

J A E R I - M

90-108

DOT3.5コードのベクトル化

— DOT3.5 NEA版, DOT3.5 FNS版,
DOT3.5炉設計版, RADHEAT-V4, DOT-DD —

1990年7月

野々宮 嶽^{*}・石黒美佐子・筒井 恒夫

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, Tckai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

©Japan Atomic Energy Research Institute, 1990

編集兼発行 日本原子力研究所
印 刷 いばらき印刷株

DOT 3.5 コードのベクトル化

—DOT 3.5 NEA 版, DOT 3.5 FNS 版, DOT 3.5 炉設計版, RADHEAT-V 4, DOT-DD—

日本原子力研究所東海研究所情報システムセンター

野々宮 嶽*・石黒美佐子・筒井 恒夫†

(1990年6月14日受理)

本報告は, Sn 法による 2 次元放射線輸送コード DOT 3.5 のベクトル化について述べる。対象としたコードは, ORNL で開発された NEA オリジナル版だけでなく日本原子力研究所内で実際に用いられている DOT 3.5 の改良版である DOT 3.5 コード FNS 版, DOT 3.5 コード核融合実験炉設計版, RADHEAT-V 4 システム ESPRIT モジュールについてもベクトル化を行った。DOT 3.5 では, エネルギー群数や空間メッシュ数が大きくなると外部ファイルとの入出力回数が極端に増加するため CPU 時間よりも入出力処理のための時間が経過時間の大部分を占めるようになる。そこで DOT 3.5 コード FNS 版と DOT 3.5 の DDX 版である DOT-DD コードのベクトル化版に対して入出力処理の高速化を実施した。

ベクトル化版のオリジナル版に対する性能向上は, DOT 3.5 コードで 1.7 ~ 1.9 倍, DOT 3.5 FNS 版で 2.2 ~ 2.3 倍, DOT 3.5 炉設計版で 1.7 倍, RADHEAT-V 4 システムで 3.1 ~ 4.4 倍である。また, 入出力処理高速化版のオリジナル版に対する経過時間は, DOT 3.5 コード FNS 版及び DOT-DD コードで 50% ~ 65% に減少した。

本報告書では, ベクトル化及び入出力処理高速化版の対象となったコードの概要, ベクトル化及び I/O 高速化の方法, 計算結果の評価及び速度向上効果について述べる。

東海研究所: 〒319-11 茨城県那珂郡東海村白方字白根2-4

+ 原子炉工学部

* 外来研究員, 富士通株式会社

Vectorization of DOT3.5 Code

— DOT3.5 NEA Version, DOT3.5 FNS Version,
DOT3.5 FER Version, RADHEAT-V4, DOT-DD —

Iwao NONOMIYA^{*}, Misako ISHIGURO and Tsuneo TSUTSUI⁺

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

(Received June 14, 1990)

In this report, we describe the vectorization of two-dimensional Sn-method radiation transport code DOT3.5.

Vectorized codes are not only the NEA original version developed at ORNL but also the versions improved by JAERI: DOT3.5 FNS version for fusion neutronics analyses, DOT3.5 FER version for fusion reactor design, and ESPRIT module of RADHEAT-V4 code system for radiation shielding and radiation transport analyses.

In DOT3.5, input/output processing time amounts to a great part of the elapsed time when a large number of energy groups and/or a large number of spatial mesh points are used in the calculated problem. Therefore, an improvement has been made for the speedup of input/output processing in the DOT3.5 FNS version, and DOT-DD (Double Differential cross section) code.

The total speedup ratio of vectorized version to the original scalar one is 1.7~1.9 for DOT3.5 NEA version, 2.2~2.3 for DOT3.5 FNS version, 1.7 for DOT3.5 FER version, and 3.1~4.4 for RADHEAT-V4, respectively.

The elapsed times for improved DOT3.5 FNS version and DOT-DD are reduced to 50~65% that of the original version by the input/output speedup.

+ Department of Reactor Engineering

* On leave from FUJITSU, LTD.

In this report, we describe summary of codes, the techniques used for the vectorization and input/output speedup, verification of computed results, and speedup effect.

Keywords: Vectorization, Nuclear Codes, Radiation Transport, Sn Method, DOT3.5, Supercomputing, Supercomputer, Computer Codes, FACOM VP-100, DOT-DD, RADHEAT-V4

目 次

1.はじめに	1
2. DOT 3.5 NEA オリジナル版	5
2.1 コードの概要	5
2.2 DOT 3.5 旧ベクトル化版	6
2.3 動的挙動分析	8
2.4 ベクトル化方法	8
2.4.1 サブルーチン GRIND	8
2.4.2 サブルーチン OUTER	9
2.4.3 ルジャンドル展開次数の無制限化	9
2.5 計算結果の評価	10
2.6 ベクトル化効果	10
3. DOT 3.5 FNS 版	26
3.1 コードの概要	26
3.2 動的挙動分析	26
3.3 ベクトル化方法	26
3.4 計算結果の評価	26
3.5 ベクトル化効果	27
3.6 I/O の高速化手法	27
3.7 I/O 高速化の効果	28
4. DOT 3.5 炉設計版	35
4.1 コードの概要	35
4.2 動的挙動分析	35
4.3 ベクトル化方法	35
4.4 計算結果の分析	35
4.4.1 計算結果の評価	35
4.4.2 計算結果の相違原因について	36
4.5 ベクトル化効果	37
5. RADHEAT-V 4	42
5.1 システムの概要	42
5.2 動的挙動分析	42
5.3 ベクトル化方法	43
5.3.1 サブルーチン CSCAT	43
5.3.2 サブルーチン GRIND	43
5.4 計算結果の評価	44

5.5 ベクトル化効果	44
6. DOT-DD	54
6.1 コードの概要	54
6.2 I/O の高速化手法	55
6.3 I/O 高速化の効果	56
7. ベクトル化版留意事項	59
7.1 使用制限事項	59
7.2 メモリの増加について	60
8. おわりに	61
謝　　辞	61
参考文献	62
付録 A DOT 3.5 NEA オリジナル版使用手引	63
付録 B DOT 3.5 FNS 版使用手引	65
付録 C DOT 3.5 炉設計版使用手引	68
付録 D RADHEAT-V 4 使用手引	71
付録 E DOT-DD 使用手引	73

Contents

1.	Introduction	1
2.	DOT3.5 NEA original version	5
2.1	Outline	5
2.2	Old vector version of DOT3.5 code	6
2.3	Dynamic behavior of DOT3.5 NEA version	8
2.4	Vectorization techniques	8
2.4.1	Subroutine GRIND	8
2.4.2	Subroutine OUTER	9
2.4.3	Eliminating the limitation of Legendre expansion degree ..	9
2.5	Verification of computed results	10
2.6	Vectorization effect	10
3.	DOT3.5 code FNS version	26
3.1	Outline	26
3.2	Dynamic behavior of DOT3.5 FNS version	26
3.3	Vectorization techniques	26
3.4	Verification of computed results	26
3.5	Vectorization effect	27
3.6	Method of I/O speedup	27
3.7	I/O speedup effect	28
4.	DOT3.5 code FER version	35
4.1	Outline	35
4.2	Dynamic behavior of DOT3.5 FER version	35
4.3	Vectorization methods	35
4.4	Comparison of computed results	35
4.4.1	Verification of computed results	35
4.4.2	Reason for numerical difference	36
4.5	Vectorization effect	37
5.	RADHEAT-V4 code system	42
5.1	Outline	42
5.2	Dynamic behavior of RADHEAT-V4	43
5.3	Vectorization methods	43
5.3.1	Subroutine CSCAT	43
5.3.2	Subroutine GRIND	43
5.4	Verification of computed results	44
5.5	Vectorization effect	44
6.	DOT-DD code	54

6.1 Outline	54
6.2 Method of I/O speedup	55
6.3 I/O speedup effect	56
7. Relevant information on the vector versions	59
7.1 Constraints in use of vector versions	59
7.2 Increase of required memory	60
8. Concluding remarks	61
Acknowledgements	61
References	62
APPENDIX A User's guide for vectorized DOT3.5 code	
NEA original version	63
APPENDIX B User's guide for vectorized DOT3.5 code	
FNS version	65
APPENDIX C User's guide for vectorized DOT3.5 code	
FER version	68
APPENDIX D User's guide for vectorized RADHEAT-V4	
code system	71
APPENDIX E User's guide for vectorized DOT-DD code	73

1. はじめに

原研情報システムセンターでは、スーパーコンピュータを有効利用するために原子力コードのベクトル化を実施してきた。過去2年間で炉物理関連コードとしては、SRAC, CITATION, CITATION-FBR, PHENIX, FPGS^{(1),(2)}, CITATION HTTR 版などが実施された。1990年初めから、原研で多くの分野でよく使用されている2次元ディストリート・オーディネート放射線輸送コード DOT 3.5 のベクトル化と入出力処理の高速化を実施した。

DOT 3.5 コードは、原研において以前石黒らによってベクトル化が実施されていた。この DOT 3.5 の旧ベクトル化版では、計算のカーネル部分の漸化式が充分ベクトル化されないまま残っていた。漸化式を完全にベクトル化する方法としては、斜め方向のメッシュを同時に計算する、つまり、 $i+j+k=一定$ のメッシュ点 (i, j, k) に対して同時に計算する Hyperplane 法が有効であり、動燃では、TWOTRAN-II コードや⁽³⁾、DOT 4.2 コード⁽⁴⁾を Hyper plane 法を採用してベクトル化し、4倍程度の高速化を図っている。

原研においては、DOT コードについては、継続性を保つ必要から DOT 4.2 よりも DOT 3.5 が多用されている。今回の DOT 3.5 ベクトル化改良版においては、本来ならば、Hyperplane 法によるベクトル化を図るべきであるが、プログラム構造を大きく変更する必要があり、作業工数を考慮して Hyperplane 法の採用には致っていない。このため、カーネル・プログラムの一部分がスカラー計算で残っており、速度向上倍率が2~4倍と低い原因となっている。今後は DOT 3.5 の後続版の利用に移っていくと思われる多様な DOT コードのベクトル化に際しては、Hyperplane 法によるベクトル化を採用していく予定である。

今回は、このベクトル化版の普及を目的として、原研で各研究室で個別に改良されたいくつのバージョンに対しベクトル化版を利用できるようにした。具体的には、旧ベクトル化バージョンに対するベクトル化の強化とエラー修正、速度向上効果の測定、ベクトル化版の計算結果の検証、主記憶内入出力機能 (VIO/F 機能⁽⁵⁾) を利用した入出力の高速化などを行った。

ベクトル化の対象となったのは、ORNL で開発され OECD/NEA から入手したオリジナル版をはじめ、核融合中性子源計算のための DOT 3.5 FNS 版、核融合炉設計のための DOT 3.5 FER 版、遮蔽安全評価群定数作成及び放射線輸送解析コードシステム RADHEAT-V 4 の ESPRIT モジュールに組み込まれている DOT 3.5 コードである。

入出力処理の短縮化は、極めて多くの群を対象とする場合にユーザにとって非常にメリットがある。そこで、DOT 3.5 FNS 版と DOT 3.5 の二重積分群定数を使用する DOT-DD に対して入出力処理の短縮化を実施した。なお、DOT-DD のベクトル化版組み込みは既に筒井によってなされていた。

本報告では、最近実施したベクトル化の改良のみならず、刊行されていなかった旧ベクトル化版のベクトル化方法についても概要を示す。

今回の作業でベクトル化されたコードの CPU 時間、速度向上倍率、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スカラー計算、下

段はベクトル化版ベクトル計算である。ここで、オリジナル版スカラー計算とはオリジナル・プログラムを VP-100 でスカラー計算した（すべてのサブルーチンを FORTRAN 77 でコンパイルしたものを実行した）ものであり、ベクトル化版ベクトル計算とはベクトル化したプログラムをベクトル計算した（すべてまたは一部のサブルーチンを FORTRAN 77 /VP でコンパイルしたものを実行した）ものである。ベクトル化により、CPU 時間が DOT 3.5 では 1.7 倍～1.9 倍に、DOT 3.5 FNS 版では 2.2 倍～2.3 倍に、DOT 3.5 炉設計版では 1.7 倍に、RADHEAT-V 4 では 3.1 倍～4.4 倍に高速化された。

さらに今回の作業で I/O 高速化が施されたコードの経過時間、CPU 時間、I/O 回数等を Table 1.2 に示す。I/O 高速化により、経過時間が、DOT 3.5 FNS 版では 51.5%～65.9% に、DOT-DD で 50.1% に減少した。

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	形状	人力データの概要				Sn数	CPU時間(秒)	VU時間(秒)	倍率	ベクトル化率(%)	メモリ(KB)	I/O回数
		計算モデル*	群数	メッシュ	ルジャンドル展開次数							
DOT 3.5	R-Z	3	63	51×54	3	48	228.91	—	—	—	1412	21826
	R-Z	4	63	51×54	3	48	119.97	45.85	1.91	66.9	2968	21825
	X-Y	3	63	51×54	3	48	121.50	51.48	1.79	68.7	2968	21825
	X-Y	4	63	51×54	3	48	110.79	38.90	1.90	66.4	2968	21825
DOT 3.5 FNS 版	R-Z	4	125	40×86	5	48	191.13	—	—	—	1412	21825
	R-Z	4	125	40×86	5	160	109.81	45.47	1.74	66.7	2968	21825
	DOT 3.5 炉内設計版	R-Z	3	42	13×10	3	48	1703.43	—	—	4556	9009
	R-Z	3	9	41×29	—	48	767.84	525.77	2.22	86.2	6680	9008
RADHEAT -V4	R-Z	0	9	41×29	—	48	4444.94	—	—	—	4556	9497
	R-Z	4	9	41×29	—	48	1951.81	1272.84	2.28	87.2	6680	9497
	R-Z	4	9	41×29	—	48	83.76	—	—	—	1776	1804
	R-Z	3	9	41×29	—	48	49.60	23.72	1.69	66.2	3328	1803
RADHEAT -V4	R-Z	0	9	41×29	—	48	186.95	—	—	—	2012	2781
	R-Z	4	9	41×29	—	48	51.47	24.75	3.63	85.7	2580	2781
	R-Z	3	9	41×29	—	48	232.43	—	—	—	2012	2781
	R-Z	4	9	41×29	—	48	75.82	11.54	3.07	72.3	2580	2781
*) ダイアモンド差分式における内挿方式 0 : mixed linear-step 3 : weighted difference 4 : mixed linear-weighted												

Table 1.2 I/O speedup effect

上段：オリジナル版

下段：I/O高速化版

コード名	入力データの概要				経過時間	CPU時間	I/O回数	拡張領域 使用量 (MB)	コメント
	形状	群数	メッシュ	ルジャンドル 展開次数					
DOT3.5 FNS 版	R-Z	125	40×86	5	48	43分 42.54秒*	17分 00.90秒	20207	0 HI/O使用
	R-Z	125	40×86	5	160	67分 53.24秒*	37分 54.20秒	9011	40 VIO/F使用
DOT-DD	R-Z	125	16×34	—	48	90分 47.41秒**	5分 23.35秒	21762	0 HI/O使用
						45分 30.27秒**	5分 11.10秒	9506	47 VIO/F使用
								166683	0
								4908	15 VIO/F使用

* : 1多重で実行

** : 日中実行したものであり1多重でない

HI/O : 高速入出力機能

2. DOT 3.5 NEA オリジナル版

2.1 コードの概要

DOT 3.5 コード⁽⁶⁾は、オークリッジ国立研究所で開発された 2 次元ディスクリート・オーディネート輸送コードである。

本コードは、多群の中性子輸送方程式をディスクリート・オーディネート法という差分法により離散化している。 $(x-y)$ 体系、 $(r-z)$ 体系、 $(r-\theta)$ 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、中性子の散乱及び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 (2.1) 式である。

$$\begin{aligned} \frac{d\phi(r, z, \theta, \eta, \psi, E)}{ds} + \Sigma^T(r, z, \theta, E) \phi(r, z, \theta, \eta, \psi, E) \\ = S(r, z, \theta, \eta, \psi, E) \\ + \int_{-1}^1 \int_0^{2\pi} \int_0^\infty \Sigma^S(r, z, \theta; E', \bar{\Omega}' \rightarrow E, \bar{\Omega}) \phi(r, z, \theta, \eta', \psi', E') dE' d\psi' d\eta' \end{aligned} \quad (2.1)$$

ここで ϕ は中性子束、 s は空間及び角度方向を表わし r, z, θ, η で与えられる。 S は中性子源を表わす。

r, z, η, ψ, E に対するインデックスを (i, j, K, n, g) とおけば、差分近似により、以下の式が導出される。

$$\begin{aligned} \phi_{G,I,J,D} = & (|\bar{\mu}_D| 2\pi \bar{r}_I \Delta z_J \left(\text{or} \frac{\phi_{G,i+1,J,D}}{\phi_{G,i,J,D}} \right) + |\bar{\eta}_D| 2\pi \bar{r}_I \Delta \bar{r}_I \left(\frac{\phi_{G,I,j+1,D}}{\phi_{G,I,J,D}} \right) \\ & + \frac{\Delta z_J}{W_D} \bar{\gamma}_N \phi_{G,I,J,n,K} + \frac{1}{2} V_{I,J} S'_{G,I,J,D}) / (|\bar{\mu}_D| \Delta z_J 2\pi \bar{r}_I + |\bar{\eta}_D| 2\pi \bar{r}_I \Delta \bar{r}_I \\ & + \frac{\Delta z_J}{W_D} \bar{\gamma}_N + \frac{1}{2} V_{I,J} \Sigma_G^T), \bar{r}_N = \frac{1}{2} (r_{n+1} + r_n) \end{aligned} \quad (2.2)$$

ここで、 I, J, D は、各々 $\left(i - \frac{1}{2}, i + \frac{1}{2}\right)$, $\left(j - \frac{1}{2}, j + \frac{1}{2}\right)$ 及び $\left(n - \frac{1}{2}, n + \frac{1}{2}; K\right)$ の区間を示す。また、どちらの端から計算されるかによって上段または下段の計算式が使用される。

(2.2) 式から $\phi_{G,I,J,D}$ を求め、以下の (2.3) 式からメッシュ (i, j, n, g) 上の ϕ を補間する。

$$\begin{aligned} \begin{bmatrix} \phi_{G,i,j,D} \\ \phi_{G,i+1,j,D} \end{bmatrix} &= 2\phi_{G,I,J,D} - \begin{bmatrix} \phi_{G,i-1,j,D} \\ \phi_{G,i,j,D} \end{bmatrix} \\ \begin{bmatrix} \phi_{G,I,j,D} \\ \phi_{G,I,j+1,D} \end{bmatrix} &= 2\phi_{G,I,J,D} - \begin{bmatrix} \phi_{G,I,j+1,D} \\ \phi_{G,I,j,D} \end{bmatrix} \\ \phi_{G,I,J,n+1,K} &= 2\phi_{G,I,J,D} - \phi_{G,I,J,n,K} \end{aligned} \quad (2.3)$$

(2.2)–(2.3) 式の計算は, “power iteration” 法により実行される。 (2.2)–(2.3) 式は, 例えば (r, z) 形状の場合には r 方向インデックス (i) , z 方向のインデックス (j) および角度方向インデックス (n) に対する漸化式を成す。

この中性子束計算における反復計算は, 多重の反復構造となっている。まず, エネルギー間の上方散乱や核分裂による中性子源については外側反復 (outer iteration) で, 自群内散乱の中性子源については内側反復 (inner iteration) で中性子束変動効果を取り入れている。プログラムでは, この 2 つのループ以外に, 群のループ, 角度 (M) のループ, メッシュ (I 及び J) のループがある (Fig. 2.1 参照)。

2.2 DOT 3.5 旧ベクトル化版

原研における DOT 3.5 コードのベクトル化は, まず 1982 年に石黒ら⁽¹⁾により行われた。ここで用いたベクトル化手法は, 次の通りである。

m : 内部反復回数
 $f \rightarrow g$: g は f を用いて計算される
 とおく。

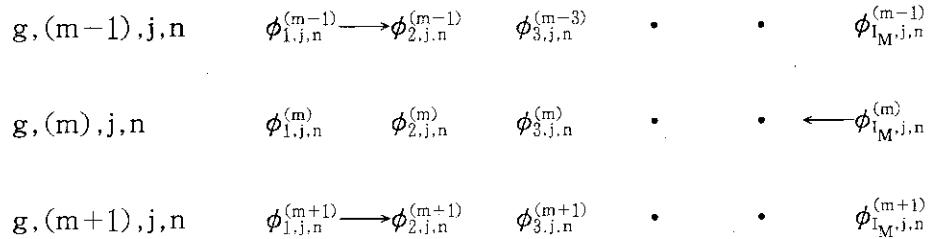
(1) スカラーフィールド計算

$$g, (m), j, n \quad \phi_{1,j,n}^{(m)} \longrightarrow \phi_{2,j,n}^{(m)} \longrightarrow \phi_{3,j,n}^{(m)} \longrightarrow \dots$$

$\left(\begin{array}{c} \text{外側ループ} \\ \text{インデックス} \end{array} \right)$

$$g, (m+1), j, n \quad \phi_{1,j,n}^{(m+1)} \leftarrow \phi_{2,j,n}^{(m+1)} \leftarrow \dots \leftarrow \phi_{I_M,j,n}^{(m+1)}$$

○の値が与えられ, 右向き左向き交互に計算される。 (2.4)

(2) ベクトル計算 ($\phi^{(m)}$ の代わりに $\phi^{(m-1)}$ を利用)

この手法は、漸化式の回帰的参照関係をなくすために、前回の反復で計算した値を用いることによりベクトル化を可能としている。しかし、前回の反復で計算した値を用いたことにより、収束が遅くなったり、正しく収束しなかったりした。そこで、2重反復計算の外側反復計算の初回をスカラー計算し、2回目からベクトル計算を行う方法を用いることにより正しく収束し、さらに臨界計算では2倍程度の高速化も期待できることが確認された。しかし、DOT 3.5 コードの計算の主流である遮蔽計算では外側反復計算は1回しか行われないために全く高速化されない。

そこで、その後この手法をやめ、回帰計算部分はそのままスカラー計算し、それ以外の部分についてベクトル計算するバージョンを作成した。

このベクトル化版が今回の作業の基となったバージョンである（今後このバージョンを旧ベクトル化版とよぶ）。以下で旧ベクトル化版のベクトル化手法を述べる。

① ループ分割によるスカラー部分の切分け

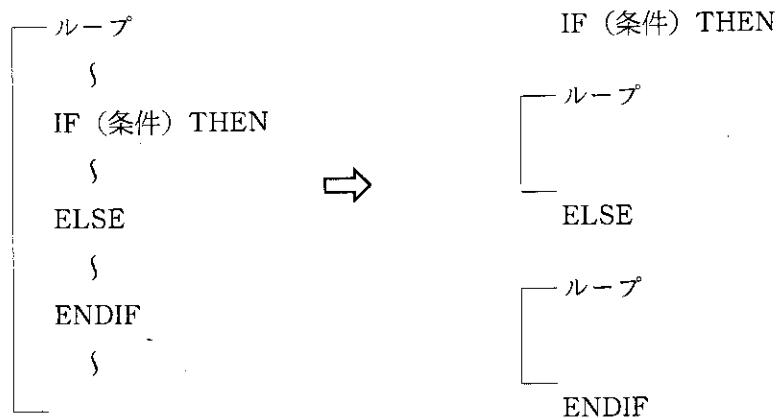
旧ベクトル化版ではサブルーチン GRIND をベクトル化するためにループを3つに分割し、このうちの2つ（1番目と3番目）についてベクトル計算し、2番目のループについてはスカラー計算している。1番目のループは、漸化式の各係数を計算する部分であり、2番目は漸化式（中性子束計算）の部分であり、3番目は中性子束を使って他の値を計算する部分である。

② ループ統合によるベクトル長の拡大

サブルーチン GRIND は、Fig. 2.1 に示すように3重ループ（外側から J ループ、M ループ、I ループ）構造になっている。このうちの内側の2重ループを1重化し、ベクトル長の拡大化を図っている。

③ サブルーチン分割による IF 文のループ外への移動

オリジナル版には多くの IF 文がループ中にあり高速化を阻害している。これらの IF 文のうちループ内で判定が変わらないものについては次のようにループ外に移動し、高速化することが可能である。



旧ベクトル化版ではルジャンドル展開次数、形状、計算モデルに対する IF 文をループの外に移動し、各々のループを 1 つのサブルーチンとした。この修正によりサブルーチン GRIND が 8 種類になった。8 種類のサブルーチンを Table 2.1 に示す。この表以外のルジャンドル展開次数、形状、計算モデルの場合、旧ベクトル化版では計算できない。プログラムではメッセージを出力して終了する。

2.3 動的挙動分析

FORTUNE ツールを用いて CPU 時間分布の動的挙動分析を実施した。動的挙動分析に使用するデータは、Table 2.2 に示すように、2 次元 R-Z 形状の遮蔽計算である。Fig. 2.2 に分析結果を示す。Fig. 2.2 を見ると、GRIRZ (GRIND を条件により分割したサブルーチンの 1 つ) が 79.8%，OUTER が 9.3%，WWESOL が 8.5% であり、これらの 3 ルーチンで全体の 97.6% を占める。

したがって、これらのサブルーチンを中心にチューニングを実施した。

2.4 ベクトル化方法

今回、新たにベクトルチューニングを試みたサブルーチンは、旧 GRIND, OUTER である。また、ベクトル化版でのルジャンドル展開次数の制限をなくした。

2.4.1 サブルーチン GRIND

オリジナル版での GRIND は、ベクトル化版では形状、計算モデル、ルジャンドル展開次数により 8 種類のサブルーチンに分かれている。

新しいベクトル化手法を GRIRZ を用いて説明する。今までに行われたベクトル化によって I ループは 3 つに分割されている。これらのループのうち、1 番目のループは完全にベクトル化されており、2 番目のループは全くベクトル化されていない。これに対して 3 番目のループ（ループ番号 13063）は、Fig. 2.3 のようにベクトル化部分と非ベクトル化部分が混じっている。このループが完全にベクトル化されない理由は、変数 NMJ の参照が定義より前にあるためである。

AIFLUX を計算しているこの部分は、他の部分と完全に切り離せる。そこで、この部分以外を今まで通り計算し、この部分だけ別のループにする。ループ分割した結果を Fig. 2.4 に示す。これを見ると、すべての部分でベクトル化されていることが解る。

他の 7 サブルーチンについても同様にベクトル化することができる。

この改良により、ループ 13063 部分は、14.23 秒から 4.65 秒に高速化された。

2.4.2 サブルーチン OUTER

今までサブルーチン OUTER は、全くベクトル化されていなかった。しかし、FORTUNE のコスト分布を見ると 9.3% と高い比率になっている。

OUTER の中でコストの高いのは、Fig. 2.5 に示すようにループ No. 7200 であり、全体の 97.48% を占めている。このループは、その内部に 2 つのループがあり、ベクトル長はそれぞれ 2 及び 1 と短い。そのため、このままでは高速化が図れない。

そこで外側の I のループ（ベクトル長 2754）を K 及び L のループの内側に入れることを考える。

手順は次の通りである。

- ① ループ内で値の変わらない変数 KO1A をループの外に出し、その値により 2 通りの処理を行うようとする。
- ② KO1A が 4624 の場合の処理を次のように修正する。まず、I のループを 4 分割（IF 文及びループが区切り）する。IF 文がある場合、真になる I の番号を保存し、それ以降は、その保存した番号の計算のみ行うようとする。また、変数 ITEMP1 が負になることはほとんどない（このデータの場合なし）ので、1 から IMJMまでの間でまったく真にならなかった場合の処理（リストベクトルを用いず、連続アクセスになる）を追加する。

以上の手順の後、ベクトル化したものを Fig. 2.6 に示す。

サブルーチン OUTER をそのままベクトル計算する（FORTRAN 77-VP で、コンパイルしたものを行なう）と計算結果が一致しない。この原因を調べたところ、OUTER の中にある総和計算が原因であった。このループの計算時間は短いので、このループの計算をスカラーで実行するように修正した（Fig. 2.7 参照）。この修正により、サブルーチン OUTER 全体をベクトル計算しても数値が一致するようになった。

上記改良の成果を Table 2.3 に示す。これによると、全 CPU 時間で 22.6 秒の短縮であり、ループ 7200 の部分では 3 倍の高速化が図られている。

2.4.3 ルジャンドル展開次数の無制限化

旧ベクトル化版では、ルジャンドル展開次数は 3 次または 5 次 ($A03 = 3 \text{ or } 5$) のみ有効であり、他の次数の計算はできなかった。新ベクトル化版では、この制限を取り除き、しかも計算時間（CPU 時間）は旧版の場合とほとんど変わらないよう修正を加えた。

修正を加えたサブルーチンは、INNER と GRIND (GRIXY, GRIXY5, GRIRZ, GRIRZ5, GRIXY4, GRIXY45, GRIRZ4, GRIRZ45) である。

① サブルーチン INNER

旧ベクトル化版のサブルーチン INNER では、ルジャンドル展開の次数 ($A03 = 3$ or 5)、形状 ($IGE = 0$ or 1)、計算モデル ($FXT = 3$ or 4) により 8 種類のサブルーチンを呼び出している (Fig. 2.8 参照)。それに対し、ルジャンドル展開次数の制限をなくした新ベクトル化版では、4 種類のサブルーチンのみを呼び出せばよい (Fig. 2.9 参照)。

② サブルーチン GRIND

他のサブルーチンでも同様であるので GRIRZ について説明する。

GRIRZ の中でルジャンドル展開次数に関係した部分は、Fig. 2.10 に示すように ZSOR の計算部分である。ZSOR に加えられる P_3 と S_3 の積の数がジャンドル展開次数により変わる。これをループにし、IQ のループの外に出したのが、ルジャンドル展開次数無制限版 (Fig. 2.11) である。このようにすると GRIRZ と GRIRZ 4 が全く同じになり、1 つを省略できる。

2.5 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 4 ケースの試計算を実施した。試計算データは、動的挙動分析に用いたものとほとんど同じであるが、形状が 2 種類、計算モデルが 2 種類に分かれている (Table 2.4, 2.5 参照)。計算結果を Table 2.6 に示す。Table 2.6 によると各値とも有効桁 3 術の範囲では完全に一致している。DOT 3.5 コードの計算が単精度であり、中性子束の収束条件が 10^{-1} であることから十分許容範囲内であると判断できる。

なお、Table 2.6 で選んだ群番号及びメッシュ番号は、任意に選んだものであるが、他の群、メッシュでも同様であり、有効桁 3 術までは完全に一致している。

今回ベクトル化したサブルーチン GRIND は、形状、計算モデルにより 4 種類ある。これらの 4 サブルーチンのテストを上記 4 ケースで実施しているので、試計算はこれで十分と考える。

2.6 ベクトル化効果

Table 2.2 で示した問題についてオリジナル版及び旧ベクトル化版に対してのベクトル化効果を測定した。その結果を Table 2.7 に示す。これによると新ベクトル化版ベクトル計算は、オリジナル版スカラー計算に比べて 1.91 倍に高速化され、旧ベクトル化版ベクトル計算に比べても 1.25 倍に高速化された。

また、Table 2.4, 2.5 で示した 4 ケースについてオリジナル版に対してのベクトル化効果を測定した。その結果を Table 2.8 に示す。これによるとベクトル化版ベクトル計算は、オリジナル版スカラー計算に比べて 1.74 ~ 1.91 倍に高速化され、ベクトル化版スカラー計算に比べて 1.76 ~ 1.93 倍に高速化された。

Table 2.1 Eight versions of subroutine GRIND

FXT	A03	IGE	サブルーチン
4	3	0	GRIXY4
4	5	0	GRIX45
4	3	1	GRIRZ4
4	5	1	GRIR45
3	3	0	GRIXY
3	5	0	GRIXY5
3	3	1	GRIRZ
3	5	1	GRIRZ5

A03：散乱断面積の最大ルジャンドル展開次数

IGE：2次元形状の指定(0:X-Y 形状, 1:R-Z 形状)

FXT：計算モデル

3: weighted-difference mode

4: linear+weighted-difference mode

Table 2.2 Input data used for dynamic profile analysis
(DOT3.5 NEA version)

項目	値
計算条件	固定線源問題
形状	R-Z 体系
計算モデル	weighted-difference モード
メッシュサイズ	51×54
最大ルジャンドル展開次数	3
S _n 角度分点数	48
領域数	15
群数	63

Table 2.3 Vectorization effect of subroutine OUTER

計算種別	CPU時間
全計算時間(旧ベクトル化版ベクトル計算+改良前OUTER スカラー計算)	2分 30.17秒
全計算時間(旧ベクトル化版ベクトル計算+改良後OUTER ベクトル計算)	2分 7.54秒
改良前ループ7200部分スカラー計算	32.67秒
改良前ループ7200部分ベクトル計算	41.36秒
改良後ループ7200部分ベクトル計算	10.78秒

Table 2.4 Common input data for test runs
(DOT3.5 NEA vector version)

項目	値
計算条件	固定線源問題
メッシュサイズ	51×54
最大ルジャンドル展開次数	3
S _N 角度分点数	48
領域数	15
群数	63
収束判定条件	0.1
pointwise flux収束判定値	0.1

Table 2.5 Input data of geometry type, and calculation model for test runs
(DOT3.5 NEA vectorized version)

ケースNo.	形状	計算モデル
1	R-Z	weighted difference mode
2	R-Z	mixed linear-weighted mode
3	X-Y	weighted difference mode
4	X-Y	mixed linear-weighted mode

Table 2.6 Comparison of computed results between the original and vector versions (DOT3.5 NEA version)

ケース No.	項 目	群番号	オリジナル版 スカラー計算	ベクトル化版 ベクトル計算
ケ ー ス 1	IN-SCATTER	1	3.27826×10^{-6}	3.27826×10^{-6}
	SELF-SCATTER	30	3.43379×10^{-1}	3.43378×10^{-1}
	OUT-SCATTER	62	1.29840×10^{-1}	1.29838×10^{-1}
	ABSORPTIONS	10	7.13479×10^{-4}	7.13475×10^{-4}
	RT-LEAKAGE	30	1.14704×10^{-5}	1.14706×10^{-5}
	VT-LEAKAGE	63	-2.25460×10^{-6}	-2.25476×10^{-6}
	TOP-LEAKAGE	20	6.40317×10^{-3}	6.40326×10^{-3}
	BOT-LEAKAGE	50	6.47104×10^{-6}	6.47102×10^{-6}
	NET-LEAKAGE	1	6.75396×10^{-1}	6.75425×10^{-1}
ケ ー ス 2	IN-SCATTER	1	3.27826×10^{-6}	3.27826×10^{-6}
	SELF-SCATTER	30	3.44395×10^{-1}	3.44377×10^{-1}
	OUT-SCATTER	62	1.29389×10^{-1}	1.29386×10^{-1}
	ABSORPTIONS	10	7.01288×10^{-4}	7.01259×10^{-4}
	RT-LEAKAGE	30	8.40841×10^{-6}	8.40323×10^{-6}
	VT-LEAKAGE	63	-2.20612×10^{-6}	-2.20621×10^{-6}
	TOP-LEAKAGE	20	6.09982×10^{-3}	6.09971×10^{-3}
	BOT-LEAKAGE	50	6.44905×10^{-6}	6.44903×10^{-6}
	NET-LEAKAGE	1	6.82328×10^{-1}	6.82349×10^{-1}
ケ ー ス 3	IN-SCATTER	1	4.17233×10^{-7}	4.17233×10^{-7}
	SELF-SCATTER	30	1.89746×10^{-1}	1.89729×10^{-1}
	OUT-SCATTER	62	7.17360×10^{-2}	7.17298×10^{-2}
	ABSORPTIONS	10	4.05912×10^{-4}	4.05892×10^{-4}
	RT-LEAKAGE	30	5.28014×10^{-6}	5.27987×10^{-6}
	VT-LEAKAGE	63	-1.28576×10^{-6}	-1.28565×10^{-6}
	TOP-LEAKAGE	20	4.23697×10^{-3}	4.23687×10^{-3}
	BOT-LEAKAGE	50	3.82237×10^{-6}	3.82218×10^{-6}
	NET-LEAKAGE	1	8.06392×10^{-1}	8.06417×10^{-1}
	FLUXメッシュNo.(1, 1)	1	2.03942×10^{-2}	2.03943×10^{-2}
	FLUXメッシュNo.(51, 54)	1	6.86982×10^{-6}	6.86805×10^{-6}
	FLUXメッシュNo.(20, 54)	2	9.85366×10^{-5}	9.85304×10^{-5}
	FLUXメッシュNo.(30, 30)	2	2.22759×10^{-4}	2.22738×10^{-4}
	FLUXメッシュNo.(1, 1)	3	1.62267×10^{-5}	1.62262×10^{-5}
	FLUXメッシュNo.(1, 54)	3	1.80282×10^{-5}	1.80278×10^{-5}

Table 2.6 (Continued)

ケース No.	項 目	群番号	オリジナル版 スカラ－計算	ベクトル化版 ベクトル計算
ケ 1 ス 3	FLUXメッシュNo.(51, 1)	4	2.63707×10^{-6}	2.63683×10^{-6}
	FLUXメッシュNo.(51,54)	4	7.75942×10^{-7}	7.75865×10^{-7}
	FLUXメッシュNo.(30,30)	5	8.94249×10^{-5}	8.94188×10^{-5}
	FLUXメッシュNo.(51, 8)	6	1.66019×10^{-6}	1.66006×10^{-6}
ケ 1 ス 4	IN-SCATTER	1	4.17233×10^{-7}	4.17233×10^{-7}
	SELF-SCATTER	30	1.93044×10^{-1}	1.93017×10^{-1}
	OUT-SCATTER	62	7.21853×10^{-2}	7.21789×10^{-2}
	ABSORPTIONS	10	4.06639×10^{-4}	4.06610×10^{-4}
	RT-LEAKAGE	30	3.97078×10^{-6}	3.97025×10^{-6}
	VT-LEAKAGE	63	-1.26002×10^{-6}	-1.26005×10^{-6}
	TOP-LEAKAGE	20	4.21276×10^{-3}	4.21250×10^{-3}
	BOT-LEAKAGE	50	3.87325×10^{-6}	3.87306×10^{-6}
	NET-LEAKAGE	1	8.06398×10^{-1}	8.06407×10^{-1}

Table 2.7 Vectorization effect (comparison among original scalar version, old vector version and new vector version of DOT3.5 NEA)

	オリジナル版 スカラー計算	旧ベクトル化版 ベクトル計算	新ベクトル化版 スカラー計算	新ベクトル化版 ベクトル計算
CPU時間(秒)	228.91	150.17	223.60	119.97
VU時間(秒)	0.0	35.00	0.0	45.85
倍率*	1.0	1.52	1.02	1.91
ベクトル化率(%)	—	未算出	—	66.9
I/O回数	21826	21828	21825	21825
メモリ使用量(KB)	1412	2616	2908	2968

*: オリジナル版スカラー計算のCPU時間を、各計算のCPU時間で割ったもの

Table 2.8 Speedup effect (original scalar version to new vector version of DOT3.5 NEA)

ケース	CPU時間 (秒)	VU時間 (秒)	倍率	ベクトル化率 (%)	I/O回数	メモリ使用量 (KB)
1	228.91	—	1.0	—	21826	1412
	223.60	—	1.02	—	21825	2908
	119.97	45.85	1.91	66.9	21825	2968
2	217.96	—	1.0	—	21825	1412
	223.83	—	0.97	—	21826	2908
	121.50	51.48	1.79	68.7	21825	2968
3	210.46	—	1.0	—	21825	1412
	213.89	—	0.98	—	21825	2908
	110.79	38.90	1.90	66.4	21825	2968
4	191.13	—	1.0	—	21825	1412
	193.44	—	0.99	—	21826	2908
	109.81	45.47	1.74	66.7	21825	2968

上段：オリジナル版スカラー計算

中段：ベクトル化版スカラー計算

下段：ベクトル化版ベクトル計算

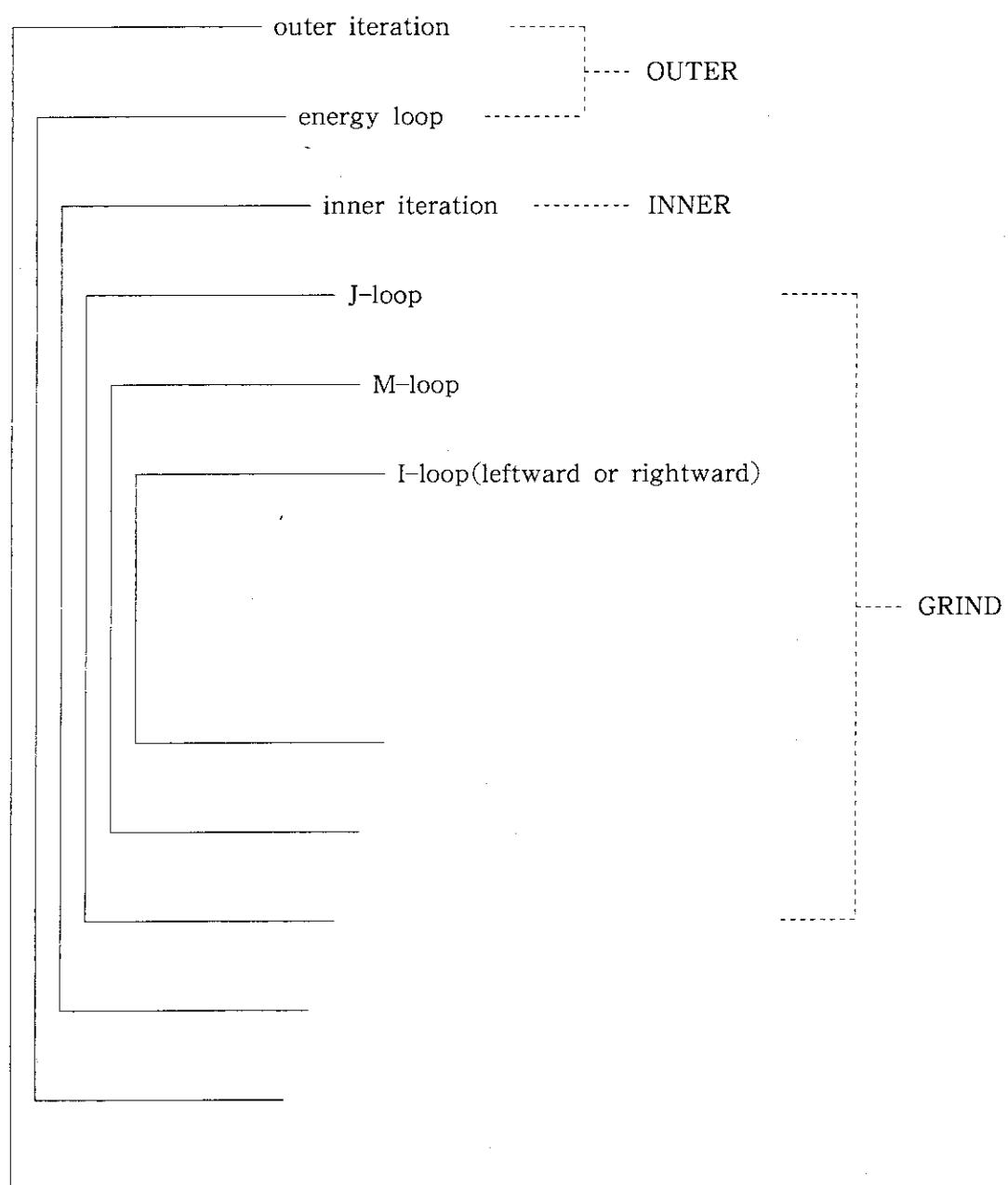


Fig. 2.1 Loop structure of time consuming subroutines
in the DOT3.5 NEA version

I	NO.	ROUTINE	UNITS	LINES	ERR.	EXECUTIONS	COST	X	0.....1....2....3....4....5....6....7.....
I	00001	GRIRZ	1	642	0	126	0.437379E+10	79.8	I*****
I	00002	OUTER	1	334	0	1	0.511245E+09	9.3	I*****
I	00003	WESOL	1	126	0	63	0.64059E+09	8.5	I*****
I	00004	INNER	1	634	0	63	0.787072E+08	1.4	I
I	00005	ACTVY	1	170	0	1	0.78309E+08	0.5	I
I	00006	INIT	1	553	0	1	0.983276E+07	0.2	I
I	00007	WANDR	1	62	0	0	0.374226E+07	0.1	I
I		WANDR4				190			
I		WANDR3				0			
I		WANDR2				2394			
I		WANDR1				1071			
I		WANDR0				0			
I		BKSPCE				0			
I		REWND				45			
I	00008	WOT	1	74	0	66	0.313492E+07	0.1	I
I		WOD				0			
I		WIT				0			
I		WID				0			
I	00009	S862	1	124	0	2	0.250807E+07	0.0	I
I	0010	CMCLR	1	12	0	8	0.192336E+07	0.0	I
I	0011	FFREAD	1	122	0	132	0.680008E+06	0.0	I
I	0012	TIMSTR	1	53	0	2	0.557383E+06	0.0	I
I		TIMON				3973			
I	0013	CLEAR	1	11	0	72	0.431697E+06	0.0	I
I	0014	WOT10	1	58	0	2	0.270337E+06	0.0	I
I	0015	MAPR	1	66	0	1	0.219224E+06	0.0	I
I	0016	ITIME	1	11	0	15892	0.206576E+06	0.0	I
I	0017	IFTIME	1	10	0	0	0.174867E+06	0.0	I
I		ICLOCK				15897			
I	0018	FIOAS	1	257	0	7	0.104708E+06	0.0	I
I	0019	PCON	1	46	0	1	0.541260E+05	0.0	I
I	0020	DOT	1	147	0	1	0.428400E+05	0.0	I
I	0021	FISCAL	1	110	0	2	0.228270E+05	0.0	I
I	0022	LOCO	1	264	0	1	0.170590E+05	0.0	I
I	0023	S8847	1	83	0	2	0.548100E+04	0.0	I
I	0024	S8850	1	137	0	1	0.505300E+04	0.0	I
I	0025	TPXCS	1	20	0	1	0.420200E+04	0.0	I
I	0026	TIMSUM	1	81	0	1	0.420200E+04	0.0	I
I	0027	INP	1	453	0	1	0.382900E+04	0.0	I
I	0028	MESSAGE	1	17	0	1	0.200800E+04	0.0	I
I	0029	CMASGN	1	10	0	95	0.133000E+04	0.0	I
I	0030	TPCOPY	1	131	0	1	0.113100E+04	0.0	I
I	0031	S8830	1	24	0	1	0.102900E+04	0.0	I
I	0032	ANFOOL	1	52	0	63	0.567000E+03	0.0	I
I		ANFOUT				0			
I	0033	IDOT	1	150	0	16	0.307000E+03	0.0	I
I	0034	CLUGE	1	119	0	1	0.206000E+03	0.0	I
I	0035	FIDA	1	17	0	2	0.120000E+03	0.0	I
I		F100				6			

Fig. 2.2 Results of the dynamic profile analysis of DOT3.5 NEA version

```

*VOCL_LOOP,NOVREC(B4,PCL,PCR,PCD,PCU,AJFLUX,AIFLUX,N4,AFLUX,N2)
M DO 13063 I=1,IM
    MIM=M1+I*MM
    IJ=INDE+I
V IF(IPTSCL.EQ.0)GO TO 4663
V IF(XM7M ) 4163,4163,4263
V 4163 IF(I.EQ.IM .AND. IBR.NE.5)GO TO 4363
V PCL(IJ)=PCL(IJ) - ZXMU1(I)*ZA1B2(I)*WOM
V GO TO 4363
V 4263 IF(I.EQ.1.AND.IBL.NE.5) GO TO 4363
V PCR(IJ)=PCR(IJ) - ZXMU1(I)*ZA1B2(I)*WOM
V 4363 IF(XM5M ) 4463,4463,4563
V 4463 IF(J.EQ.JM .AND. IBT.NE.5)GO TO 4663
V PCD(IJ)=PCD(IJ) - ZETAB4(I)*WOM
V GO TO 4663
V 4563 IF(J.EQ.1.AND.IBB.NE.5) GO TO 4663
V PCU(IJ)=PCU(IJ) - ZETAB4(I)*WOM
V 4663 CONTINUE
V IF(WOM .NE.0.0 ) THEN
V FLUXV=BBC1(I)*(ZNBAR(I)-B4(MIM)) + B4(MIM)
V N4(I)=CCC1(I)*(ZNBAR(I)-N4(I))+ N4(I)
V ELSE
V FLUXV=BBC2(I)*(ZNBAR(I)-B4(MIM)) + B4(MIM)
V N4(I)=ZNBAR(I)
V ENDIF
V B4(MIM)=FLUXV
CCCCC N4(I)=FLUXA
***** STORE BOUNDARY ANGULAR FLUX *****
V 1065 IF(IZZ.EQ.0),GO TO 1365
V IF(FLAG.GT.0.0) GO TO 1465
V IF(IZZ.EQ.J) FLUXV=FLUXV+V5(MIM)
V GO TO 1365
V 1465 IF(IZZ.EQ.(J+1)) FLUXV=FLUXV+V5(MIM)
V IF(IB6) 2963,3163,3263
V 1365 CONTINUE
V 2963 IF(FLAG.GT.0.0) GO TO 3163
V IF(J.NE.IB6TMP) GO TO 3163
V IJKL=IT15*I-IT15+NMI
V AJFLUX(IJKL)=FLUXV
V GO TO 3163
V 3263 IF(FLAG.LT.0.0) GO TO 3163
V IF(J.NE.IB6TMP) GO TO 3163
V IJKL=IT14*I-IT14+NMI
V AJFLUX(IJKL)=FLUXV
V 3163 IF(IB5) 3563,1963,3663
V 3563 IF(XM7M.GT.0.0) GO TO 1963
V IF(I.NE.IB5TMP) GO TO 1963
M IJKL=IMMJ+NMJ
V AIFLUX(IJKL)=ZB2MJ(I)
S NMJ=NMJ+1
V GO TO 1963
V 3663 IF(XM7M.LT.0.0) GO TO 1963
V IF(I.NE.IB5TMP) GO TO 1963
M IJKL=JMMJ+NMJ
V AIFLUX(IJKL)=ZB2MJ(I)
M NMJ=NMJ+1
***** WRITE ANGULAR FLUXES *****
V 1963 IF(IAFP.NE.2) GO TO 2463
V IF(XM5M .GT.0.0) GO TO 2263
V AFLUX(I,M)=ZNBAR(I)
V GO TO 2463
V 2263 IMT15=M-T15
V AFLUX(I,IMT15)=ZNBAR(I)
V 2463 CONTINUE
V IJGJI=IJGJ + I
V N2(IJGJI)=N2(IJGJI) + ZNBAR(I)*WOM
V 13063 CONTINUE

```

Fig. 2.3 Program list of DO 13063 in subroutine GRIND of old vector version

```

*VOCL LOOP,NOVREC(B4,PCL,PCR,PCD,PCU,AJFLUX,AIFLUX,N4,AFLUX,N2)
V   DO 13063 I=1,IM
    MIM=M1+I*MM
    IJ=INDE+I
V     IF(IPTSCL.EQ.0)GO TO 4663
V     IF(XM7M ) 4163,4163,4263
V 4163 IF(I.EQ.IM .AND. IBR.NE.5)GO TO 4363
V     PCL(IJ)=PCL(IJ) - ZXMUA1(I)*ZA1B2(I)*WOM
V     GO TO 4363
V 4263 IF(I.EQ.1.AND.IBL.NE.5) GO TO 4363
V     PCR(IJ)=PCR(IJ) - ZXMUA1(I)*ZA1B2(I)*WOM
V 4363 IF(XM5M ) 4463,4463,4563
V 4463 IF(J.EQ.JM .AND. IBT.NE.5)GO TO 4663
V     PCD(IJ)=PCD(IJ) - ZETAB4(I)*WOM
V     GO TO 4663
V 4563 IF(J.EQ.1.AND.IBB.NE.5) GO TO 4663
V     PCU(IJ)=PCU(IJ) - ZETAB4(I)*WOM
 4663 CONTINUE
    IF(WOM .NE.0.0          ) THEN
V     FLUXV=BBC1(I)*(ZNBAR(I)-B4(MIM)) + B4(MIM)
V     N4(I)=CCC1(I)*(ZNBAR(I)-N4(I))+ N4(I)
V     ELSE
V     FLUXV=BBC2(I)*(ZNBAR(I)-B4(MIM)) + B4(MIM)
V     N4(I)=ZNBAR(I)
      ENDIF
    B4(MIM)=FLUXV
CCCCC N4(I)=FLUXA
***** STORE BOUNDARY ANGULAR FLUX *****
V 1065 IF(IZ2.EQ.0) GO TO 1365
V     IF(FLAG.GT.0.0) GO TO 1465
V     IF(IZ2.EQ.J) FLUXV=FLUXV+V5(MIM)
V     GO TO 1365
V 1465 IF(IZ2.EQ.(J+1)) FLUXV=FLUXV+V5(MIM)
V     IF(IB6) 2963,3163,3263
V 1365 CONTINUE
V 2963 IF(FLAG.GT.0.0) GO TO 3163
V     IF(J.NE.IB6TMP) GO TO 3163
      IJKL=IT15*I-IT15+NMI
      AJFLUX(IJKL)=FLUXV
V     GO TO 3163
V 3263 IF(FLAG.LT.0.0) GO TO 3163
V     IF(J.NE.IB6TMP) GO TO 3163
      IJKL=IT14*I-IT14+NMI
      AJFLUX(IJKL)=FLUXV
CNN
V 3163 CONTINUE
C3163 IF(IB5) 3563,1963,3663
C3563 IF(XM7M.GT.0.0) GO TO 1963
C     IF(I.NE.IB5TMP) GO TO 1963
C     IJKL=IMMJ+NMJ
C     AIFLUX(IJKL)=ZB2MJ(I)
C     NMJ=NMJ+1
C     GO TO 1963
C3663 IF(XM7M.LT.0.0) GO TO 1963
C     IF(I.NE.IB5TMP) GO TO 1963
C     IJKL=JMMJ+NMJ
C     AIFLUX(IJKL)=ZB2MJ(I)
C     NMJ=NMJ+1
***** WRITE ANGULAR FLUXES *****

```

Fig. 2.4 Program list of DO 13063 in subroutine GRIND of new vector version

```

V 1963 IF(IAFP.NE.2) GO TO 2463
V      IF(XM5M .GT.0.0) GO TO 2263
V      AFLUX(I,M)=ZNBAR(I)
V      GO TO 2463
2263 IMT15=M-T15
V      AFLUX(I,IMT15)=ZNBAR(I)
2463 CONTINUE
      IJGJI=IJGJ + I
V      N2(IJGJI)=N2(IJGJI) + ZNBAR(I)*WOM
V 13063 CONTINUE
      IF(IB5) 93563,91963,93663
93563 IF(XM7M.GT.0.0) GO TO 91963
      IF(I.NE.IB5TMP) GO TO 91963
*VOCL LOOP,NOVREC(AIFLUX)
V      DO 13070 I=1,IM
      IJKL=IMMJ+NMJ
V      AIFLUX(IJKL)=ZB2MJ(I)
V      NMJ=NMJ+1
V 13070 CONTINUE
      GO TO 91963
93663 IF(XM7M.LT.0.0) GO TO 91963
      IF(I.NE.IB5TMP) GO TO 91963
*VOCL LOOP,NOVREC(AIFLUX)
V      DO 13071 I=1,IM
      IJKL=JMMJ+NMJ
V      AIFLUX(IJKL)=ZB2MJ(I)
V      NMJ=NMJ+1
V 13071 CONTINUE
      91963 CONTINUE

```

Fig. 2.4 (Continued)

Fig. 2.5 Program list of loop structure in subroutine OUTER of original version

```

      GO TO K01A, (2324,4624)
V 2324   DO 7200 I = 1,IMJM
V       ITEMP1=M0(I)
V       ITEMP1=M2(ITEMP1)
V       ITEMP=IABS(ITEMP1)
V       TEMP=CO(PBAR,ITEMP)
V       IF(ITEMP1)2224,2323,2323
V 2224     ITEMP1=-ITEMP1
V 2323     S2(I) = S2(I) + N2(I,IG1)*TEMP
V 7200    CONTINUE
GO TO 45
4624   II = 0
V       DO 10 I = 1,IMJM
V           IZZZ1(I)=M2(M0(I))
V           IZZZ=IABS(IZZZ1(I))
V           ZZZ(I)=CO(PBAR,IZZZ)
V           IF(IZZZ1(I).LT.0) THEN
V               IZZZ1(I)=-IZZZ1(I)
V               II = II + 1
V               IIIN(II) = I
V               ENDIF
V 10    CONTINUE
C
S       DO 20 K=1,K01
S           JJ = 0
S           IF(JJ.EQ.IMJM) THEN
V             DO 23 I=1,IMJM
V                 IZZZ1(I) = IZZZ1(I) + 1
V                 ZZZ1(I)=CO(PBAR,IZZZ1(I))
V                 IF (ZZZ1(I).NE.0.0) THEN
V                     JJ = JJ + 1
V                     JJIN(JJ) = I
V                     ENDIF
V 23    CONTINUE
S           ELSE
*VOCL LOOP,NOVREC
V             DO 21 IW=1,II
V                 I = IIIN(IW)
V                 IZZZ1(I) = IZZZ1(I) + 1
V                 ZZZ1(I)=CO(PBAR,IZZZ1(I))
V                 IF (ZZZ1(I).NE.0.0) THEN
V                     JJ = JJ + 1
V                     JJIN(JJ) = I
V                     ENDIF
V 21    CONTINUE
ENDIF
S           LL=K+1
S           DO 30 L=1,LL
V               JKL=(K*(K+1))/2 +L -1
*VOCL LOOP,NOVREC
V             DO 31 IW=1,JJ
V                 I = JJIN(IW)
V                 S4(I,JKL) = S4(I,JKL) + J3(I,JKL,IG1)*ZZZ1(I)
V 31    CONTINUE
S 30    CONTINUE
S 20    CONTINUE
V             DO 40 I=1,IMJM
V                 S2(I) = S2(I) + N2(I,IG1)*ZZZ(I)
V 40    CONTINUE
45    CONTINUE

```

Fig. 2.6 Program list of loop structure in subroutine OUTER of vector version

```

3524 V11=0.
*VOCL LOOP,SCALAR
DO 3624 I=1,IMJM
S2(I)=S2(I)*V0(I)
3624 V11=V11+S2(I)

```

Fig. 2.7 Vectorization control against summation
in subroutine OUTER

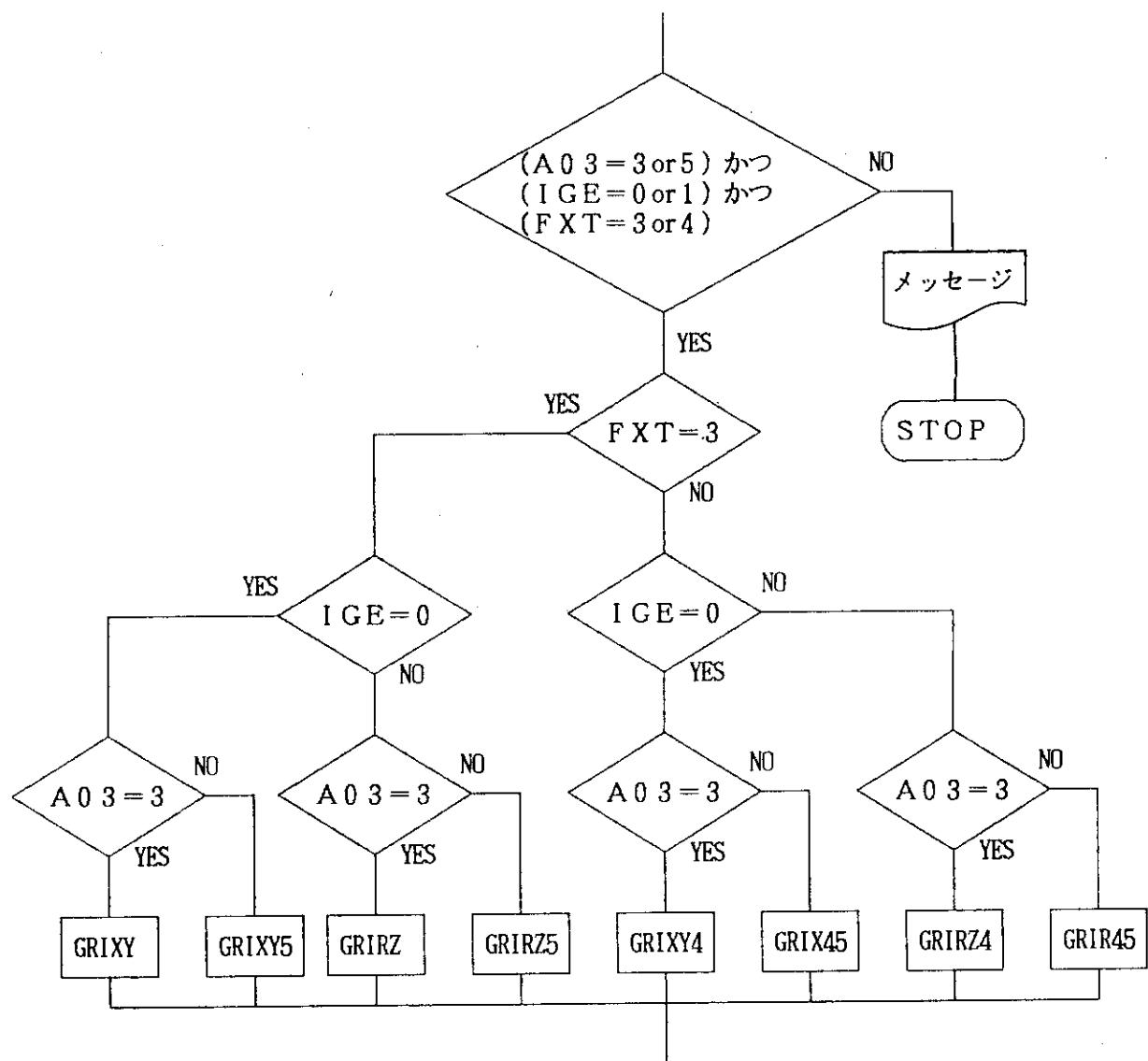


Fig. 2.8 Program flow of calling sequence to subroutine GRIND
in INNER subroutine of old vector version

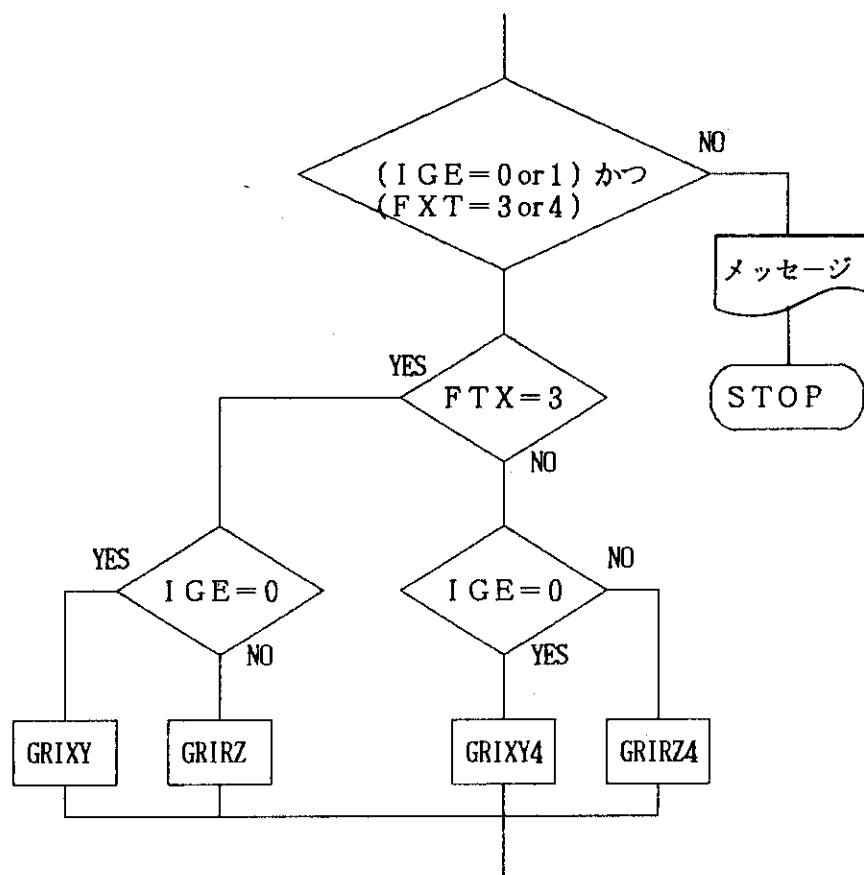


Fig. 2.9 Program flow of calling sequence to subroutine GRIND
in INNER subroutine of new vector version

```

      IF(XM7M.LE.0.) THEN
*VOCL LOOP,NOVREC(ZSOR)
  DO 1084 IQ=1,IM
    I=IP-IQ
    IJ =INDE+I
    ZSOR(I)=ZSOR(I)+P3(M      )*S3(IJ      )
    1      +P3(M+      MM)*S3(IJ+      IMJM)
    2      +P3(M+      2*MM)*S3(IJ+      2*IMJM)
    3      +P3(M+      3*MM)*S3(IJ+      3*IMJM)
    4      +P3(M+      4*MM)*S3(IJ+      4*IMJM)
    5      +P3(M+      5*MM)*S3(IJ+      5*IMJM)
    6      +P3(M+      6*MM)*S3(IJ+      6*IMJM)
    7      +P3(M+      7*MM)*S3(IJ+      7*IMJM)
    8      +P3(M+      8*MM)*S3(IJ+      8*IMJM)

1084 CONTINUE
      ELSE
*VOCL LOOP,NOVREC(ZSOR)
  DO 1085 IQ=1,IM
    I=IQ
    IJ =INDE+I
    CCCCCC ZSOR(I)=ZSOR(I)+P3(M+(K-1)*MM)*S3(IJ+(K-1)*IMJM)
    ZSOR(I)=ZSOR(I)+P3(M      )*S3(IJ      )
    1      +P3(M+      MM)*S3(IJ+      IMJM)
    2      +P3(M+      2*MM)*S3(IJ+      2*IMJM)
    3      +P3(M+      3*MM)*S3(IJ+      3*IMJM)
    4      +P3(M+      4*MM)*S3(IJ+      4*IMJM)
    5      +P3(M+      5*MM)*S3(IJ+      5*IMJM)
    6      +P3(M+      6*MM)*S3(IJ+      6*IMJM)
    7      +P3(M+      7*MM)*S3(IJ+      7*IMJM)
    8      +P3(M+      8*MM)*S3(IJ+      8*IMJM)

1085 CONTINUE
      ENDIF

```

Fig. 2.10 Program list of Legendre expansion in subroutine GRIRZ
of old vector version

```

      IF(XM7M.LE.0.) THEN
  DO 1084 K=1,NOMA
    P3WK=P3((K-1)*MM+M)
    K1WK=(K-1)*IMJM+INDE
*VOCL LOOP,NOVREC(ZSOR)
  DO 1084 IQ=1,IM
    I=IP-IQ
    ZSOR(I)=ZSOR(I)+P3WK*S3(K1WK+I)
1084 CONTINUE
      ELSE
  DO 1085 K=1,NOMA
    P3WK=P3((K-1)*MM+M)
    K1WK=(K-1)*IMJM+INDE
  DO 1085 I=1,IM
    ZSOR(I)=ZSOR(I)+P3WK*S3(K1WK+I)
1085 CONTINUE
      ENDIF

```

Fig. 2.11 Program list of Legendre expansion in subroutine GRIRZ
of new vector version

3 DOT 3.5 FNS 版

3.1 コードの概要

DOT 3.5 FNS 版は、DOT 3.5 コードを原研臨界プラズマ研究部炉設計研究室で改良したものであり、主に原子炉工学部核融合炉物理研究室（FNS）及び遮蔽研究室で使われている。

主な改良点は次の通りである。

- (1) APPLE-2 コード⁽⁸⁾で必要なデータ（DOT 3.5 の形状入力データ及び中性子束）をファイルに出力できるようにした。APPLE-2 コードでは、これらのデータを基に計算領域（ゾンマップ）を表示し、その中性子束、反応率分布等を 2 次元等高線表示する。
- 改良サブルーチンは、OUTER, TPSAVE, TPXF である。
- (2) 群独立配列の ANISN 用断面積を処理して DOT 3.5 で読める形に変換するプリプロセッサーが付加されている。⁽⁹⁾
- (3) 上記(1)と同様な改良により、誘導放射能計算コード THIDA-2⁽⁹⁾で必要なデータ（DOT 3.5 の形状入力データ及び中性子束）をファイルに出力できる機能が付加されている。

DOT 3.5 FNS 版は、ある時期の DOT 3.5 炉設計版である。DOT 3.5 FNS 版はその後改良は全くされていないが DOT 3.5 炉設計版はその後も改良がなされている。

3.2 動的挙動分析

DOT 3.5 FNS 版の主要部分は、DOT 3.5 コードのサブ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。

3.3 ベクトル化方法

DOT 3.5 コードのベクトル化技法と高速化技法を DOT 3.5 FNS 版に適用した。具体的な方法は、2.3 節を参照のこと。

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、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 3.1 に示す。

2 ケースの試計算結果を Table 3.2 に示す。これによると各値とも有効桁 4 術の範囲で完全に一致している。DOT 3.5 コードの計算が单精度であり、中性子束の収束条件が 10^{-2} 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。

また、今回ベクトル化したサブルーチン GRIND は、形状、計算モデルによりベクトル化版で

は4種類ある。今回の試計算では、これらのうち1種類のみについてテストした。他の3種類については、今回はテストしなかったが、NEAオリジナル版では行っている。DOT 3.5 FNS版とNEAオリジナル版では、サブルーチンGRINDは完全に同じである。したがって、試計算は、これで十分と考える。

3.5 ベクトル化効果

Table 3.1で示した問題についてベクトル化効果を測定した。その結果をTable 3.3に示す。これによると新ベクトル化版ベクトル計算は、オリジナル版スカラー計算に比べてケース1で2.22倍、ケース2で2.28倍に高速化され、旧ベクトル化版ベクトル計算に比べてもケース1で1.32倍、ケース2で1.17倍に高速化された。

3.6 I/O の高速化手法

DOT 3.5 FNS版は、DOT 3.5と同様に、2次元形状(X-Y, R-Z, R-θ)で表現したモデルをダイヤモンド差分式によって解いている。

解は、与えられた線源、境界条件等を用いて初期推定から計算を開始し、ある収束条件を満たすまで繰り返し計算を行って求める。この繰り返し操作は、内側反復(inner iteration)と外側反復(outer iteration)の2つがあり、内側反復はそれぞれの空間セル内のすべてのエネルギー群のフラックスについて行われる。

オリジナルでは、計算されたそれぞれの空間セル内のすべてのエネルギー群でのスカラー・フラックスは、プログラム内で保存できないので一時ファイル(論理ユニット番号: NSCRAT)に出力し、使用する毎に入力している。

このファイルの入出力回数は、データにもよるが全入出力回数の5割以上を占め、しかも1回の入出力でのデータ数が他とは比べものにならないほど多い。

そこで、このファイルの全部又は一部についてVIO/F化することを考えた。

スカラー・フラックスの出力回数は群数回(各群とも1回)であるが、入力回数nは

$$n = m(m-1)/2 \quad (m: \text{群数})$$

であり、群毎の入力回数は、1群は(m-1)回、2群は(m-2)回・・・(m-1)群は1回となる。そこで全群がVIO/F化できない場合、1群からできる範囲でVIO/F化する(Fig. 3.1参照)。

VIO/F化する論理ユニット番号は90であり、これを使っているサブルーチンはOUTERとS8850であり、両サブルーチンについて修正を加えた(Fig. 3.2, Fig. 3.3)。

論理ユニット番号90番の1回のI/Oでのレコード長は次式で与えられる。

$$M90 = (IM * JM * (1 + A03 * (A03 + 3) / 2) + MM * (JM + IM)) * IGM * 4$$

IM : R方向メッシュ数

JM : Z方向メッシュ数

A03 : 散乱次数

MM : S_N数

IGM : 群数

また、1群から何群まで VIO/F 化したかを示す情報を付加した (Fig. 3.4, Fig. 3.5 参照)。この出力情報を見れば、VIO/F 化された群数と VIO/F 化された群が I/O 回数に示す割合 (VIO/F 化率) が解る。VIO/F 化率は次式で計算される。

$$V = \frac{(g - g_v) \times (g - g_v - 1)}{g \times (g - 1)} \times 100$$

ここで

V : VIO/F化率(%)

g : 全群数

g_v : VIO/F化された群数

3.7 I/O 高速化の効果

試計算を 2 ケースについて実施した。

各ケースの入力条件を Table 3.4 に示す。

試計算結果を Table 3.5 に示す。

これによるとケース 1 では、I/O 回数が 21762 回から 9506 回へと 56.3% 減少し、CPU Time を除いた Elapsed Time も 1799.04 秒から 406.46 秒へ 77.4% 減少した。

同様にケース 2 でも、I/O 回数が 20207 回から 9011 へと 55.4% 減少し、CPU Time を除いた Elapsed Time も 1601.6 秒から 340.68 秒へ 78.7% 減少した。

ここで、Elapsed Time は、1 多重で実行したものであるので、正確な値になっている。

以上の改良により、I/O 回数は半分以下に減少し、Elapsed Time も CPU Time に対して高々 1.12 倍 (ケース 1, ケース 2 : 1.33 倍) の増加にとどまっているので、今回の修正の効果は大きいと考えられる。

Table 3.1 Base input to test the DOT3.5 FNS vector version

項 目	値
計 算 条 件	first collision source問題
形 状	R-Z 形 状
計 算 モ デ ル	mixed linear-weighted mode
メ ッ シ ュ サ イ ズ	40×86
最大ルジャンドル展開次数	5
群 数	125
S _N 角 度 分 点 数	ケース 1 48 ケース 2 160
pointwise flux 収束判定値	0.01

Table 3.2 Comparison of computed results between the original and vector versions (DOT3.5 FNS version)

ケース No.	項 目	群番号	オリジナル版 スカラ－計算	ベクトル化版 ベクトル計算
ケ ース 1	IN-SCATTER	125	1.13757×10^{-1}	1.13760×10^{-1}
	SELF-SCATTER	7	2.80994×10^{-3}	2.80994×10^{-3}
	OUT-SCATTER	60	2.36340×10^{-2}	2.36339×10^{-2}
	ABSORPTIONS	125	1.55576×10^{-2}	1.55598×10^{-2}
	RT-LEAKAGE	7	3.87453×10^{-3}	3.87454×10^{-3}
	VT-LEAKAGE	60	1.29521×10^{-3}	1.29522×10^{-3}
	TOP-LEAKAGE	125	3.25346×10^{-3}	3.25397×10^{-3}
	BOT-LEAKAGE	7	-7.33060×10^{-6}	-7.33051×10^{-6}
	NET-LEAKAGE	60	3.99307×10^{-3}	3.99310×10^{-3}
ケ ース 2	IN-SCATTER	125	1.14308×10^{-1}	1.14309×10^{-1}
	SELF-SCATTER	7	2.80442×10^{-3}	2.80441×10^{-3}
	OUT-SCATTER	60	2.36652×10^{-2}	2.36650×10^{-2}
	ABSORPTIONS	125	1.57603×10^{-2}	1.57619×10^{-2}
	RT-LEAKAGE	7	3.81899×10^{-3}	3.81893×10^{-3}
	VT-LEAKAGE	60	1.27588×10^{-3}	1.27590×10^{-3}
	TOP-LEAKAGE	125	3.27183×10^{-3}	3.27222×10^{-3}
	BOT-LEAKAGE	7	-9.64854×10^{-5}	-9.64827×10^{-5}
	NET-LEAKAGE	60	3.98617×10^{-3}	3.98619×10^{-3}

Table 3.3 Speedup effect (DOT3.5 FNS version)

ケ ース	項 目	オリジナル版 スカラー計算	旧ベクトル化版 ベクトル計算	新ベクトル化版 スカラー計算	新ベクトル化版 ベクトル計算
ケ ース 1	C P U 時間	28分 23.43秒	16分 49.88秒	31分 39.58秒	12分 47.84秒
	V U 時間	—	5分 41.35秒	—	8分 25.77秒
	倍 率	1.0	1.69	0.90	2.22
	ベクトル化率 (%)	—	未測定	—	86.2
	I/O 回数	9009	9010	9008	9008
	メモリ使用量 (KB)	4556	5764	6052	6680
ケ ース 2	C P U 時間	74分 4.94秒	37分 56.81秒	88分 43.52秒	32分 31.81秒
	V U 時間	—	18分 10.66秒	—	21分 12.84秒
	倍 率	1.0	1.95	0.83	2.28
	ベクトル化率 (%)	—	未測定	—	87.2
	I/O 回数	9497	9506	9497	9497
	メモリ使用量 (KB)	4556	5708	6052	6680

Table 3.4 Base input data to test the I/O speedup effect
(DOT3.5 FNS version)

	ケース 1	ケース 2
形 状	R-Z	R-Z
R 方向メッシュ数	40	40
Z 方向メッシュ数	86	86
ルジャンドル展開次数	5	5
Sn 数	48	160
群 数	125	125

Table 3.5 I/O speedup effect (DOT3.5 FNS version)

計算条件		I/O 回数	CPU Time	VU Time	Elapsed Time
ケ ス 1	①H I/O (Original)	20207	17分 00.99秒	5分 36.44秒	43分 42.54秒
	②V I O/F (\leq 69群)	11322	16分 48.54秒	5分 38.33秒	26分 52.54秒
	③V I O/F (全群)	9011	16分 49.16秒	5分 41.19秒	22分 29.84秒 (注1)
ケ ス 2	④H I/O (Original)	21762	37分 54.20秒	18分 02.64秒	67分 53.24秒
	⑤V I O/F (\leq 42群)	14970	37分 49.12秒	18分 03.67秒	55分 02.95秒
	⑥V I O/F (全群)	9506	37分 56.81秒	18分 10.66秒	44分 43.27秒 (注2)

②: V I O/F 実行 (ケース1, \leq 69群をV I O/F化)

I/O : 44.0%減少

Elapsed : 38.5%減少

③: V I O/F 実行 (ケース1, \leq 125群をV I O/F化)

I/O : 55.4%減少

Elapsed : 48.5%減少

⑤: V I O/F 実行 (ケース2, \leq 42群をV I O/F化)

I/O : 31.2%減少

Elapsed : 18.9%減少

⑥: V I O/F 実行 (ケース1, \leq 125群をV I O/F化)

I/O : 56.3%減少

Elapsed : 34.1%減少

(注1) Elapsed Time 推計式 :

①, ②, ③の I/O 回数及び Elapsed Time より

$$(20207 - 11322) : (20207 - 9011)$$

$$= (2622.54 - 1612.54) : (2622.54 - x)$$

$$x = 1349.84 \text{ (22min} 29.84\text{sec)}$$

(注2) Elapsed Time 推計式 :

④, ⑤, ⑥の I/O 回数及び Elapsed Time より

$$(21762 - 14970) : (21762 - 9506)$$

$$= (4073.24 - 3302.95) : (4073.24 - x)$$

$$x = 2683.27 \text{ (44min} 43.27\text{sec)}$$

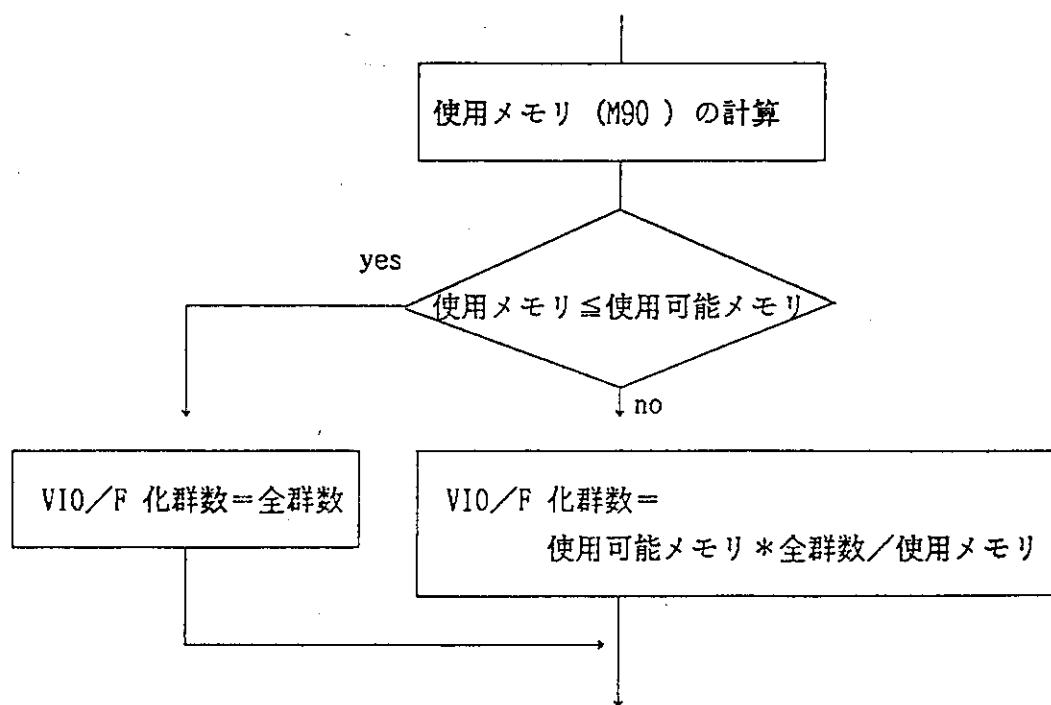


Fig. 3.1 Algorithm to set the number of groups
to reside the coefficients in core for
using VIO/F (DOT3.5 FNS version)

SUBROUTINE OUTER

```

        }

C.CS    COMMON /VIOF$/ IG,VIOF
C.CS
        }

C.CS
      DATA MVIOF / 92000000 /
      M90 = ( IM*JM*( 1 + A03*( A03 + 3 )/2 ) +
      1           MM*( JM + IM ) )*IGM*4
      IF( M90 . LE . MVIOF ) THEN
        IG = IGM
      ELSE
        IG   = MVIOF*IGM/M90
        VIOF = REAL( ( IGM - IG )*( IGM - IG - 1 ) )/
      1           REAL( IGM*( IGM - 1 ) )*100.0
      ENDIF
C.CS
        }

C.CS
      IF( IGV . LE . IG ) THEN
        NSCRT1 = NSCRAT - 1
      ELSE
        NSCRT1 = NSCRAT
      ENDIF
C.CS
        }

C.CS
C2641 CALL WANDR2(NSCRAT,B2(1),MMJM,B4(1),MMIM,1)
C      IF(IGV.NE.IGM) GO TO 2541
C 741 CALL REWND(NSCRAT)
2641 CALL WANDR2(NSCRT1,B2(1),MMJM,B4(1),MMIM,1)
      IF(IGV.NE.IGM) GO TO 2541
    741 IF( IGV . LE . IG ) THEN
        CALL REWND (NSCRT1)
      ELSE
        CALL REWND (NSCRT1)
        CALL REWND (NSCRT1-1)
      ENDIF
C.CS
        }

```

Fig. 3.2 Modification of subroutine OUTER to speedup the I/O processing

SUBROUTINE DOT

```

        }

C.CS    CALL WRITE$
C.CS
        }

```

Fig. 3.4 Modification of subroutine DOT to speedup the I/O processing

SUBROUTINE S8850

```

        }

C.CS   COMMON / VIOF* / IG , VIOF
C.CS
        }

C.CS
C.CS   CALL WANDR4(NFLUX1,N2(1,1,IG1),IMJM,J3(1,1,IG1),NOMG(IIG),
C.CS   B2(1),MMJM,B4(1),MMIM,2)
C.CS   CALL WANDR1(NPS0,S2(1,1),IMJM,2)
      IF( NFLUX1 . EQ . 90. ) THEN
        IF( IIG . LE . IG ) THEN
          CALL WANDR4(89,N2(1,1,IG1),IMJM,J3(1,1,IG1),NOMG(IIG),
1           B2(1),MMJM,B4(1),MMIM,2)
1       ELSE
          CALL WANDR4(90,N2(1,1,IG1),IMJM,J3(1,1,IG1),NOMG(IIG),
1           B2(1),MMJM,B4(1),MMIM,2)
1       ENDIF
      ELSE
        CALL WANDR4(NFLUX1,N2(1,1,IG1),IMJM,J3(1,1,IG1),NOMG(IIG),
1           B2(1),MMJM,B4(1),MMIM,2)
1       ENDIF
      IF( NPS0 . EQ . 90 ) THEN
        IF( IIG . LE . IG ) THEN
          CALL WANDR1(89,S2(1,1),IMJM,2)
        ELSE
          CALL WANDR1(90,S2(1,1),IMJM,2)
        ENDIF
      ELSE
        CALL WANDR1(NPS0,S2(1,1),IMJM,2)
      ENDIF
C.CS
        }
    
```

Fig. 3.3 Modification of subroutine S8850 to speedup the I/O processing

SUBROUTINE WRITE*

```

SUBROUTINE WRITE*
COMMON / VIOF* / IG , VIOF
VIOF1 = 100.0 - VIOF
WRITE( 6,6000 ) IG,VIOF1
6000 FORMAT( 1H ,
1 '===' NUMBER OF GROUP USED IN VIO/F : ' , I5 , '     ===' , / ,
2 ' === RATE OF VIO/F    ( X )      : ' , F8.2 , ' ===' )
      RETURN
END

```

Fig. 3.5 Addition of subroutine WRITE* to speedup the I/O processing

4. DOT 3.5 炉設計版

4.1 コードの概要

DOT 3.5 炉設計版は、DOT 3.5 コードを原研臨界プラズマ研究部炉設計研究室で改良したものである。DOT 3.5 FNS 版は DOT 3.5 炉設計版から派生後していったものである。DOT 3.5 炉設計版は、DOT 3.5 FNS 版派生後も改良が加えられ、現在に至っている。

DOT 3.5 コードに対する主な改良点は DOT 3.5 FNS 版と同様であるので、詳細はそちらの項を参照のこと。

4.2 動的挙動分析

DOT 3.5 炉設計版の主要部分は、DOT 3.5 コードのサブ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。

4.3 ベクトル化方法

DOT 3.5 コードのベクトル化技法と高速化技法を DOT 3.5 炉設計版に適用した。具体的な方法は、2.3 節を参照のこと。

4.4 計算結果の分析

4.4.1 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、1 ケースの試計算を実施した。試計算データを Table 4.1 に示す。

試計算結果を Table 4.2, 4.3 に示す。これによると VT-LEAKAGE と TOP-LEAKAGE の値を除き、有効桁 3 桁の範囲で一致している。VT-LEAKAGE と TOP-LEAKAGE の相違原因については次節で詳しく述べる。他の数値については、DOT 3.5 コードが单精度であり、中性子束の収束条件が 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。

また、今回ベクトル化したサブルーチン GRIND は、形状、計算モデルによりベクトル化版では 4 種類ある。今回の試計算では、これらのうち 1 種類のみについてテストした。他の 3 種類については、今回テストしなかったが、NEA オリジナル版では行っている。DOT 3.5 炉設計版と NEA オリジナル版では、サブルーチン GRIND は完全に同じである。したがって試計算はこれで十分と考える。

4.4.2 計算結果の相違原因について

オリジナル版スカラー計算とベクトル化版ベクトル計算（ベクトル化版スカラー計算でも同様）の群ごとの TOP-LEAKAGE を Table 4.4 に示す。これによると各群とも 1～3% の違いがオリジナル版とベクトル化版で生じている。なお、今回のデータでは、BOT-LEAKAGE がすべてゼロなので、VT-LEAKAGE も TOP-LEAKAGE と同じ値になっている。以下で、この数値相違原因について述べる。

TOP-LEAKAGE は、群毎にサブルーチン INNER で計算される。変数名は VVL であり、その値を決定する変数は B4 である。配列 B4 は軸方向の角度依存中性子束であり、サブルーチン GRIND で計算される。B4 の値は、変数 B2 MJ, B4 MI, N4I (配列では B2 (MJM), B4 (MIM), N4(I)) がゼロかそうでないかにより、大きく変わることがある。

B4 は、いくつかの一時変数の四則演算により求められる。それらの一時変数には、XN1, XN2, XN3, XD1, XD2, XD3 などがあり、オリジナル版では Fig. 4.1 のように計算している。Fig. 4.1 から解るように、XN1, XD1 の計算式は B2 MJ により変わり、XN2, XD2 の計算式は B4 MI により変わり、XN3, XD3 の計算式は N4I により変わる。

オリジナル版とベクトル化版で精度、計算順序が変わらなければ XN1 等の計算式も同じになるが、実際は精度、計算順序共に変わっている。

一次変数 NBAR について、オリジナル版とベクトル化版の違いを調べた。

オリジナル版の NBAR の計算式は次の通りである。

$$\text{NBAR} = (\text{XN1} + \text{XN2} + \text{XN3} + \text{SOR}) / (\text{XD1} + \text{XD2} + \text{XD3} + \text{SIGT})$$

これに対しベクトル化版では、次のように 3 回に分けて計算している。3 回に分けて計算している理由は、XN1, XD1 の計算が回帰的参照となりベクトル計算されないためであり、ベクトル計算ができる他の部分 (XN2 + XN3 + SOR, XD2 + XD3 + XD4) をベクトル化した。

$$\text{WK20} = \text{XN2} + \text{XN3} + \text{SOR}$$

$$\text{WK21} = \text{XD2} + \text{XD3} + \text{SIGT}$$

$$\text{NBAR} = (\text{XN1} + \text{WK20}) / (\text{XD1} + \text{WK21})$$

このように計算すると、場合により計算結果が異なる（有効桁の 7～9 術目）。この異った結果を用いると、B2 MJ, B4 MI, N4I がゼロになったり、ゼロにならなかったりする。例を 2 ケース（実際のもの）示し、B4 の値がどのように変化するか示す。

(例 1)

$$\begin{cases} \text{オリジナル版 } \text{N4I} = -0.243355197 \times 10^{-10} \neq 0 \\ \text{ベクトル化版 } \text{N4I} = 0 \end{cases}$$

① オリジナル版計算式

$$\text{XD3} = \text{CCC} * \text{GAMP}$$

$$\text{XN3} = (\text{GAMM} - \text{GAMP} + \text{XD3}) * \text{N4I}$$

② ベクトル化版計算式

$$\text{XD3} = \text{GAMP2}$$

XD3=0.0

この例の場合、XN3はN4Iの絶対値が小さいのでほとんど差はないが、XD3はオリジナル版が142.558594、ベクトル化版が197.493790と大きな差となった。その結果B4の値もオリジナル版が $0.179637718 \times 10^{-4}$ 、ベクトル化版が $0.161375792 \times 10^{-4}$ であり、10.2%の差を生じた。

(例2)

$$\begin{cases} \text{オリジナル版} & B2MJ = 0 \\ \text{ベクトル化版} & B2MJ = -0.909494702 \times 10^{-12} \neq 0 \end{cases}$$

B2MJは、NBARより計算される。NBARは、オリジナル版が $0.195541270 \times 10^{-5}$ 、ベクトル化版が $0.195541179 \times 10^{-5}$ であり、差は8桁目が9違うだけである。このNBARから計算したB2MJが上記の値である。B2MJが変わるとXN1、XD1の計算式が変わり、B4の値もオリジナル版0.0、ベクトル化版 $-0.366146458 \times 10^{-12}$ となった。

上記2例から解るようにDOT3.5コードでは、わずかな式の順序の変更が最終結果に影響を及ぼす。しかし、それでも高々数パーセントの違いであり、しかも値が異なる所は他に比べて数桁オーダーの小さい所である。

今回の数値相違はベクトル化したために起きた問題ではなく、DOT3.5コード・オリジナル版に内在していた問題がベクトル化により、顕在化してきただけのことである。

したがって、DOT3.5コードベクトル化版は、使用する上で問題はないと言える。

4.5 ベクトル化効果

Table4.1で示した問題についてベクトル化効果を測定した。その結果をTable4.5に示す。これによると、ベクトル化版ベクトル計算は、オリジナル版スカラー計算に比べて1.69倍に高速化され、ベクトル化版スカラー計算に比べて1.56倍に高速化された。

Table 4.1 Input data for test the DOT3.5 FER vector version

項 目	値
計 算 条 件	固定線源問題
形 状	R-Z
メ ッ シ ュ サ イ ズ	13×10
最大ルジャンドル展開次数	3
S _N 角 度 分 点 数	48
領 域 数	2
群 数	42
計 算 モ デ ル	weighted difference mode
pointwise flux 収束判定値	0.001

Table 4.2 Comparison of computed results between the original and vector versions (DOT3.5 FER version)

項 目	群番号	オリジナル版 スカラー計算	ベクトル化版 ベクトル計算
IN-SCATTER	1	5.36442×10^{-7}	5.36442×10^{-7}
SELF-SCATTER	20	3.02660	3.02661
OUT-SCATTER	41	2.29007×10^{-3}	2.28956×10^{-3}
ABSORPTIONS	1	1.40607×10^{-1}	1.40609×10^{-1}
LFT-LEAKAGE	20	0.0	0.0
RT-LEAKAGE	41	1.03553×10^{-4}	1.03528×10^{-4}
VT-LEAKAGE	1	5.86558×10^{-6}	6.00886×10^{-6}
TOP-LEAKAGE	20	6.95907×10^{-6}	7.15512×10^{-6}
BOT-LEAKAGE	41	0.0	0.0
NET-LEAKAGE	1	7.74052×10^{-3}	7.74067×10^{-3}

Table 4.3 Comparison of neutron fluxes between the original and vector versions (DOT3.5 FER version)

メッシュ番号		群番号	中性子束	
			オリジナル版 スカラー計算	ベクトル化版 ベクトル計算
R方向	Z方向			
1	1	1	3.21539×10^{-3}	3.21539×10^{-3}
13	10	1	5.12605×10^{-7}	5.12605×10^{-7}
3	3	2	4.19651×10^{-5}	4.19651×10^{-5}
13	3	2	1.37778×10^{-7}	1.37778×10^{-7}
5	5	10	1.08699×10^{-6}	1.08698×10^{-6}
10	10	10	6.09149×10^{-8}	6.09141×10^{-8}
1	1	41	1.34291×10^{-6}	1.34263×10^{-6}
3	5	41	1.23275×10^{-6}	1.23248×10^{-6}
7	5	42	7.22574×10^{-11}	7.22392×10^{-11}
13	10	42	2.02577×10^{-11}	2.02520×10^{-11}

Table 4.5 Speedup effect between the original and vector versions (DOT3.5 FER version)

	オリジナル版 スカラー計算	ベクトル化版 スカラー計算	ベクトル化版 ベクトル計算
C P U 時間(秒)	83.76	76.52	49.60
V U 時間(秒)	—	—	23.72
倍率	1.0	1.09	1.69
ベクトル化率(%)	—	—	66.2
I/O 回数	1804	1803	1803
メモリ使用量(KB)	1776	3268	3328

Table 4.4 Comparison of group dependent top leakage between
the original and vector versions (DOT3.5 FER version)

GROUP	TOP LEAKAGE	
	オリジナル版	ベクトル化版
1	5.86558E-06	6.00886E-06
2	1.40211E-06	1.43289E-06
3	4.35243E-07	4.42413E-07
4	2.39841E-07	2.45702E-07
5	2.07594E-07	2.11637E-07
6	6.22052E-08	6.30023E-08
7	3.67616E-08	3.70841E-08
8	4.52061E-08	4.54505E-08
9	6.97699E-08	7.16576E-08
10	1.01472E-07	1.04372E-07
11	1.35574E-07	1.40144E-07
12	1.76593E-07	1.80272E-07
13	5.88291E-07	5.92492E-07
14	1.40604E-06	1.44508E-06
15	3.81803E-06	3.91266E-06
16	5.72783E-06	5.80619E-06
17	6.53783E-06	6.65311E-06
18	7.40952E-06	4.94454E-06
19	5.04754E-06	5.03168E-06
20	6.95907E-06	7.17392E-06
21	5.21256E-06	6.08965E-06
22	7.25221E-06	7.48769E-06
23	7.08018E-06	7.20300E-06
24	5.60029E-06	5.58596E-06
25	6.29900E-06	6.54286E-06
26	-3.61388E-05	-3.55081E-05
27	5.22621E-06	5.34528E-06
28	3.07771E-06	3.26325E-06
29	4.76018E-06	4.91820E-06
30	5.31172E-06	5.34584E-06
31	3.05817E-06	3.06327E-06
32	4.32444E-06	4.34341E-06
33	3.92261E-06	3.95015E-06
34	3.53839E-06	3.55638E-06
35	2.99684E-06	3.01162E-06
36	2.35648E-06	2.36610E-06
37	1.73395E-06	1.75053E-06
38	1.14891E-06	1.15627E-06
39	6.63533E-07	6.69786E-07
40	3.22470E-07	3.23582E-07
41	1.26961E-07	1.27981E-07
42	-2.05586E-10	-2.01310E-10
合計	8.41456E-05	8.51355E-05

Fig. 4.1 Program list of a part of subroutine GRIND

5. RADHEAT-V 4

5.1 システムの概要

遮蔽安全評価用群定数作成および放射線輸送解析コードシステムRADHEAT-V4⁽¹⁾は、遮蔽安全評価および中性子と光子の輸送解析を精確に行うために、1987年に原研でRADHEAT-V3システムの発展版として作成された。

RADHEAT-V 4 システムは、中性子-光子の結合多群定数を作成し、中性子と光子の輸送解析を行い、さらに原子炉または遮蔽材中の放射線による発熱と原子のはじき出しを計算する種々の機能モジュールで構成されている (Fig. 5.1 参照)。この中で、計算の中心になる（計算時間が多くのかかる）部分は、中性子およびガンマ線輸送解析モジュールである。さらに、このモジュールの中でも 2 次元 S_N 輸送計算モジュール ESPRIT が計算時間が多くのかかる。3 次元モンテカルロ計算モジュール MCASE も計算時間が多くのかかるが今回のベクトル化では除いた。

RADHEAT-V 4 システムでは、各モジュールが完全に独立しており、しかも入力データも独立である。したがって、今回ベクトル化を行う ESPRIT モジュールも独立である。

DOT 3.5 コードでは、断面積および放射線束の角度分布を評価する際に、有限項ルジャンドル展開法を用いている。このため、高エネルギー領域における非等方性が顕著な場合には展開誤差が無視できなくなり、負の角度フラックスが生じるケースがある。これを避けるため ESPRIT モジュールは、DOT 3.5 コードにおける有限項ルジャンドル展開法の部分を、直接角度表示 (DAR) 法と呼ばれる精確な手法を用いている。

5.2 動的挙動分析

RADHEAT-V 4 システムは、多くのモジュールの集まりであり、計算の目的により使用するモジュールが異なる。しかし計算時間が多くのかかるモジュールという点から見ると中性子およびガンマ線輸送解析モジュールが 1 番である。今回は、その中でも計算時間が多くのかかる 2 次元 S_N 計算モジュール ESPRIT のベクトル化を実施した。

Fig. 5.1 は、ESPRIT モジュールの主要サブルーチンの木構造である。

FORTUNE ツールを用いて CPU 時間分布の動的挙動分析を実施した。動的挙動分析に使用するデータは、Table 5.1 に示すように円筒形状 10 群の問題である。Fig. 5.3 に分析結果を示す。Fig. 2.3 を見ると、DAR 法の中心となる散乱線源を計算するサブルーチン CSCAT と中性子束の収束計算の核となるサブルーチン GRIND で全体の 95.4% を占めている。

したがって、CSCAT, GRIND を中心にベクトル化を実施した。

5.3 ベクトル化方法

ベクトル化は、サブルーチン CSCAT, GRIND を中心に実施し、必要があれば他のサブルーチンの修正も行う。

5.3.1 サブルーチン CSCAT

サブルーチン CSCAT は、散乱確率関数を用いて散乱線源を計算するサブルーチンであり、DAR 法の核となる部分である。

Fig. 5.2 に示すようにサブルーチン CSCAT は、3重のループ構造になっている。この中で計算時間が多くかかるのはこのループの最内の部分（作業配列へのたしあげ計算部分）であり、サブルーチン CSCAT の計算時間の 95% 以上を占めている。この部分の最内ループは角度メッシュのループで今回のデータの場合ループ長は 48 である。この部分のプログラムリストを Fig. 5.3 に示す。最内ループの中の 2つの配列は共に連続アクセスであるので、ベクトル長が 48 とそれほど多くない今回のケースでもコンパイラによる自動ベクトル化により 10 倍以上の高速化が可能であった。

自動ベクトル化以外の高速化手法として次の 2つが考えられる。

- ① 最内ループともう 1つ外側のループを 1重化し…ベクトル長を長くする。
- ② 1番外側の ij メッシュのループを最内に持ってきてベクトル長を長くする。

上記 2 方法のうち①の方法は、外側のループがたしあげ計算となり、2重ループを 1重化すると回帰的参照となってしまいベクトル化できない。また②の方法は、ij メッシュのループの中に 1つ判定文があり、この判定文が真になると、以降の処理を行わない。そのため、この ij メッシュのループを最内に持ってくると、この判定文がネックになり、リストベクトルを使わざるを得なくなり、自動ベクトル化より高速化できる可能性は小さい。このような理由から自動ベクトル化により高速化した。

5.3.2 サブルーチン GRIND

サブルーチン GRIND は、中性子束を計算する部分であり、内側反復計算の中心に位置するサブルーチンである。

このサブルーチンは名称からも推測できるように、DOT 3.5 コードのサブルーチン GRIND と全く同じ機能を有している。そこでベクトル化した DOT 3.5 コードのサブルーチン GRIND を ESPRIT モジュールに組み込むことを考える。組み込み手順は次の通りである。

- ① DOT 3.5 のオリジナル GRIND と ESPRIT モジュールの GRIND のソース・プログラムを比較する。
- ② 異なっている部分をベクトル化 GRIND に反映させる。
- ③ サブルーチン INNER の GRIND を呼んでいる部分を DOT 3.5 ベクトル化版のそれに合わせる。

上記手順により作業を行った。①の比較の結果を Fig. 5.4 に示す。このように、相違箇所は 3ヶ所であり、この部分を DOT 3.5 ベクトル化版に反映させた。DOT 3.5 ベクトル化版の GRIND

は、形状、ルジャンドル展開次数、計算モードにより 8 種類に分かれている。このうちルジャンドル展開次数は、DAR 法を用いた ESPRIT モジュールの GRIND では使わないので半分の 4 種類になる。3ヶ所の修正箇所のうち、1番目と 3番目は、このルジャンドル展開に関する部分であり、4種類のサブルーチンに共通である。また 2番目の修正は、ESPRIT モジュール固有の部分を DOT 3.5 の形に修正し、ベクトル化サブルーチンは今まで通りとした。

ベクトル化版では判定文をループの外に出しており計算部分が 2ヶ所になっている場合があるので、修正には注意を要した。4種類のサブルーチンのうち GRIRZ (R-Z 形状、計算モデル=3) について修正箇所を Fig. 5.5 に示す。

また、サブルーチン INNER の GRIND を呼んでいる部分を DOT 3.5 のベクトル化版と同じように修正した。ただし、ルジャンドル展開次数に関連した判定文は除く。修正リストを Fig. 5.6 に示す。このように ESPRIT モジュールの INNER では、計算モデル (FXT) と形状 (IGE) によらず、すべての場合について計算できるようにした。

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 5 ケースの試計算を実施した。試計算データは、動的挙動分析に用いたものとほとんど同じであるが、形状が 2 種類、計算モデルが 3 種類に分かれている (Table 5.2, 5.3 参照)。計算結果を Table 5.4 に示す。Table 5.4 によると各値とも有効桁 3 術の範囲では完全に一致している。RADHEAT-V4 システムが単精度であり、中性子束の収束条件が 10^{-2} であることから判断して、十分許容範囲内であると判断できる。

今回ベクトル化したサブルーチン GRIND は、形状、計算モデルにより 5 種類 (ケース 2 の mixed linear-step mode については、GRIND はベクトル化していない) ある。これらの 5 サブルーチンのテストを上記 5 ケースで実施しているので、試計算はこれで十分と考えられる。

5.5 ベクトル化効果

Table 5.2, 5.3 で示した問題 5 ケースについてベクトル化効果を測定した。その結果を Table 5.5 に示す。これによるとベクトル化版ベクトル計算は、オリジナル版スカラー計算に比べて 3.1 倍から 4.3 倍に高速化された。ここでケース 2 はサブルーチン GRIND 部分がベクトル化されていないので、他のケースより遅くなっている。

Table 5.1 Base input data to test RADHEAT-V4 system

形 状	R-Z
R 方向メッシュサイズ	41
Z 方向メッシュサイズ	29
エネルギー群数	9
計算条件	first collision source問題
計算モデル	weighted-difference mode
収束判定値	0.01
pointwise flux 収束判定値	0.01

Table 5.2 Common input data for test runs of RADHEAT-V4 system

R 方向メッシュサイズ	41
Z 方向メッシュサイズ	29
エネルギー群数	9
計算条件	first collision source問題
収束判定条件	0.01
pointwise flux 収束判定値	0.01

Table 5.3 Input data of geometry type and calculation model for test runs (RADHEAT-V4 system)

ケースNo.	形 状	計 算 モ デ ル
1	R-Z	weighted difference mode
2	R-Z	mixed linear-step mode
3	R-Z	mixed linear-weighted mode
4	X-Y	weighted difference mode
5	X-Y	mixed linear-weighted mode

Table 5.4 Comparison of computed results between the original and vector versions (RADHEAT-V4 system)

ケース	項目	群番号	オリジナル版 スカラー計算	ベクトル化版 ベクトル計算
ケ 1 ス 1	IN-SCATTER	2	8.77917×10^{-3}	8.77917×10^{-3}
	SELF-SCATTER	5	5.79749×10^{-2}	5.79745×10^{-2}
	OUT-SCATTER	8	2.84077×10^{-1}	2.84075×10^{-1}
	ABSORPTIONS	1	1.24274×10^{-4}	1.24273×10^{-4}
	TOP-LEAKAGE	5	3.14244×10^{-4}	3.14245×10^{-4}
	BOT-LEAKAGE	9	-2.46946×10^{-2}	-2.46939×10^{-2}
	NET-LEAKAGE	8	2.42627×10^{-2}	2.42624×10^{-2}
	FLUXメッシュNo.(10,10)	1	1.70521×10^{-9}	1.70521×10^{-9}
	FLUXメッシュNo.(41,29)	9	2.02929×10^{-13}	2.02913×10^{-13}
ケ 1 ス 2	IN-SCATTER	2	8.78209×10^{-3}	8.78209×10^{-3}
	SELF-SCATTER	5	5.79148×10^{-2}	5.79144×10^{-2}
	OUT-SCATTER	8	2.84535×10^{-1}	2.84530×10^{-1}
	ABSORPTIONS	1	1.24227×10^{-4}	1.24226×10^{-4}
	TOP-LEAKAGE	5	2.79918×10^{-4}	2.79915×10^{-4}
	BOT-LEAKAGE	9	-2.40332×10^{-2}	-2.40318×10^{-2}
	NET-LEAKAGE	8	2.32506×10^{-2}	2.32501×10^{-2}
	FLUXメッシュNo.(10,10)	1	1.69872×10^{-9}	1.69872×10^{-9}
	FLUXメッシュNo.(41,29)	9	1.95414×10^{-13}	1.95378×10^{-13}
ケ 1 ス 3	IN-SCATTER	2	8.78108×10^{-3}	8.78114×10^{-3}
	SELF-SCATTER	5	5.80732×10^{-2}	5.80730×10^{-2}
	OUT-SCATTER	8	2.86137×10^{-1}	2.86127×10^{-1}
	ABSORPTIONS	1	1.24287×10^{-4}	1.24287×10^{-4}
	TOP-LEAKAGE	5	3.02714×10^{-4}	3.02715×10^{-4}
	BOT-LEAKAGE	9	-2.41679×10^{-2}	-2.41672×10^{-2}
	NET-LEAKAGE	8	2.30572×10^{-2}	2.30570×10^{-2}
	FLUXメッシュNo.(10,10)	1	1.66713×10^{-9}	1.66713×10^{-9}
	FLUXメッシュNo.(41,29)	9	2.02639×10^{-13}	2.02626×10^{-13}
ケ 1 ス 4	IN-SCATTER	2	1.59763×10^{-5}	1.59763×10^{-5}
	SELF-SCATTER	5	1.05227×10^{-4}	1.05226×10^{-4}
	OUT-SCATTER	8	4.94064×10^{-4}	4.94051×10^{-4}
	ABSORPTIONS	1	2.23567×10^{-7}	2.23567×10^{-7}
	TOP-LEAKAGE	5	5.56628×10^{-7}	5.56624×10^{-7}
	BOT-LEAKAGE	9	-5.37460×10^{-5}	-5.37477×10^{-5}

Table 5.4 (Continued)

ケース	項目	群番号	オリジナル版 スカラー計算	ベクトル化版 ベクトル計算
ケ ー ス 1 5	NET-LEAKAGE	8	5.64998×10^{-5}	5.64990×10^{-5}
	FLUXメッシュNo.(10,10)	1	1.75637×10^{-9}	1.75403×10^{-9}
	FLUXメッシュNo.(41,29)	9	6.11438×10^{-11}	6.11352×10^{-11}
	IN-SCATTER	2	1.59762×10^{-5}	1.59761×10^{-5}
	SELF-SCATTER	5	1.05323×10^{-4}	1.05322×10^{-4}
	OUT-SCATTER	8	4.95384×10^{-4}	4.95371×10^{-4}
	ABSORPTIONS	1	2.23567×10^{-7}	2.23566×10^{-7}
	TOP-LEAKAGE	5	5.37607×10^{-7}	5.37707×10^{-7}
	BOT-LEAKAGE	9	-5.39346×10^{-5}	-5.39313×10^{-5}
	NET-LEAKAGE	8	5.55030×10^{-5}	5.55019×10^{-5}
	FLUXメッシュNo.(10,10)	1	1.85012×10^{-9}	1.85012×10^{-9}
	FLUXメッシュNo.(41,29)	9	6.32050×10^{-11}	6.31968×10^{-11}

Table 5.5 Speedup effect (RADHEAT-V4 system)

ケース	CPU時間 (秒)	VU時間 (秒)	倍 率	ベクトル化率 (%)	I/O回数	メモリ使用量 (KB)
1	186.95	—	1.0	—	2781	2012
	189.94	—	0.98	—	2781	2528
	51.47	24.75	3.63	85.7	2781	2580
2	232.43	—	1.0	—	2781	2012
	232.14	—	1.00	—	2781	2528
	75.82	11.54	3.07	72.3	2781	2576
3	182.50	—	1.0	—	2781	2012
	210.38	—	0.87	—	2781	2528
	57.42	30.48	3.18	85.2	2781	2580
4	298.93	—	1.0	—	2786	2012
	248.75	—	1.20	—	2781	2528
	68.74	32.06	4.35	87.7	2781	2580
5	185.13	—	1.0	—	2781	2012
	195.72	—	0.95	—	2781	2528
	51.09	27.79	3.62	87.4	2781	2580

上段：オリジナル版スカラー計算

中段：ベクトル化版スカラー計算

下段：ベクトル化版ベクトル計算

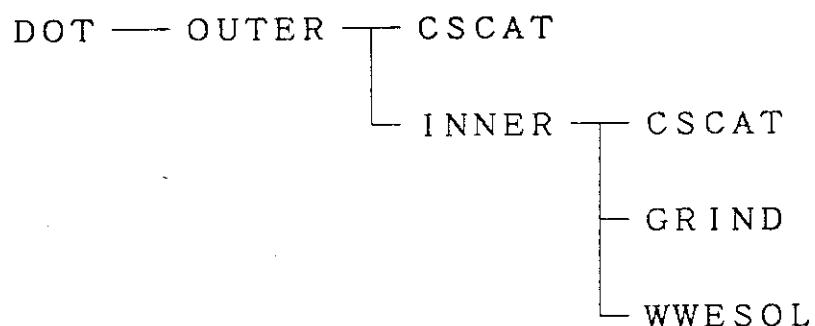


Fig. 5.1 Tree structure of time-consuming subroutines of RADHEAT-V4

```
NN=MM*MM*(ITEMP1-1)
2-----DO 300 K=1,MM
2           F=J3(I,K)*WO(K)
2 3-----DO 300 L=1,MM
2 3           NN=NN+1
2 3           WORK(L)=WORK(L) + F*P3(NN)
+-----300 CONTINUE
```

Fig. 5.3 Time consuming loops of subroutine CSCAT of RADHEAT-V4

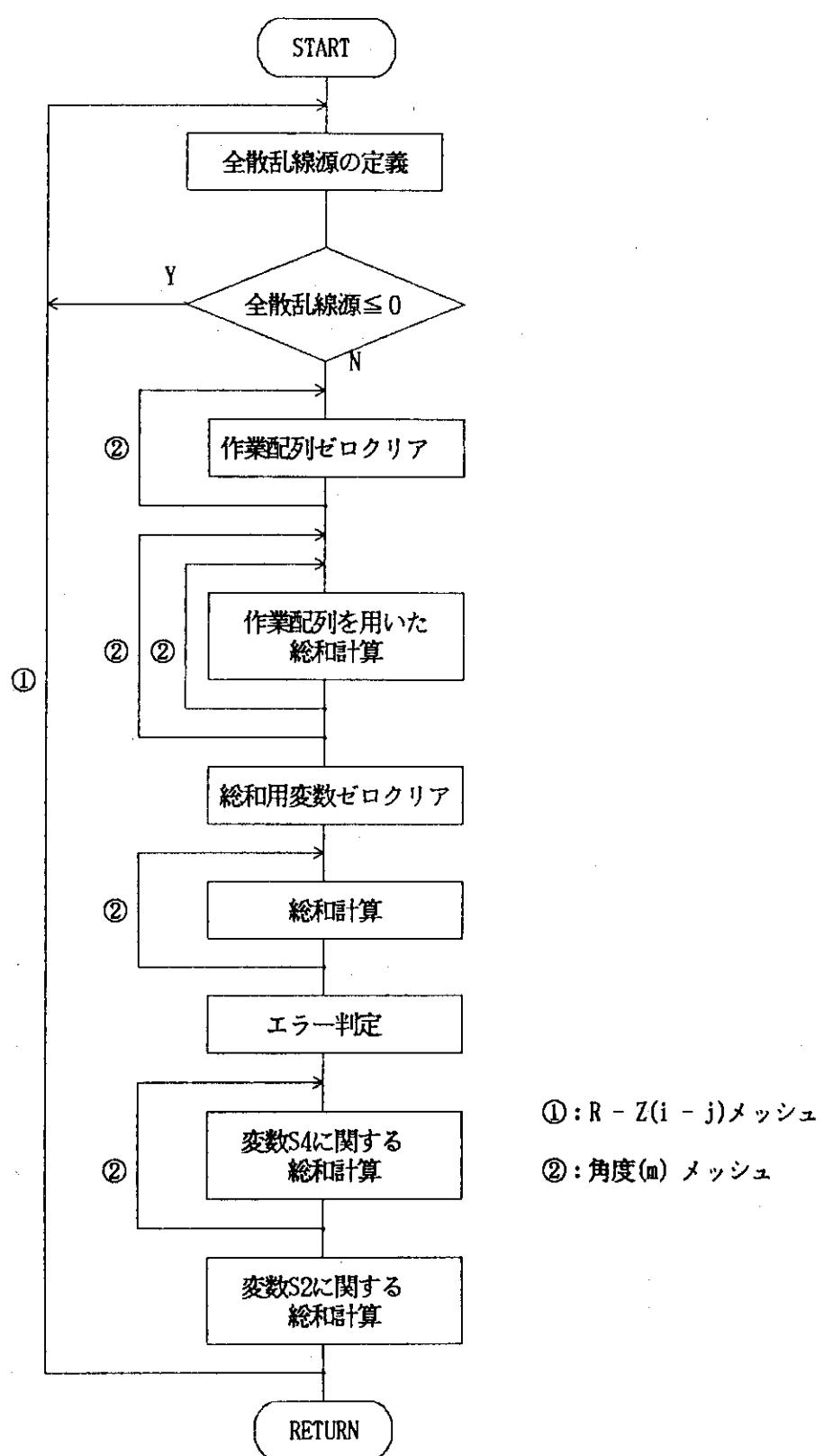


Fig. 5.2 Loop structure of subroutine CSCAT of RADHEAT-V4

```

*DELETE 163 - 170
(( SOR=SO(IJ)
(( IF(A03.EQ.0)GO TO 1080
(( MK=M
(( IJK=IJ
(( DO 1082 K=1,NOMA
(( SOR=SOR + P3(MK)*S3(IJK)
(( MK=MK + MM
(( 1082 IJK=IJK + IMJM
*INSERT 170
(( IJK=IJ+IMJM*(M-1)
(( SOR=S3(IJK)

*DELETE 304 - 304
(( IF(B2MJ.EQ.0.0)GO TO 6035
*INSERT 304
(( IF(ABS(B2MJ).LT.1.0E-10)GO TO 6035
*DELETE 400 - 405
(( IJKIG=KIG + IJ
(( MK=M
(( DO 3063 K=1,NOMA
(( J3(IJKIG)=J3(IJKIG) + P3(MK)*WNBAR
(( IJKIG=IJKIG + IMJM
(( 3063 MK=MK + MM
*INSERT 405
(( IJKIG=KIG + IJ + IMJM*(M-1)
(( J3(IJKIG) = NBAR

```

相違 1

相違 2

相違 3

Fig. 5.4 Coding difference of subroutine GRIND between DOT3.5 and RADHEAT-V4

```

      IF(XM7M.LE.0.) THEN
*VOCL LOOP,NOVREC(ZSOR)
  DO 1081 IQ=1,IM
    I=IP-IQ
    IJ      =INDE+I
  CNN   ZSOR(I)=SO(IJ)
  CNN
    IJK      =IJ+IMJM*(M-1)
  ZSOR(I)=S3(IJK)
  CNN
  1081 CONTINUE
    ELSE
*VOCL LOOP,NOVREC(ZSOR)
  DO 1082 IQ=1,IM
    I=IQ
    IJ      =INDE+I
  CNN   ZSOR(I)=SO(IJ)
  CNN
    IJK      =IJ+IMJM*(M-1)
  ZSOR(I)=S3(IJK)
  CNN
  1082 CONTINUE
    ENDIF

```

(1) 修正 その 1

```

      IF(A03.EQ.0)GO TO 30631
  CNN   DO 3063 K=1,NOMA
*VOCL LOOP,NOVREC(J3)
  DO 3063 I=1,IM
    IJ=INDE+I
  CNN
  CNN   J3(KIG+IJ+(K-1)*IMJM)=
  CNN   1     J3(KIG+IJ+(K-1)*IMJM)+P3(M+(K-1)*MM)*ZNBAR(I)*WOM
        IJKIG=KIG+IJ+IMJM*(M-1)
        J3(IJKIG)=ZNBAR(I)
  CNN
  3063 CONTINUE
  30631 CONTINUE

```

(3) 修正 その 2

Fig. 5.5 Modification of subroutine GRIND of RADHEAT-V4 for vectorization

CNN

```

IF(FXT .EQ. 4) THEN
  IF(IGE.EQ.0) THEN
    CALL GRIXY4(M0,M2,X0,VO,CO,T7,M7,W1,B2,B0,BSR,W0,N4,M6,Z5,M3,A0,
1 P3,S3,G2,B4,S0,M5,AFLUX,N2,J3,MMP,ITLP,IMP,JMP,MTF,IT15,IMJMP,
2 NOMAP,ISIZE1,FLAG,M11,M21,IGG,ALBDOR,IIGM,ISIZE2,CBAN(1,1),
3 CBAN(1,2),CBAN(1,3),CBAN(1,4),IPTSCL,BSL,AIFLUX,AJFLUX,V4,V5)
    ELSEIF(IGE.EQ.1) THEN
      CALL GRIRZ4(M0,M2,X0,VO,CO,T7,M7,W1,B2,B0,BSR,W0,N4,M6,Z5,M3,A0,
1 P3,S3,G2,B4,S0,M5,AFLUX,N2,J3,MMP,ITLP,IMP,JMP,MTF,IT15,IMJMP,
2 NOMAP,ISIZE1,FLAG,M11,M21,IGG,ALBDOR,IIGM,ISIZE2,CBAN(1,1),
3 CBAN(1,2),CBAN(1,3),CBAN(1,4),IPTSCL,BSL,AIFLUX,AJFLUX,V4,V5)
    ELSE
      CALL GRIND (M0,M2,X0,VO,CO,T7,M7,W1,B2,B0,BSR,W0,N4,M6,Z5,M3,A0,
1 P3,S3,G2,B4,S0,M5,AFLUX,N2,J3,MMP,ITLP,IMP,JMP,MTF,IT15,IMJMP,
2 NOMAP,ISIZE1,FLAG,M11,M21,IGG,ALBDOR,IIGM,ISIZE2,CBAN(1,1),
3 CBAN(1,2),CBAN(1,3),CBAN(1,4),IPTSCL,BSL,AIFLUX,AJFLUX,V4,V5)
    ENDIF
  ELSEIF(FXT.EQ.3) THEN
    IF(IGE.EQ.0) THEN
      CALL GRIXY(M0,M2,X0,VO,CO,T7,M7,W1,B2,B0,BSR,W0,N4,M6,Z5,M3,A0,
1 P3,S3,G2,B4,S0,M5,AFLUX,N2,J3,MMP,ITLP,IMP,JMP,MTF,IT15,IMJMP,
2 NOMAP,ISIZE1,FLAG,M11,M21,IGG,ALBDOR,IIGM,ISIZE2,CBAN(1,1),
3 CBAN(1,2),CBAN(1,3),CBAN(1,4),IPTSCL,BSL,AIFLUX,AJFLUX,V4,V5)
      ELSEIF(IGE.EQ.1) THEN
        CALL GRIRZ(M0,M2,X0,VO,CO,T7,M7,W1,B2,B0,BSR,W0,N4,M6,Z5,M3,A0,
1 P3,S3,G2,B4,S0,M5,AFLUX,N2,J3,MMP,ITLP,IMP,JMP,MTF,IT15,IMJMP,
2 NOMAP,ISIZE1,FLAG,M11,M21,IGG,ALBDOR,IIGM,ISIZE2,CBAN(1,1),
3 CBAN(1,2),CBAN(1,3),CBAN(1,4),IPTSCL,BSL,AIFLUX,AJFLUX,V4,V5)
      ELSE
        CALL GRIND (M0,M2,X0,VO,CO,T7,M7,W1,B2,B0,BSR,W0,N4,M6,Z5,M3,A0,
1 P3,S3,G2,B4,S0,M5,AFLUX,N2,J3,MMP,ITLP,IMP,JMP,MTF,IT15,IMJMP,
2 NOMAP,ISIZE1,FLAG,M11,M21,IGG,ALBDOR,IIGM,ISIZE2,CBAN(1,1),
3 CBAN(1,2),CBAN(1,3),CBAN(1,4),IPTSCL,BSL,AIFLUX,AJFLUX,V4,V5)
      ENDIF
    ELSE
      CALL GRIND (M0,M2,X0,VO,CO,T7,M7,W1,B2,B0,BSR,W0,N4,M6,Z5,M3,A0,
1 P3,S3,G2,B4,S0,M5,AFLUX,N2,J3,MMP,ITLP,IMP,JMP,MTF,IT15,IMJMP,
2 NOMAP,ISIZE1,FLAG,M11,M21,IGG,ALBDOR,IIGM,ISIZE2,CBAN(1,1),
3 CBAN(1,2),CBAN(1,3),CBAN(1,4),IPTSCL,BSL,AIFLUX,AJFLUX,V4,V5)
    ENDIF
  ENDIF

```

CNN

Fig. 5.6 Modification of subroutine INNER of RADHEAT-V4 for vectorization

6. DOT-DD

6.1 コードの概要

DOT-DD コード⁽¹⁾は、DOT 3.5 コードに次の機能を付加したものである。

- (1) 二重微分型断面積 (DDX) を用いる S_N 計算
- (2) リスタート計算の効率化
- (3) 高速入出力機能への対応
- (4) UNCL-DD による 1 回衝突線源の計算

DOT-DD コードは、DDX を用いる S_N 計算部分と PL 展開を用いる S_N 計算部分に分かれており、主に使用しているのは、DDX を用いる S_N 計算部分である。以下 DDX を用いる S_N 計算について述べる。

輸送計算を行うのに DDX を使用する場合、輸送方程式中の衝突線源項は、次のように記述することができる。

$$q_s(r, \Omega) = \sum_{g'} \int d\Omega' \phi_{s'}(r, \Omega') \frac{\Sigma_s^l(k; g' \rightarrow g)}{2\pi\Delta\mu_k} \quad (6.1)$$

ここで、添字 k は $\mu_{k+1} < \Omega \cdot \Omega' < \mu_k$ となる角度の番号である。また、 $\Delta\mu_k = \mu_k - \mu_{k+1}$ であり、 μ_k は DDX の角度である。そして、 $\Sigma_s^l(k; g' \rightarrow g)$ は、次式で計算される物質 J の巨視的 2 重微分断面積である。

$$\begin{aligned} \Sigma_s^l(k; g' \rightarrow g) &= \sum_j N \rho_s^j(k; g' \rightarrow g) \\ &= \sum_j N \rho_{pr, g'}^j I^j(k; g' \rightarrow g) \end{aligned} \quad (6.2)$$

S_N 輸送コードにおいて式 (6.1) は次のように計算される。

$$\begin{aligned} q_{g, m} &= \sum_{g'} \sum_{m'} \phi_{g', m'} \cdot W_{m'} \sum_k \frac{\Sigma_s^l(k; g' \rightarrow g)}{2\pi\Delta\mu_k} \times \\ &\quad \frac{1}{4\pi W_{m'} W_m} \int_{\Delta\Omega_{m'}} d\Omega' \int_{\Delta\Omega_m} d\Omega \delta_{\mu^*, k} \end{aligned} \quad (6.3)$$

ここで

$$\delta_{\mu^*, k} = \begin{cases} 1 & (\mu_{k+1} \leq \mu^* = \Omega \cdot \Omega' \leq \mu_k) \\ 0 & (\text{その他}) \end{cases}$$

そして W_m は、方向 m の重みである。 $\Delta\Omega_m$ は、方向 m の立体角であり、 $\Delta\Omega_m = 4\pi W_m$ で与えられる。 $q_{g,m}$ と $\phi_{g',m'}$ は散乱中性子源と角度依存中性子束である。式(6.3)は次のように書き直すことができる。

$$q_{g,m} = \sum_{g'} \sum_{m'} \phi_{g',m'} \cdot W_{m'} \sum_k \Sigma_S^l(k; g' \rightarrow g) \times P(k, m', m) \quad (6.4)$$

ここで $P(k, m', m)$ は、 $\Omega_{m'}$ で入射した中性子が Ω_m 方向へ散乱されるとき Ω_m と $\Omega_{m'}$ の間の角の余弦が $\mu_{k+1} \leq \mu \leq \mu_k$ になる確率である。この P は、別に作成したプログラム MONTATMによって求め、それを DOT-DD で使用している。

6.2 I/O の高速化手法

DOT-DD コードも DOT 3.5 と同様に、オリジナルでは、計算されたそれぞれの空間セル内のすべてのエネルギー群でのスカラーフラックスは、プログラム内で保存できないので一時ファイル（論理ユニット番号：NFLUX 1, NSCRAT）に出力し、使用する毎に入力している。

このファイルの入出回数は、データにもよるが DDX 計算の場合全入出回数の 9 割以上を占めるケースが多く、しかも 1 回の入出力でのデータ数が非常に多い。

そこで、DOT 3.5 FNS 版と同様な方法(3.6 節)で、このファイルの全部又は一部について VIO/F 化することを考えた。

スカラーフラックスの出回数は群数回（各群とも 1 回）であるが、入回数 n は

$$n = m(m-1)/2 \quad (m: \text{群数})$$

であり、群毎の入回数は、1 群は $(m-1)$ 回、2 群は $(m-2)$ 回、 $(m-1)$ 群は 1 回となる。そこで全群が VIO/F 化できない場合、1 群からできる範囲で VIO/F 化する (Fig. 6.1 参照)。

VIO/F 化する論理ユニット番号は 3 (実際には、入力データで与えた NFLUX 1) であり、これを使っているサブルーチンは OUTERD, OUTER と S 8850 であり、各サブルーチンについて修正を加えた。

論理ユニット番号 3 番のレコード長は DDX 計算の場合(1), PL 計算の場合(2)でそれぞれ与えられる。

(1) DDX 計算

$$\text{MEMRY3} = (\text{IM} * \text{JM} + \text{MM} * (\text{IM} * \text{JM} + \text{JM} * \text{IM})) * \text{IGM} * 4$$

IM : R 方向メッシュ数

JM : Z 方向メッシュ数

MM : Sn 数

IGM : 群数

(2) PL 計算

$$\text{MEMRY3} = (\text{IM} * \text{JM} * (1 + \text{A03} * (\text{A03} + 3) / 2) + \\ \text{MM} * (\text{JM} + \text{IM})) * \text{IGM} * 4$$

IM : R方向メッシュ数

JM : Z方向メッシュ数

A03 : 散乱次数

MM : Sn数

IGM : 群数

6.3 I/O 高速化の効果

試計算は DDX 計算 1 ケースについて実行した。

入力条件を Table 6.1 に示す。

試計算結果を Table 6.2 に示す。

これによると、I/O 回数が 166683 回から 4908 回へと 97.1% 減少し、CPU Time は 323 秒が 311 秒へと 3.7% 減少し、Elapsed Time は 90 分から 45 分へと約半分に減少した。ここで Elapsed Time は、1 多重でランしたものではなく実際に日中ランしたものであるので正確さに欠けるがある程度の目安にはなると思う。

従って、今回の作業により I/O 回数は大幅な削減が図られ、Elapsed Time の減少も期待できる。

Table 6.1 Base input data to test
the DOT-DD I/O speedup

形 状	R-Z
R 方向メッシュ数	16
Z 方向メッシュ数	34
Sn 数	48
群数	125

Table 6.2 I/O speedup effect for DOT-DD code

	オリジナル版	VIO/F 化版
I / O 回 数	166683	4908
CPU Time	5 分 23.35 秒	5 分 11.10 秒
VU Time	2 分 10.36 秒	2 分 5.88 秒
メモリ	4776 KB	4872 KB
拡張メモリ	0	15 MB
Elapsed Time	90 分 47.41 秒	45 分 30.27 秒

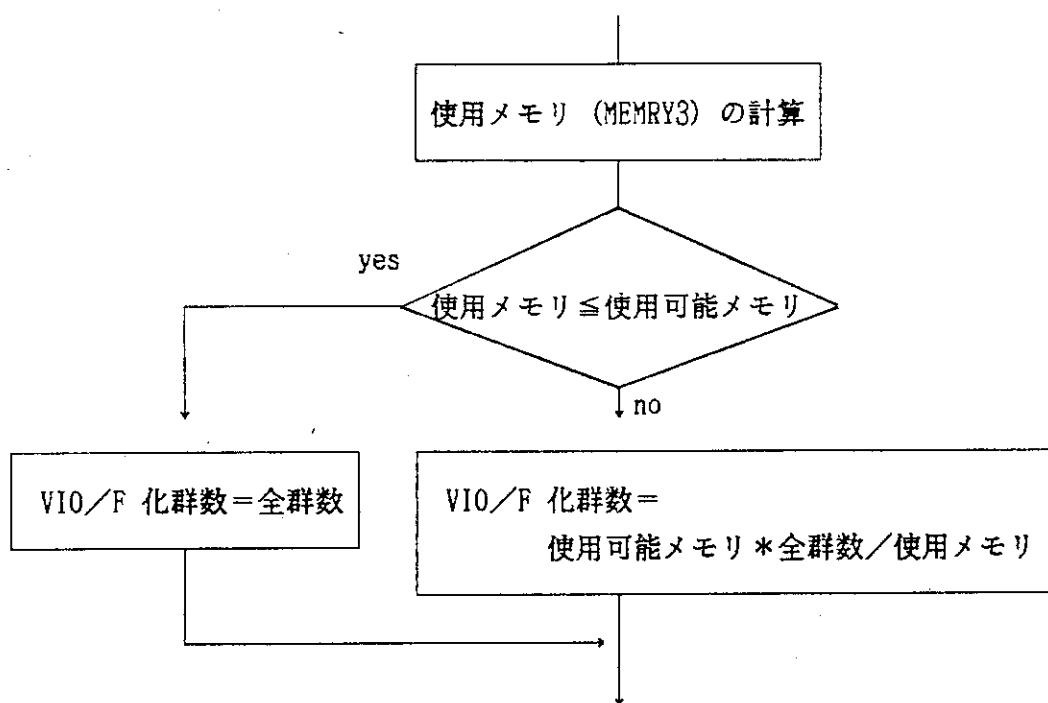


Fig. 6.1 Algorithm to set the number of groups to reside the coefficients in core for using VIO/F (DOT-DD code)

7. ベクトル化版留意事項

ベクトル化版を使用する際の制限事項とベクトル化版のメモリの増加について以下に記す。

7.1 使用制限事項

ベクトル化版使用制限事項を以下に示す。DOT 3.5 NEA オリジナル版, DOT 3.5 FNS 版, DOT 3.5 炉設計版と RADHEA-V 4 では制限事項が異なるので別々に記す。

(1) DOT 3.5 NEA オリジナル版, DOT 3.5 FNS 版, DOT 3.5 炉設計版

(a) 形 状

ベクトル化版で使用できる形状は、X-Y 形状と R-Z 形状だけであり、R-θ 形状は使用できない。

(b) 計算モデル

ベクトル化版で使用できる計算モデルは、weighted difference ($FXT=3$) と mixed linear-weighted ($FXT=4$) だけである。

(c) ルジャンドル展開次数

新しいベクトル化版では制限はない。以前のベクトル化版では、3 次及び 5 次のみ計算可能であった。

(d) i 方向メッシュサイズ

i 方向メッシュサイズ (X-Y 形状では X 方向, R-Z 形状では R 方向のメッシュサイズ) の上限は 1500 である。

(e) ij メッシュサイズ

i 方向と j 方向のメッシュサイズの積の上限は、50000 である。

(2) RADHEAT-V 4

(a) 形 状

ベクトル化版でも全形状について計算可能である。しかし、R-θ 形状については、GRIND 部分がベクトル化されていないので、他の形状よりも遅い。

(b) ベクトル化版でも全計算モデルについて計算可能である。しかし、mixed linear-step mode ($FXT=0$), linear mode ($FXT=1$) 及び step mode ($FXT=2$) については、GRIND 部分がベクトル化されていないので、他の計算モデルよりも遅い。

(c) i 方向メッシュサイズ

i 方向メッシュサイズ (X-Y 形状では X 方向, R-Z 形状では R 方向のメッシュサイズ) の上限は、1500 である。

7.2 メモリの増加について

DOT 3.5 NEA オリジナル版, DOT 3.5 FNS 版, DOT 3.5 炉設計版と RADHEAT-V 4 ではメモリの増加量が異なるので別々に記す。

(1) DOT 3.5 NEA オリジナル版, DOT 3.5 FNS 版, DOT 3.5 炉設計版

DOT 3.5 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いる。サブルーチン GRIND をベクトル化する際に 1500 の大きさの配列を各サブルーチンで 20 個ずつ, サブルーチン OUTER をベクトル化する際に 50000 の大きさの配列を 5 個必要とする。

したがって増加メモリは次式のように約 1.5 メガバイトになる。

$$\begin{aligned}\text{増加メモリ} &= (1500 \times 20 \times 4 + 50000 \times 5) \times 4 \text{バイト} \\ &= 1480\text{Kバイト}\end{aligned}$$

(2) RADHEAT-V 4

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いる。サブルーチン GRIND をベクトル化する際に 1500 の大きさの配列を各サブルーチンで 20 個ずつ作成した。

したがって増加メモリは次式のように約 0.5 メガバイトになる。

$$\begin{aligned}\text{増加メモリ} &= 1500 \times 20 \times 4 \times 4 \text{バイト} \\ &= 480\text{Kバイト}\end{aligned}$$

8. おわりに

2次元ディスクリート・オーディネーツ輸送コード DOT 3.5 系列のベクトル化を実施した。利用者の多い DOT 3.5 コードのベクトル化版の普及を目的としてこのバージョンに対して種々のベクトル化改良、I/O 高速化を施し、また多くのテスト計算を実施することによりベクトル化版を検証した。

今までベクトル化整備が十分でなかった各研究室の個別の、あるいは、一部修正、改良して使用しているバージョン（FNS 版、炉設計版）に対してもベクトル化を実施し、計算結果についても十分な評価、検証を行った。さらに DOT 3.5 コードの改良版が組み込まれている遮蔽安全評価用群定数作成及び放射線輸送解析コードシステム RADHAT-V 4 の 2 次元輸送計算部分（ESPRIT モジュール）のベクトル化を実施した。また、DOT 3.5 コードを用いて計算する場合の大きな問題点であった経過時間（elapsed time）及び入出力（I/O）回数についても主記憶ファイル入出力（VIO/F）機能を用いて大幅に削減することができた。

ベクトル化により、CPU 時間が DOT 3.5 では 1.7 倍～1.9 倍に、DOT 3.5-FNS 版では 2.2 倍～2.3 倍に、DOT 3.5 炉設計版では 1.7 倍に、RADHEAT-V 4 では 3.1 倍～4.4 倍に高速化された。さらに I/O 高速化により、経過時間が、DOT 3.5-FNS 版では 51.5%～65.9% に、DOT-DD では 50.1% に減少した。

これらの高速化率は、決して高いものではないが、原研においては新しいスーパーコンピュータ（FACOM VP-2600）も近々導入される予定であるので、これらのベクトル化版を大いに利用してスーパーコンピュータを有効に活用していただきたいと思う。

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、富士通㈱南多善氏に感謝します。

DOT 3.5 コード FNS 版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、原研核融合炉物理研究室小迫和明氏に感謝します。DOT 3.5 コード炉設計版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、原研炉設計研究室関泰氏に感謝します。RADHEAT-V 4 システムのベクトル化作業の際に、貴重な助言をいただきました原研プラント安全解析研究室長内藤倣孝氏に感謝します。また、RADHEAT-V 4 システム及びデータの提供と貴重な助言をいただきました原研プラント安全研究室増川史洋氏に感謝します。DOT-DD コードの I/O 高速化作業の際に、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中川正幸氏に感謝します。DOT-DD コードの I/O 高速化作業の際に、貴重な助言をいただきました日本総合研究所佐々木誠氏に感謝します。

8. おわりに

2次元ディスクリート・オーディネーツ輸送コード DOT 3.5 系列のベクトル化を実施した。利用者の多い DOT 3.5 コードのベクトル化版の普及を目的としてこのバージョンに対して種々のベクトル化改良、I/O 高速化を施し、また多くのテスト計算を実施することによりベクトル化版を検証した。

今までベクトル化整備が十分でなかった各研究室の個別の、あるいは、一部修正、改良して使用しているバージョン（FNS 版、炉設計版）に対してもベクトル化を実施し、計算結果についても十分な評価、検証を行った。さらに DOT 3.5 コードの改良版が組み込まれている遮蔽安全評価用群定数作成及び放射線輸送解析コードシステム RADHAT-V 4 の 2 次元輸送計算部分（ESPRIT モジュール）のベクトル化を実施した。また、DOT 3.5 コードを用いて計算する場合の大きな問題点であった経過時間（elapsed time）及び入出力（I/O）回数についても主記憶ファイル入出力（VIO/F）機能を用いて大幅に削減することができた。

ベクトル化により、CPU 時間が DOT 3.5 では 1.7 倍～1.9 倍に、DOT 3.5-FNS 版では 2.2 倍～2.3 倍に、DOT 3.5 炉設計版では 1.7 倍に、RADHEAT-V 4 では 3.1 倍～4.4 倍に高速化された。さらに I/O 高速化により、経過時間が、DOT 3.5-FNS 版では 51.5%～65.9% に、DOT-DD では 50.1% に減少した。

これらの高速化率は、決して高いものではないが、原研においては新しいスーパーコンピュータ（FACOM VP-2600）も近々導入される予定であるので、これらのベクトル化版を大いに利用してスーパーコンピュータを有効に活用していただきたいと思う。

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、富士通㈱南多善氏に感謝します。

DOT 3.5 コード FNS 版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、原研核融合炉物理研究室小迫和明氏に感謝します。DOT 3.5 コード炉設計版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、原研炉設計研究室関泰氏に感謝します。RADHEAT-V 4 システムのベクトル化作業の際に、貴重な助言をいただきました原研プラント安全解析研究室長内藤淑孝氏に感謝します。また、RADHEAT-V 4 システム及びデータの提供と貴重な助言をいただきました原研プラント安全研究室増川史洋氏に感謝します。DOT-DD コードの I/O 高速化作業の際に、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中川正幸氏に感謝します。DOT-DD コードの I/O 高速化作業の際に、貴重な助言をいただきました日本総合研究所佐々木誠氏に感謝します。

参考文献

- (1) 野々宮巖, 原田裕夫; 原子力コードのベクトル化 88-1—SRAC, CITATION, TWO-TRAN-II, COREBN, CITATION-FBR—, JAERI-M 89-030 (1989)
- (2) 野々宮巖, 折居茂夫, 平塚 篤, 原田裕夫; 原子力コードのベクトル化 89-1—PHENIX, FPGS—, JAERI-M 89-124 (1989)
- (3) 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, Tokoyo, Japan, July 15-17, 1986
- (4) DOT 4.2 コードチューニング報告書, 私信 (1985) 動力炉・核燃料開発事業団
- (5) FACOM OS IV/F 4 MSP FORTRAN 77/VP 使用手引書 V 10 用 (1988), 富士通(株)
- (6) F. R. Mynatt, et al.; DOT III a Two-Dimensional Discrete Ordinates Transport Code, ORNL-TM-4280 (June 1973)
- (7) 石黒美佐子, 筒井恒夫; 中性子輸送コードのベクトル化 (DOT 3.5, TWOTRAN, ANISN, PALLAS, BERMUDA), JAERI-M 82-199 (1982)
- (8) H. Kawasaki and Y. Seki; APPLE-2: An Improved Version of APPLE Code for Plotting Neutron and Gamma Ray Spectra and Reaction Rates, JAERI-M 82-091 (1982)
- (9) Y. Seki, H. Iida, H. Kawasaki and K. Yamada; THIDA-2: An Advanced Code System for Calculation of Transmutation, Activation, Decay Heat and Dose Rate, JAERI 1301 (1986)
- (10) N. Yamano, K. Minami, K. Koyama and Y. Naito; RADHEAT-V 4: A Code System to Generate Multigroup Constants and Analyze Radiation Transport for Shielding Safety Evaluation, JAERI 1316 (May. 1989)
- (11) T. Mori, M. Nakagawa, M. Sasaki; One-, Two-, Three- Dimensional Transport Codes Using Multi-Group Double-Differential Form Cross Sections, JAERI 1314 (November 1988)

付録 A DOT 3.5 NEA オリジナル版使用手引

(1) JCL

DOT 3.5 コード NEA オリジナル版ベクトル化版のコンパイル, リンク, 実行の JCL は次の通りであり, J0002.DOT35NEA.CNTL (CLG) に登録した。

```
//JCLG JOB
//      EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 20579202,IW.NONOMIYA,0341.02
//      T.6 C.5 W.4 I.7
//      OPTP  PASSWORD=????,CLASS=5
// EXEC    FORT77VP,SO='J0002.DOT35VP1',DISP=MOD,
//      A='ELM(*)',RGN=5000K
//SYSPRINT DD DUMMY
// EXEC    FORT77,SO='J0002.DOT350R',DISP=MOD,OPT=3,
//      A='ELM(*)',RGN=5000K
//SYSPRINT DD DUMMY
// EXEC LKED77
// EXEC GO
//FT06F001 DD SYSOUT=*
//FT01F001 DD DISP=(,DELETE),DSN=&&FT01,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT02F001 DD DISP=(,DELETE),DSN=&&FT02,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT03F001 DD DISP=(,DELETE),DSN=&&FT03,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT04F001 DD DISP=(,DELETE),DSN=&&FT04,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT14F001 DD DISP=(,DELETE),DSN=&&FT14,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT90F001 DD DISP=(,DELETE),DSN=&&FT90,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT08F001 DD DSN=J0002.DOTRZLIB.DATA,DISP=SHR
//FT09F001 DD DISP=(,DELETE),DSN=&&FT09,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(900,500))
//FT10F001 DD DISP=(,DELETE),DSN=&&FT10,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT11F001 DD DISP=(,DELETE),DSN=&&FT11,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT12F001 DD DISP=(,DELETE),DSN=&&FT12,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT13F001 DD DISP=(,DELETE),DSN=&&FT13,
```

```

// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
// FT15F001 DD DISP=(,DELETE),DSN=&&FT15,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
// FT20F001 DD DISP=(,DELETE),DSN=&&FT20,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
// FT21F001 DD DISP=(,DELETE),DSN=&&FT21,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
// FT50F001 DD DISP=(,DELETE),DSN=&&FT50,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
// FT91F001 DD DISP=(,DELETE),DSN=&&FT91,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
// EXPAND DISKTO,DDN=SYSIN,DSN='J0002.DOT35.FEM2',Q='DATA(COREXY)', ++
// 

```

(2) 使用制限事項

(a) 形 状

ベクトル化版で使用できる形状は、X-Y 形状と R-Z 形状だけであり、R-θ 形状は使用できない。

(b) 計算モデル

ベクトル化版で使用できる計算モデルは、weighted difference (FXT = 3) と mixed linear-weighted (FXT = 4) だけである。

(c) ルジャンドル展開次数

新しいベクトル化版では制限はない。以前のベクトル化版では、3 次及び 5 次のみ計算可能であった。

(d) i 方向メッシュサイズ

i 方向メッシュサイズ (X-Y 形状では X 方向、R-Z 形状では R 方向のメッシュサイズ) の上限は 1500 である。

(e) ij メッシュサイズ

i 方向と j 方向のメッシュサイズの積の上限は、50000 である。

(3) メモリの増加について

DOT 3.5 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いる。サブルーチン GRIND をベクトル化する際に 1500 の大きさの配列を各サブルーチンで 20 個ずつ、サブルーチン OUTER をベクトル化する際に 50000 の大きさの配列を 5 個作成した。

したがって増加メモリは次式のように約 1.5 メガバイトになる。

$$\begin{aligned}
 \text{増加メモリ} &= (1500 \times 20 \times 4 + 50000 \times 5) \times 4 \text{ バイト} \\
 &= 1480 \text{ Kバイト}
 \end{aligned}$$

付録 B DOT 3.5 FNS 版使用手引

(1) JCL

DOT 3.5 FNS 版ベクトル化版のコンパイル、リンク、実行の JCL は次の通りであり、J0002.DOT35FNS.CNTL (CLG) に登録した。

```
//JCLG JOB
//      EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 20579202,IW.NONOMIYA,0341.02
//          T.8 C.7 I.5 W.4 E.55 SRP
//          OOPTP  PASSWORD=????,CLASS=4
// EXEC      FORT77VP,SO='J0001.DOT35V10',DISP=MOD,
//          A='ELM(DUTERVOY)',RGN=5000K
//SYSPRINT DD DUMMY
// EXEC      FORT77VP,SO='J0002.DOT35VP1',DISP=MOD,
//          A='ELM(*)',RGN=5000K
//SYSPRINT DD DUMMY
//* ----- DOT3.5 2DN SN-CALCULATION --- VECTOR-VERSION -----
// EXEC LKEDIT77,LM=J0001.DOT35.KKLOAD,CNTL=NO
//     INCLUDE OLDLM(DOT35V10)
//     ENTRY MAIN
//     NAME TEMPNAME
//*SYSPRINT DD DUMMY
//*
//DOT35 EXEC GO,
//          A='HIO=(2,3,4,14,90)'
//FT06F001 DD SYSOUT=*,DCB=(RECFM=FBA,LRECL=137,BLKSIZE=8220)
//FT01F001 DD UNIT=WK10,SPACE=(TRK,(300,100)),DISP=NEW,
//          DCB=(RECFM=FB,LRECL=32756,BLKSIZE=32760)
//* ----- HIO-VERSION OF I/O PROCESS -----
//FT02F001 DD UNIT=WK10,SPACE=(CYL,150,,CONTIG),DISP=NEW,DSN=&&FT02
//FT03F001 DD UNIT=WK10,SPACE=(CYL,150,,CONTIG),DISP=NEW,DSN=&&FT03
//FT04F001 DD UNIT=WK10,SPACE=(CYL,150,,CONTIG),DISP=NEW,DSN=&&FT04
//FT14F001 DD UNIT=WK10,SPACE=(CYL,150,,CONTIG),DISP=NEW,DSN=&&FT14
//FT90F001 DD UNIT=WK10,SPACE=(CYL,150,,CONTIG),DISP=NEW,DSN=&&FT90
//* -----
//FT08F001 DD DSN=J0001.DOT35.JACKAS10.DATA,DISP=SHR,LABEL=(,,IN)
//* ----- DOT3.5 OUTPUT FLUX FILE --> FT09F001 -----
//FT09F001 DD UNIT=WK10,SPACE=(TRK,(2000,200)),DISP=NEW,
//          DCB=(DSORG=PS,RECFM=VBS,LRECL=32756,BLKSIZE=32760)
//* -----
//FT10F001 DD UNIT=WK10,SPACE=(TRK,(10,10)),DISP=NEW,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=32760)
//FT11F001 DD UNIT=WK10,SPACE=(TRK,(300,100)),DISP=NEW,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=32760)
//FT12F001 DD UNIT=WK10,SPACE=(TRK,(10,10)),DISP=NEW,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=32760)
//FT13F001 DD UNIT=WK10,SPACE=(TRK,(10,10)),DISP=NEW,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=32760)
//FT15F001 DD UNIT=WK10,SPACE=(TRK,(2400,200)),DISP=NEW,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=32760)
//FT18F001 DD DSN=J9202.DOTCROSW.DATA,DISP=SHR
//FT20F001 DD UNIT=WK10,SPACE=(TRK,(10,10)),DISP=NEW,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=32760)
```

```
//FT91F001 DD UNIT=WK10,SPACE=(TRK,(100,50)),DISP=NEW,
//          DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=32760)
//*
//FT89F001 DD SUBSYS=(VPCS,'SPACE=50M')
//*
//* ----- DOT3.5 INPUT DATA-SET -----
//SYSIN    DD DSN=J0001.DOT35.SLABASSY.DATA(BESL4DT),
//          DCB=(DSORG=PS),LABEL=(,,,IN),DISP=SHR
++
//
```

(2) VIO/F 化領域の変更

VIO/F 化領域を変更したい場合、プログラムと JCL の修正が必要である。現在、プログラムでは VIO/F 化領域の大きさを 92000000 で定義している。これをバイトで表すと 36.8 MB である。JCL では、これより大きな値が定義してあればよい。現在は 50 MB である。また、拡張領域の大きさを示す E. は、JCL で定義している値より大きな値であればよい。現在は、E. 55 (拡張領域 55 MB) である。

(a) プログラム (サブルーチン OUTER)

```
CNN
  DIMENSION IZZZ1(50000),ZZZZ(50000),ZZZ1(50000)
  DIMENSION IIIN(50000),JJIN(50000)
CNN
C.CS
  DATA MVI0F / |92000000| /
  M90 = ( IM*JM*( 1 + A03*( A03 + 3 )/2 ) +
  1           MM*( JM + IM ) ) *IGM*4
```

(b) JCL

```
//JCLG JOB
//      EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 20579202,IW.NONOMIYA,0341.02
//          T.8 C.7 I.5 W.4 [E.55] SRP
//          OOPTP PASSWORD=????,CLASS=4
// EXEC FORT77VP,SD='J0001.DOT35VIO',DISP=MOD,
//          A='ELM(OUTERVOY)',RGN=5000K
//SYSPRINT DD DUMMY
}

/*
//FT89F001 DD SUBSYS=(VPCS,'SPACE=50M')
//*
//* ----- DOT3.5 INPUT DATA-SET -----
//SYSIN    DD DSN=J0001.DOT35.SLABASSY.DATA(BESL4DT),
//          DCB=(DSORG=PS),LABEL=(,,,IN),DISP=SHR
++
//
```

(3) 使用制限事項

(a) 形 状

ベクトル化版で使用できる形状は、X-Y 形状と R-Z 形状だけであり、R-θ 形状は使用できない。

(b) 計算モデル

ベクトル化版で使用できる計モデルは、weighted difference ($FXT = 3$) と mixed linear-weighted ($FXT = 4$) だけである。

(c) ルジャンドル展開次数

新しいベクトル化版では制限はない。以前のベクトル化版では、3 次及び 5 次のみ計算可能であった。

(d) i 方向メッシュサイズ

i 方向メッシュサイズ (X-Y 形状では X 方向、R-Z 形状では R 方向のメッシュサイズ) の上限は 1500 である。

(e) ij メッシュサイズ

i 方向と j 方向のメッシュサイズの積の上限は、50000 である。

(4) メモリの増加について

DOT 3.5 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いる。サブルーチン GRIND をベクトル化スル際に、1500 の大きさの配列を各サブルーチンで 20 個ずつ、サブルーチン OUTER をベクトル化する際に 50000 の大きさの配列を 5 個作成した。

したがって増加メモリは次式のように約 1.5 メガバイトになる。

$$\begin{aligned}\text{増加メモリ} &= (1500 \times 20 \times 4 + 50000 \times 5) \times 4 \text{ バイト} \\ &= 1480 \text{K バイト}\end{aligned}$$

付録 C DOT 3.5 炉設計版使用手引

(1) JCL

DOT 3.5 炉設計版ベクトル化版のコンパイル, リンク, 実行の JCL は次の通りであり, J0002.
DOT35FER.CNTL (CLG) に登録した。

```
//JCLG JOB
//      EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 20579202,IW.NONOMIYA,0341.02
// T.5 C.7 W.3 I.4 SRP
// OPTP PASSWORD=????,CLASS=0
// EXEC FORT77,SO='J2372.DOT35JCL',Q='CNTL',A='ELM(GICX40DT)'
// EXEC LKED77
// EXEC GO
// EXPAND DISKTO,DDN=FT01F001,DSN='J2372.GICX40'
// FT02F001 DD DISP=(NEW,PASS),DSN=&&GICX,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(900,500))
//SYSIN DD *
2 3 42 1
/*
//***** DOT3.5 CALC.          J2372.THIDA3PK.DATA(DOTRUN) *****
//*      FT05 : READ OF SIMPLE INPUT DATA CARDS.           *
//*      FT08 : READ OF GICX40 CROSS SECTION.             *
//*      FT09 : WRITE OF FLUX FOR RESTART.                *
//*      FT11 : WRITE OF ANGULAR FLUX FOR DOMINO.        *
//*      FT18 : READ OF FLUX GEUSS FOR REATART.         *
//***** EXEC FORT77VP,SO='J0002.DOT35VP1',DISP=MOD,
// A='ELM(*)',RGN=7000K
//SYSPRINT DD DUMMY
// EXEC FORT77,SO='J2372.DOT35',Q='FORT',A='ELM(OUTER,TPSAVE)',DISP=MOD
// EXEC FORT77,SO='J2372.DOT35',Q='FORT',A='ELM(TPXF)',DISP=MOD
//SYSPRINT DD DUMMY
// EXEC FORT77,SO='J2372.DOT35',Q='FORT',A='ELM(ACTVTY)',DISP=MOD
//SYSPRINT DD DUMMY
// EXEC LKEDIT77,LM='J2372.DOT35',A='LREP(JMF,JMP)'
// EXEC GO
//FT06F001 DD DCB=(BLKSIZE=137)
//FT01F001 DD DISP=(,DELETE),DSN=&&FT01,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT02F001 DD DISP=(,DELETE),DSN=&&FT02,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(300,100))
//FT03F001 DD DISP=(,DELETE),DSN=&&FT03,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(300,100))
//FT04F001 DD DISP=(,DELETE),DSN=&&FT04,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(300,100))
```

```

//FT08F001 DD DISP=(OLD,DELETE),DSN=&&GICX,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(900,500))
//FT09F001 DD DISP=(,CATLG,DELETE),DSN=&&FT09,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(20,10),RLSE)
//FT10F001 DD DISP=(,DELETE),DSN=&&FT10,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT11F001 DD DISP=(,DELETE),DSN=&&FT11,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT12F001 DD DISP=(,DELETE),DSN=&&FT12,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT13F001 DD DISP=(,DELETE),DSN=&&FT13,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT14F001 DD DISP=(,DELETE),DSN=&&FT14,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT15F001 DD DISP=(,DELETE),DSN=&&FT15,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT20F001 DD DISP=(,DELETE),DSN=&&FT20,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT21F001 DD DISP=(,DELETE),DSN=&&FT21,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
//FT50F001 DD DISP=(,DELETE),DSN=&&FT50,
// DCB=(RECFM=VBS,LRECL=19064,BLKSIZE=19068),UNIT=WK10,
// SPACE=(TRK,(100,50))
// EXPAND DISKTO,DDN=SYSIN,DSN='J0002.DOT35.FEM2',Q=' .DATA(SEKIDAT1)'
++
//

```

(2) 使用制限事項

(a) 形 状

ベクトル化版で使用できる形状は、X-Y 形状と R-Z 形状だけであり、R-θ 形状は使用できない。

(b) 計算モデル

ベクトル化版で使用できる計算モデルは、weighted difference (FXT=3) と mixed linear-weighted (FXT=4) だけである。

(c) ルジャンドル展開次数

新しいベクトル化版では制限はない。以前のベクトル化版では、3次及び5次のみ計算可能であった。

(d) i 方向メッシュサイズ

i 方向メッシュサイズ (X-Y 形状では X 方向、R-Z 形状では R 方向のメッシュサイズ) の上限は 1500 である。

(e) ij メッシュサイズ

i 方向と j 方向のメッシュサイズの積の上限は、50000 である。

(3) メモリの増加について

DOT 3.5 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いる。サブルーチン GRIND をベクトル化する際に 1500 の大きさの配列を各サブルーチンで 20 個ずつ、サブルーチン OUTER をベクトル化する際に 50000 の大きさの配列を 5 個作成した。

したがって増加メモリは次式のように約 1.5 メガバイトになる。

$$\begin{aligned}\text{増加メモリ} &= (1500 \times 20 \times 4 + 50000 \times 5) \times 4 \text{ バイト} \\ &= 1480 \text{K バイト}\end{aligned}$$

付録 D RADHEAT-V4 使用手引

(1) JCL

RADHEAT-V4 システム ESPRIT モジュールベクトル化版のコンパイル、リンク、実行の JCL は次の通りであり、J0002.RADHEAT4.CNTL (CLG) に登録した。

```
//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLM='++'
// JUSER 20579202,IW.NONOMIYA,0341.02
T.6 I.4 W.3 C.3 SRP
OPTP PASSWORD=????,CLASS=0
// EXEC ASM,SO='J1446.DPOOL2',Q='ASM(GETDCB)'
// EXEC ASMLK
//SYSLMOD DD DSN=&&ASMLLOAD,DISP=(NEW,PASS,DELETE),SPACE=(TRK,(5,3,2)),
//           UNIT=WK10
//SYSIN DD *
  NAME GETDCB
/*
//FORT  EXEC FORT77
//SYSIN DD *
  SUBROUTINE ALOCAT(LOC,SUB)
  DIMENSION D(300000)
  IF(LOC.GT.0) GO TO 10
  LOC=300000
  CALL DTLIST
  GO TO 999
10  CONTINUE
  CALL SUB(D,LOC)
999  RETURN
E   N   D
/*
// EXEC FORT77VP,SO='J0002.RADV4VP',DISP=MOD,
// A='ELM(*)',RGN=3000K
//SYSPRINT DD DUMMY
// EXEC FORT77,SO='J1446.ESPRIT2X',A='ELM(*)',DISP=MOD,OPT=3,
// RGN=3000K
//SYSPRINT DD DUMMY
// EXEC FORT77,SO='J1446.DPOOL2',A='ELM(*)',DISP=MOD,OPT=3,
// RGN=3000K
//SYSPRINT DD DUMMY
// EXEC LKED77,PRVLIB='&&ASMLLOAD',PRVQ=
//ESPRIT EXEC GO,ORECFM=FA,OBSIZE=137
//FT01F001 DD DSN=&&F1,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT02F001 DD DSN=&&F2,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT03F001 DD DSN=&&F3,UNIT=WK10,SPACE=(TRK,(300,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT04F001 DD DSN=&&F4,UNIT=WK10,SPACE=(TRK,(300,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT09F001 DD DSN=&&F9,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT10F001 DD DSN=&&FA,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
```

```

//FT12F001 DD DSN=&&FB,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT13F001 DD DSN=&&FC,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT14F001 DD DSN=&&FD,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT15F001 DD DSN=&&FE,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT30F001 DD DSN=&&FF,UNIT=WK10,SPACE=(TRK,(100,100)),
// DCB=(LRECL=18632,BLKSIZE=18636,RECFM=VBS)
//FT91F001 DD DSN=J1446.POOL87.DATA,DISP=SHR,LABEL=(,,,IN)
//SYSIN DD *

```

コントロールデータ

```

++  
//

```

(2) 制限事項

(a) 形 状

ベクトル化版でも全形状について計算可能である。しかし、R-θ形状については、GRIND 部分がベクトル化されていないので、他の形状よりも遅い。

(b) 計算モデル

ベクトル化版でも全計算モデルについて計算可能である。しかし、mixed linear-step mode (FXT=0), linear mode (FXT=1) 及び step mode (FXT=2) については、GRIND 部分がベクトル化されていないので、他の計算モデルよりも遅い。

(c) i 方向メッシュ

i 方向メッシュサイズ (X-Y 形状では X 方向, R-Z 形状では R 方向のメッシュサイズ) の上限は、1500 である。

(3) メモリの増加について

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いる。サブルーチン GRIND をベクトル化する際に 1500 の大きさの配列を各サブルーチンで 20 個ずつ作成した。

したがって増加メモリは次式のように約 0.5 メガバイトになる。

$$\begin{aligned}
 \text{増加メモリ} &= 1500 \times 20 \times 4 \times 4 \text{ バイト} \\
 &= 480 \text{ Kバイト}
 \end{aligned}$$

付録 E DOT-DD 使用手引

(1) JCL

DOT-DD コード I/O 高速化版のコンパイル, リンク, 実行の JCL は次の通りであり, J0002.DOTDD.CNTL に登録した。

```

//FT71F001 DD DSN=J3803.DDXLIB3.DATA,DISP=SHR,LABEL=(,,,IN)
/*
/* ----- DIRECT ACCESS FILE FOR MIXING (NDO) -----
//FT50F001 DD UNIT=TSSWK,SPACE=(TRK,(60,0)),DCB=RECFM=F,DISP=(,DELETE)
/* ----- WORK FILE FOR MIXING (NDOA) -----
//FT51F001 DD UNIT=WK10,SPACE=(TRK,(100,50),RLSE),
// DCB=BLKSIZE=19069
/*
/* ----- WORK FILE FOR MIXING (NDM) -----
//FT52F001 DD UNIT=WK10,SPACE=(TRK,(100,50),RLSE),
// DCB=BLKSIZE=19069
//FT65F001 DD *

    インプットデータ

/*
//SYSIN DD *

    コントロールデータ

/*
/*
/*     DOT-DDX LI20 SLAB #1A AIR(20)+LI20(60CM)
/*
//FORT7 EXEC FORT77VP,SO=J0002.DOTDDVIO,
//   A='ELM(*),NOS',
//   RGN=5120K,DISP=MOD
//SYSPRINT DD DUMMY
/*
//FORT7 EXEC FORT77VP,SO=J0002.DOTDD,A='ELM(*),NOS',
//   RGN=5120K,DISP=MOD
//SYSPRINT DD DUMMY
/*
//ASM EXEC ASM,SO=J2350.0.SASAKI,Q='ASM(GRTUNCL)',
//   DISP=MOD
//SYSPRINT DD DUMMY
/*
//LKED EXEC LKED77,CNTL=NO,PRVLIB='J2031.LIB431'
//SYSIN DD *
  ENTRY MAIN
  NAME TEMPNAME(R)
/*
//DOTSAN EXEC GO,A='HIO=(3,4,62)',OBSIZE=137
//FT06F001 DD SYSOUT=*
/*
/* ----- SCARATCH FILE -----
//FT02F001 DD UNIT=WK10,SPACE=(CYL,80),           < NCR1 >
//   DCB=BLKSIZE=32760
/*
/* ----- SCRATCH FILE -----
//FT03F001 DD UNIT=WK10,SPACE=(CYL,80,,CONTIG)      < NFLUX1 >
//FT99F001 DD SUBSYS=(VPCS,'SPACE=29M')
/*
/* ----- SCARTCH FILE -----
//FT04F001 DD UNIT=WK10,SPACE=(CYL,80,,CONTIG)      < NSCRAT >
/*
//FT07F001 DD UNIT=WK10,SPACE=(TRK,(150,50),RLSE),
//   DCB=(RECFM=VBS,BLKSIZE=19069)
/*
/* ----- MIXED CROSS SECTION -----
//FT08F001 DD DSN=&&WK4,DISP=SHR
/*
/* ----- RESART OUTPUT -----
//FT09F001 DD DUMMY

```

```

/*
/* ----- RESTART INPUT -----
//FT20F001 DD DUMMY
/*
/* ----- ANGULAR FLUX OUTPUT -----
//FT10F001 DD DUMMY
/*
/* ----- TOTAL FLUX OUTPUT -----
//FT41F001 DD DUMMY
/*
----- FT11F001 DD UNIT=WK10,SPACE=(TRK,(90,10),RLSE), < NBFT >
// DCB=(RECFM=VBS,BLKSIZE=19069,LRECL=X)
//FT12F001 DD UNIT=WK10,SPACE=(TRK,(30,10),RLSE) < NGAM >
/*
/* ----- WORK FILE FOR ZONE BALANCE TABLE -----
//FT13F001 DD UNIT=WK10,SPACE=(TRK,(30,10),RLSE) < NZBT >
/*
/* ----- BOUNDARY SOURCE -----
//FT14F001 DD UNIT=WK10,SPACE=(TRK,(30,10),RLSE) < NBSO >
/*
/* ----- FIRST COLLISION SOURCE & UNCOLLIDED FLUX -----
//FT15F001 DD DSN=&&WK15,DISP=SHR < NPSO >
/*
//FT50F001 DD UNIT=WK10,SPACE=(TRK,(150,50),RLSE),
// DCB=(RECFM=F,DSORG=DA)
//FT51F001 DD UNIT=WK10,SPACE=(TRK,(150,50),RLSE),
// DCB=(RECFM=VBS,BLKSIZE=19069)
//FT52F001 DD UNIT=WK10,SPACE=(TRK,(150,50),RLSE),
// DCB=(RECFM=VBS,BLKSIZE=19069)
/*
/* ----- ANGULAR TRANSPORT FUNCTION (NDP) -----
//FT61F001 DD DSN=J9021.IGL0048.DATA,DISP=SHR,LABEL=(,,,IN) (S48)
/*
/* ----- XPMM = X-SEC * ANGULAR TRANSPORT FUNCTION (NP) -----
//FT62F001 DD UNIT=WK10,SPACE=(CYL,60,,CONTIG) < NP >
//FT63F001 DD UNIT=WK10,SPACE=(TRK,(30,10)) < NPP >
/*
//FT70F001 DD DSN=J3803.DDXLIB3.DATA,DISP=SHR,LABEL=(,,,IN)
//***** */
//SYSIN DD *
コントロールデータ

++
//
```

(2) VIO/F 化領域の変更

VIO/F 化領域を変更したい場合、プログラムと JCL の修正が必要である。現在、プログラムでは VIO/F 化領域の大きさを 29000000 で定義している。これをバイトで表すと 11.6 MB である。JCL では、これより大きな値が定義してあればよい。現在は 29 MB である。また、拡張領域の大きさを示す E. は、JCL で定義している値より大きな値であればよい。現在は、E.30 (拡張領域 30 MB) である。

(a) プログラム (サブルーチン ALOCAT)

```

SUBROUTINE ALOCAT(LOC,SUB)
PARAMETER( LIMIT = 1000000 )
COMMON /DCOMMN/ D(LIMIT)
CNN
COMMON /VIDEX/ IGVIOF,VIOF,NFLUX2,MVIOF,MEMORY3
MVIOF = 29000000
CNN
1 IF(LOC.GT.0) GO TO 10
LOC = LIMIT
GO TO 999
10 CONTINUE
CALL SUB(D,LOC)
999 RETURN
END

```

この値を変更する

(b) JCL

```

//JCLG JOB
//JCLG EXEC JCLG
//SYSIN DD DATA,DLME='++'
// JUSER 30479202,IW,NONOMIYA,0341.02
    T.6 C.5 I.6 W.4 E.30
OPTP PASSWORD=? ,CLASS=6
                                     ↓ これらの値を変更する
{
}

//* ----- SCARATCH FILE -----
//FT02F001 DD UNIT=WK10,SPACE=(CYL,80),
//           DCB=BLKSIZE=32760
//*
//* ----- SCRATCH FILE -----
//FT03F001 DD UNIT=WK10,SPACE=(CYL,80,,CONTIG)
//FT99F001 DD SUBSYS=(VPCS,'SPACE=29M')
//*
//* ----- SCARTCH FILE -----
//FT04F001 DD UNIT=WK10,SPACE=(CYL,80,,CONTIG)
//*

```

(3) メッセージ

VIO/F 化された群数のメッセージが機番 6 番から次のように出力される。

```

== NUMBER OF GROUP USED IN VIO/F : 125 ===
== RATE OF VIO/F ( % ) : 100.00 ===
== LIMIT MEMORY SIZE (BYTE) : 29000000 ===
== USED MEMORY SIZE (BYTE) : 14528000 ===

```

上から順に

- VIO/F 化された群数
- VIO/F 化された割合
- メモリの上限
- 使われたメモリ

である。