRVP. FORT77, J0002. CITFBR. FORT77, J0002. CITATION. FORT77)。そこで，ここではロードモジュール作成のJCLを2種類と実行JCLを2種類示す。ロードモジュール作成JCLのうち(a)は基本領域を使用したオーバーレイ構造のJCLであり，(b)は拡張領域を使用したJCLである。実行JCLの2種類のうち(c)は基本領域を使用した実行ステップのJCLであり,(d)は拡張領域を使用した実行ステップのJCLである。

(a)　ロードモジュール（基本領域使用）作成JCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.4 C.8 I.4 W.2
  OPTP PASSWORD      ,CLASS=O
//*  CITATION LOAD MODULE UPDATE STEP --------------------------
//FORT7 EXEC FORT77VP,RGN=8000K,
//   SO=J0002.CITFBRVP,Q='.FORT77',
//   A='ELM(*),VPXTBL',DISP=MOD
//SYSPRINT DD DUMMY
//*
//FORT7 EXEC FORT77,
//   SO=J0002.CITFBR,Q='.FORT77',
//   A='ELM(*)',DISP=MOD
//SYSPRINT DD DUMMY
//*
//FORT7 EXEC FORT77,
//   SO=J0002.CITATION,Q='.FORT77',DISP=MOD,RGN=3000K,
//   A='ELM(*),NOS'
//SYSPRINT DD DUMMY
//*
//*
//   EXEC LKEDCT77,LM='J9202.CITFCAV2',UNIT=D0340,MODS='50,30,10',
//   PARM='OVLY,LIST,LET',CNTL=NO
//SYSIN DD *
 ENTRY MAIN
 INSERT INPT,GRIT,RQED,CALR,ICLOCK,ITTIME,MODEL,RSET,IDAY,TITET
 OVERLAY NODE3000
 INSERT IPTM,BKLR
 OVERLAY NODE30A1
 INSERT OPT1,GETC,GETE,CSTC,CRDR,UPDT,RONE,RALL,COPY,WALL,              /
        WART,RAEN,SNSN,ORDE,PUNS,DTFP,FLTF
 OVERLAY NODE30A1
 INSERT SETV,CNTR,HIST,GEOM,LVMX,MESH,COMP,CMOT,KOMP,KMOT,              /
        OVER,MACR,SSET,KXNX,KSIG,TAPE
 OVERLAY NODE30A1
 INSERT CLAS,DENS,BKLE,FXSO,BEER,SRCH,RODI,DCAY,YELD,CHAN,IPRT,DYPD,    /
        TAPX,NSRT,CNIO,BNSB,CPNC,DISK,SIZE,RSTR,TRAN,SHOX,WIO3,IMXS,    /
        MYSH,RODO,KRST
 OVERLAY NODE30A1
 INSERT GEDT,GRIV,GNTL,GION,FMIP,PLIN,PUTA,JUNK,SHUF,SHIN,STSH,CHCK,    /
        CSRT,MBST,STFM
 OVERLAY NODE3000
 INSERT EIGN,BIGS,XSET,EXTR,CYCR,GINS,ITED,UDTE,SSZU
 OVERLAY NODE30A2
 INSERT WFCC,WFAC,STVR,YNAM,WNSS,RNSS,HOWE,INFX,KNFX,RODX
 OVERLAY NODE30A2
 INSERT FLUX,DNSD,ABPR,LOOP,FINS
 OVERLAY NODE30B2
 INSERT CNST
 OVERLAY NODE30B2
 INSERT BEGN,RDUE
 OVERLAY NODE30B2
 INSERT FWRD,FXRD
 OVERLAY NODE30B2
 INSERT DPER
 OVERLAY NODE30B2
```

```
 INSERT HWRD,HXRD
 OVERLAY NODE30B2
 INSERT WFLX
 OVERLAY NODE30B2
 INSERT FTRI
 OVERLAY NODE30A2
 INSERT KLUX,KNSD,KBPR,KOOP,KINS
 OVERLAY NODE30B3
 INSERT KNST
 OVERLAY NODE30B3
 INSERT KEGN,KDUE
 OVERLAY NODE30B3
 INSERT KWRD,KXRD,KZRD
 OVERLAY NODE30B3
 INSERT KPER
 OVERLAY NODE30B3
 INSERT MWRD
 OVERLAY NODE30B3
 INSERT KTRI
 OVERLAY NODE30A2
 INSERT NMBL,WSTR,DISH,DIRT,FASP,DASH,KASH,DODA,KRAN,CRSH,MASH
 OVERLAY NODE3000
 INSERT OUTC,POUT,KOUT,LEAKZ
 OVERLAY NODE30A3
 INSERT PDWT,KDWT,HEAT,PTAB,KTAB,KUDN,DTOR,EDIN,TABL,DLOP,RERT,NUDN,    /
        DNFC,CMXS
 OVERLAY NODE30A3
 INSERT PERT,PIOS,TCOF,PURT,KOKN,BEFF,VMAP,NMAP,RFLX
 OVERLAY NODE3000
 INSERT TSCL,BURN,CYED,NUCY
 OVERLAY NODE3000
 INSERT MEDT,DRIV,XION,INTL,IFCE,RADE,IFVX,ODER,MNGE,WCNC,ACCT,SECO,    /
        CONT,CFNT,DPOT,LRCY,LRTR,GETV,RIO2,DCAX,MANG,I120,SADD,CORT,    /
        MBED,INCO,EQTS,I2T4
 NAME VP
/*
++
//
```

(b) ロードモジュール（拡張領域使用）作成JCL

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
   T.5 C.8 I.4 W.2
  OPTP PASSWORD=     ,CLASS=0
//*  CITATION LOAD MODULE UPDATE STEP -------------------------
//FORT7 EXEC FORT77,B='AE'
//SYSPRINT DD DUMMY
//SYSIN DD *
      COMMON /ARRAY / A(3500000)
      MEMORY = 3500000
      CALL INPT(A,MEMORY)
      STOP
      END
/*
//*
//FORT7 EXEC FORT77VP,RGN=8000K,B='AE',
//  SO=J0002.CITFBRVP,Q='.FORT77',
//  A='ELM(*),VPXTBL',DISP=MOD
//SYSPRINT DD DUMMY
//*
//FORT7 EXEC FORT77,B='AE',
//  SO=J0002.CITFBR,Q='.FORT77',
//  A='ELM(*)',DISP=MOD
//SYSPRINT DD DUMMY
//*
//FORT7 EXEC FORT77,B='AE',
//  SO=J0002.CITATION,Q='.FORT77',DISP=MOD,RGN=3000K,
//  A='ELM(*),NOS'
//SYSPRINT DD DUMMY
//*
//*
//  EXEC LKEDCT77,LM='J9202.CITFCAVE',UNIT=D0340,MODS='50,30,10',
//  PARM='LIST,LET,MAP',SECTION=31
```

(c) 実行JCL（基本領域使用）

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
  T.6 C.8 W.4 I.4 E.12
  OPTP PASSWORD=     ,CLASS=3
//*  JOINT LOAD MODULE UPDATE STEP ----------------------------
//  EXEC FORT77,SO=J9202.CITFCAJN,
//  A='ELM(*)',B='LANGLVL(66)'
//  EXEC LKED77,A='LREP(JMF,JMP)',
//       PRVLIB='J2505.J2187.LIB433'
// EXEC LMGOCIT,LM='J9202.CITFCAVP',
//       PDS='J3622.ZPPR17A.FCA.PDS.G25.DATA',
//       PNM=VP
//JOINTRUN.STEPLIB DD DSN=&&LM,DISP=SHR
//JOINTRUN.USERPDS DD DSN=J3622.ZPPR17A.FCA.PDS.G25.DATA,DISP=SHR,
//      LABEL=(,,,IN)
//JOINTRUN.SYSIN DD DSN=J3622.ZPPR17.FCA.DATA(CIT3DFCA),DISP=SHR,
//      LABEL=(,,,IN)
//CITATION.FT07F001 DD UNIT=WK10,SPACE=(TRK,(300,50))
//CITATION.FT15F001 DD SUBSYS=(VPCS,'SPACE=10M')
//CITATION.FT34F001 DD UNIT=WK10,SPACE=(TRK,(300,50))
++
//
```

(d) 実行JCL（拡張領域使用）

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 30479202,IW.NONOMIYA,0341.02
  T.6 C.8 W.4 I.4 E.15
  OPTP PASSWORD=     ,CLASS=4
//*  JOINT LOAD MODULE UPDATE STEP ------------------------------
//   EXEC FORT77,SO=J9202.CITFCAUP,
//   A='ELM(MACSET,CITATI,GET008,MCSTOR)'
//   EXEC LKEDIT77,LM=J2505.J2187.JOINTUP,A='LREP(JMF,JMP)',
//        PRVLIB='J2505.J2187.LIB433'
// EXEC LMGOCIT,LM='J9202.CITFCAVE',
//        PDS='J3622.ZPPR17A.FCA.PDS.G25.DATA',
//        PNM=TEMPNAME
//JOINTRUN.STEPLIB DD DSN=&&LM,DISP=SHR
//JOINTRUN.USERPDS DD DSN=J3622.ZPPR17A.FCA.PDS.G25.DATA,DISP=SHR,
//      LABEL=(,,,IN)
//JOINTRUN.SYSIN DD DSN=J3622.ZPPR17.FCA.DATA(CIT3DFCA),DISP=SHR,
//      LABEL=(,,,IN)
//CITATION.FT07F001 DD UNIT=WK10,SPACE=(TRK,(300,50))
//CITATION.FT15F001 DD UNIT=WK10,SPACE=(TRK,(300,50))
//*CITATION.FT15F001 DD SUBSYS=(VPCS,'SPACE=10M')
//CITATION.FT34F001 DD UNIT=WK10,SPACE=(TRK,(300,50))
++
//
```

(2) メモリとI／O

CITATION－FBRコードベクトル化版もCITATIONコードと同様に，オリジナル版に比べてメモリが増加しており（メモリ増加量の計算は本文2.6.2節を参照），その分もメインルーチンで定義している共通配列（配列名：A）に組み入れた。

現ロードモジュールの共通配列は基本領域を使用する場合A（1800000）（8メガバイトで入るほぼ最大のサイズ），拡張領域を使用する場合A（3500000）で定義しており，特別に大きなメッシュサイズのデータでない限り十分に入る大きさである。今回のテストデータにおける入力条件とメモリの大きさの関係をTable E.1に示す。これによると最大で110万であり通常の使用には十分たえられる。

また，CITATION－FBRコードはI／O回数が多いというのが一般ユーザの考えである。I／Oが多くかかるのはEquation constantsをインコアで処理できずdisk I／O で行うのが原因である。今回の計算でもdisk I／Oをインコア処理に変えることにより28600回から700回にI／O回数が減少している。メインで大きく共通配列をとることによりEquation constantsをインコアで処理できるようになり，I／O回数を減らすことができる。今回のデータにおけるEquation constantsの大きさをTable E.1に示す。これによると両ケース共350万でおさまっている。Equation constantsがインコアで処理されたかdisk I／Oになったか，インコアにするためにはどれだけ必要か，の情報は出力リストに表示される。出力例をFig. E.1に示す。これより，出力リストのこの部分を見れば次の計算のとき，どれだけ共通配列が必要かわかる。修正箇所をFig. E. 2に示す。

(3) ベクトル化版の内側反復回数と初期加速係数の推奨値

オリジナル版とベクトル化版では収束計算の手法が異なるので，最適な内側反復回数，初期加速係数も異なる。最適な値は，手法だけでなくデータにも依存するので必ずしもここで示す値が最適となるとは限らないが，今回テストに使用したデータを感度解析して得られた最適な値を推奨値としてTable E. 2に示す。

Table E.1 Sample length of common array for vectorized CITATION-FBR code

形状	メッシュサイズ	群数	Disk I/Oモード			インコアモード	
			オリジナル版で必要な共通配列の大きさ (A)	ベクトル化版で必要な共通配列の大きさ (B)	増加分 (B)−(A)	equation constantsをインコアで処理する為に必要な大きさ (C)	合計 (B)+(C)
2次元円筒形状	35×25	70	170792	187331	16539	185265	372596
3次元XYZ形状	32×32×21	25	899338	1098975	199637	2121216	3220191

Table E.2 The recommended number of inner iterations and initial relaxation factor for vectorized CITATION-FBR code

形状	2 次 元	3 次 元
内側反復回数	10〜15	3〜4
初期加速係数	1.8	1.8

```
CORE STORAGE DIFFERENCE (WORDS) EQUATION CONSTANTS I/O ~ 734896

EQUATION CONSTANTS WILL BE STORED IN CORE
NUMBER OF---COLUMNS, ROWS, PLANES, GROUPS, UPSCAT, DOWNSCAT ~
MEMORY LOCATIONS RESERVED FOR DATA STORAGE---1800000
MEMORY LOCATIONS USED FOR THIS PROBLEM-------1075804
MEMORY LOCATIONS NOT USED-------------------- 724196
```

Fig. E.1 Sample output list of used area for CITATION-FBR code

```
COMMON /ARRAY / A(1800000)  <-- この２ケ所を修正する
MEMORY = 1800000
CALL INPT(A,MEMORY)
STOP
END
```

Fig. E.2 Modification of common array size for CITATION-FBR code