ISSN 0389-4010
UDC 681.31

航空宇宙技術研究所報告

TECHNICAL REPORT OF NATIONAL AEROSPACE LABORATORY

TR-909

プログラム高速化技術とスーパコンピュータSXシステムによる検証

中村絹代・吉田正廣・峯尾真一

1986年7月

航空宇宙技術研究所
NATIONAL AEROSPACE LABORATORY

プログラム高速化技術とスーパコンピュータ SXシステムによる検証*

中村絹代** 吉田正廣** 峯尾真一***

PROGRAMMING TECHNIQUES FOR HIGH-SPEED PROCESSING AND VERIFICATION WITH THE NEC SUPERCOMPUTER SX SYSTEM

Kinuyo NAKAMURA, Masahiro YOSHIDA
and Shinichi MINEO

ABSTRACT

In this report various methods of effective FORTRAN programming to increase vectorization rates and conduct high-speed processing of FORTRAN programs for supercomputers are presented. The methods are used to tune up the NAL application software package, AFMESH, which is a two-dimensional grid generation program, and INVERSE, which is a two-dimensional airfoil design program. The tuned-up programs are executed with the NEC supercomputer SX-2 system and computing times compared with the times of the original programs show the usefulness of the methods.

1. まえがき

近年，科学技術計算の大規模化に伴って計算の高速化の要求が高まっている。これに応え，半導体技術を中心としたコンピュータハードウェア技術のめざましい進歩の基盤のもと，計算速度，記憶容量の両面において，これまでの大型汎用計算機を大幅に上まわるスーパコンピュータの開発がなされている。

大規模な科学技術計算の高速化の実現のためには，

(1) ハードウェアのスカラ演算，ベクトル演算の性能

(2) プログラム実行におけるベクトル演算の比率（ベクトル化率）

が重要である。スカラ演算およびベクトル演算の性能はハードウェアおよび計算機アーキテクチャにより決定される。一方，ベクトル化率は

(a) プログラムすべき問題自体の並列処理可能性およびアルゴリズムの選択等の設計レベル

(b) プログラムコーディング・レベル

(c) コンパイラ・レベル

の各段階での検討および改善により大きくすることが可能である。

スーパコンピュータを有効に使用するためには並列処理可能な範囲をできるだけ広くすることが基本であるが，上記の(a)，(b)，(c)の過程でその並列性は様々な要因により減少し，その後の過程で復元されることはない。このように，計算の高速化の実現のためには，問題の持っている並列性をできるだけ保持することが重要である。特に，計算機ユーザにと

*昭和61年4月1日 受付
**計算センタ
***日本電気株式会社

っては(a)および(b)，すなわちプログラム設計およびコーディングの段階での並列性の保持が重要である。

本報告では，スーパコンピュータSXシステムの使用による航空宇宙技術研究所（以下航技研という）汎用プログラムAFMESHおよびINVERSEの高速化に関する成果について述べる。この成果は，航空宇宙技術の研究開発に必要な数値シミュレーションを高速に実行するための並列計算法および並列処理プログラム技術の研究を目的とする，航技研と日本電気株式会社（以下日電という）との共同研究によるものである。

第2章ではスーパコンピュータSXシステムの概要を説明する。ハードウェア性能やプログラム高速化のためのツールの紹介を行う。

第3章ではプログラムを高速化するための方法について説明する。

第4章ではAFMESHおよびINVERSEプログラムの高速化を示す。

2. SXシステムの概要

2.1 ハードウェアの機能と性能[1) 2)]

スーパコンピュータSXシステムの機能および性能について以下に説明する。

2.1.1 ハードウェア構成

図1はSXシステムハードウェア構成図である。図2は演算プロセッサの部分を拡大して描いたSX－2システムハードウェア構成図である。

SXシステムの本体系装置は

(1) 科学演算処理装置
　　制御プロセッサ（CP）
　　演算プロセッサ（AP）
(2) 主記憶装置（MMU）
　　制御プロセッサメモリ（CPM）
　　演算プロセッサメモリ（AM）
(3) 拡張記憶装置（XMU）
(4) 入出力処理装置（IOP）

から成る。

CPはシステム全体の制御を行うプロセッサであり，オペレーティングシステム機能の大部分がCP上で実行される。コンパイルジョブステップやリンケージジョブステップはCPで実行される。

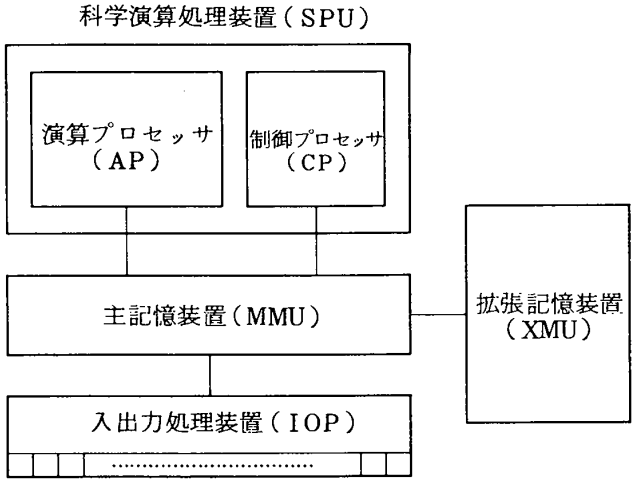


図1　SXシステムハードウェア構成図

APはベクトル演算を主体とするFORTRANのオブジェクト・プログラムを専用に実行するプロセッサである。すなわち，ゴージョブステップはAPで実行される。

MMUはCPMとAMから成り，CPMはCP上で実行される制御プログラムおよびそれらが使用するデータの格納に使われ，AMはAP上で実行されるプログラムおよびそれらが使用するデータの格納に使われる。

256KビットのMOSダイナミック・メモリ素子を使用したXMUは，磁気ディスク装置と主記憶装置の間のアクセスギャップ（アクセス時間の差）を埋めるための高速大容量記憶装置である。

IOPはCPから発行される入出力命令の解釈，実行，および周辺装置と主記憶装置間のデータ転送の制御を行う。

2.1.2 ハードウェア性能

表1はSX－2システムの性能諸元であり，表2は基本ベクトル性能表である。

SX－2システムは最大性能1.3GFLOPS*の高速スーパコンピュータである。以下にその特徴を示す。

(1) SXシステムではゲート遅延が250PSで集積度が1000ゲートの高速論理LSIや液冷を採用した新型LSI高密度パッケージをはじめとする

* GFLOPS＝giga floating operations per second

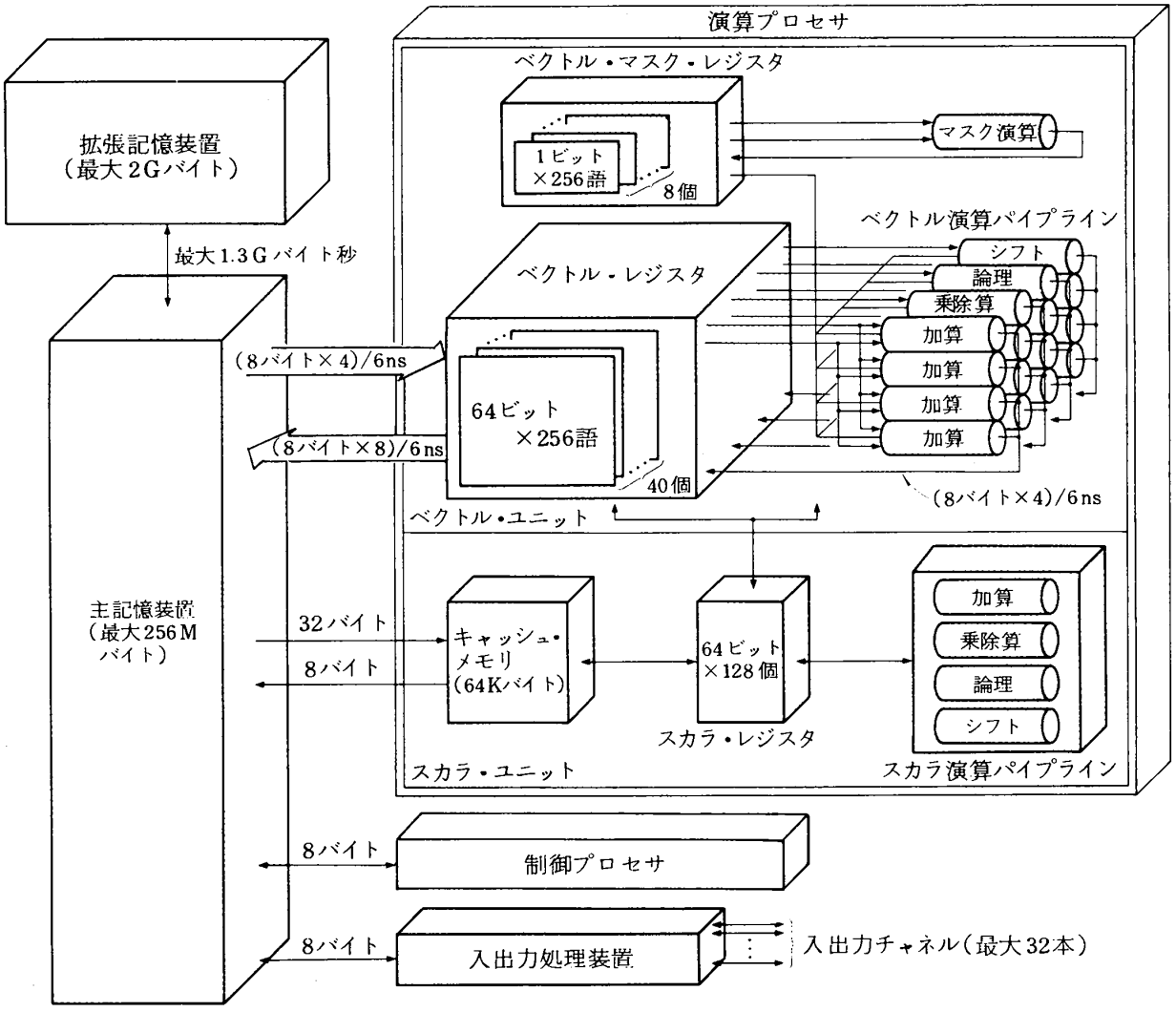


図2　SX−2システムハードウェア構成図

高密度実装技術により，ベクトル，スカラともに6nsの高速基本マシンサイクルが実現されている。

(2) ベクトル処理能力を高めるため，ベクトル用演算パイプラインを16本備え，同時並行して動作させる多重並列パイプライン方式となっている。また，MMUからパイプライン演算器へのベクトルデータの供給能力は11Gバイト／秒である。

(3) スカラ処理能力を高めるため，科学技術計算に適した命令セットを採用したRISCアーキテクチャ* が採用されている。さらに128個の汎用レジスタが用意され，演算パイプライン方式がとられている。

(4) 演算処理装置をCPとAPの二つの機能に分け，独立並行して実行させる機能分散構成にして処理能力の向上が図られている。

(5) キャッシュメモリやベクトルレジスタにはチップ当り1Kビットのバイポーラ RAMが使用され，アクセス時間3.5nsが実現されている。

(6) MMUはチップ当り64Kビット，アクセス時間40nsの大容量スタティックRAMが使用されている。

(7) XMUは256KビットのMOSダイナミック・メモリ素子を使用して，2Gバイトまで実装でき，磁気ディスクの転送速度の約400倍の転送

---

* RISCアーキテクチャ（reduced instruction set computer）
命令制御を簡単にし，使用頻度の高い基本的な命令だけを用意することにより，ハードウェアの簡単化とマシン・サイクルの高速化を図ったコンピュータである。

表1 SXシステムの諸元

項目			SX-2
最大性能(浮動小数点演算)			1.3 GFLOPS
マシン・サイクル			6 ns
命令数		スカラ	81
		ベクトル	86
科学演算処理装置(SPU)	レジスタ	ベクトル・レジスタ	80Kバイト
		ベクトル・マスク	256ビット×8個
		スカラ	64ビット×128個
	データ形式	固定小数点	32ビット
		浮動小数点	32/64/128ビット
		論理	64ビット
	ベクトル演算パイプライン		16本
	スカラ演算パイプライン		1セット
	キャッシュ・メモリ		64Kバイト
主記憶装置(MMU)		容量	128/256Mバイト
		インタレース	512ウエイ
		記憶素子	64KビットMOSスタチックRAM
入出力処理装置(IOP)		総合データ転送能力	50Mバイト/秒
		入出力チャネル本数	最大32本
拡張記憶装置(XMU)		容量	128Mバイト～2Gバイト
		転送速度	最大1.3Gバイト/秒
		記憶素子	256KビットMOSダイナミックRAM

表2 基本ベクトル性能

項目	SX-2
最大演算性能 (加算＋乗算＋論理＋シフト)	2.6 GOPS[*1]
最大浮動小数点演算性能 (加算＋乗算)	1.3 GFLOPS[*2]
加算性能	667 MFLOPS
乗算性能	667 MFLOPS
除算性能	83MFLOPS

[*1] GOPS＝giga operations per second
[*2] GFLOPS＝giga floating operations per second
[*3] MFLOPS＝mega floating operations per second

速度1.3Gバイト／秒をもつ。

2.2 言語の概要[3)4)]

SXシステムの主プログラミング言語はFORTRAN言語であり，ユーザはFORTRANプログラムを作成することによりスーパコンピュータSXシステムの超高速性能を容易に引き出すことが可能である。

図3はFORTRAN 77／SXを中心とするソフトウェア体系図である。図からわかるように

(1) FORTRAN処理系

(2) FORTRAN支援系

(3) ベクトル処理用科学技術計算ライブラリ

の三つの機能に分けられる。

FORTRAN処理系のFORTRAN 77，77／SXはJIS FORTRAN*上位水準に準拠し，次に述べる多くの拡張機能をもつ。

(a) 型の拡張

(b) プログラムの自由形式記述

(c) 組込み関数の追加

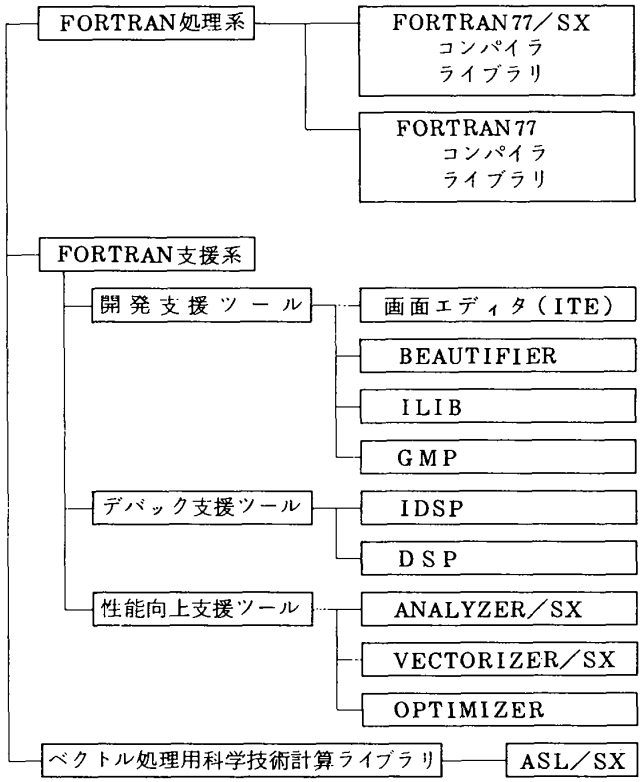


図3　ソフトウェア体系図

(d) 組込みサブルーチンの追加

(e) 入出力文の追加

(f) INCLUDE文の追加

(g) 日本語処理機能の追加

(h) 他FORTRANとの互換機能

また，汎用機のFORTRAN 77との互換性もある。FORTRAN77／SXは，FORTRAN77に加えて，

1) 自動ベクトル化機能

2) ベクトル化指示行

3) 最適化機能

4) 高速入出力機能および拡張記憶装置に対する入出力機能

の四つの機能を有するFORTRANコンパイラである。

FORTRAN支援系は，プログラム開発過程で必要なツール類である。以下に各ツールの簡単な説明を行う。

(1) 開発支援

[画面エディタ]

TSSによるプログラム開発保守作業のための開発支援サービスプログラム

[BEAUTIFIER]

プログラム清書ユティリティ

[ILIB]

ライブラリ管理のためのサービスプログラム

[GMP]

汎用マクロプロセッサ

(2) デバッグ支援

[IDSP]

TSSによる会話型デバッギングのためのサービスプログラム

[DSP]

BATCH処理形態によるデバッギングのためのサービスプログラム

(3) 性能向上支援

[ANALYZER／SX]

プログラム動的特性解析および静的構造解析のためのサービスプログラム

---

* JIS FORTRAN
一般にFORTRAN 77と呼ばれ，国際規格「ISO 1539−1980 programming language FORTRAN」並びに米国規格「ANSI X 3.9−1978 programming language FORTRAN」に等しいものである。

［VECTORIZER／SX］

SXシステムベクトル化促進用チューニングサービスプログラム

［OPTIMIZER］

FORTRAN原始プログラムに最適化処理を施すサービスプログラム

FORTRAN77／SX言語および性能向上支援ツールに関してはプログラムの高速化と密接な関係があるので，項を設けてその機能について説明を行う。

2.2.1 FORTRAN77／SXの機能

FORTRAN77／SXはFORTRAN77言語で書かれた利用者プログラムの実行において，スーパコンピュータSXシステムの超高速性を十分に引き出すための言語処理系であり，次の機能をもつものである。

(1) ベクトル化機能

行列の行要素，列要素，あるいは対角要素など規則的に並んだデータ列をベクトルデータといい，ベクトルデータに対する演算を一度に実行する命令をベクトル命令という。これに対して単一データ（スカラデータ）に対して演算を実行する命令をスカラ命令という。コンパイラが原始プログラムを解析してオブジェクトプログラムを作成する過程において，ベクトル命令で実行可能な部分を自動的に検出し，その部分に対してベクトル命令を生成することを自動ベクトル化という。表3－1にFORTRAN77／SXの自動ベクトル化の対象を示し，表3－2にFORTRAN77／SXの拡張ベクトル化機能を示す。

自動ベクトル化について理解を深めるため，簡単な例を示し説明する。

例1 単純なループ

```
      DO   10   I=1, N
      T(I)=X * C(I)
      A(I)=B(I)+T(I)
10    CONTINUE
```

⇩ ベクトル化

$T_i \leftarrow X * C_i \quad (i=1, 2, \cdots, n)$

$A_i \leftarrow B_i + T_i \quad (i=1, 2, \cdots, n)$

例1のDOループはベクトル化されることにより上のような演算命令列におきかえられる。その結果，実行順序がベクトル化されない場合と次のように異なるので注意が必要である。

表3-1 FORTRAN77／SXの自動ベクトル化の対象

対象となるループ	DOループ
対象となるデータの型	整数型，論理型 実数型(単精度，倍精度) 複素数型(単精度，倍精度)
対象となる文	代入文，CONTINUE文 IF文(算術，論理，ブロック) ELSE IF文，ELSE文，END IF文，単純GO TO文
対象となる演算	加減乗除算，べき算 型変換 組み込み関係 内積，総和，累積 漸化式 最大，最小 サーチ
対象となるベクトル	連続，等間隔ベクトル 間接指標ベクトル 指標変数

表3-2 FORTRAN77／SXの拡張ベクトル化機能

ベクトル化不能部分を含むループ	ループ分割によりベクトル化
外部関数の引用を含むループ	ベクトル↔スカラ中継ルーチンの挿入により疑似ベクトル関数化してベクトル化
データの参照関係が適合しないループ	文の入れ替えまたは作業用ベクトルの導入によりベクトル化
ベクトル化の可否が判断不能のループ	ベクトル化指示行の指示によりベクトル化

・ベクトル化されない場合

```
T(1)=X * C(1)
A(1)=B(1)+T(1)
  ⋮
T(N)=X * C(N)
A(N)=B(N)+T(N)
```

ベクトル化される場合

```
T(1)＝X ＊ C(1)
　　⋮
T(N)＝X ＊ C(N)
A(1)＝B(1)＋T(1)
　　⋮
A(N)＝B(N)＋T(N)
```

例2　マスク付ベクトル演算命令

```
DO　10　I=1，N
IF(A(I).GE. 0.0) THEN
  X(I)＝Y(I)＋A(I)＊Z(I)
ENDIF
10　CONTINUE
```

⇩ ベクトル化

$$M_i \quad \begin{cases} 1 \ (\text{if} \ A_i \geq 0.) \\ 0 \ (\text{if} \ A_i < 0.) \end{cases}$$

$T_i \leftarrow A_i * Z_i$ （if $M_i = 1$）

$X_i \leftarrow Y_i + T_i$ （if $M_i = 1$）

M_i：ベクトル演算のマスク制御に使用するマスクベクトル

(2)　ベクトル化指示行

ベクトル化指示行は，原始プログラムの解析によって得ることのできない情報をユーザがコンパイラに与えることにより，コンパイラによるベクトル化の効果を促進させるためのものである。逆にベクトル化を抑止するためにスカラ指定を行うことも可能である。

例

```
＊VDIR　NODEP
　　DO　10　I=1，N
　　A(I)＝3.0
　　B(I)＝A(I＋L)
10　CONTINUE
```

例のDO 10ループにおいて，L≦0ならばA(I)とA(I＋L)の定義・引用関係を正しく保つことができ，L＞0ならばA(I)とA(I＋L)の定義・引用関係を正しく保つことができない。定義・引用関係を正しく保つことが可能な場合には，＊行のベクトル化指示行をDOループの直前に挿入することによりDOループをベクトル化することができる。

(3)　最適化機能

FORTRAN77／SXは以下のような最適化を行い，プログラム実行時間の短縮化が図られている。

(a)　乗除算の特殊演算化

例

```
DO　10　I=1，N
X(I)＝A(I)＊0.5＋B(I)＊C(I)
10　CONTINUE
```

⇩ ベクトル化

① $T1_i \leftarrow A_i * 0.5$

② $T2_i \leftarrow B_i * C_i$

$X_i \leftarrow T1_i + T2_i$

⇩ 最適化

③ $T1_i \leftarrow \mathrm{Half}(A_i)$

④ $T2_i \leftarrow B_i * C_i$

$X_i \leftarrow T1_i + T2_i$

①，②は同じ乗算パイプラインを使用するので並列実行できないが，①をHALF命令を使用して③に置きかえることにより，③，④は並列実行可能となり処理時間の短縮を図ることができる。

(b)　共通式の削除

ある計算が同一プログラム中で既に行われている場合，不要な再計算を除去し，以前の計算結果の引用でおきかえることをいう。

(c)　スカラ演算のくくりだし

スカラデータ同志で演算可能なものはスカラ演算でおきかえ，その後ベクトル演算を行う。これによりベクトル演算の演算量が減少し，実行時間を短縮することができる。

(d)　除算の乗算化

ベクトル乗算命令はベクトル除算命令に比べ高速なので，除算を乗算化することにより演算時間を短縮することができる。

(e)　マスク演算の最適化

同一のマスク演算は共通式として処理することにより実行効率低下をさけている。

(f)　不要終値保証の除去

例

```
DO　10　I=1，N
X＝A(I)
10　B(I)＝C(I) ＊ X
```

DOループ終了後，X＝A(N)の値を保証しなくてもよい場合に，終値を保証する命令を削除して実行時間の短縮がはかられている。

(g) レジスタ割り当ての最適化

ベクトル演算の中間結果を可能な限りベクトルレジスタ上に保持するようにし，主記憶とベクトルレジスタ間の移送回数を減少させる。ベクトルマスクレジスタに関しても最適な割り当てを行い，主記憶とのロードおよびストアの回数を必要最少限にしている。

(h) 命令の並べかえによる最適化

1) 演算の並列実行可能性

2) 演算のチェイニング可能性

3) ベクトルレジスタの有効利用

を考慮して命令の順序を適切に並べかえ，パイプラインが並列に動作するようにする。

(4) 高速入出力機能および拡張記憶装置に対する入出力機能

入出力高速化のために以下の機能を設けている。

(a) 非同期入出力機能

一つのプログラム中の入出力処理と演算処理を並列に行わせる機能である。プログラム実行中に非同期入出力文があらわれると入出力処理のための別のプロセス(タスク)をCP上で起動させ，AP上のプログラムはCPのそのタスクを起動しおえるとすぐに次の文の実行に移る。これにより，CPとAPは並列動作可能となる。図4に非同期入出力処理の概要を示す。

(b) 並行入出力

データをブロック単位に複数の磁気ディスク装置に分散しておき，一つの入出力処理に対してそれらの装置と主記憶間で並行にデータ転送を行うものである。図5に並行入出力の概要を示す。

(c) 並列編集出力

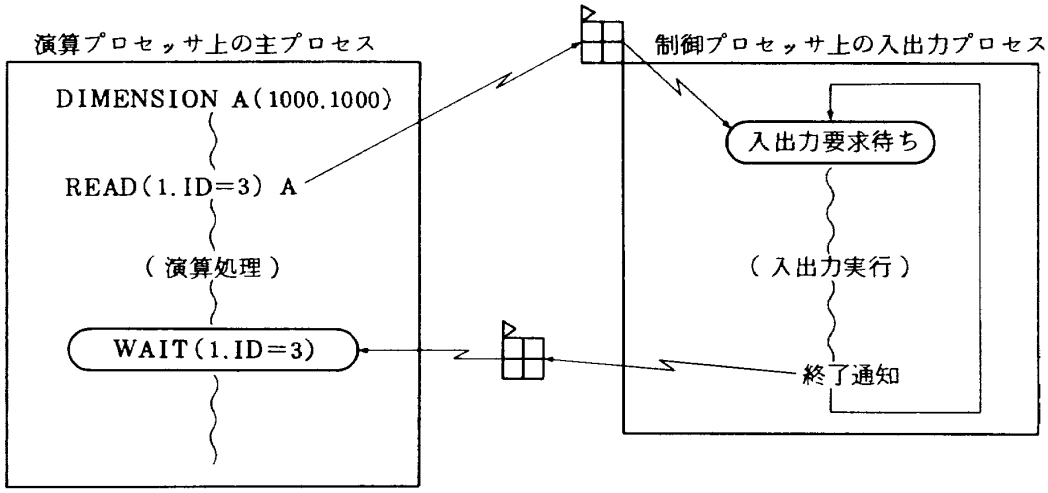


図4 非同期入出力処理

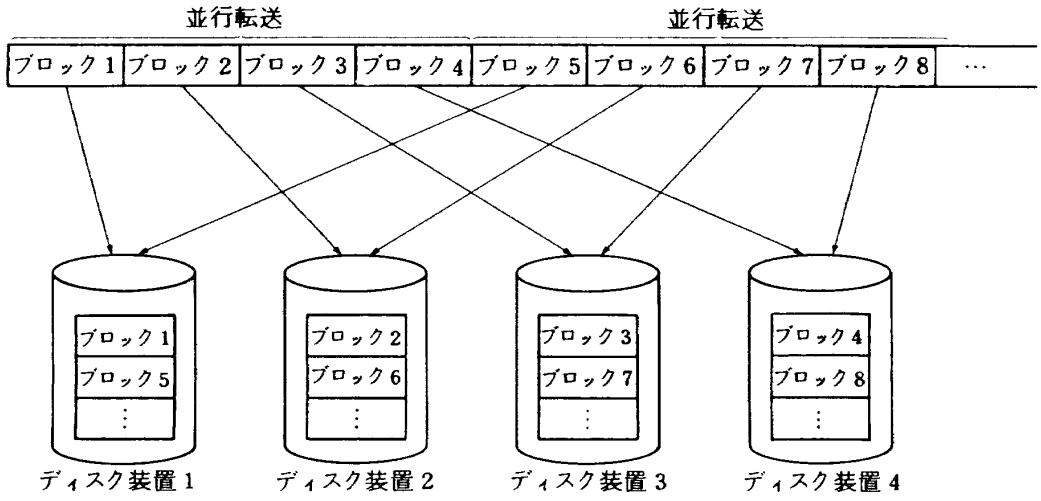


図5 並行入出力

書式付きWRITE 文中の出力並びの要素とFORMAT情報を一括して渡すことにより，以後の実行時入出力ルーチンによる編集出力処理と演算処理とを独立させ，CPとAPとで並列に実行させている。図6に並列編集出力の概要を示す。

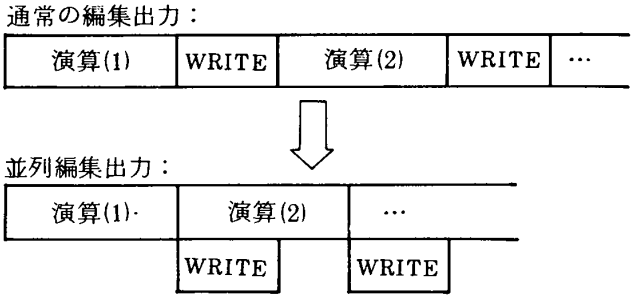


図6　並列編集出力

(d)　拡張記憶装置に対する入出力

拡張記憶装置は大量データの入出力を高速に処理するためにあり，その使用法はジョブ制御文においてファイル割り当てを拡張記憶装置に指定するだけで可能である。

ファイルの属性は一時ファイルと永久ファイルの二種類あり，一時ファイル利用の場合にはFORTRANの実行時入出力ルーチンにより直接，主記憶と拡張記憶装置の間でデータ転送するので，データ転送にかかわるオペレーティングシステムのオーバヘッドの削減およびバッファ長の制限撤廃による大量データの一括転送が可能になるため，ハードウェア性能が最大限に引き出されている。

図7に拡張記憶装置の入出力方式の概要を示す。

2.2.2　ANALYZER／SXの機能

ANALYZER／SXはプログラムの動的特性情報および静的構造情報を解析するサービスプログラムである。プログラムを高速化するために必要なチューニングを行うための検討資料を与えるものであり，次の機能を有する。

(1)　動的特性情報

(a)　プログラム全体の輪郭情報

1)　各プログラム単位毎にその引用回数，実行コスト比*，ベクトル化対象DOループの個数とその中でベクトル化されたループの個数，およびそれらの実行コスト比，そのプログラム単位内でのベクトル化率

2)　プログラム全体でのベクトル化率

(b)　各プログラム単位の詳細な輪郭情報

1)　各実行文の実行回数と実行コスト比

2)　条件付き分岐の分岐先毎の選択回数とその比率

3)　ベクトル化されたDOループの表示

(c)　DOループに関するベクトル化情報

1)　各DOループ（ベクトル化されたもの，ベクトル化されないもの別に実行コストの比率

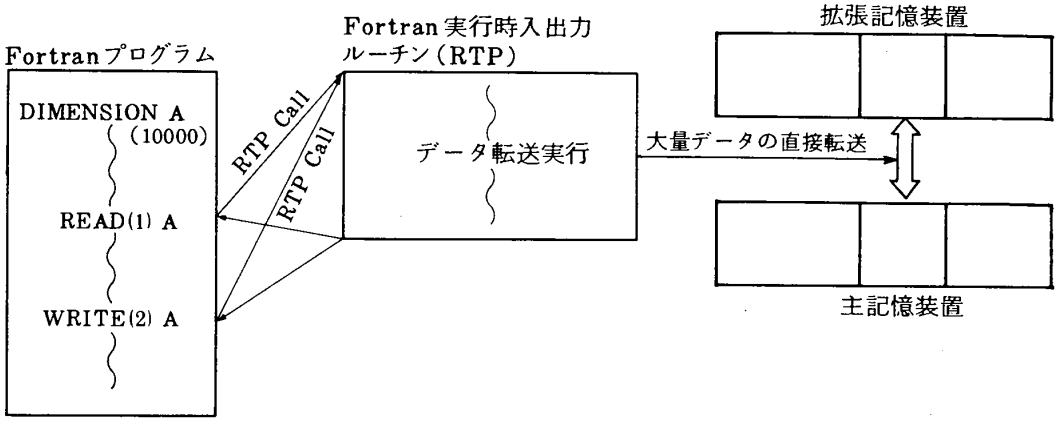


図7　拡張記憶装置の入出力方式（一時ファイル）

＊　実行時間の全体比を近似的に表わしたものである。

の高い順にソートされる）の実行コスト比，
ベクトル化されないものについては不可理由，
平均ベクトル長，原始プログラム上での位置

(2) 静的構造情報

(a) プログラム全体に対する情報

1) プログラム構成図

2) プログラム単位間の呼び出しに関する相互参照リスト

3) 仮引数と実引数の対応表

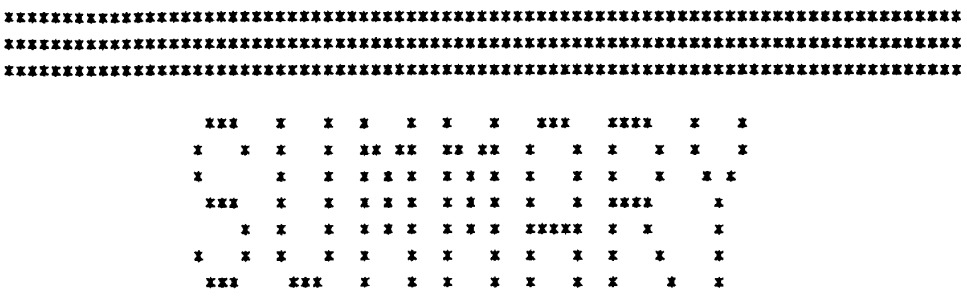
4) 共通ブロックの要素に関する相互参照リスト

(b) 各プログラム単位毎の情報

1) 英字名と行番号に関する相互参照リスト

図8にANALYZER／SXの出力例を示す。

・ プログラム全体の輪郭情報



TOTAL EXECUTION (CPU) TIME = 0: 1' 2" 150 (62150 MSEC)

TOTAL EXECUTION FREQUENCY = 3876592

TOTAL VECTORIZATION RATIO = 84.72%

ATR	PROGRAM	FREQUENCY	EXEC COST(%)	V.RATIO	LOOP	V.LOOP	V.LOOP RATIO
-->	MAIN	1	0.07	61.25	15	9	80.36
SUB	SUBGTE	1	0.34	68.94	20	12	84.25
SUB	KITLUX	1	1.21	89.51	5	4	92.62
FNC	MNRDS5	1	0.66	83.87	6	5	92.55
SUB	NJYI09	109	5.67	99.99	6	6	100.00
SUB	KJTD06	981	76.62	82.69	5	5	100.00
FNC	KRTJUR	109	2.15	23.68	4	3	73.24
SUB	KJTD07	327	13.27	99.94	1	1	100.00
SUB	KRTJUX	1	0.01	99.99	1	1	100.00

図8 ANALYZER／SXの出力例

・ プログラム単位の輪郭情報

```
   ELN   ILN  EXECUTION  COST(%) LOOP        FORTRAN STATEMENT

000100     1          1                            SUBROUTINE SUB(A,KO,N,EPS,ILL)
000110     2                                       DIMENSION  A(KO,KO)
000120     3                                       INTEGER    X(150)
000130     4                                       LOGICAL    B
000140     5          1 (  0.00)                   ILL=0
000150     6          1 (  0.00)                   IF((KO.GE.N).AND.(N.GE.2).AND.(N.LE.1500))GOTO1
000160     7          0 (  0.00)                   ILL=30000
000170     8          0 (  0.00)                   RETURN
000180     9          1 (  0.00)                 1 DO 10 I = 1 , N
000190    10        150 (  0.00) V---------->       | X(I) = I
000200    11        150 (  0.00) V----------      10 CONTINUE
000210    12          1 (  0.00)                   DO 110 K = 1 , N
000220    13        150 (  0.00) 1---------->       | M = K
000230    14        150 (  0.00) |                  | W = 0.0
000240    15        150 (  0.00) |                  | DO 20 I = K , N
000250    16      11325 (  0.11) | V--------->     |   IF(ABS(A(I,K)).LE.W) GO TO 20
000260    17        447 (  0.00) | |               |   W = ABS(A(I,K))
000270    18        447 (  0.00) | |               |   M = I
000280    19      11325 (  0.11) | V----------    20 | CONTINUE
000290    20        150 (  0.00) |                  | IF(M.EQ.K) GO TO 50
000300    21        148 (  0.00) |                  | L = X(M)
000310    22        148 (  0.00) |                  | X(M) = X(K)
000320    23        148 (  0.00) |                  | X(K) = L
000330    24        148 (  0.00) |                  | DO 40 J = 1 , N
000340    25      22200 (  0.36) | V--------->     |   W = A(K,J)
000350    26      22200 (  0.52) | |               |   A(K,J) = A(M,J)
000360    27      22200 (  0.26) | |               |   A(M,J) = W
000370    28      22200 (  0.42) | V----------    40 | CONTINUE
000380    29        150 (  0.00) 1----------      50 | IF(ABS(A(K,K)).GE.EPS) GO TO 60
000390    30          0 (  0.00)                   | ILL = K
000400    31          0 (  0.00)                   | RETURN
000410    32        150 (  0.00) 1----------      60 | P = 1.0/A(K,K)
000420    33        150 (  0.00) |                  | DO 70 I = 1 , N
000430    34      22500 (  0.53) | V--------->     |   A(K,J) = A(K,J) * P
000440    35      22500 (  0.05) | V----------    70 | CONTINUE
000450    36        150 (  0.00) |                  | DO 100 I = 1 , N
000460    37      22500 (  0.11) | 2--------->     |   T = -A(I,K)
000470    38      22500 (  0.53) | |               |   B = (I.NE.K).AND.(T.NE.0.0)
000480    39      22500 (  0.11) | |               |   IF(.NOT.B) GO TO 100
000490    40      22330 (  0.10) | |               |   DO 90 J = 1 , N
000500    41    3349500 ( 94.15) | | V------->     |   | A(I,J) = A(I,J) + A(K,J) * T
000510    42    3349500 (  0.00) | | V--------    90 |   CONTINUE
000520    43      22500 (  0.37) | 2----------  100 |   A(I,K) = P * T
000530    44        150 (  0.00) |                  | A(K,K) = P
000540    45        150 (  0.01) 1----------     110 CONTINUE
```

図8（続き）

・ DOループに関するベクトル化情報

```
DO PROFILE LIST

PROGRAM : PROGRAM1

ST NO.    ELN  ILN  COST(%)     AV.LOOP LENGTH     VECTORIZATION INFORMATION

UNVECTORIZED DO LOOP

   2  000300   30    1.05              18.0   24 UNSUITABLE TYPE OF DO VARIABLE OR
                                                    DO PARAMETER

PARTIALLY VECTORIZED DO LOOP

   3  000410   41    4.55              10.0   ITERATION COUNT      :  UNKNOWN
                                              TEMPORARY VECTOR     :  20000 BYTES
                                              INDEX VARIABLES      :  I
                                              RELATIVE CONSTANTS   :  P1       K
                                              VECTOR ITERATION     :  45
                                              VECTOR PART          :  41-48

                                              49          30       :  UNSUITABLE DATA
                                                                      REFERENCE RELATION

VECTORIZED DO LOOP

   7  000730   73   81.02             180.0   INTERCHANGED BY      :  72
                                              ITERATION COUNT      :  UNKNOWN
                                              TEMPORARY VECTOR     :  20000 BYTES
                                              INDEX VARIABLES      :  J
                                              RELATIVE CONSTANTS   :  P1       K
   1  000050    5    5.35              18.0   ITERATION COUNT      :  UNKNOWN
                                              TEMPORARY VECTOR     :  20000 BYTES
                                              INDEX VARIABLES      :  I
                                              RELATIVE CONSTANTS   :  P1       K
   5  000550   55    0.02              18.0   SINGULARIZED LOOP    :  54 55
                                              ITERATION COUNT      :  UNKNOWN
                                              TEMPORARY VECTOR     :  20000 BYTES
                                              INDEX VARIABLES      :  I
                                              RELATIVE CONSTANTS   :  P1       K
```

図8（続き）

・ プログラム構成図

```
PROGRAM STRUCTURE LIST      ( :INT.FUNC:  'INT.SUBR' (EXT.PROC) /MAIN ENTRY/ )

****:MAINPRG1
     !
     !__ASTFUNCK
     !  !
     !  !__:COS:
     !
     !__'SETSW'
     !
     !__SUB10
     !  !
     !  !__FUNC01
     !  !
     !  !__SUB101
     !     !
     !     !__SUB102:<--D001
     !        !
     !        !__:FLOAT:
     !           !
     !           !__FUNC02:<--D002
     !              !
     !              !__:ABS:
     !
     !__SUB20
        !
        !__ENT10/SUB10/
        !
        !__SUB102:-->D001
        !
        !__SUB103
           !
           !__SUB104
           !  !
           !  !__FUNC02:-->D002
           !  !
           !  !__SUBROU(DUMMY)
           !
           !__(SUB105)
           !
           !__(SUB106)
```

・ 仮引数と実引数の対応を示すリスト

ARGUMENT LIST

PROGRAM	REF.LN	ARGUMENT						
*** SUBROUT1 ***		IX	IY	IZ	WEIGHT	VOL		
FTMAIN	000690	KX	KY	KZ	0.154	RVOL		
	000960	KX	KY	*1	*2	RVOL		
		1 : RJS*STU						
		2 : WGT*0.3						
SUB2	000540	HX	HY	HZ	1.25	VOL		
SUB3	000360	JX	JY	JZ	3.56	JVOL		
*** SUBROUT2 ***		PARAM01	PARAM02	PARAM03	PARAM04	PARAM05	PARAM06	PARAM07
FTMAIN	000430	3.14159	DELT	IX	*1	1.984	IY	85
		1 : ARY(DX(IX)) * YSHADE						
	000700	RPAI	DLT	2.5334	D	RJET	IYYH	150
GETSYL	001430	NASA	KAI	HFT	ARY(100)	1.954	HY	65
*** SUBROUT3 ***		NOARGUMENT						
POST		NOARGUMENT						

図8（続き）

3. プログラム高速化の概要

FORTRAN 77 言語で書かれたプログラムはFORTRAN 77／SXコンパイラを通すことにより自動的にベクトル化される。しかし，チューニングを施すことによりさらに実効性能の改善を図ることができる。そのためには，以下に述べることが必要である。

3.1 プログラム分析

一般的に，科学技術計算プログラムは多くのサブルーチンから成っているが，そのプログラムを処理するための中央処理装置使用時間（CPUT）の90%以上は数個のサブルーチンに費やされていることが多い。また，一つのサブルーチンについてもその中の一部の実行文にCPUTの負荷が集中している場合が多い。プログラム全体のサブルーチンやDOループをベクトル化することは不可能であるし，CPUT負荷の集中している場所に対するチューニングが最も効果的であるので，高速化対象プログラムを分析して負荷の集中している場所を把握する必要がある。これはANALYZER／SXを利用して行うことができる。

また，プログラムをコンパイルすることによりFORTRAN77／SXからベクトル化に関する情報が得られる。コンパイル時のソースプログラムリストに

V 矢印で示すDOループがベクトル化されたことの表示

S 矢印で示すDOループの中でベクトル化されないFORTRAN文の表示

M 矢印で示すDOループの中で一部分がベクトル化されないFORTRAN文の表示

n 矢印で示すDOループがベクトル化されないことの表示（n：整数）

によりベクトル化情報が示される。図9にDOループのベクトル化表示を示す。図10にベクトル化に関するメッセージを示す。このメッセージにより完全にベクトル化されない理由を知ることができるので，これらを解消することがベクトル化率を高めることになる。

上記のプログラム分析やベクトル化情報から，高速化対象プログラムの実効性能改善のための資料を得て，その実現のためのプログラム改善方法の決定が必要となる。

3.2 スーパコンピュータの効率的利用の促進

ベクトル化率を高めること，すなわち，プログラムのベクトル命令で処理される部分の比率を大きくすることは，ベクトル命令がスカラ命令に比較して，格段に高速なことによる。

図11はベクトル化率50%のプログラムとベクトル化率90%のプログラムのベクトル化前およびベクトル化後の実行時間の比較を視覚的に示したもので

```
                    DO 10300 IDXJ = JS , JE
V------------>      |  XXSETA(IDXJ) = ( XETS( IDXJ+1 ) - XETS( IDXJ-1 ) ) * 0.5
|                   |  YXSETA(IDXJ) = ( YETS( IDXJ+1 ) - YETS( IDXJ-1 ) ) * 0.5
V------------ 10300 CONTINUE
```

(1) ベクトル化されたDOループ

```
                3130 DO 3140 J = 3 , JWS
V------------>S     |  XX           = XX - DXSW0
|             S     |  X(J , KEND+1) = XX
|                   |  Y(J , KEND+1) = YMIN
V------------ 3140 CONTINUE
```

(2) 部分ベクトル化されたDOループ

```
                    DO  3200 J = 1 , JJJ-1
4------------>      |  GO TO (3192 , 3193) IBFRNW
|           3192    |  DX01  = DXSW0 * ( 1 + EPS / SQRT( FLOAT(J) ) ) ** (J - 1)
|                   |  GO TO 3195
|           3193    |  DX01  = DXSW0 * ( 1 + EPS )** (J - 1)
|           3195    |  XX    = XX - DX01
|                   |  JJ    = J
|                   |  IF ( XX .LT. BFRNT )                          GO TO  3300
|                   |  X(JWS+J , KEND+1) = XX
|                   |  Y(JWS+J , KEND+1) = YMIN
4------------ 3200 CONTINUE
```

(3) ベクトル化されないDOループ

図9 ベクトル化情報

```
3      97          10300   DOループ全体をベクトル化する．DO変数は IDXJ
                           *  繰返し数                    :  不明
                           *  作業用ベクトル領域          :  0 バイト
                           *  指標変数                    :  IDXJ
```

(1) ベクトル化されたDOループに関するメッセージ

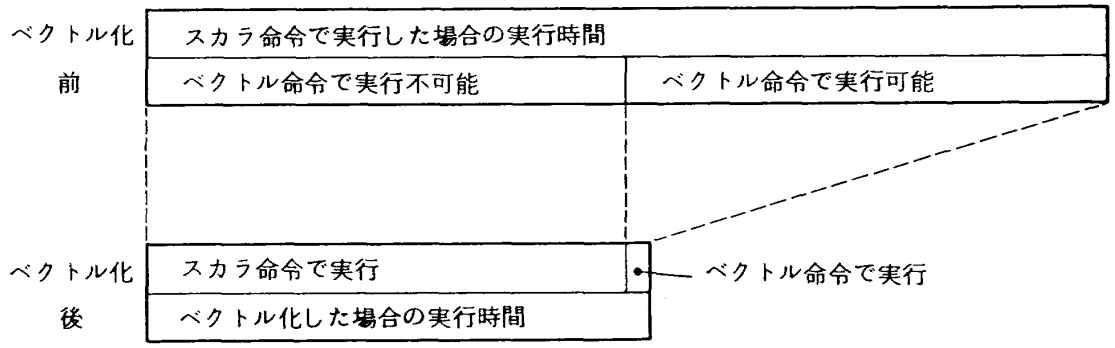
```
2      130         3140   DOループを部分的にベクトル化する．DO変数は J
                           *  ベクトル化範囲              :  133--134
                           *  繰返し数                    :  不明
                           *  作業用ベクトル領域          :  0 バイト
                           *  指標変数                    :  J
                           *  相対定数                    :  YMIN      KEND      DXSWO
                    131      123  :  変数が定義前に引用されている  :  XX  132
```

(2) 部分ベクトル化されたDOループに関するメッセージ

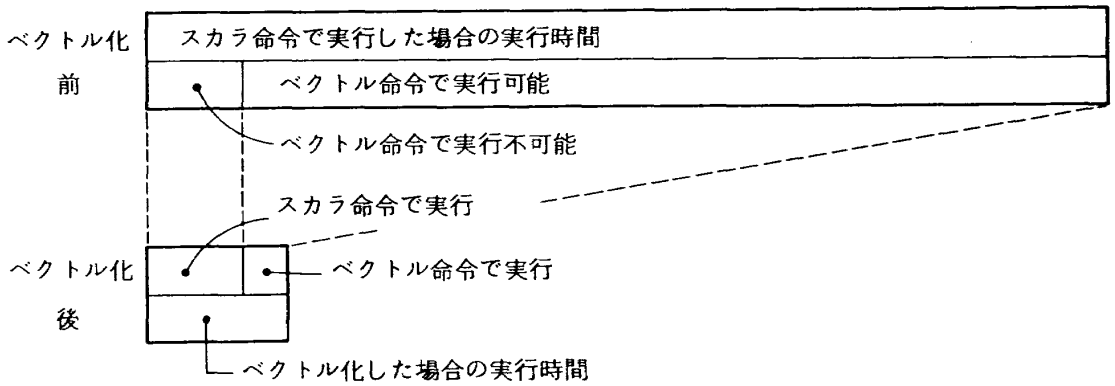
```
3      148         3200   DOループはベクトル化できない
                    148      305  :  ループの途中からの脱出が含まれている
                    149      112  :  ベクトル化の対象外の文である
                    150      123  :  変数が定義前に引用されている  :  DX01  153
                    152      123  :  変数が定義前に引用されている  :  DX01  153
                    153      123  :  変数が定義前に引用されている  :  XX  156
                    153      123  :  変数が定義前に引用されている  :  XX  155
```

(3) ベクトル化されないDOループに関するメッセージ

図10 ベクトル化情報メッセージリスト



(a) ベクトル化率50％



(b) ベクトル化率90％

図11 ベクトル化率による性能差

ある。ベクトル化率の大きいプログラムの方がベクトル化後の実行時間の短縮が図られている様子がよく理解できる。

また，図11から，プログラムのベクトル命令で実行可能な部分が効率的なベクトル命令で処理されるならば実行時間の短縮が図られ，そうでない場合には高速化がさほど期待できないことが理解できる。

したがって，スーパコンピュータの効果を発揮させるためには，ベクトル化率を高めることおよびベクトル命令の効率を高めることが重要である。以下に，それを実現する方法を述べる。

3.2.1 ベクトル化の促進

ベクトル化率を高めるためには以下の三つの方法がある。

(1) ベクトル化不可原因の除去によるベクトル化率の改善

ベクトル化処理対象であるにもかかわらずベクトル化されないDOループあるいは一部分しかベクトル化されないDOループのベクトル化不可原因を取り除き，ベクトル化されない部分がベクトル化されるようにする。

たとえばDOループ中のデータの定義・引用関係が正しく保たれるか否かをコンパイラが判断できない場合でも正しく保たれるならばその旨のベクトル化指示行をDOループの直前に挿入することによりベクトル化することが可能である。

```
例      DO  10  J=1, N
        X(L-1)=X(L-1)-X(LW)*Y(J)
   10   LW=LW+1
```

例では，LWの初期値が不明なため，コンパイラは左辺のX(L−1)と右辺のX(LW)との間の定義・引用関係が，ベクトル化することにより正しく保たれるか否か判断できない。もし正しく保たれるならば

```
*VDIR NODEP(X)
```

のベクトル化指示行をDOループの直前に挿入することによりそのDOループをベクトル化可能にできる。

(2) ベクトル化処理対象の拡大によるベクトル化率の改善

ベクトル化の処理対象ではないが，ベクトル化可能となる部分を探し出し，ベクトル化されるように書き直す。

たとえばベクトル化の対象となるのはDOループであるので，IF文とGOTO文により構成されるループをDOループに書き直すというコーディングスタイルの変更によりベクトル化を促進できる。

(3) ベクトル化に適した計算法またはアルゴリズムの選択によるベクトル化率の改善

プログラムの全体あるいは一部分を，ベクトル化に適した計算法またはアルゴリズムを用いて書き直す。

プログラム設計において計算すべき問題の持っている並列処理可能性を考慮に入れずに，プログラムコーディングが行われると，後からその並列処理可能な構造を復元できない。スーパコンピュータを効率的に使うためにプログラム設計の段階で十分な検討を行うことが必要である。

3.2.2 ベクトル命令の効率化の促進

(1) ベクトル長の拡大

ベクトル化されたDOループを実行する際には，そのための前準備の処理が必要である。また，個々のベクトル命令についても，演算処理に入る前にある程度の準備が必要である。これらの処理のための時間を立ち上がり時間という。立ち上がり時間はベクトル長によらずほぼ一定であるため，ベクトル長が短く，実際のベクトル演算に要する時間が小さい場合には，立ち上がり時間の影響が相対的に大きくなり，高速化は期待できない。

図12に立ち上がり時間と交叉長の関係を示す。交叉長とはベクトル化した場合とベクトル化しない場

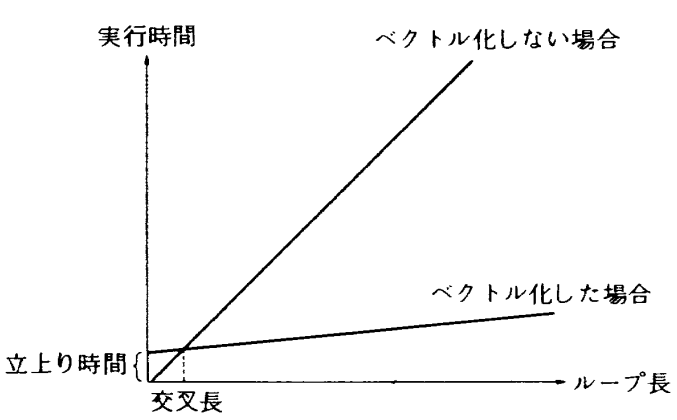


図12 立ち上がり時間と交叉長

合とで実行時間が等しくなるベクトル長をいう。SXシステムの場合，条件にもよるが交叉長は5～10である。

したがって，ベクトル化の効果を十分に引き出すためにはベクトル長を長くすべきであるが，ある限界以上長くしても得るところは少ないので，効率的にベクトル長を拡大すべきである。

たとえば，ベクトル長の拡大を図るためにn次元の配列を低次元化することを考える。低次元化とはn次元の配列を$n-1$次元以下の配列にすることをいう。理解を容易にするため2次元の配列X(I,J)を1次元の配列A(K)に低次化することを考える。

```
例      DIMENSION  X(100, 100),
        XX(100, 100)
        DO  10  J=1, N
        DO  10  I=1, M
        X(I, J)=XX(I, J)
   10   CONTINUE
```

例においてMおよびNが100に等しく，それがプログラム上に記述されていれば，拡張ベクトル化機能により外側のDOループもベクトル化され，DOループが一重化され，ベクトル長を拡大できる。しかし，一般的にプログラム上でDOループの繰り返し数は変数で与えられていることが多いので，外側のDOループのベクトル化は期待できない。そのため，次のように配列の次元を低次元化してベクトル長を拡大することが考えられる。

```
        ⋮
        DO  100  J=1, N          ┐
        K=(J-1) * M              │
        DO  100  I=1, M          │
        K=K+I                    │ (a)
        A(K)=X(I, J)             │
        AA(K)=XX(I, J)           │
  100   CONTINUE                 ┘
        ⋮
        ⋮
        DO  10  K=1, M * N       ┐
        A(K)=AA(K)               │ (b)
   10   CONTINUE                 ┘
        ⋮
```

(a)は配列を1次元化するために予め一度行わねばならない前処理であり，(b)は例において二重ループとなっていたのを一重ループにしたものである。

このように，二重ループをすべてベクトル化するために配列を低次元化してベクトル長を拡大できるが，それを行うために前処理となる計算を行わねばならないので，(a)と(b)の演算時間の比により効率的な場合とそうでない場合がある。

(2) 配列の参照パターンの改善

ベクトルデータは並び方により連続ベクトル，等間隔ベクトルおよび間接指標ベクトルの三種類に分けられる。図13にベクトルの種類を示す。ベクトル命令によりこれらのベクトルデータを処理するため

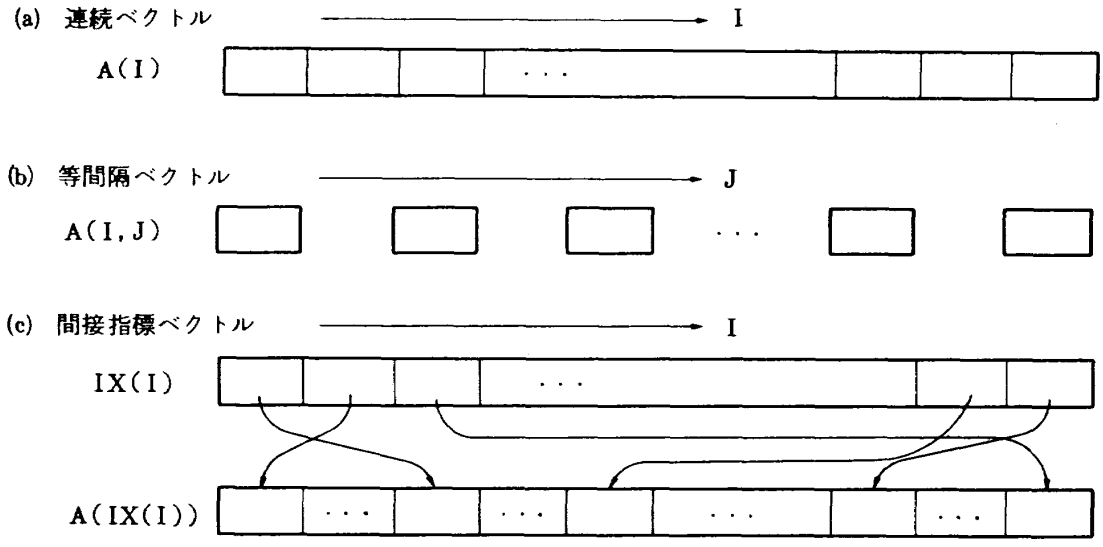


図13　ベクトルの種類

には，最初に記憶域からベクトルデータを取り出し，処理したあと再び記憶域に格納する必要がある。この記憶域からの取り出しや記憶域への格納に要する時間は必ずしも一様ではない。

三種類のベクトルデータのうち，連続ベクトルおよび要素間距離が奇数の等間隔ベクトルデータが最も効率の良い並び方である。

(3) 並列実行可能な演算数の増加

ベクトル演算に関して，加減算，乗算，シフト演算（実数の2倍・$\frac{1}{2}$倍を含む）および論理演算は互いに並列実行が可能である。したがって，同一ループ中の演算数の多い程，また，演算の種類も多い程効率が良い。次の例のようにDOループはなるべく一つのDOループに統合すべきである。

```
例      DO  10  I=1, N
        A(I)=B(I)+C(I)
   10   CONTINUE
        DO  20  I=1, N
        X(I)=Y(I)*Z(I)
   20   CONTINUE
          ⇩
        DO  10  I=1, N
        A(I)=B(I)+C(I)
        X(I)=Y(I)*Z(I)
   10   CONTINUE
```

(4) DOループ内の演算の効率化

DOループ中にIF文が含まれている場合も，ベクトルマスク機能や圧縮，伸長機能を用いてDOループはベクトル化される。

しかし，マスク付きベクトル演算命令では，マスクされている要素に対する演算そのものは抑止されるが，マスクされていない場合と同じ実行時間がかかり，さらにマスク生成のための時間が実行時間に加算される。

```
例      DO  10  I=1, 1000
          IF(e)  THEN
            A(I)=B(I)*C(I)        ①
          ENDIF
   10   CONTINUE

        DO  10  I=1, 1000
            A(I)=B(I)*C(I)        ②
   10   CONTINUE
```

例において，①の実行時間は②の実行時間＋マスク生成のための実行時間と同じである。eの真率が1％の場合，ベクトル化しない場合には乗算および代入は10回実行される。しかし，ベクトル化される場合には1000個の要素すべてについてベクトル演算を行うのに必要な時間がかかる。

圧縮，伸長機能を用いる場合にも，圧縮後のベクトル長が短くなる可能性や圧縮および伸長のための時間が必要となるため，大きな効率向上は期待できない。

したがって，DOループ中のIF文は可能ならば除去または別の表現に変更すべきである。

この他，除算のベクトル命令は他のベクトル命令に比較して遅いので，乗算化または使用頻度を下げるなどして回避する方がよい。

(5) ベクトル化オプションおよびベクトル化指示行の有効利用

ベクトル化オプションやベクトル化指示行の有効利用によりベクトル化効率を高めることができる。

たとえば，ベクトル長が短くベクトル化すると逆に遅くなる場合やコンパイラによるループ入れ換えを抑止する場合に，ベクトル化を抑止できる。

```
例      DIMENSION  A(M, N), B(M, N),
                   C(M, N)
          ⋮
  *VDIR NOVECTOR
        DO  10  I=1, IM
        DO  10  J=1, JM
        A(I, J)=B(I, J)*C(I, J)
   10   CONTINUE
```

IM＝10，JM＝10000とする。

例において，ベクトル化抑止指示行によりループの入れ換えは行われない。

このように

1) DOループのベクトル長の拡大
2) DOループ内の演算数および演算種類の増大
3) 効率的な配列参照パターン
4) DOループ内の効率の悪い演算の回避

5) ベクトル化指示行の有効利用

により，実行効率を上げることができる。

図14に性能向上のための作業手順を示す。

4. 対象プログラムの高速化

日本電気との共同研究において航技研汎用プログラムAFMESHおよびINVERSEをSXシステムを使用して高速化したので，本章ではその二本のプログラムに対する高速化技術について示す。二本の高速化対象プログラムは図15のSX−2システム環境の下で実行された。使用したコンパイラは日電のREVISION 022である。

なお，GOジョブステップ実行時間には演算処理を行うためのAP実行時間と入出力処理を行うためのCP実行時間があるが，ここでは，GOジョブステップの実行時間としてAP実行時間を採用した。ただし，汎用計算機との比較においてはAP＋CP実行時間を採用した。

また，本報告で行われた高速化のためのチューニングはいくつかを除き，パイプライン方式のスーパコンピュータに適用可能である。SXシステム特有（コンパイラも含む）のチューニングについてはその都度，示すものとする。

4.1 AFMESHプログラム

4.1.1 AFMESHプログラムの概要[5]

AFMESHは翼および計算領域を設定し，翼まわりの格子座標系を構成するプログラムであり[6]，幾何学的構成法，連立非線型偏微分方程式の差分解による構成法および後方領域構成法を組み合わせることにより翼まわりの格子網を形成するプログラムである。

図16(1)の物理面の翼および線分JEを切り開いて図16(2)の計算面をつくり，差分解による構成領域では次に示す二種類の偏微分方程式[7]を解くことにより格子の位置を求めている。

物理面において次の連立楕円型非斉次方程式（ポアソン方程式）

$$\Delta\xi = \xi_{xx} + \xi_{yy} = P(\xi, \eta)$$

$$\Delta\eta = \eta_{xx} + \eta_{yy} = Q(\xi, \eta)$$

および斉次方程式（ラプラス方程式）

$$\Delta\xi = \xi_{xx} + \xi_{yy} = 0$$

$$\Delta\eta = \eta_{xx} + \eta_{yy} = 0$$

を計算面に逆変換し，これらの方程式から差分方程式を作成し，ξ方向の三項方程式をη方向にスィープさせて解を求め，格子網を発生させている。

AFMESHプログラムは約50のサブルーチンから成る。モジュール構造を図17に示す。

なお，AFMESHプログラムの実行において，格子点数は125×51を，図16に示す三つの構成領域を構成するパラメータを用いた。

4.1.2 AFMESHプログラムの高速化I

汎用プログラムAFMESHはPLOT出力のためのサブルーチンを含んでおり，FACOM M 380計算機システム（以下M380システムという）におけるGOジョブステップの実行時間は54.8秒である。高速化のために煩雑さを避けるためPLOT出力に関するサブルーチンを削除したAFMESHプログラムのM 380システムにおけるGOジョブステップの実行時間は52.93秒であり，1.87秒実行時間が減少するが大勢に影響を与える時間ではないので，PLOT出力関係サブルーチンを削除したAFMESHプログラムをオリジナルプログラムとして高速化対象プログラムとする。

AFMESHオリジナルプログラムをそのままSX−2システムで実行すると（スカラ指定），GOジョブステップの実行時間は22.615秒であり，ベクトル化指定を行った場合の実行時間は9.492秒である。但し，これには日電のREVISION 010のコンパイラを用いた。このように，スーパコンピュータを単に用いただけでは効果は少なく，プログラムがベクトル化されて初めてスーパコンピュータの効果を発揮する。

図18はANALYZER／SXにより得られたAFMESHオリジナルプログラムのベクトル化に関する出力結果である。図の①はサブルーチンか関数かの類別，②はサブルーチン名，③はサブルーチンの実行回数，④はサブルーチンの実行コストの比率，⑤はサブルーチンのベクトル化率，⑥はサブルーチン内のDOループの数，⑦は⑥のDOループの内，ベクトル化されたDOループの数，⑧は⑦のDOループの実行コストの比率を示す。図18からSOLVEサブルーチンに全CPUTの99.08％が費やされているこ

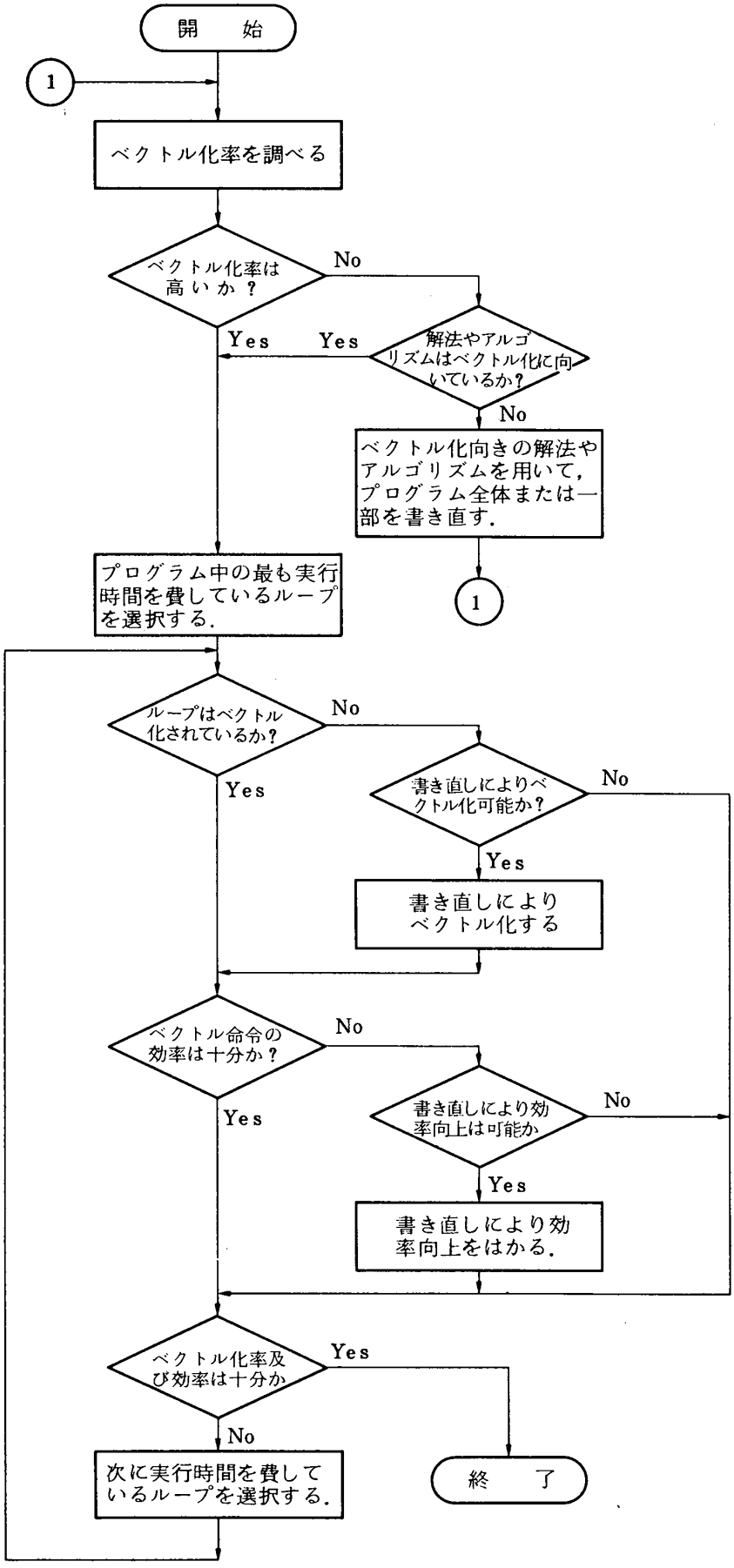


図14　性能向上のための作業手順

AP：演算プロセッサ
CP：制御プロセッサ
CPM：制御プロセッサメモリ
MMU：主記憶装置
IOP：入出力処理装置
URP：ユニットレコード処理装置
OPS：操作卓
PPR：ページプリンタ
FDD：フロッピディスク装置
MTP：磁気テープ処理装置
MTU：磁気テープ装置
MSP：磁気ディスク処理装置
DKU：磁気ディスク装置
FNP：通信処理装置

PSQ：電源制御機構
AOC：自動運転装置

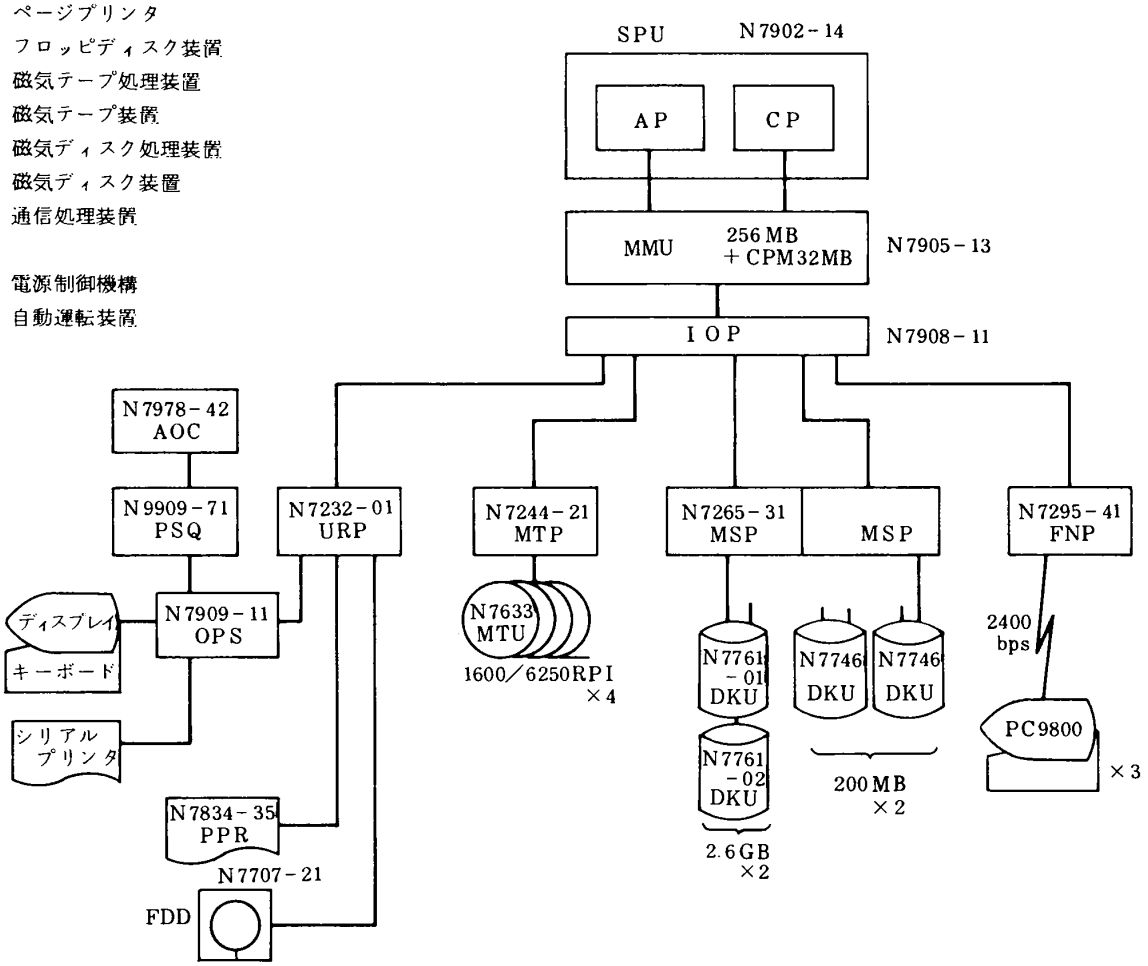


図15　共同研究に使用したSX−2システム

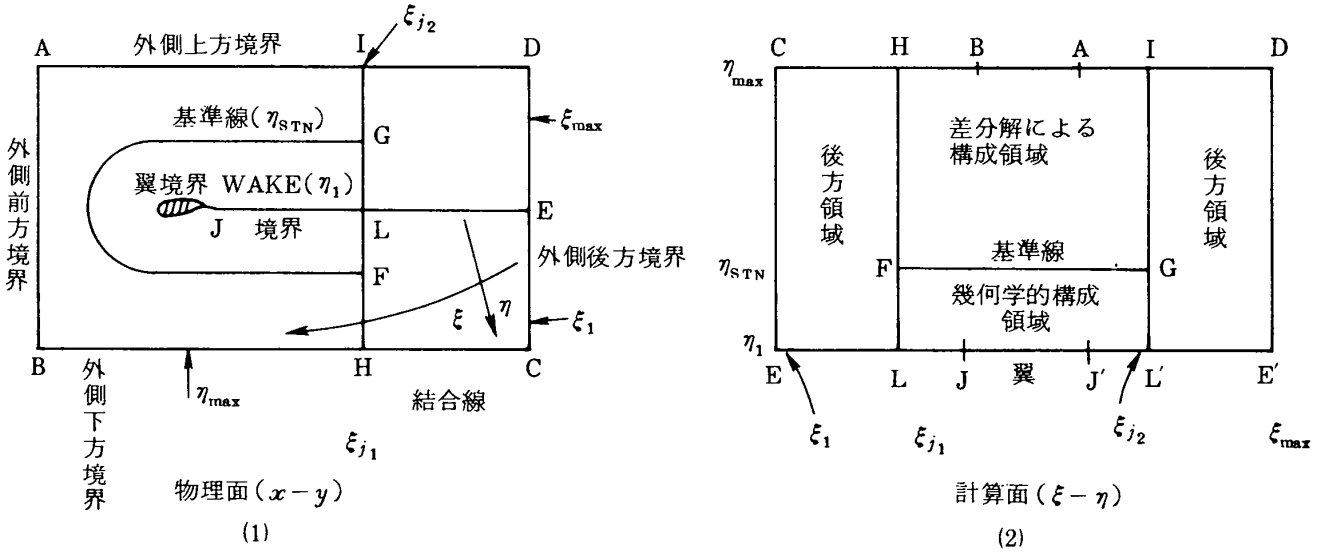


図16　座標系

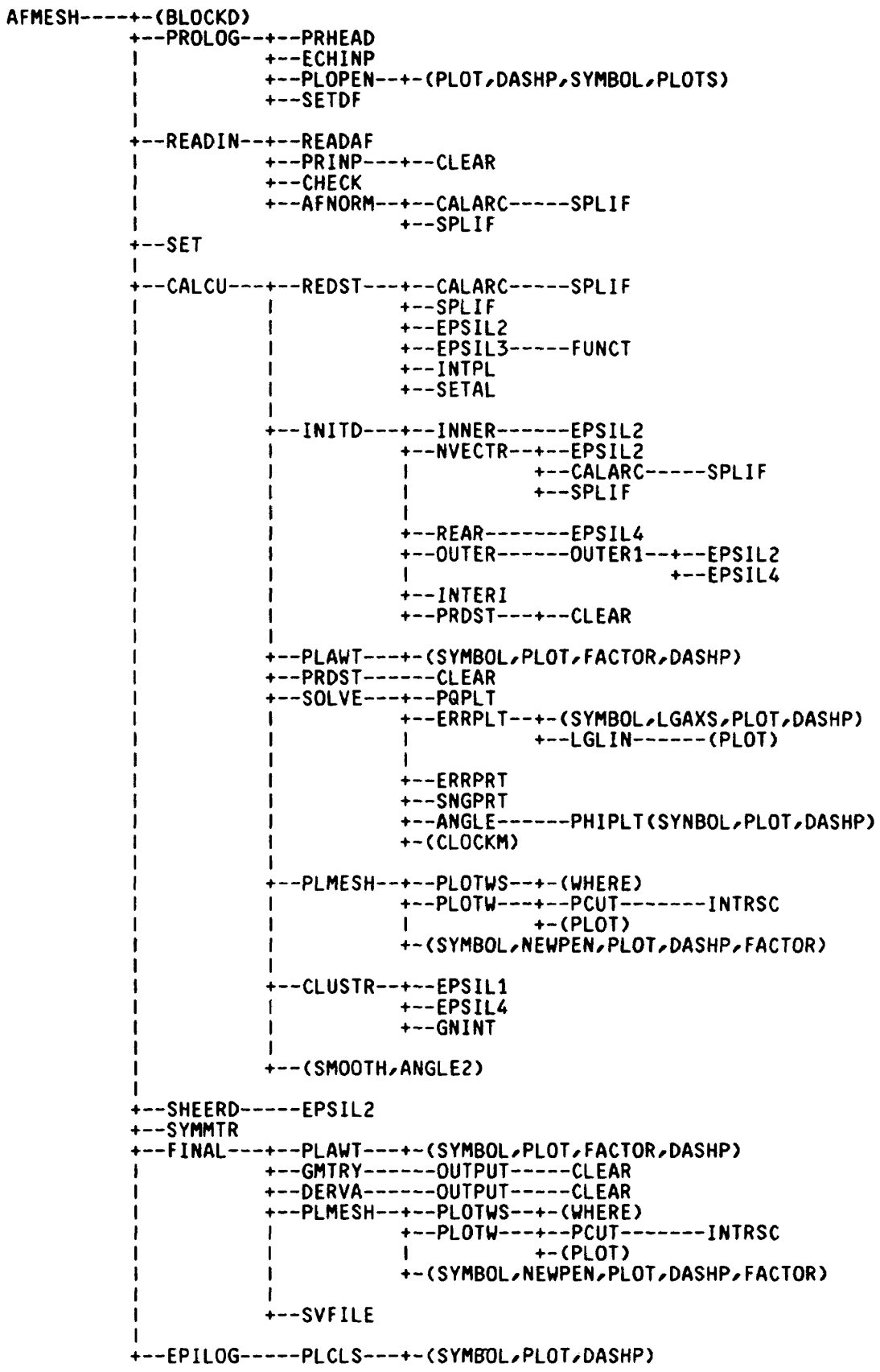


図17 AFMESHプログラム構造図

```
*****
*****  TOTAL EXECUTION ( CPU ) TIME =   0 :  1 :  9 . 246   (    69248 MSEC)
*****  TOTAL EXECUTION FREQUENCY    =          216336514
*****  TOTAL VECTORIZED RATIO       =   82.61%
*****
```

ATR	PROGRAM	FREQUENCY	EXEC COST(%)	V.RATIO	LOOP	V.LOOP	V.LOOP RATIO
-->	MAIN	1	0.01	99.89	3	3	100.00
SUB	AFNORM	1	0.00	95.58	11	9	100.00
SUB	ANGLE	1	0.00	99.66	2	2	100.00
BLK	BLKD						
SUB	CALARC	15	0.01	98.39	7	7	100.00
SUB	CALCU	1	0.01	99.73	4	4	100.00
SUB	CHECK	1	0.00	0.00	4	0	0.00
SUB	CLEAR	22	0.00	0.00	1	0	0.00
SUB	CLUSTR	2	0.08	55.49	12	11	55.51
SUB	DERVA	0	0.00	0.00	6	0	0.00
SUB	ECHINP	1	0.00	0.00	0	0	0.00
SUB	EPILOG	1	0.00	0.00	0	0	0.00
SUB	EPSIL1	2	0.00	79.52	4	3	80.14
FNC	EPSIL2	103	0.02	94.37	6	6	100.00
FNC	EPSIL3	2	0.00	0.00	0	0	0.00
FNC	EPSIL4	6	0.00	0.00	1	0	0.00
SUB	ERRPRT	2	0.00	0.00	0	0	0.00
SUB	FINAL	1	0.00	0.00	0	0	0.00
FNC	FUNCT	41	0.01	99.39	5	5	100.00
SUB	GMTRY	1	0.04	99.99	15	12	100.00
SUB	GNINT	17304	0.66	0.00	2	0	0.00
SUB	INITD	1	0.00	0.00	0	0	0.00
SUB	INNER	1	0.00	69.77	3	1	84.22
SUB	INTERI	1	0.00	0.88	2	1	0.90
SUB	INTPL	2	0.01	0.00	4	0	0.00
SUB	MVECTR	1	0.00	98.41	9	9	100.00
SUB	OUTER	1	0.00	0.00	1	0	0.00
SUB	OUTERI	0	0.00	0.00	7	3	0.00
SUB	OUTPUT	0	0.00	0.00	2	1	0.00
SUB	PRDST	0	0.00	0.00	1	0	0.00
SUB	PRINP	1	0.00	0.00	2	0	0.00
SUB	PROLOG	1	0.00	0.00	0	0	0.00
SUB	READAF	1	0.00	0.00	0	0	0.00
SUB	READIN	2	0.00	0.00	0	0	0.00
SUB	REAR	1	0.00	13.80	2	1	17.30
SUB	REDST	1	0.00	92.25	23	21	100.00
SUB	SET	2	0.00	0.00	0	0	0.00
SUB	SETAL	0	0.00	0.00	2	2	0.00
SUB	SETDF	1	0.00	40.89	3	2	84.26
SUB	SHEERD	1	0.02	97.42	19	16	98.88
SUB	SNGPRT	0	0.00	0.00	0	0	0.00
SUB	SOLVE	2	99.08	83.18	37	31	83.46
SUB	SPLIF	56	0.04	71.02	8	6	72.42
SUB	SYMMTR	0	0.00	0.00	4	4	0.00
	①	②	③	④	⑤	⑥	⑦	⑧

図18 チューニング前のAFMESHプログラムのベクトル化情報

とがわかり，ベクトル化率は83.18％であることがわかる。

以上のことから，AFMESHプログラムを高速化するためにはSOLVEサブルーチンを高速化することが効果があると思われるので，高速化の対象をSOLVEサブルーチンにしぼって行った。

表4はAFMESHプログラムに対するチューニングの一覧表である。チューニング項目，修正モジュール，オリジナルサブルーチンに対してチューニングを施した時のGOジョブステップの実行時間，オリジナルプログラムとチューニングプログラムの実行時間の差（効果），累積効果およびベクトル化率を示した。オリジナルプログラムのSX-2システムにおけるGOジョブステップの実行時間は8.852秒であり，表4のチューニングを施すことにより6.412秒になる。AP＋CP実行時間は6.596秒であるので，8倍の高速化を得たことになる。

以下にチューニング内容の具体的説明およびその効果について述べる。チューニング事項については例の一つ一つについて付録にその内容を示す。

［チューニング1］

SOLVEサブルーチン内には次の例で説明するものと同じDOループが存在する。

```
例      DO  100  I=I1, I2
            A(I)=B(I) * C(I)
100     CONTINUE
        DO  200  I=I1, I2
            D(I)=A(I)+F(I)
200     CONTINUE
        DO  300  I=I1, I2
            G(I)=D(I) * H(I)
300     CONTINUE
```

これらのDOループはいずれもI1からI2まで実行され，各DOループの間でI1およびI2の値は変化しないので，三つのDOループを合併し，次のように一つのDOループにすることが可能である。DOループを合併することにより，DOループの立ち上がり時間の短縮と，DOループ内の演算の数と種類の増大によるパイプラインの並列走行性が高くなるという効果が得られる。

```
        DO  100  I=I1, I2
        A(I)=B(I) * C(I)
        D(I)=A(I) + F(I)
        G(I)=D(I) * H(I)
```

表4 AFMESHプログラムチューニング表

No	チューニング項目	修正モジュール	GOジョブステップの実行時間 (s)	効果 (s)	累積効果 (s)	ベクトル化率 (%)
	なし(オリジナル)	—	8.852	—	—	82.61
1	DOループの合併配列のテンポラリ化式の変形	SOLVE	8.458	0.394	—	—
2	組込関数MAXのベクトル化	SOLVE	7.147	1.705	—	—
3	ベクトル関数LVMXAの使用	SOLVE	8.842	0.010	—	—
4	一次漸化式の部分ベクトル化	SOLVE	9.247	−0.395	—	—
5	総合チューニング(1)	SOLVE	6.666	2.186	—	—
6	総合チューニング(2)	SOLVE	6.412	0.254	2.440	86.46

M380計算機システムGOジョブステップ実行時間 52.93秒

```
100  CONTINUE
```

さらに，配列AおよびDはDOループを合併することにより配列変数である必要がなくなり，テンポラリ変数としても差し支えないので次のようにテンポラリ変数とする。これを配列のテンポラリ化という。配列AおよびDは主記憶装置とレジスタ間の入出力の対象となり，テンポラリ変数はレジスタ上の仮変数となるので主記憶とレジスタ間の入出力の対象とならない。従って，配列変数のテンポラリ化は不要のLOAD／STORE命令を削除する効果がある。

```
      DO  100  I=I1, I2
      AA = B(I) * C(I)
      DD = AA + F(I)
      G(I) = DD * H(I)
100   CONTINUE
```

SOLVEサブルーチンには上に述べたようなDOループ群が六つ存在し，それらのDOループ群に対してDOループの合併，DOループ内の配列のテンポラリ化，および計算式の簡単化の修正を行った。便宜的に六つのDOループ群をG1～G6ループとし，各DOループ群のチューニングの説明を行う。

付録A1(1)はG1ループに対するチューニングである。四つのDOループを合併して一つのDOループにした。このループ群の実行回数は1回，ベクトル長は101であり，共に少ないので高速化に対する寄与率は少ない。

付録A1(2)はG2ループに対するチューニングである。G1ループに対するチューニングと全く同等である。

付録A1(3)はG3ループに対するチューニングである。11個のDOループを合併して一つのDOループにまとめた結果を付録A1(3)の(I)に示す。さらに，(I)のDOループの前半に対して配列R1，R2，XXI，YXIのテンポラリ化，および配列XETS2，YETS2をテンポラリ化して参照場所に展開する修正を行った。また，(I)のDOループの後半に相当する付録A1(3)の(II)の部分は

```
      R1(IDXJ) = SIGN(XXI(IDXJ),
     1                R1(IDXJ))
      IF(P1(IDXJ).NE.0.0)THEN
```

```
      R1(IDXJ)＝R1(IDXJ)／
1                 P1(IDXJ)
      ENDIF
```

とする計算が配列GおよびLMSKを使用する複雑なコーディングであるので，簡単なコーディングに修正し，演算量を減少させた。配列Q1に関しても配列R1に関すると同様の修正を行った。オリジナルプログラムのコーディングの複雑性はFACOM 230－75 AP計算機システム（以下APシステムという）からM 380システムへの移行作業によるものである。これらの修正を施すことによりG3ループは付録A1(3)の(Ⅲ)に示すように書き換えられた。G3ループの実行回数は451回，ベクトル長は101である。

付録A1(4)はG4ループに対するチューニングであり，二つのDOループを一つに合併したものである。G4ループの実行回数は451回であり，ベクトル長は99である。

付録A1(5)はG5ループに対するチューニングである。四つのDOループを合併し，一つにまとめたDOループを付録A1(5)の(Ⅰ)に示す。さらに，(Ⅰ)に示すDOループ内の配列XXI，YXI，XETA，YETA，GD，およびBDのテンポラリ化の修正を施したものを付録A1(5)の(Ⅱ)に示す。さらに，(Ⅲ)に示すDOループ内で不変の計算式を予めDOループ外で計算しておき，DOループ内では参照する形式に修正を行った。FLOAT(K−(KSTART−1))はDOループ内で2度使用されるが，この演算は1度行われ，2度目は参照される。このDOループの実行回数は56112回であり，ベクトル長は101であるので

$$56112 \times 101 = 5667312 \text{ 回}$$

の計算が実行される。計算される値は不変であるので予め計算しておくことによりその演算の実行回数は42回で済む。この修正の効果は大きく，表には示さないが，チューニング1に対する効果0.394秒の2／3を占める。

このチューニングにより，FLOAT(K−(KSTART−1))を格納するための配列を作成しなければならない。これは主記憶量を必要とすることなので，利用可能な主記憶量が小さい場合には格納領域をとることができないので毎回計算を行わねばならない。本プログラムはAPシステムにおいて作成されたプログラムであり，利用可能な最大主記憶量は1MBであった。M 380システムにおける利用可能な最大主記憶量は32MBであり，SX−2計算機システムにおける利用可能な最大主記憶量は256 MBである。従って，本チューニングの効果は計算機の性能が上がると共に増大する最大主記憶量により得られる。

これらの修正を施すことによりG5ループは付録A1(5)の(Ⅲ)に示すように書き換えられた。

付録A1(6)はG6ループに対するチューニングである。五つのDOループを合併したが，最後のDO 12340ループはベクトル化されていないので，合併されたDOループのその部分に相当するステートメントには“S”表示で，そのステートメントがベクトル化されないことが示されている。したがって，合併されたDOループは部分ベクトル化されている。G6ループの実行回数は56112回であり，ベクトル長は101である。G1～G5ループに比較してG6ループの実行回数は多いので，DOループが完全にベクトル化されればかなりの効果が期待できる。なお，“S”表示のステートメントがベクトル化されない理由は組み込み関数MAXのパラメータが三つあることによる。詳細はチューニング2で述べる。

［チューニング2］

オリジナルのSOLVEサブルーチンのDO 12340ループでは組み込み関数MAXを使用して最大値を求めている。現時点では組み込み関数MAXのパラメータは二つを対象としている。オリジナルでは三つの値の最大値を求めているのでこのDOループはベクトル化されない。これをベクトル化するために，変数WMAXを新しくつくり，最初に二つの値の大きい方を求め，次にRMAXとWMAXの比較を行うようにコーディングを変更した。この修正によりDOグループがベクトル化された。

さらに，MAX関数のパラメータにOMEGAという常数を毎回乗じているが，

$$\text{OMEGA} > 0$$
$$\text{ABS}(\text{SUB}(\text{IDXJ})) \geqq 0$$
$$\text{ABS}(\text{SUP}(\text{IDXJ})) \geqq 0$$

であることにより，最後にOMEGAを乗じてRMAXの値としても差し支えないので，コーディングをか

えた。この修正により乗算の演算量を減少させることができた。これらの修正の効果は大きく，表4により1.705秒速くなったことがわかる。付録A 2に修正前および修正後のコーディングを示す。

［チューニング3］

SOLVEサブルーチン内に付録A3に示すコーディングステートメントがある。ここでは特異点の値を除いて，配列R1およびR2の最大値とその最大となる位置を求めている。その方法は特異点の値を除いて仮配列XXIおよびYXIをつくり，配列R1およびR2の個数より一つ少ない個数の配列に対して最大値とその位置を求めている。これを組み込み関数LVMXAを用いて修正した。LVMXAは指定された配列の絶対値の最大値の位置を求める関数であり，スーパコンピュータ向きの効率の良い関数である。LVMXAにより位置が求められたら，指定した配列のその位置の値を最大値として定義すればよい。この修正によりベクトル化されないDOループがベクトル関数LVMXAに置き換えられ，ベクトル効率の良いステートメントになった。LVMXAはFORTRAN77/SX言語特有の組み込み関数である。

また，配列R1およびR2の絶対値の最大値を求めるので，特異点の値を0.0に設定し，R1およびR2の最大値を求めても差し支えない。この修正は仮配列XXIおよびYXIの削除，すなわち仮配列へのLOAD/STORE命令を削除できるので，無駄な演算量をなくすことができる。

なお，計算結果に支障はないが，最大値を求めるDOループが二重になっているために演算量が2倍になっていたので余分なコーディングは削除した。

最大値を求めるステートメントに対する修正を行ったチューニング2に比較して，本チューニングの効果は少ないことが表4の効果よりわかる。これは実行回数がチューニング2の場合は56112回であるのに対し，チューニング3の場合は451回と圧倒的に少ないことが原因である。

［チューニング4］

方程式を解くDO 2100ループがベクトル化されないので部分ベクトル化されるようにコーディングを変更した。付録A 4にチューニング前およびチューニング後のコーディングを示す。

DO 2100ループの実行回数は56112回であり，かなり多い実行回数にもかかわらず，表4からは0.395秒の実行時間の増加となりベクトル化の効果のないことがわかる。

チューニング前とチューニング後のコーディングを比較する。

```
Z=1.0/(DIAG(J)-SUB(J)*SUP(J-1))
SUP(J)=SUP(J)*Z
```

の2ステートメントはチューニング前もチューニング後もスカラ命令で処理される。しかし，

```
F(J)=(F(J)-SUB(J)*F(J-1))*Z        ①
```

は

```
F(J)=F(J)*Z-SUB(J)*Z*F(J-1)        ②
```

とコーディングを変更することによりベクトリ命令で処理されることになる。①の場合は乗算2，減算1のスカラ命令で処理されるが，②の場合は乗算3，減算1のベクトル命令で処理される。

DOループが部分ベクトル化されることによる立ち上がり時間がベクトル長100のベクトル演算の処理に比較して相対的に大きいことや，漸化式演算*のベクトル命令は他のベクトル命令に比較して遅いことが，チューニング効果のない理由として考えられる。

従って，ベクトル化率を高めるためにDOループのベクトル化を促進することが，逆の演算量やオーバヘッドを増加させる場合があるので，効果的にベクトル化を行う必要がある。

なお，本チューニングではベクトル化の促進のためのコーディング変更によるチューニング効果は得られなかったが，ベクトル長の長いDOループ，DOループ内のベクトル演算の効率の良いものなどを対象としたベクトル化であれば効果を発揮し得ることを注意しておく。

［チューニング5］

修正を行っても効果のないチューニング4を除くすべての修正をSOLVEサブルーチンに対して施した。これにより，8.852秒から6.666秒になり，2.186秒CPUTを減少させることができた。

---

* 漸化式演算
FORTRAN 77/SX言語特有のマクロ演算である。

［チューニング6］

SOLVEサブルーチンではラプラスおよびポアソン方程式が解かれているが，これをラプラス方程式を解くサブルーチンSOLVELと，ポアソン方程式を解くサブルーチンSOLVEPの二つに分けた。この分割により，ラプラス方程式かポアソン方程式かの判定のIF文の削除，各方程式に不要のIF文の削除がなされた。また，SOLVELサブルーチンでは不要のDO 10100ループで行われている配列の零クリアの削除，およびDO 2000ループ内のラプラス方程式では不要の計算式を削除した。これらは基礎方程式においてラプラス方程式はP＝Q＝0となるため削除できる部分である。付録A5に修正前および修正後のコーディングを示す。

これらの修正により0.254秒実行時間を減少させることができた。この効果はサブルーチンSOLVELにおける不要の方程式計算の削除によるものである。

（まとめ）

AFMESHプログラムの最もCPU負荷のかかるサブルーチンSOLVEに対して次のチューニングを行った。

1) いくつかのDOループを合併し，テンポラリ化可能な配列はテンポラリ変数に置き換えた。これによりDOループの立ち上がり時間の節約，余分なLOAD／STORE命令の削除およびパイプラインの並列処理による高速化効果があった。
2) 簡単なコーディングスタイルに変更して余分な計算量（演算量）を減少させた。
3) DOループ内で不変な計算式をDOループ外に排出して予め計算しておき，以後，参照する形式にしてその演算量を減少させた。
4) 組み込み関数MAXをベクトル化可能な形に変更してDOループをベクトル化可能にした。これにより，DOループは完全ベクトル化となりベクトル計算機の効果を発揮した。
5) 組み込み関数LVMXAを使用するコーディングスタイルに変更して高速化を図った。
6) ベクトル化されないDOループに対してコーディングスタイルを変えてDOループを部分ベクトル化可能にした。これは高速化に対してさほど効果がなかった。
7) SOLVEサブルーチンを二つに分けることにより不要計算を削除し，その演算量を減少させて高速化をはかった。

これらのチューニングの中でAFMESHプログラムに最も効果のあったものは組み込み関数MAXのベクトル化である。この実行頻度は高く，この部分がベクトル化されるか否かは全体のCPUTを左右する。また，DOループを合併してテンポラリ化可能な配列をテンポラリ変数にすることは効果がある。予め不要であることがわかっている計算は削除しておく効果も大きい。

様々なチューニングを施すことによりベクトル化率は82.61％から86.46％になった（図19）。オリジナルプログラムのベクトル化率が高いので，ベクトル化率は4％の増加にとどまった。

AFMESHプログラムはAPシステムにおいて，AP－FORTRAN言語を用いて作成された。その後，M380システムへの移行作業においてAP－FORTRAN言語からFORTRAN 77言語に書き換えられた。この時期の短期間の大量のプログラムの移行作業のために，AP－FORTRANで書かれたプログラムをFORTRAN 77コンパイラで実行可能にするための作業をAPTRAN*プログラムを用いて行った。AP－FORTRAN言語では一行のコーディングで数次元の配列すべてを計算する機能をもたせていたので，APTRANプログラムを通すことにより一行のコーディングはDOループを使うコーディングに書き換えられた。このようなコーディングに対してはDOループを統合することでかなりのCPUT減少をもたらす。AFMESHプログラムの場合も上記と同様のチューニングを施すことにより，M380システムのGOジョブステップの実行時間は52.93秒から38.31秒になり14.62秒節約することができる。このように，1ケース当りの実行時間の短縮化が計られれば全ケースを処理する時間も少なくて済み，経済的である。また，計算機システムが変わった場合には，

---

* APTRANプログラム
AP－FORTRAN言語で書かれたプログラムを汎用計算機で実行可能となるようにコンバートするプログラムである。

```
*****
*****  TOTAL EXECUTION ( CPU ) TIME =   0 :  0 ' 47 " 547   (      47547 MSEC)
*****  TOTAL EXECUTION FREQUENCY    =                 161113147
*****  TOTAL VECTORIZED RATIO       =   86.46%
*****
```

ATR	PROGRAM	FREQUENCY	EXEC CONT(%)	V.RATIO	LOOP	V.LOOP	V.LOOP RATIO
-->	MAIN	1	0.02	99.89	3	3	100.00
SUB	AFNORM	1	0.00	95.58	11	9	100.00
SUB	ANGLE	1	0.00	99.66	2	2	100.00
BLK	BLKD						
SUB	CALARC	15	0.01	98.39	7	7	100.00
SUB	CALCU	1	0.01	99.73	4	4	100.00
SUB	CHECK	1	0.00	0.00	4	0	0.00
SUB	CLEAR	22	0.00	0.00	1	0	0.00
SUB	CLUSTR	2	0.11	55.49	12	11	55.51
SUB	DERVA	0	0.00	0.00	6	0	0.00
SUB	ECHINP	1	0.00	0.00	0	0	0.00
SUB	EPILOG	1	0.00	0.00	0	0	0.00
SUB	EPSIL1	2	0.00	99.22	4	4	100.00
FNC	EPSIL2	103	0.03	96.37	6	6	100.00
FNC	EPSIL3	2	0.00	0.00	0	0	0.00
FNC	EPSIL4	6	0.00	0.00	1	0	0.00
SUB	ERRPRT	2	0.00	0.00	0	0	0.00
SUB	FINAL	1	0.00	0.00	0	0	0.00
FNC	FUNCT	41	0.01	99.39	5	5	100.00
SUB	GMTRY	1	0.05	99.99	15	12	100.00
SUB	GMINT	17304	0.88	0.00	2	0	0.00
SUB	INITD	1	0.00	0.00	0	0	0.00
SUB	INNER	1	0.00	69.77	3	1	84.22
SUB	INTERI	1	0.01	0.88	2	1	0.90
SUB	INTPL	2	0.01	4.33	4	2	4.55
SUB	MVECTR	1	0.00	98.41	9	9	100.00
SUB	OUTER	1	0.00	0.00	1	0	0.00
SUB	OUTERI	0	0.00	0.00	1	0	0.00
SUB	OUTPUT	0	0.00	0.00	2	1	0.00
SUB	PRDST	0	0.00	0.00	1	1	0.00
SUB	PRINP	1	0.00	0.00	2	0	0.00
SUB	PROLOG	1	0.00	0.00	0	0	0.00
SUB	READAF	1	0.00	0.00	0	0	0.00
SUB	READIN	2	0.00	0.00	0	0	0.00
SUB	REAR	1	0.00	13.80	2	1	17.30
SUB	REDST	1	0.00	92.25	23	21	100.00
SUB	SET	2	0.00	0.00	0	0	0.00
SUB	SETAL	0	0.00	0.00	2	2	0.00
SUB	SETDF	1	0.00	40.89	3	2	84.28
SUB	SHEERD	1	0.03	97.42	19	18	98.88
SUB	SMGPRT	0	0.00	0.00	0	0	0.00
SUB	SOLVEP	1	38.35	88.43	18	13	88.73
SUB	SOLVEL	1	60.42	86.52	6	5	86.85
SUB	SPLIF	56	0.05	71.02	8	8	65.65
SUB	SYMMTR	0	0.00	0.00	4	4	0.00
SUB	CLOCK	4	0.00	0.00	0	0	0.00

図19 チューニング後のAFMESHプログラムのベクトル化情報

プログラムの移行作業を順次行って，新システムにおいてプログラムが効率的に実行されるかの見直しを図るべきである。

4.1.3 AFMESHプログラムの高速化Ⅱ

AFMESHプログラムをSX−2システムで実行すると8.852秒かかるが，高速化Ⅰで述べた様々なチューニングを施して6.412秒になった。本節ではさらに高速化するために方程式の解法をスーパコンピュータ向きの解法[8) 9)]に変更することを試みた。なお，本節で述べるプログラムは日電のREVISION 018のコンパイラで実行されたものである。

オリジナルAFMESHプログラムのラプラスおよびポアソン方程式はSLOR法で解かれている。解法をZEBRAI法やVOLUME CHEKERBOARD法[6) 7)]に変更することによって高速化できる。

ZEBRAI法はSLOR法にODD－EVEN法を取り入れた解法であり，回帰的演算がなくベクトル化可能であるためスーパコンピュータ向きの解法である。三次元(I, J, K)で説明する。I方向の連立方程式をJ方向にスイープし，Kのループでまわす時に，J＋K：ODDならば黒，J＋K：EVENならば白とし，はじめ黒のI方向の方程式を一度に並列に計算した後，白の場合を同様に計算して1回のイタレーションとする。

VOLUME CHEKERBOARD法はSLOR法をSOR法に変えODD－EVEN法を取り入れた解法であり，ZEBRAI法と同様にスーパコンピュータ向きの解法である。三次元(I, J, K)で説明する。I＋J＋K：ODDならば黒，I＋J＋K：EVENならば白とし，はじめに黒を一度に並列に計算した後，白を同様に計算して1回のイタレーションとする。

AFMESHプログラムをSLOR法からZEBRAI法に変えると，高速化Ⅰでチューンアップされたプログラムの実行時間をさらに約2秒縮めて4.586秒となる。AP+CP実行時間は4.811秒であるので，汎用計算機における実行時間の11倍の高速化が得られた。SLOR法で解いた場合の収束回数は，ラプラス方程式解法では885回，ポアソン方程式解法では451回であるが，ZEBRAI法を用いた場合ラプラス方程式では852回，ポアソン方程式では426回となり，収束回数はいずれの場合も30回程少なくなるがそのオーダは変わらない。

SLOR法からVOLUME CHEKERBOARD法に変更するとその実行には6.609秒要し，高速化Ⅰで

チューンアップした場合の実行時間とほぼ同じである。ラプラスおよびポアソン方程式とも収束回数に大きな変化がみられ，ラプラス方程式は4347回，ポアソン方程式は3338回となり，それぞれSLOR法の収束回数の4.9倍および7.4倍にもなる。収束回数は適当な緩和係数の設定により小さくできるが，今回は適当な緩和係数の設定までは至らなかった。

表5はAFMESHプログラムの解法別の1イタレーションに要する時間，収束回数，および実行時間の比を示したものである。表から以下のことがわかる。

1) ベクトル計算機向きの解法に変更すると1イタレーションに要する時間が少なくなる。
2) 同じ収束回数で収束するならば，VOLUME CHEKERBOARD法が最も速い解法である。
3) ラプラス方程式の場合，収束回数がSLOR法の5～6倍でもVOLUME CHEKERBOARD解法の方が実行時間が少なくてすむ。ポアソン方程式の場合も同様の事が言える。

このように，解法を変更することによる高速化は，たとえ収束回数が増加したとしても十分に効果があることがわかる。しかし，解法を変えることによりオリジナルプログラムの初期条件などの条件を同条件で実行した場合でも，収束しなくなる可能性も出てくるので注意が必要である。

4.2 INVERSEプログラム

4.2.1 INVERSEプログラム[10)11)]の概要

INVERSEプログラムは圧力分布（C_P^S）を与える事によりその圧力分布をもつ翼型を求めるという，言わゆる逆問題を解くプログラムであり，翼型設計に使われるものである。その基本概念はTranenに依るものである。

プログラムは直接解法ステップ[12)13)]および逆解法ステップから成る。それらのステップを交互に繰り返し実行し，C_P^Sと仮翼型から求めた圧力分布（C_P^D）が一致した時点の翼型を求めるものである。

直接解法ステップでは仮翼型を与え，そのまわりで圧縮完全ポテンシャル流の偏微分方程式を解く（Neumann問題）。

$$(c^2-U^2)\phi_{xx}-2UV\phi_{xy}+(c^2-V^2)\phi_{yy}=0$$

$$U=\phi_x,\quad V=\phi_y$$

$$c^2=1/M_\infty^2+(\gamma-1)(1-U^2-V^2)/2$$

γ：比熱比

M_∞：一様流マッハ数

c：音速

逆解法ステップでは仮翼型位置で圧力分布C_Pを与え同じ偏微分方程式を解き（Dirichlet問題），その解から求まる仮翼型位置における外部法線方向の流速成分V_nを用いて流量計算を行い，流線を求め，それを修正された仮翼型とする。

図20にINVERSEプログラムの処理フローの概略を示す。なお，本プログラムの実行は格子点数81×16で行った。

4.2.2 INVERSEプログラムの高速化

図21はANALYZER／SXにより得られたINVERSEプログラムのベクトル化率等の情報であ

表5　AFMESHプログラムのSXシステム処理による1イタレーションに要する時間

解法	ラプラス方程式				ポアソン方程式			
	1イタレーション当りの実行時間*	収束回数（回）	比率（1）**	比率（2）***	1イタレーション当りの実行時間*	収束回数（回）	比率（1）**	比率（2）***
SLOR（高速化1）	4.55	885	1	—	4.75	451	1	—
ZEBRA 1	3.21	852	0.71	1	3.59	426	0.76	1
CHEKER BOARD	0.67	4347	0.15	0.21	1.00	3338	0.21	0.28

*　単位：ミリ秒／イタレーション

**　SLOR法を1とした場合の各解法に要する時間の比率

***　ZEBRAI法を1とした場合のCHEKERBOARD法に要する時間の比率

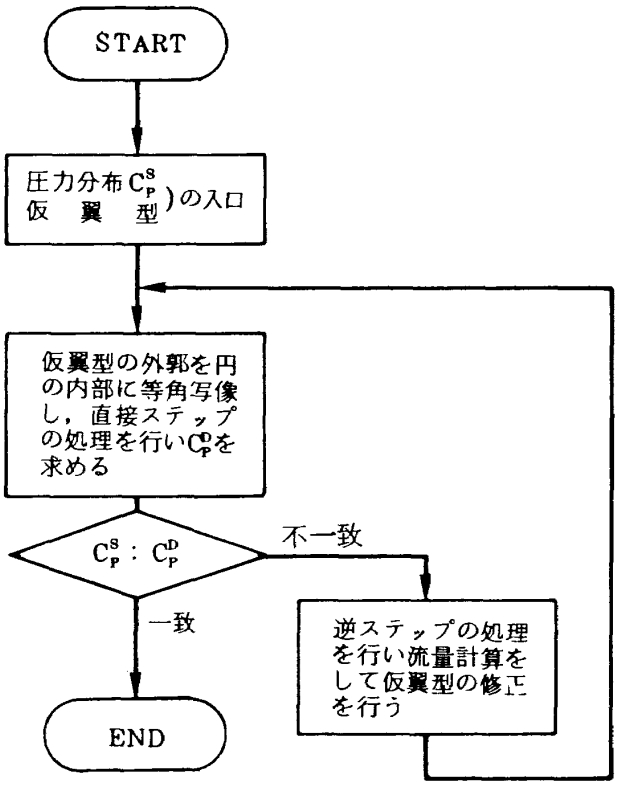


図20 INVERSEプログラム処理フロー概略

る。プログラム全体のベクトル化率は34.45%であり，低い。プログラム全体に対する実行コストの比率はMURMANサブルーチンが最も高く47.81%であり，FFORMおよびTRIDサブルーチンが12.77%，11.0%と続いている。そして，その三つのサブルーチン内のベクトル化率は4.49%から36.81%であり，いずれも低い。従って，INVERSEプログラムの高速化のためには，ベクトル化率を高くすること，実行コストの比率の高い上記の三つのサブルーチンに加えて，4.07～6.1%の実行コストの比率をもつINVRSEおよびSWEEPサブルーチンを対象にチューニングを施すことが良い。

表6はINVERSEプログラムに対するチューニングの一覧表である。チューニング項目，修正モジュール，オリジナルサブルーチンにチューニングを施した時のGOジョブステップの実行時間，オリジナルプログラムとチューニングプログラムの実行時間の差（効果），累積効果，およびベクトル化率を示した。

INVERSEプログラムのM380システムにおけるGOジョブステップの実行時間は11.82秒である。オリジナルプログラムのSX-2システムにおける実

```
*****
*****  TOTAL EXECUTION ( CPU ) TIME =   0 :  0 '33 " 801    (     33801 MSEC)
*****  TOTAL EXECUTION FREQUENCY    =          50206832
*****  TOTAL VECTORIZED RATIO       =  34.45%
*****
```

ATR	PROGRAM	FREQUENCY	EXEC COST(%)	V.RATIO	LOOP	V.LOOP	V.LOOP RATIO
-->	MAIN	1	0.10	49.74	85	43	52.29
BLK	BLKD						
FNC	IBTOD	46	0.00	0.00	0	0	0.00
FNC	INDEXR	12	0.00	0.00	1	0	0.00
SUB	AIRFOL	8	0.82	32.84	38	17	33.09
SUB	CONJ	102	0.21	77.43	4	4	100.00
SUB	COSI	273	0.23	0.00	1	0	0.00
SUB	CPTOLP	4	0.18	3.49	5	2	3.71
SUB	FAIR1	0	0.00	0.00	2	1	0.00
SUB	FAIR3	0	0.00	0.00	1	0	0.00
SUB	FFORM	1502	12.77	4.49	5	1	14.10
SUB	FIGURE	1	0.01	28.93	24	9	29.43
SUB	FOUCF	8	0.01	23.53	2	1	23.71
SUB	GETCP	8	0.09	88.45	3	3	89.38
SUB	GOPRIN	8	0.03	48.63	5	3	49.17
SUB	GRAFIC	4	0.02	87.01	12	10	88.73
SUB	GTURB	8	0.04	80.92	14	12	83.59
SUB	HTURB	0	0.00	0.00	16	12	0.00
SUB	HI3A	0	0.00	0.00	21	15	0.00
SUB	INTPL	226	0.49	9.61	1	1	9.62
SUB	INVRSE	3	4.07	97.64	48	43	99.78
SUB	KMY02	0	0.00	0.00	12	9	0.00
SUB	LEXY	4	0.00	83.92	4	2	89.28
SUB	MAP	8	2.14	18.93	15	11	19.11
SUB	MAXXY	10	0.00	26.06	3	1	26.95
SUB	MFNC	0	0.00	0.00	0	0	0.00
SUB	MURMAN	45120	47.81	23.72	3	3	24.51
SUB	MURMA1	6720	4.91	80.32	1	1	83.37
SUB	MURMOC	0	0.00	0.00	4	4	0.00
SUB	NASHMC	6	0.09	0.00	1	0	0.00
SUB	PHILP	22	0.15	61.42	1	1	63.20
SUB	RESTRT	8	0.13	45.15	10	8	45.23
SUB	SET	62	0.00	0.00	0	0	0.00
SUB	SET1	72	0.00	0.00	0	0	0.00
SUB	SOLV1	84	1.62	48.12	7	5	49.26
SUB	SONIC	4	0.32	0.00	2	0	0.00
SUB	SPLIF	171	0.43	0.00	0	0	0.00
SUB	SWEEP	342	6.10	99.06	15	15	100.00
SUB	SWEEP1	57	1.36	99.30	11	11	100.00
SUB	SWEEP2	27	0.43	99.36	8	8	100.00
SUB	TRDPHA	0	0.00	0.00	3	3	0.00
SUB	TRDPHB	0	0.00	0.00	3	3	0.00
SUB	TRDPHC	17760	1.30	0.00	0	0	0.00
SUB	TRDPOC	0	0.00	0.00	0	0	0.00
SUB	TRID	45120	11.00	36.81	3	1	37.62
SUB	TRID1	3444	1.40	0.00	2	0	0.00
SUB	TWOFFT	1290	1.72	41.27	4	2	41.76
SUB	VALSUM	4	0.02	99.44	7	7	100.00

図21 チューニング前のINVERSEプログラムのベクトル化情報

表6　INVERSEプログラムチューニング表

No.	チューニング項目	修正モジュール	GOジョブステップの実行時間 (s)	効果 (s)	累積効果 (s)	ベクトル化率 (%)
	なし(オリジナル)	—	5.178	—	—	34.45
1	テンポラリ変数の配列化	MURMAN	4.148	1.030	—	70.11
2	テンポラリ変数の配列化	MURMA1	5.143	0.035	—	—
3	サブルーチンのインライン化	TRID	5.071	0.107	—	—
4	サブルーチン呼び出し回数の減少	SWEEP	5.148	0.030	—	—
5	サブルーチン呼び出し回数の減少	SWEEP	5.165	0.013	—	—
6	サブルーチン呼び出し回数の減少	INVRSE	5.149	0.029	—	—
7	サブルーチン呼び出し回数の減少	INVRSE	5.181	−0.003	—	—
8	総合チューニング		3.907	1.241	1.271	71.06

M 380計算機システムGOジョブステップ実行時間　11.82秒

行時間は5.178秒であり，表6のチューニングを施すことにより3.907秒になる。

以下にチューニング内容の具体的説明およびその効果について述べる。AFMESHプログラム同様チューニング事項を付録Bに示す。

[チューニング1]

MURMANサブルーチン内に付録B1に示すDOループが存在する。このDOループは部分ベクトル化されてはいるもののDOループ内の演算はほとんどスカラ演算で行われ，ベクトル化されているステートメントはわずか三行である。これではDOループがベクトル化されてもその効果はない。そこでこのDOループ全体がベクトル化されるように次の修正を行った。簡単のために次の例で説明する。

```
      ⋮
      X0=A(I+1, 1)−A(I−1, 1)
      X1=A(I+1, 2)−A(I−1, 2)
      DO 10 J=2, N
      X2=X1
      X1=A(I+1, J+1)−A(I−1, J+1)
      Y(J)=X1−X0+E(J)
      X0=X2
10    CONTINUE
      ⋮
```

上のDOループはテンポラリ変数X0，X1，X2のデータの定義引用関係が問題となってベクトル化されない。これらのテンポラリ変数を配列X(I)と書きかえることにより，ループ内において，定義が引用に先行している関係をつくることができる。これによりこのDOループ全体をベクトル化することができる。その修正により上のDOループは次のようになる。

```
      DIMENSION X(0:N)
      ⋮
      X(0)=A(I+1, 1)−A(I−1, 1)
      X(1)=A(I+1, 2)−A(I−1, 2)
      DO 10 J=2, N
      X(J)=A(I+1, J+1)−A(I−1, J+1)
      Y(J)=X(J)−X(J−2)+E(J)
10    CONTINUE
      ⋮
```

付録B1に示されるDOループ内のPHIXP,

PHIYY，PHIYPの三つのテンポラリ変数を配列PHIXP(J)，PHIYY(J)およびPHIYP(J)とすることによりDO 60ループ全体をベクトル化することができた。

オリジナルプログラムでテンポラリ変数となっていた理由は，本プログラムが主記憶容量の小さい計算機システムで作成されたことによる。限られたリソースのためにテンポラリ変数にせざるを得なかった。現在では256MBの主記憶量の使用が可能となっているのでそれを十分に使用してベクトル化を図ることができる。

[チューニング2]

付録B2に示すように，MURMA1サブルーチンに対してチューニング1と同様の修正を施した。

[チューニング3]

実行頻度の高いプログラムの中で呼ばれるサブルーチンを直接，呼び出し元に埋め込むことにより，手続き呼び出しの回数を減少させることができ，時間の短縮を計ることができる。例を次に示す。

例

```
      ⋮
CALL SUB(A(I), B(I), C(I), X)
      ⋮
END
SUBROUTINE SUB(D, E, F, Y)
Y＝D** 3＋E** 2＋F
RETURN
END

  ⇩

DO 10 I=1, N
      ⋮
X= A(I)** 3＋B(I)** 2＋C(I)
      ⋮
10 CONTINUE
      ⋮
END
```

本チューニングでは，付録B3に示すように，最も実行頻度の高いMURMANサブルーチン内にTRIDサブルーチンを展開する修正を行った。図18よりMURMANサブルーチンの実行コストの割合は47.81%，TRIDサブルーチンは45120回呼び出されていることがわかる。

[チューニング4]

理解を容易にするため次の例で説明する。

```
      ⋮
DO 10 I=1, N
IF(IQ.EQ.1) THEN
  CALL SUB1(X(I), Y(I), Z(1))
ELSE
  CALL SUB2(X(1), Y(I), Z(I))
ENDIF
10 CONTINUE
      ⋮
END
SUBROUTINE SUB1(A, B, C)
A＝B＋C
RETURN
END
SUBROUTINE SUB2(A, B, C)
A＝B－C
RETURN
END
```

DO 10ループ内でIQは不変であるのでIF文をDOループ外に排出することにより，IF文の実行回数をN回から1回に，サブルーチンSUB1およびSUB2の呼び出し回数をM回(M≦N)から1回に減少させることができる。この修正によりサブルーチンSUB1およびSUB2を配列を処理するサブルーチンに変更する。従って，次のコーディングスタイルに変わる。

```
      ⋮
IF(IQ.EQ.1) THEN
  CALL SUBX1(X, Y, Z, N)
ELSE
  CALL SUBX2(X, Y, Z, N)
ENDIF
      ⋮
END
SUBROUTINE SUBX1(A, B, C, N)
DIMENSION A(N), B(N), C(N)
```

```
      DO  10  I=1, N
      A(I) = B(I) + C(I)
10    CONTINUE
      RETURN
      END
      SUBROUTINE  SUBX2(A,B,C,N)
      DIMENSION  A(N), B(N), C(N)
      DO  10  I=1, N
      A(I) = B(I) - C(I)
10    CONTINUE
      RETURN
      END
```

SWEEPサブルーチン内に付録B 4のDO 30ループがあり，IQCはDOループ内で不変であるのでループ外に排出可能である。従って，DO 30 ループをIF文のみのコーディングに変更して，PHIの配列計算は呼び出されるサブルーチン内で計算されるように変更した。ここではMURMANサブルーチンの呼び出しとPHIの配列計算を行うMURMAN 1 サブルーチンを新たに作成した。MURMAN 1 サブルーチン内ではDO 92ループの外側のループはベクトル化されていない。しかし，例に示したSUBX 1およびSUBX 2サブルーチン内のDOループの形式であればベクトル化可能となるので，その場合にはチューニングの効果もさらに大きくなる。

［チューニング5］

SWEEPサブルーチン内に付録B 5に示すようなループがある。ステートメント番号80からGO TO 80のステートメントまではループになっている。これもチューニング4と同様の修正を行うことによりIF文の実行回数およびサブルーチンの呼び出し回数を1回にできるという効果がある。修正によって新たにMURMAN 2 サブルーチンを作成した。

［チューニング6］

INVRSEサブルーチン内に付録B 6のコーディングがあるので，チューニング4と同様の修正を行った。

［チューニング7］

INVRSEサブルーチン内に付録B 7のコーディングがあるので，チューニング5と同様の修正を行った。

（まとめ）

表6のチューニング表から，チューニング1の修正が最も効果のあることがわかる。ベクトル化率も34.45%から70.11%になり，2倍になっている。チューニング1～7の総合チューニングを施すことによりINVERSEプログラムのGOジョブステップの実行時間は3.907秒となる。しかし，INVERSEプログラムはラインプリンタ出力が多いためにその出力編集に時間がかかるので，AFMESHプログラムに比較してCP実行時間が大きい。AP＋CP実行時間は5.319秒であるので，汎用計算機における実行時間の2.2倍の高速化になる。図22にINVERSEプログラムのチューニング後のベクトル化情報を示す。チューニング後のベクトル化率は71.06%であり，チューニング1以外の修正ではベクトル化率を上げる効果はほとんどない。

このようにベクトル化率が高くなったのにもかかわらず高速化効果が得られないのは次の理由による。

1) 汎用大型計算機で実行しても12秒程度の時間で実行可能な規模の小さいジョブである。
2) 実行コストの比率の高いサブルーチンが一つだけでなく分散している。
3) 頻繁に使われるベクトル長が16であり，スーパコンピュータのベクトル効率を引き出すベクトル長としては短すぎる。
4) 入出力が多い。

5. むすび

本報告において，高速化対象プログラムのベクトル化率を上げ，効率的にベクトル化されるようにチューニングを施した結果，比較的小規模なプログラムながらその効果を確認できた。AFMESHプログラムについては最も実行コストの比率の高いSOLVEサブルーチンに様々なチューニングを施した結果，8倍の高速化を得ることができ，スーパコンピュータに適したアルゴリズムを使用した計算法の利用による高速化の実行においては11倍の高速化が確認できた。また，INVERSEプログラムについては2.2倍の高速化を得た。

AFMESHおよびINVERSEプログラムをスーパコンピュータで効率的に実行するために行われたチ

```
*****
*****  TOTAL EXECUTION ( CPU ) TIME =    0 :  0 ' 31 " 813   (     31813 MSEC)
*****  TOTAL EXECUTION FREQUENCY    =          47784404
*****  TOTAL VECTORIZED RATIO       =    71.06%
*****
```

PROGRAM	FREQUENCY	EXEC COST(%)	V.RATIO	LOOP	V.LOOP	V.LOOP RATIO
MAIN	1	0.10	49.74	85	43	52.29
BLKD						
IBTOD	46	0.00	0.00	0	0	0.00
INDEXR	12	0.00	0.00	1	0	0.00
AIRFOL	8	0.80	32.84	38	17	33.09
CONJ	102	0.21	77.43	4	4	100.00
COSI	273	0.22	0.00	1	0	0.00
CPTOLP	4	0.18	3.49	5	2	3.71
FAIR1	0	0.00	0.00	2	1	0.00
FAIR3	0	0.00	0.00	1	0	0.00
FFORM	1502	12.43	4.49	6	1	14.10
FIGURE	1	0.01	28.93	24	9	29.43
FOUCF	8	0.01	23.53	2	1	23.71
GETCP	8	0.09	88.45	3	3	89.38
GOPRIN	8	0.02	48.63	5	3	49.17
GRAFIC	4	0.02	87.01	12	10	88.73
GTURB	8	0.03	80.92	14	12	83.59
HTURB	0	0.00	0.00	16	12	0.00
HI3A	0	0.00	0.00	21	15	0.00
INTPL	226	0.48	9.01	1	1	9.82
INVRSE	3	2.29	99.21	46	41	99.62
KMYO2	0	0.00	0.00	12	9	0.00
LEXY	4	0.00	83.92	4	4	89.28
MAP	8	2.09	18.93	15	11	19.11
MAXXY	10	0.00	26.06	3	1	26.95
MPMC	0	0.00	0.00	0	0	0.00
MURMAN1	342	18.07	85.83	7	5	88.64
MURMAN2	342	17.30	85.28	7	5	88.27
MURMAN3	222	14.51	89.02	7	5	90.78
MURMAN4	222	13.26	88.99	7	5	90.88
MURMQC	0	0.00	0.00	4	4	0.00
NASHMC	8	0.09	0.00	1	0	0.00
PHILP	22	0.15	61.42	1	1	63.20
RESTRT	8	0.12	45.15	10	8	45.23
SET	82	0.00	0.00	0	0	0.00
SETI	72	0.00	0.00	0	0	0.00
SOLVI	84	1.58	48.12	7	5	49.28
SONIC	4	0.31	0.00	2	0	0.00
SPLIF	171	0.42	0.00	0	0	0.00
SWEEP	342	3.33	99.82	13	13	100.00
SWEEP1	57	1.32	99.30	11	11	100.00
SWEEP2	27	0.42	99.36	8	8	100.00
TRDPHA	0	0.00	0.00	3	3	0.00
TRDPHB	0	0.00	0.00	3	3	0.00
TRDPHC	17760	1.27	0.00	0	0	0.00
TRDPQC	0	0.00	0.00	0	0	0.00
TRID	0	0.00	0.00	3	1	0.00
TRIDI	3444	1.36	0.00	2	0	0.00
TWOFFT	1290	1.87	41.27	4	2	41.76
VALSUM	4	0.02	99.44	7	7	100.00
MURMAN	684	0.82	84.82	8	4	87.37
MURMA1	8720	9.01	96.49	1	1	100.00

図22 チューニング後のINVERSEプログラムのベクトル化情報

ューニングおよびプログラムコーディングにおいて考えられる効率的コーディング法を以下にまとめる。

1) ベクトル化の対象はDOループであるので，ベクトル化率を高めるため，なるべくDOループを用いてコーディングする。

2) 並列実行可能な演算数の増加を図るため，DOループはなるべく統合する。

3) 立ち上がり時間の影響を小さくするため，ベクトル長の拡大を図る。

4) データの記憶域からの取り出しや記憶域への格納に要する時間が最高速で行われるように，効率的な配列参照を行うようにする。

5) 余分なLOAD／STORE命令を削減するため，DOループの統合などにより，テンポラリ化可能な配列はテンポラリ変数におきかえる。

6) オーバヘッドの減少等を図るため，プログラム構造を単純化する。

7) 演算量の減少を図るため，実行回数の多い不変な計算式は予め計算しておき，以後，参照する形式にする。また，実行回数の多い不要計算は削除する。

8) 手続き呼び出し回数の減少を図るため，呼び出し回数の多い関数やサブルーチンは呼び出し元に展開するか，呼び出し回数の減少を図る。

9) DOループ中のベクトル命令の効率化を図るため，IF文のように大きな効率向上の期待できない記述は除去または他の表現に変更する。

10) スカラ部分とベクトル部分を分けて効率的にベクトル化を図る。

高速化対象プログラムはいずれも二次元の問題を扱かったものであり，現在盛んに行われている大規模な三次元問題に対する本報告と同様のチューニングの効果により，さらに効率の良いものとなり，高速化率を高めることが予想できる。高速化対象プログラムは計算機のリソースの制限のきつい時に作成されている。既存プログラムにはリソース制限のために主記憶を十分に使用できない状態で作成されたものが多い。このために，一度の実行で済ませられる計算を主記憶にストアできないために，実際には

何度も計算されている。しかし，現在では計算機のハードウェアおよびソフトウェア技術の発展により最先端のスーパコンピュータでは最大256MBの主記憶使用が可能であり，その能力を十分に活用して高速化を図ることが可能である。また，スーパコンピュータを効果的に活用するためには，並列計算可能という特性を十分に引き出すことが必要である。そのためには，ベクトル化されたDOループのベクトル長の拡大およびループ内の演算の並列処理性の増大を図ることが必要である。したがって，大規模なプログラムの場合には，256MBの主記憶容量およびスーパコンピュータの並列処理能力がより活用できるため，数十倍の高速化は十分に可能である。

なお，チューニングに用いたコンパイラは開発途上のものであり，本報告でチューニングの対象としたループのうちのいくつかは正式に提供されるコンパイラでは自動ベクトル化の対象となること，およびコンパイラのオブジェクトもさらに高速化されていることを付け加えておく。

最後に，高速化対象プログラムとして，AFMESHプログラムは広瀬直喜技官，河合伸坦技官（空気力学第二部）に提供していただき，INVERSEプログラムは石黒登美子技官（計算センタ），神谷信彦技官（新型航空機研究グループ），河合伸坦技官（空気力学第二部）に提供していただいた。

また，高速化対象プログラムのSXシステムにおけるチューニング作業を担当された日本電気株式会社の大道学氏，高畠典克氏，並びに関係各部門の方々に紙上を借りて感謝の意を表する。

参考文献

1) 古勝紀誠，渡辺　貞，近藤良三；スーパコンピュータSXシステム／高出力半導体レーザ，日経エレクトロニクス　1984 11-19

2) 日本電気；NECスーパコンピュータSX－1／SX-2演算プロセッサ機能説明書GBC10-1 1985年9月

3) 日本電気；SXソフトウェアFORTRAN 77言語説明書GGB11-1，1985年6月

4) 日本電気；SXソフトウェアFORTRAN77，77/SXプログラミング手引書GGB12-1 1985年6月

5) 広瀬直喜，河合伸坦；AFMESH 利用の手引き，航空宇宙技術研究所報告（出版予定）

6) 広瀬直喜，河合伸坦，伊沢隆男，菊池路子；遷音速翼型解析法のための格子形成コードAFMESHの開発，第13回日本航空宇宙学会年会講演会講演集p.158～161，昭和57年4月

7) Thompson, J.F., Thames, F.C. & Mastin, C.W.; Boundary-Fitted Curvilinear Coordinate Systems For Solution of Partial Differential Equations on Fields Containing Any Number of Arbitrary Two-Dimensional Bodies, NASA CR-2729, (1977)

8) Jerry C. South Jr. and James D.Keller; Vector Processor Algorithms for Transonic Flow Calculations AIAA JOURNAL VOL.18, No.7 79-1457R

9) N.Duane Melson and James D.Keller; Use of CYBER 203 and CYBER 205 Computer for Three-Dimensional Transonic Flow Calculations

10) 石黒登美子，神谷信彦，河合伸坦；完全ポテンシャル流の遷音速翼型設計Ⅰ，数値解法とその適用計算例，航空宇宙技術研究所報告　TR-672

11) 石黒登美子，神谷信彦，河合伸坦，小口慶子；完全ポテンシャル流の遷音速翼型設計Ⅱ，汎用プログラムの開発，航空宇宙技術研究所報告TR-673

12) F.Bauer, P.Garabedian, D.Korn & A. Jameson; Supercritical Wing Sections Ⅱ, Lecture Notes in Econom. and Math. Systems, VOL. 108, Springer-Verlag (1975)

13) F. Bauer, P. Garabedian & D. Korn; Super Critical Wing Sections (Ⅲ), Lecture Notes in Econom. and Math. Systems, VOL.150, Springer-Verlag (1977)

付録A

A 1 DOループの統合

A 1 (1)

```
                    DO 10200 IDXJ = JS , JE
V------------>      |  XXS(IDXJ) = 0.5 * (X(IDXJ+1,KSM) - X(IDXJ-1,KSM))
|                   |  YXS(IDXJ) = 0.5 * (Y(IDXJ+1,KSM) - Y(IDXJ-1,KSM))
V------------ 10200 CONTINUE
                    DO 10300 IDXJ = JS , JE
V------------>      |  U(IDXJ) = DS * DS
|                   |  W(IDXJ) = XXS(IDXJ) * XXS(IDXJ) + YXS(IDXJ) * YXS(IDXJ)
|                   |  V(IDXJ) = 2.0 * DS * SQRT(W(IDXJ)) * COS(THETA)
V------------ 10300 CONTINUE
                    DO 10400 IDXJ = JS , JE
V------------>      |  XETS(IDXJ) = DS * (-XXS(IDXJ) * COS(THETA) -
|                  1|                    YXS(IDXJ) * SIN(THETA)) /
|                  2|                SQRT( W(IDXJ))
V------------ 10400 CONTINUE
                    DO 10450 IDXJ = JS , JE
V------------>      |  YETS(IDXJ) = DS * (-YXS(IDXJ) * COS(THETA) +
|                  1|                    XXS(IDXJ) * SIN(THETA)) /
|                  2|                SQRT( W(IDXJ))
V------------ 10450 CONTINUE
```

⇩ (実行回数：1, ベクトル長：101)

```
                    DO 10200 IDXJ = JS , JE
V------------>      |  XXS(IDXJ) = 0.5 * (X(IDXJ+1,KSM) - X(IDXJ-1,KSM))
|                   |  YXS(IDXJ) = 0.5 * (Y(IDXJ+1,KSM) - Y(IDXJ-1,KSM))
|                   |  U(IDXJ) = DS * DS
|                   |  W(IDXJ) = XXS(IDXJ) * XXS(IDXJ) + YXS(IDXJ) * YXS(IDXJ)
|                   |  V(IDXJ) = 2.0 * DS * SQRT(W(IDXJ)) * COS(THETA)
|                   |  XETS(IDXJ) = DS * (-XXS(IDXJ) * COS(THETA) -
|                  1|                    YXS(IDXJ) * SIN(THETA)) /
|                  2|                SQRT( W(IDXJ))
|                   |  YETS(IDXJ) = DS * (-YXS(IDXJ) * COS(THETA) +
|                  1|                    XXS(IDXJ) * SIN(THETA)) /
|                  2|                SQRT( W(IDXJ))
V------------ 10200 CONTINUE
```

A 1 (2)

```
                    DO 10470 IDXJ = JS , JE
V------>            |  JACOS(IDXJ) = XXS(IDXJ) * YETS(IDXJ) - XETS(IDXJ) * YXS(IDXJ)
V------ 10470 CONTINUE
                    DO 10480 IDXJ = JS , JE
V------>            |  XXS2(IDXJ) = X(IDXJ+1,KSM) - 2.0 * X(IDXJ,KSM) +
|                  1|               X(IDXJ-1,KSM)
|                   |  XXSETA(IDXJ) = (XETS(IDXJ+1) - XETS(IDXJ-1)) * 0.5
|                   |  YXS2(IDXJ) = Y(IDXJ+1,KSM) - 2.0 * Y(IDXJ,KSM) +
|                  1|               Y(IDXJ-1,KSM)
|                   |  YXSETA(IDXJ) = (YETS(IDXJ+1) - YETS(IDXJ-1)) * 0.5
|                   |
|                   |  JACOS(IDXJ) = 1.0 / JACOS(IDXJ)
|                   |  JACOS(IDXJ) = JACOS(IDXJ) ** 3
V------ 10480 CONTINUE
```

⇩ (実行回数：1, ベクトル長：101)

```
                    DO 10470 IDXJ = JS , JE
V------>            |  XXS2(IDXJ) = X(IDXJ+1,KSM) - 2.0 * X(IDXJ,KSM) +
|                  1|               X(IDXJ-1,KSM)
|                   |  XXSETA(IDXJ) = (XETS(IDXJ+1) - XETS(IDXJ-1)) * 0.5
|                   |  YXS2(IDXJ) = Y(IDXJ+1,KSM) - 2.0 * Y(IDXJ,KSM) +
|                  1|               Y(IDXJ-1,KSM)
|                   |  YXSETA(IDXJ) = (YETS(IDXJ+1) - YETS(IDXJ-1)) * 0.5
|                   |  JACOS(IDXJ) =1.0/(XXS(IDXJ)*YETS(IDXJ)-XETS(IDXJ)*YXS(IDXJ))**3
V------ 10470 CONTINUE
```

A 1 (3)

```
9----> 1000 CONTINUE
|            ITER = ITER + 1
|            RMAX = 0.0
|            RSUM = 0.0
|            IF (.NOT. POIS)                                  GO TO 1050
|            DO 11020 IDXJ = JS , JE
| V-->       |  XETS2(IDXJ) = (-7.0 * X(IDXJ,KSM) + 8.0 * X(IDXJ,KS)
| |          ||                - X(IDXJ,KS1)) * 0.5 - 3.0 * XETS(IDXJ)
| |          |  YETS2(IDXJ) = (-7.0 * Y(IDXJ,KSM) + 8.0 * Y(IDXJ,KS)
| |          ||                - Y(IDXJ,KS1)) * 0.5 - 3.0 * YETS(IDXJ)
| |          |  "
| |          |  R1(IDXJ) = (-1.0) * (U(IDXJ) * XXS2(IDXJ) + V(IDXJ) * XXSETA(IDXJ)
| |          ||                                            + W(IDXJ) * XETS2(IDXJ))
| |          |  R2(IDXJ) = (-1.0) * (U(IDXJ) * YXS2(IDXJ) + V(IDXJ) * YXSETA(IDXJ)
| |          ||                                            + W(IDXJ) * YETS2(IDXJ))
| V--- 11020 CONTINUE
|            DO 11030 IDXJ = JS , JE
| V-->       |  E(IDXJ)  = JACOS(IDXJ) * (YETS(IDXJ) * R1(IDXJ) -
| |          ||                          XETS(IDXJ) * R2(IDXJ))
| |          |  F(IDXJ)  = JACOS(IDXJ) * (-YXS(IDXJ) * R1(IDXJ) +
| |          ||                          XXS(IDXJ) * R2(IDXJ))
| V--- 11030 CONTINUE
|            DO 11040 IDXJ = JS , JE
| V-->       |  R1(IDXJ) = E(IDXJ) - P1(IDXJ)
| |          |  R2(IDXJ) = F(IDXJ) - Q1(IDXJ)
| V--- 11040 CONTINUE
|            "
|            DO 11050 IDXJ = JS , JE
| V-->       |  XX1(IDXJ)  = OMEGP1 * ABS(R1(IDXJ))
| |          |  YX1(IDXJ)  = OMEGQ1 * ABS(R2(IDXJ))
| V--- 11050 CONTINUE
|            DO 1100 J = JS , JE
| V-->       |  XETA(J) = PLIM * MAX(ABS(P1(J)),1.0)
| |          |  XX1(J)  = MIN(XX1(J),XETA(J))
| |          |  YETA(J) = QLIM * MAX(ABS(Q1(J)),1.0)
| |          |  YX1(J)  = MIN(YX1(J),YETA(J))
| V---  1100 CONTINUE
|            "
|            DO 11130 IDXJ = JS , JE
| V-->       |  P1(IDXJ)   = P1(IDXJ) + SIGN(XX1(IDXJ),R1(IDXJ))
| |          |  Q1(IDXJ)   = Q1(IDXJ) + SIGN(YX1(IDXJ),R2(IDXJ))
| V--- 11130 CONTINUE
|            "
|            DO 11140 IDXJ = JS , JE
| V-->       |  G(IDXJ)    = 1.0
| |          |  LMSK(IDXJ) = P1(IDXJ) .EQ. 0.0
| V--- 11140 CONTINUE
|            "
|            DO 11150 IDXJ = JS , JE
| V-->       |  IF (LMSK(IDXJ)) THEN
| |          |  |  E(IDXJ) = G(IDXJ)
| |          |  ELSE
| |          |  |  E(IDXJ) = P1(IDXJ)
| |          |  END IF
| V--- 11150 CONTINUE
|            DO 11160 IDXJ = JS , JE
| V->        |  R1(IDXJ)   = SIGN(XX1(IDXJ),R1(IDXJ)) / E(IDXJ)
| |          |  LMSK(IDXJ) = Q1(IDXJ) .EQ. 0.0
| V-- 11160 CONTINUE
|            DO 11170 IDXJ = JS , JE
| V->        |  IF (LMSK(IDXJ)) THEN
| |          |  |  F(IDXJ) = G(IDXJ)
| |          |  ELSE
| |          |  |  F(IDXJ) = Q1(IDXJ)
| |          |  END IF
| V-- 11170 CONTINUE
|            DO 11180 IDXJ = JS , JE
| V->        |  R2(IDXJ)   = SIGN(YX1(IDXJ),R2(IDXJ)) / F(IDXJ)
| V-- 11180 CONTINUE
```

⇩ (実行回数：451 / ベクトル長：101)

(I)

```
 1000 CONTINUE
      ITER = ITER + 1
      RMAX = 0.0
      RSUM = 0.0
      IF (.NOT. POIS)                                   GO TO 1050
      DO 11020 IDXJ = JS , JE
V---- XETS2(IDXJ) = (-7.0 * X(IDXJ,KSM) + 8.0 * X(IDXJ,KS)
     1                    - X(IDXJ,KS1)) * 0.5 - 3.0 * XETS(IDXJ)
      YETS2(IDXJ) = (-7.0 * Y(IDXJ,KSM) + 8.0 * Y(IDXJ,KS)
     1                    - Y(IDXJ,KS1)) * 0.5 - 3.0 * YETS(IDXJ)
      R1(IDXJ) = (-1.0) * (U(IDXJ) * XXS2(IDXJ) + V(IDXJ) * XYSET.(IDXJ)
     1                                           + W(IDXJ) * XETS2(IDXJ))
      R2(IDXJ) = (-1.0) * (U(IDXJ) * YXS2(IDXJ) + V(IDXJ) * YXSET.(IDXJ)
     1                                           + W(IDXJ) * YETS2(IDXJ))
      E(IDXJ)  = JACOS(IDXJ) * (YETS(IDXJ) * R1(IDXJ) -
     1                          XETS(IDXJ) * R2(IDXJ))
      F(IDXJ)  = JACOS(IDXJ) * (-YXS(IDXJ) * R1(IDXJ) +
     1                          XXS(IDXJ) * R2(IDXJ))
      R1(IDXJ) = E(IDXJ) - P1(IDXJ)
      R2(IDXJ) = F(IDXJ) - Q1(IDXJ)
      XX1(IDXJ)  = OMEGP1 * ABS(R1(IDXJ))
      YX1(IDXJ)  = OMEGQ1 * ABS(R2(IDXJ))
      XETA(IDXJ) = PLIM * MAX(ABS(P1(IDXJ)),1.0)
      XX1(IDXJ)  = MIN(XX1(IDXJ),XETA(IDXJ))
      YETA(IDXJ) = QLIM * MAX(ABS(Q1(IDXJ)),1.0)
      YX1(IDXJ)  = MIN(YX1(IDXJ),YETA(IDXJ))
      P1(IDXJ)   = P1(IDXJ) + SIGN(XX1(IDXJ),R1(IDXJ))
      Q1(IDXJ)   = Q1(IDXJ) + SIGN(YX1(IDXJ),R2(IDXJ))
      G(IDXJ)    = 1.0
      LMSK(IDXJ) = P1(IDXJ) .EQ. 0.0
      IF (LMSK(IDXJ)) THEN
      E(IDXJ) = G(IDXJ)
      ELSE
      E(IDXJ) = P1(IDXJ)
      END IF
      R1(IDXJ)   = SIGN(XX1(IDXJ),R1(IDXJ)) / E(IDXJ)
      LMSK(IDXJ) = Q1(IDXJ) .EQ. 0.0
      IF (LMSK(IDXJ)) THEN
      F(IDXJ) = G(IDXJ)
      ELSE
      F(IDXJ) = Q1(IDXJ)
      END IF
      R2(IDXJ)   = SIGN(YX1(IDXJ),R2(IDXJ)) / F(IDXJ)
L---- 11020 CONTINUE
```

(II)

```
|                 |  G(IDXJ)    = 1.0
|                 |  LMSK(IDXJ) = P1(IDXJ) .EQ. 0.0
|                 |  IF (LMSK(IDXJ)) THEN
|                 |  |  E(IDXJ) = G(IDXJ)
|                 |  ELSE
|                 |  |  E(IDXJ) = P1(IDXJ)
|                 |  END IF
|                 |  R1(IDXJ)   = SIGN(XX1(IDXJ),R1IDXJ) / E(IDXJ)
|                 |  LMSK(IDXJ) = Q1(IDXJ) .EQ. 0.0
|                 |  IF (LMSK(IDXJ)) THEN
|                 |  |  F(IDXJ) = G(IDXJ)
|                 |  ELSE
|                 |  |  F(IDXJ) = Q1(IDXJ)
|                 |  END IF
|                 |  R2(IDXJ)   = SIGN(YX1(IDXJ),R2IDXJ) / F(IDXJ)
```

⇩ (実行回数：451, ベクトル長：101)

```
|                 |  IF (P1(IDXJ).NE.0.0) THEN
|                 |  |  R1(IDXJ)   = SIGN(XX1(IDXJ),R1IDXJ) / P1(IDXJ)
|                 |  ELSE
|                 |  |  R1(IDXJ)   = SIGN(XX1(IDXJ),R1IDXJ) / 1.0
|                 |  END IF
|                 |  IF (Q1(IDXJ).NE.0.0) THEN
|                 |  |  R2(IDXJ)   = SIGN(YX1(IDXJ),R2IDXJ) / Q1(IDXJ)
|                 |  ELSE
|                 |  |  R2(IDXJ)   = SIGN(YX1(IDXJ),R2IDXJ) / 1.0
|                 |  END IF
```

```
                DO 11020 IDXJ = JS , JE
V-->            |  XETS2(IDXJ) = (-7.0 * X(IDXJ,KSM) + 8.0 * X(IDXJ,KS)
|           11|                              - X(IDXJ,KS1)) * 0.5 - 3.0 * XETS(IDXJ)
|               |  YETS2(IDXJ) = (-7.0 * Y(IDXJ,KSM) + 8.0 * Y(IDXJ,KS)
|           11|                              - Y(IDXJ,KS1)) * 0.5 - 3.0 * YETS(IDXJ)
|               |  R1IDXJ = (-1.0) * (U(IDXJ) * XXS2(IDXJ) + V(IDXJ) * XXSETA(IDXJ  )
|           11|                                                    + W(IDXJ) * XETS2(IDXJ))
|               |  R2IDXJ = (-1.0) * (U(IDXJ) * YXS2(IDXJ) + V(IDXJ) * YXSETA(IDXJ  )
|           11|                                                    + W(IDXJ) * YETS2(IDXJ))
|               |  E(IDXJ)  = JACOS(IDXJ) * (YETS(IDXJ) * R1IDXJ -
|           11|                                          XETS(IDXJ) * R2IDXJ)
|               |  F(IDXJ)  = JACOS(IDXJ) * (-YXS(IDXJ) * R1IDXJ +
|           11|                                          XXS(IDXJ) * R2IDXJ)
|               |  R1IDXJ = E(IDXJ) - P1(IDXJ)
|               |  R2IDXJ = F(IDXJ) - Q1(IDXJ)
|               |  XX1IDXJ  = OMEGP1 * ABS(R1IDXJ)
|               |  YX1IDXJ  = OMEGQ1 * ABS(R2IDXJ)
|               |  XETA(IDXJ) = PLIM * MAX(ABS(P1(IDXJ)),1.0)
|               |  XX1(IDXJ)  = MIN(XX1IDXJ,XETA(IDXJ))
|               |  YETA(IDXJ) = QLIM * MAX(ABS(Q1(IDXJ)),1.0)
|               |  YX1(IDXJ)  = MIN(YX1IDXJ,YETA(IDXJ))
|               |  P1(IDXJ)   = P1(IDXJ) + SIGN(XX1(IDXJ),R1IDXJ)
|               |  Q1(IDXJ)   = Q1(IDXJ) + SIGN(YX1(IDXJ),R2IDXJ)
|               |  IF (P1(IDXJ).NE.0.0) THEN
|               |  |  R1(IDXJ)   = SIGN(XX1(IDXJ),R1IDXJ) / P1(IDXJ)
|               |  ELSE
|               |  |  R1(IDXJ)   = SIGN(XX1(IDXJ),R1IDXJ) / 1.0
|               |  END IF
|               |  IF (Q1(IDXJ).NE.0.0) THEN
|               |  |  R2(IDXJ)   = SIGN(YX1(IDXJ),R2IDXJ) / Q1(IDXJ)
|               |  ELSE
|               |  |  R2(IDXJ)   = SIGN(YX1(IDXJ),R2IDXJ) / 1.0
|               |  END IF
|               |  "
V-- 11020 CONTINUE
```

⇩ (実行回数：451 / ベクトル長：101)

(Ⅲ)

```
                DO 11020 IDXJ = JS , JE
V->             |  R1IDXJ = (-1.0) * (U(IDXJ) * XXS2(IDXJ) + V(IDXJ) * XXSETA(IDXJ  )
|           11|     + W(IDXJ) *((-7.0 * X(IDXJ,KSM) + 8.0 * X(IDXJ,KS)
|         2|           - X(IDXJ,KS1)) * 0.5 - 3.0 * XETS(IDXJ)))
|               |  R2IDXJ = (-1.0) * (U(IDXJ) * YXS2(IDXJ) + V(IDXJ) * YXSETA(IDXJ  )
|           11|           + W(IDXJ) * ((-7.0 * Y(IDXJ,KSM) + 8.0 * Y(IDXJ,KS)
|         2|                       - Y(IDXJ,KS1)) * 0.5 - 3.0 * YETS(IDXJ)))
|               |  E(IDXJ)  = JACOS(IDXJ) * (YETS(IDXJ) * R1IDXJ -
|           11|                                          XETS(IDXJ) * R2IDXJ)
|               |  F(IDXJ)  = JACOS(IDXJ) * (-YXS(IDXJ) * R1IDXJ +
|           11|                                          XXS(IDXJ) * R2IDXJ)
|               |  R1IDXJ = E(IDXJ) - P1(IDXJ)
|               |  R2IDXJ = F(IDXJ) - Q1(IDXJ)
|               |  XX1IDXJ  = OMEGP1 * ABS(R1IDXJ)
|               |  YX1IDXJ  = OMEGQ1 * ABS(R2IDXJ)
|               |  XETA(IDXJ) = PLIM * MAX(ABS(P1(IDXJ)),1.0)
|               |  XX1IDXJ  = MIN(XX1IDXJ,XETA(IDXJ))
|               |  YETA(IDXJ) = QLIM * MAX(ABS(Q1(IDXJ)),1.0)
|               |  YX1IDXJ  = MIN(YX1IDXJ,YETA(IDXJ))
|               |  P1(IDXJ)   = P1(IDXJ) + SIGN(XX1IDXJ,R1IDXJ)
|               |  Q1(IDXJ)   = Q1(IDXJ) + SIGN(YX1IDXJ,R2IDXJ)
|               |  R1(IDXJ)   = SIGN(XX1IDXJ,R1IDXJ)
|               |  IF (P1(IDXJ).NE.0.0) THEN
|               |  |  R1(IDXJ)   = R1(IDXJ) / P1(IDXJ)
|               |  END IF
|               |  R2(IDXJ)   = SIGN(YX1IDXJ,R2IDXJ)
|               |  IF (Q1(IDXJ).NE.0.0) THEN
|               |  |  R2(IDXJ)   = R2(IDXJ) / Q1(IDXJ)
|               |  END IF
|               |  "
V- 11020 CONTINUE
```

A 1 (4)

```
|                       RP(ITER) = ABS(XX1(JS))
|                       DO 11400 IDXJ = JS+1 , JE
| V---------->          |  RP(ITER) = MAX(RP(ITER) , ABS(XX1(IDXJ)))
| V---------- 11400 CONTINUE
|                       RQ(ITER) = ABS(YX1(JS))
|                       DO 11500 IDXJ = JS+1 , JE
| V---------->          |  RQ(ITER) = MAX(RQ(ITER) , ABS(YX1(IDXJ)))
| V---------- 11500 CONTINUE
|                       RPQS(ITER) = 0.0
|                       DO 11600 IDXJ = JS , JE
| V---------->          |  RPQS(ITER) = RPQS(ITER) + ABS(XX1(IDXJ)) + ABS(YX1(IDXJ))
| V---------- 11600 CONTINUE
```

⇩ (実行回数：451 / ベクトル長： 99)

```
                    RP(ITER) = ABS(XXI(JS))
                    RQ(ITER) = ABS(YXI(JS))
                    DO 11400 IDXJ = JS+1 , JE
V------------>      |  RP(ITER) = MAX(RP(ITER) , ABS(XXI(IDXJ)))
|                   |  RQ(ITER) = MAX(RQ(ITER) , ABS(YXI(IDXJ)))
V------------ 11400 CONTINUE
                    RPQS(ITER) = 0.0
                    DO 11600 IDXJ = JS , JE
V------------>      |  RPQS(ITER) = RPQS(ITER) + ABS(XXI(IDXJ)) + ABS(YXI(IDXJ))
V------------ 11600 CONTINUE
```

A 1 (5)

```
          DO 2000 K = KSTART , KEND
;-->      |  DO 11950 IDXJ = JS , JE
V->       |  |  XXI(IDXJ)  = 0.5 * (X(IDXJ+1,K) - X(IDXJ-1,K))
|         |  |  YXI(IDXJ)  = 0.5 * (Y(IDXJ+1,K) - Y(IDXJ-1,K))
|         |  |  XETA(IDXJ) = 0.5 * (X(IDXJ,K+1) - X(IDXJ,K-1))
|         |  |  YETA(IDXJ) = 0.5 * (Y(IDXJ,K+1) - Y(IDXJ,K-1))
V-- 11950 |  CONTINUE
          |  DO 11960 IDXJ = JS , JE
V->       |  |  SUP(IDXJ)  = XETA(IDXJ) * XETA(IDXJ) + YETA(IDXJ) * YETA(IDXJ)
|         |  |  GD(IDXJ)   = XXI(IDXJ) * XXI(IDXJ) + YXI(IDXJ) * YXI(IDXJ)
|         |  |  SUB(IDXJ)  = SUP(IDXJ)
|         |  |  DIAG(IDXJ) = (-2.0) * (SUP(IDXJ) + GD(IDXJ))
|         |  |  BD(IDXJ)   =    0.5  * ((X(IDXJ+1,K) - X(IDXJ-1,K)) *
|       1 |  |                          (X(IDXJ,K+1) - X(IDXJ,K-1))
|       2 |  |                        + (Y(IDXJ+1,K) - Y(IDXJ-1,K)) *
|       3 |  |                          (Y(IDXJ,K+1) - Y(IDXJ,K-1)))
V-- 11960 |  CONTINUE
          |
          |  DO 11970 IDXJ = JS , JE
V->       |  |  JACOB(IDXJ) = XXI(IDXJ) * YETA(IDXJ) - YXI(IDXJ) * XETA(IDXJ)
V-- 11970 |  CONTINUE
          |  DO 11980 IDXJ = JS , JE
V->       |  |  F(IDXJ) =      BD(IDXJ)
|       1 |  |               * 0.25 * (X(IDXJ+1,K+1) - X(IDXJ+1,K-1) -
|       2 |  |                          X(IDXJ-1,K+1) + X(IDXJ-1,K-1))
|       3 |  |               + (-GD(IDXJ)) * (X(IDXJ,K+1) + X(IDXJ,K-1))
|       4 |  |               + (-JACOB(IDXJ) ** 2)
|       5 |  |               * (P1(IDXJ) * EXP(-A*FLOAT(K-(KSTART-1))) * XXI(IDXJ) +
|       6 |  |                  Q1(IDXJ) * EXP(-B*FLOAT(K-(KSTART-1))) * XETA(IDXJ))
|         |  |  G(IDXJ) =      BD(IDXJ)
|       1 |  |               * 0.25 * (Y(IDXJ+1,K+1) - Y(IDXJ+1,K-1) -
|       2 |  |                          Y(IDXJ-1,K+1) + Y(IDXJ-1,K-1))
|       3 |  |               + (-GD(IDXJ)) * (Y(IDXJ,K+1) + Y(IDXJ,K-1))
|       4 |  |               + (-JACOB(IDXJ) ** 2)
|       5 |  |               * (P1(IDXJ) * EXP(-A*FLOAT(K-(KSTART-1))) * YXI(IDXJ) +
|       6 |  |                  Q1(IDXJ) * EXP(-B*FLOAT(K-(KSTART-1))) * YETA(IDXJ))
V-- 11980 |  CONTINUE
```

⇩ (実行回数：56112, ベクトル長：101)

(I)

```
          DO 2000 K = KSTART , KEND
-->       |  DO 11950 IDXJ = JS , JE
V->       |  |  XXI(IDXJ)  = 0.5 * (X(IDXJ+1,K) - X(IDXJ-1,K))
|         |  |  YXI(IDXJ)  = 0.5 * (Y(IDXJ+1,K) - Y(IDXJ-1,K))
|         |  |  XETA(IDXJ) = 0.5 * (X(IDXJ,K+1) - X(IDXJ,K-1))
|         |  |  YETA(IDXJ) = 0.5 * (Y(IDXJ,K+1) - Y(IDXJ,K-1))
|         |  |  SUP(IDXJ)  = XETA(IDXJ) * XETA(IDXJ) + YETA(IDXJ) * YETA(IDXJ)
|         |  |  GD(IDXJ)   = XXI(IDXJ) * XXI(IDXJ) + YXI(IDXJ) * YXI(IDXJ)
|         |  |  SUB(IDXJ)  = SUP(IDXJ)
|         |  |  DIAG(IDXJ) = (-2.0) * (SUP(IDXJ) + GD(IDXJ))
|         |  |  BD(IDXJ)   =    0.5  * ((X(IDXJ+1,K) - X(IDXJ-1,K)) *
|       1 |  |                          (X(IDXJ,K+1) - X(IDXJ,K-1))
|       2 |  |                        + (Y(IDXJ+1,K) - Y(IDXJ-1,K)) *
|       3 |  |                          (Y(IDXJ,K+1) - Y(IDXJ,K-1)))
|         |  |  JACOB(IDXJ) = XXI(IDXJ) * YETA(IDXJ) - YXI(IDXJ) * XETA(IDXJ)
|         |  |  F(IDXJ) =      BD(IDXJ)
|       1 |  |               * 0.25 * (X(IDXJ+1,K+1) - X(IDXJ+1,K-1) -
|       2 |  |                          X(IDXJ-1,K+1) + X(IDXJ-1,K-1))
|       3 |  |               + (-GD(IDXJ)) * (X(IDXJ,K+1) + X(IDXJ,K-1))
|       4 |  |               + (-JACOB(IDXJ) ** 2)
|       5 |  |               * (P1(IDXJ) * EXP(-A*FLOAT(K-(KSTART-1))) * XXI(IDXJ) +
|       6 |  |                  Q1(IDXJ) * EXP(-B*FLOAT(K-(KSTART-1))) * XETA(IDXJ))
|         |  |  G(IDXJ) =      BD(IDXJ)
|       1 |  |               * 0.25 * (Y(IDXJ+1,K+1) - Y(IDXJ+1,K-1) -
|       2 |  |                          Y(IDXJ-1,K+1) + Y(IDXJ-1,K-1))
|       3 |  |               + (-GD(IDXJ)) * (Y(IDXJ,K+1) + Y(IDXJ,K-1))
|       4 |  |               + (-JACOB(IDXJ) ** 2)
|       5 |  |               * (P1(IDXJ) * EXP(-A*FLOAT(K-(KSTART-1))) * YXI(IDXJ) +
|       6 |  |                  Q1(IDXJ) * EXP(-B*FLOAT(K-(KSTART-1))) * YETA(IDXJ))
V-- 11950 |  CONTINUE
```

(II)

```
                DO 2000 K = KSTART , KEND
--->            |  DO 11950 IDXJ = JS , JE
V->             |  |  XX(IDXJ  = 0.5 * (X(IDXJ+1,K) - X(IDXJ-1,K))
|               |  |  YX(IDXJ  = 0.5 * (Y(IDXJ+1,K) - Y(IDXJ-1,K))
|               |  |  XETA(IDXJ = 0.5 * (X(IDXJ,K+1) - X(IDXJ,K-1))
|               |  |  YETA(IDXJ = 0.5 * (Y(IDXJ,K+1) - Y(IDXJ,K-1))
|               |  |  SUP(IDXJ)  = XETA(IDXJ * XETA(IDXJ + YETA(IDXJ * YETA(IDXJ
|               |  |  GD(IDXJ   = XX(IDXJ * XX(IDXJ + YX(IDXJ * YX(IDXJ
|               |  |  SUB(IDXJ)  = SUP(IDXJ)
|               |  |  DIAG(IDXJ) = (-2.0) * (SUP(IDXJ) + GD(IDXJ)
|               |  |  BD(IDXJ   =    0.5  * ((X(IDXJ+1,K) - X(IDXJ-1,K)) *
|             1 |  |                              (X(IDXJ,K+1) - X(IDXJ,K-1))
|             2 |  |                          + (Y(IDXJ+1,K) - Y(IDXJ-1,K)) *
|             3 |  |                              (Y(IDXJ,K+1) - Y(IDXJ,K-1)))
|               |  |  JACOB(IDXJ) = XX(IDXJ * YETA(IDXJ - YX(IDXJ * XETA(IDXJ
|               |  |  F(IDXJ) =     BD(IDXJ
|             1 |  |                * 0.25 * (X(IDXJ+1,K+1) - X(IDXJ+1,K-1) -
|             2 |  |                          X(IDXJ-1,K+1) + X(IDXJ-1,K-1))
|             3 |  |                + (-GD(IDXJ) * (X(IDXJ,K+1) + X(IDXJ,K-1))
|             4 |  |                + (-JACOB(IDXJ) ** 2)
|             5 |  |                * (P1(IDXJ) * EXP(-A*FLOAT(K-(KSTART-1))) * XX(IDXJ +
|             6 |  |                  Q1(IDXJ) * EXP(-B*FLOAT(K-(KSTART-1))) * XETA(IDXJ)
|               |  |  G(IDXJ) =     BD(IDXJ
|             1 |  |                * 0.25 * (Y(IDXJ+1,K+1) - Y(IDXJ+1,K-1) -
|             2 |  |                          Y(IDXJ-1,K+1) + Y(IDXJ-1,K-1))
|             3 |  |                + (-GD(IDXJ) * (Y(IDXJ,K+1) + Y(IDXJ,K-1))
|             4 |  |                + (-JACOB(IDXJ) ** 2)
|             5 |  |                * (P1(IDXJ) * EXP(-A*FLOAT(K-(KSTART-1))) * YX(IDXJ +
|             6 |  |                  Q1(IDXJ) * EXP(-B*FLOAT(K-(KSTART-1))) * YETA(IDXJ)
V-- 11950     |  CONTINUE
```

⇩ (実行回数：56112, ベクトル長：101)

(III)

```
                DIMENSION      XFLOAT(84)

                DO 505 K=KSTART,KEND
V--->           |  XFLOAT(K)=FLOAT(K-(KSTART-1))
V---  505 CONTINUE
                DO 2000 K = KSTART , KEND
--->            |  DO 11950 IDXJ = JS , JE
V->             |  |  XX(IDXJ  = 0.5 * (X(IDXJ+1,K) - X(IDXJ-1,K))
|               |  |  YX(IDXJ  = 0.5 * (Y(IDXJ+1,K) - Y(IDXJ-1,K))
|               |  |  XETA(IDXJ = 0.5 * (X(IDXJ,K+1) - X(IDXJ,K-1))
|               |  |  YETA(IDXJ = 0.5 * (Y(IDXJ,K+1) - Y(IDXJ,K-1))
|               |  |  SUP(IDXJ)  = XETA(IDXJ * XETA(IDXJ + YETA(IDXJ * YETA(IDXJ
|               |  |  GD(IDXJ   = XX(IDXJ * XX(IDXJ + YX(IDXJ * YX(IDXJ
|               |  |  SUB(IDXJ)  = SUP(IDXJ)
|               |  |  DIAG(IDXJ) = (-2.0) * (SUP(IDXJ) + GD(IDXJ)
|               |  |  BD(IDXJ   =    0.5  * ((X(IDXJ+1,K) - X(IDXJ-1,K)) *
|             1 |  |                              (X(IDXJ,K+1) - X(IDXJ,K-1))
|             2 |  |                          + (Y(IDXJ+1,K) - Y(IDXJ-1,K)) *
|             3 |  |                              (Y(IDXJ,K+1) - Y(IDXJ,K-1)))
|               |  |  JACOB(IDXJ) = XX(IDXJ * YETA(IDXJ - YX(IDXJ * XETA(IDXJ
|               |  |  F(IDXJ) =     BD(IDXJ
|             1 |  |                * 0.25 * (X(IDXJ+1,K+1) - X(IDXJ+1,K-1) -
|             2 |  |                          X(IDXJ-1,K+1) + X(IDXJ-1,K-1))
|             3 |  |                + (-GD(IDXJ) * (X(IDXJ,K+1) + X(IDXJ,K-1))
|             4 |  |                + (-JACOB(IDXJ) ** 2)
|             5 |  |                * (P1(IDXJ) * EXP(-A*XFLOAT(K)) * XX(IDXJ +
|             6 |  |                  Q1(IDXJ) * EXP(-B*XFLOAT(K)) * XETA(IDXJ)
|               |  |  G(IDXJ) =     BD(IDXJ
|             1 |  |                * 0.25 * (Y(IDXJ+1,K+1) - Y(IDXJ+1,K-1) -
|             2 |  |                          Y(IDXJ-1,K+1) + Y(IDXJ-1,K-1))
|             3 |  |                + (-GD(IDXJ) * (Y(IDXJ,K+1) + Y(IDXJ,K-1))
|             4 |  |                + (-JACOB(IDXJ) ** 2)
|             5 |  |                * (P1(IDXJ) * EXP(-A*XFLOAT(K)) * YX(IDXJ +
|             6 |  |                  Q1(IDXJ) * EXP(-B*XFLOAT(K)) * YETA(IDXJ)
V-- 11950     |  CONTINUE
```

A 1 (6)

```
           |  DO 12300 IDXJ = JS , JE
V-->       |  |  SUB(IDXJ) = F(IDXJ) - X(IDXJ,K)
|          |  |  SUP(IDXJ) = G(IDXJ) - Y(IDXJ,K)
V--- 12300 |  CONTINUE
           |  DO 12310 IDXJ = JS , JE
V-->       |  |  X(IDXJ,K) = X(IDXJ,K) + SUB(IDXJ) * OMEGA
V--- 12310 |  CONTINUE
           |
           |  DO 12320 IDXJ = JS , JE
V-->       |  |  Y(IDXJ,K) = Y(IDXJ,K) + SUP(IDXJ) * OMEGA
V--- 12320 |  CONTINUE
           |  RSUM1 = 0.0
           |  RSUM2 = 0.0
           |  DO 12330 IDXJ = JS , JE
V-->       |  |  RSUM1 = RSUM1 + ABS(SUP(IDXJ))
|          |  |  RSUM2 = RSUM2 + ABS(SUB(IDXJ))
V--- 12330 |  CONTINUE
           |  RSUM = RSUM + OMEGA * (RSUM1 + RSUM2)
           |
           |  DO 12340 IDXJ = JS , JE
39->       |  |  RMAX = MAX(RMAX , OMEGA*ABS(SUB(IDXJ)) , OMEGA*ABS(SUP(IDXJ)))
39-- 12340 |  CONTINUE
---- 2000 CONTINUE
```

⇩ (実行回数：56112, ベクトル長： 100)

```
           |  RSUM1 = 0.0
           |  RSUM2 = 0.0
           |  DO 12300 IDXJ = JS , JE
V-------->    |  SUBIDXJ = F(IDXJ) - X(IDXJ,K)
|          |  |  SUPIDXJ = G(IDXJ) - Y(IDXJ,K)
|          |  |  X(IDXJ,K) = X(IDXJ,K) + SUBIDXJ * OMEGA
|          |  |  Y(IDXJ;K) = Y(IDXJ,K) + SUPIDXJ * OMEGA
|          |  |  RSUM1 = RSUM1 + ABS(SUPIDXJ)
|          |  |  RSUM2 = RSUM2 + ABS(SUBIDXJ)
|     S    |  |  RMAX = MAX(RMAX , OMEGA*ABS(SUBIDXJ) , OMEGA*ABS(SUPIDXJ))
V--------S12300 |  CONTINUE
           |  RSUM = RSUM + OMEGA * (RSUM1 + RSUM2)
---------  2000 CONTINUE
```

A 2 関数MAXのベクトル化

```
           |  DO 12340 IDXJ = JS , JE
25------>  |  |  RMAX = MAX(RMAX , OMEGA*ABS(SUB(IDXJ)) , OMEGA*ABS(SUP(IDXJ)))
25------- 12340 |  CONTINUE
```

⇩ (実行回数：56112, ベクトル長： 101)

```
           |  DO 12340 IDXJ = JS , JE
V-------->    |  WMAX = MAX(OMEGA*ABS(SUB(IDXJ)) , OMEGA*ABS(SUP(IDXJ)))
|          |  |  RMAX = MAX(RMAX , WMAX)
V-------- 12340 |  CONTINUE
```

```
|          |  DO 12340 IDXJ = JS , JE
| V------->    |  WMAX = MAX(OMEGA*ABS(SUB(IDXJ)) , OMEGA*ABS(SUP(IDXJ)))
| |        |  |  RMAX = MAX(RMAX , WMAX)
| V-------- 12340 |  CONTINUE
28---------  2000 CONTINUE
```

⇩ (実行回数：56112, ベクトル長： 101)

```
|          |  DO 12340 IDXJ = JS , JE
| V------->    |  WMAX = MAX(ABS(SUB(IDXJ)) , ABS(SUP(IDXJ)))
| |        |  |  RMAX = MAX(RMAX , WMAX)
| V-------- 12340 |  CONTINUE
28---------  2000 CONTINUE
           "                **NEC**2 85/11/10  AFMESH_MAX2 : PUT OMEGA OUTSIDE
           RMAX=RMAX*OMEGA
```

A 3　ベクトル関数LVMXAの使用

```
                    JS = 3
                    JE = JMM1
                    MP(ITER) = JS - 1
                    XXIMAX   = ABS(XXI(JS))
                    DO 11200 IDXJ = JS+1 , JE
9---------->        |  IF (XXIMAX .GE. ABS(XXI(IDXJ))) GO TO 11200
|                   |  MP(ITER) = IDXJ - 1
|                   |  XXIMAX   = ABS(XXI(IDXJ))
9---------- 11200  CONTINUE

                    MQ(ITER) = JS - 1
                    YXIMAX   = ABS(YXI(JS))
                    DO 11300 IDXJ = JS+1 , JE
10--------->        |  IF (YXIMAX .GE. ABS(YXI(IDXJ))) GO TO 11300
|                   |  MQ(ITER) = IDXJ - 1
|                   |  YXIMAX   = ABS(YXI(IDXJ))
10--------- 11300  CONTINUE
```

⇩ (実行回数： 451, ベクトル長： 99)

```
|                   JS = 3
|                   JE = JMM1
|                   MP(ITER) = JS - 1
|                   XXIMAX   = ABS(XXI(JS))
|                   MQ(ITER) = JS - 1
|                   YXIMAX   = ABS(YXI(JS))
|                   MP(ITER)=LVMXA(XXI(JS),JE-JS,1)+JSS
|                   XXIMAX=ABS(XXI(MP(ITER)+1))
|                   MQ(ITER)=LVMXA(YXI(JS),JE-JS,1)+JSS
|                   YXIMAX=ABS(YXI(MQ(ITER)+1))
```

```
      JJJ = JLEAD
      JSNGLA(ITER) = JJJ - 1
      SNGLAP(ITER) = R1(JJJ)
      SNGLAQ(ITER) = R2(JJJ)
      JS = 3
      JE = JJJ - 1
      DO 11185 IDXJ = JS , JE
      XXI(IDXJ) = R1(IDXJ)
      YXI(IDXJ) = R2(IDXJ)
11185 CONTINUE
      JS = JJJ + 1
      JE = JMAX
      DO 11190 IDXJ = JS , JE
      XXI(IDXJ-1) = R1(IDXJ)
      YXI(IDXJ-1) = R2(IDXJ)
11190 CONTINUE
      JS = 3
      JE = JMM1
      MP(ITER) = JS - 1
      XXIMAX   = ABS(XXI(JS))
      MQ(ITER) = JS - 1
      YXIMAX   = ABS(YXI(JS))
      MP(ITER)=LVMXA(XXI(JS),JE-JS,1)+JSS
      XXIMAX=ABS(XXI(MP(ITER)+1))
      MQ(ITER)=LVMXA(YXI(JS),JE-JS,1)+JSS
      YXIMAX=ABS(YXI(MQ(ITER)+1))
      IF (MP(ITER) .GE. JJJ)   MP(ITER) = MP(ITER) + 1
      IF (MQ(ITER) .GE. JJJ)   MQ(ITER) = MQ(ITER) + 1
      RP(ITER) = ABS(XXI(JS))
      DO 11400 IDXJ = JS+1 , JE
      |  RP(ITER) = MAX(RP(ITER) , ABS(XXI(IDXJ)))
11400 CONTINUE
      RQ(ITER) = ABS(YXI(JS))
      DO 11500 IDXJ = JS+1 , JE
      |  RQ(ITER) = MAX(RQ(ITER) , ABS(YXI(IDXJ)))
11500 CONTINUE
```

⇩ (実行回数： 451, ベクトル長： 50)

```
      JJJ = JLEAD
      JSNGLA(ITER) = JJJ - 1
      SNGLAP(ITER) = R1(JJJ)
      SNGLAQ(ITER) = R2(JJJ)
      R1(JJJ)=0
      R2(JJJ)=0
      MMP=LVMXA(R1(JS),JE-JS+1,1)+JS
      MMQ=LVMXA(R2(JS),JE-JS+1,1)+JS
      RP(ITER)=ABS(R1(MMP))
      RQ(ITER)=ABS(R2(MMQ))
      MP(ITER)=MMP
      MQ(ITER)=MMQ
```

A 4 一次漸化式のベクトル化

```
                |  DO 2100 J = JS + 1 , JE
19------>       |  |  Z     = 1.0 / (DIAG(J) - SUB(J) * SUP(J-1))
|               |  |  SUP(J) = SUP(J) * Z
|               |  |  F(J)   = (F(J) - SUB(J) * F(J-1)) * Z
|               |  |  G(J)   = (G(J) - SUB(J) * G(J-1)) * Z
19------- 2100  |  CONTINUE
```

⇩ (実行回数：56112, ベクトル長： 100)

```
                |  DO 2100 J = JS + 1 , JE
V------->S      |  |  Z     = 1.0 / (DIAG(J) - SUB(J) * SUP(J-1))
|        S      |  |  SUP(J) = SUP(J) * Z
|               |  |  F(J)   = F(J)*Z  - SUB(J) *Z * F(J-1)
|               |  |  G(J)   = G(J)*Z  - SUB(J) *Z * G(J-1)
V-------- 2100  |  CONTINUE
```

A 5 総合チューニング(II)

SOLVEサブルーチンをSOLVELおよびSOLVEPサブルーチンに分割した。

(1)

```
                      DO 10100 I = 1 , MAXIT2
V------------>        |  RMAXT(I)  = 0.0
|                     |  RSUMT(I)  = 0.0
|                     |  RP(I)     = 0.0
|                     |  RQ(I)     = 0.0
|                     |  RPQS(I)   = 0.0
|                     |  SNGLAP(I) = 0.0
|                     |  SNGLAQ(I) = 0.0
|                     |  JSNGLA(I) = 0
V------------ 10100 CONTINUE
```

⇩ (実行回数： 2, ベクトル長：1002)

削 除

(2)

```
IF (.NOT. POIS)                    GO TO 1050
```

⇩ (実行回数：1336)

削 除

(3)

```
      IF (.NOT. POIS)                                  GO TO 2400
      IF (MAX(RP(ITER),RQ(ITER)) .GT. CVPQ .OR.
     1    RPQS(ITER) .GT. CVPQS)                       GO TO 2500
 2400 CONTINUE
```

⇩ (実行回数：1336)

SOLVEL (POIS=.FALSE.) では削除
SOLVEL (POIS=.TRUE.) ではIF文のみ削除

(4)

```
      IF (IPRINT(8) .EQ. 1 .AND. ISYMME .EQ. 1 .AND. POIS)
     1    CALL SNGPRT(SNGLAP,SNGLAQ,JSNGLA,MSIZE)
      IF (POIS)  CALL ANGLE(X,Y,JM2,KM2,P1,Q1,R1,R2,KSTART)
```

⇩ (実行回数： 2)

SOLVELでは削除
SOLVEPではIF文のみ削除

(5)　SOLVELサブルーチンについて

```
      DO 2000 K = KSTART , KEND
      DO 11950 IDXJ = JS , JE
      XXIIDXJ  = 0.5 * (X(IDXJ+1,K) - X(IDXJ-1,K))
      YXIIDXJ  = 0.5 * (Y(IDXJ+1,K) - Y(IDXJ-1,K))
      XETAIDXJ = 0.5 * (X(IDXJ,K+1) - X(IDXJ,K-1))
      YETAIDXJ = 0.5 * (Y(IDXJ,K+1) - Y(IDXJ,K-1))
      SUP(IDXJ)  = XETAIDXJ * XETAIDXJ + YETAIDXJ * YETAIDXJ
      GDIDXJ   = XXIIDXJ * XXIIDXJ + YXIIDXJ * YXIIDXJ
      SUB(IDXJ)  = SUP(IDXJ)
      DIAG(IDXJ) = (-2.0) * (SUP(IDXJ) + GDIDXJ)
      BDIDXJ   =    0.5  * ((X(IDXJ+1,K) - X(IDXJ-1,K)) *
     1                        (X(IDXJ,K+1) - X(IDXJ,K-1))
     2                   +  (Y(IDXJ+1,K) - Y(IDXJ-1,K)) *
     3                        (Y(IDXJ,K+1) - Y(IDXJ,K-1)))
      JACOB(IDXJ) = XXIIDXJ * YETAIDXJ - YXIIDXJ * XETAIDXJ
      F(IDXJ) =     BDIDXJ
     1          * 0.25 * (X(IDXJ+1,K+1) - X(IDXJ+1,K-1) -
     2                    X(IDXJ-1,K+1) + X(IDXJ-1,K-1))
     3          + (-GDIDXJ) * (X(IDXJ,K+1) + X(IDXJ,K-1))
(a)  4          + (-JACOB(IDXJ) ** 2)
     5          * (P1(IDXJ) * EXP(-A*XFLOAT(K)) * XXIIDXJ +
     6             Q1(IDXJ) * EXP(-B*XFLOAT(K)) * XETAIDXJ)
      G(IDXJ) =     BDIDXJ
     1          * 0.25 * (Y(IDXJ+1,K+1) - Y(IDXJ+1,K-1) -
     2                    Y(IDXJ-1,K+1) + Y(IDXJ-1,K-1))
     3          + (-GDIDXJ) * (Y(IDXJ,K+1) + Y(IDXJ,K-1))
(b)  4          + (-JACOB(IDXJ) ** 2)
     5          * (P1(IDXJ) * EXP(-A*XFLOAT(K)) * YXIIDXJ +
     6             Q1(IDXJ) * EXP(-B*XFLOAT(K)) * YETAIDXJ)
11950 CONTINUE
```

⇩ (実行回数：56112, ベクトル長：101)

```
      DO 2000 K = KSTART , KEND
      DO 11950 IDXJ = JS , JE
      XXIIDXJ  = 0.5 * (X(IDXJ+1,K) - X(IDXJ-1,K))
      YXIIDXJ  = 0.5 * (Y(IDXJ+1,K) - Y(IDXJ-1,K))
      XETAIDXJ = 0.5 * (X(IDXJ,K+1) - X(IDXJ,K-1))
      YETAIDXJ = 0.5 * (Y(IDXJ,K+1) - Y(IDXJ,K-1))
      SUP(IDXJ)  = XETAIDXJ * XETAIDXJ + YETAIDXJ * YETAIDXJ
      GDIDXJ   = XXIIDXJ * XXIIDXJ + YXIIDXJ * YXIIDXJ
      SUB(IDXJ)  = SUP(IDXJ)
      DIAG(IDXJ) = (-2.0) * (SUP(IDXJ) + GDIDXJ)
      BDIDXJ   =    0.5  * ((X(IDXJ+1,K) - X(IDXJ-1,K)) *
     1                        (X(IDXJ,K+1) - X(IDXJ,K-1))
     2                   +  (Y(IDXJ+1,K) - Y(IDXJ-1,K)) *
     3                        (Y(IDXJ,K+1) - Y(IDXJ,K-1)))
      JACOB(IDXJ) = XXIIDXJ * YETAIDXJ - YXIIDXJ * XETAIDXJ
      F(IDXJ) =     BDIDXJ
     1          * 0.25 * (X(IDXJ+1,K+1) - X(IDXJ+1,K-1) -
     2                    X(IDXJ-1,K+1) + X(IDXJ-1,K-1))
     3          + (-GDIDXJ) * (X(IDXJ,K+1) + X(IDXJ,K-1))
      G(IDXJ) =     BDIDXJ
     1          * 0.25 * (Y(IDXJ+1,K+1) - Y(IDXJ+1,K-1) -
     2                    Y(IDXJ-1,K+1) + Y(IDXJ-1,K-1))
     3          + (-GDIDXJ) * (Y(IDXJ,K+1) + Y(IDXJ,K-1))
11950 CONTINUE
```

上のコーディングの(a)および(b)のステートメントを削除した。

付録B

B 1 スカラ変数の配列化

MURMANサブルーチン

```
                    PHIYP  = PHI(  1  ,2) - PHI(  1  ,1)
                    PHIYY  = PHIYP + PHI0 - PHI(1,1)
                    PHIXX  = PHI(I+1,1) + PHI(I-1,1) - PHI(I,1) - PHI(1,1)
                    PHIXM  = PHI(I+1,1) - PHI(I-1,1)
                    PHIXP  = PHI(I+1,2) - PHI(I-1,2)
                    "
                  DO 60 J = 2, N
 V----------->      PHIXX  = PHI(I+1,J) + PHI(I-1,J) - PHI(I,J) - PHI(I,J)
 |          S      OU     = PHIXP
 |          S      PHIXP  = PHI(I+1,J+1) - PHI(I-1,J+1)
 |          S      PHIXY  = PHIXP - PHIXM + ( E(J+1) - E(J-1) )*FAC
 |          S      PHIXM  = DU
 |                 DU     = DU*DELTH
 |          S      PHIYYM = PHIYY
 |          S      PHIYM  = PHIYP
 |          S      PHIYP  = PHI(I,J+1) - PHI(I,J)
 |          S      PHIYY  = PHIYP - PHIYM
 |          S      U      = R(J)*DU - SI(I)
 |          S      DV     = R(J)*( PHI(I,J+1) - PHI(I,J-1) )*DELR
 |          S      V      = DV*R(J) - CO(I)
 |          S      RAV    = R(J)*RA*V
 |          S      BQ     = 1./FP(I,J)
 |          S      BQU    = BQ*U
 |          S      US     = BQU*U
 |          S      UV     = ( BQU+BQU )*V
 |          S      VS     = BQ*V*V
 |          S      QS     = US + VS
 |          S      CS     = C1 - C2*QS
 |          S      CMVS   = CS - VS
 |          S      CMUS   = CS - US
 |          S      PHIXT  = BETA*ABS( U )
 |          S      PHIYT  = BETA*ABS(RAV)
 |                 "
 |          S      D(J)   = RA4*( .( CMVS + US - VS )*DV - UV*DU )
 |                        + RI(J)*QS*BQ*(   U *( FP(I-1,J) - FP(I+1,J) )
 |                 1                      + RAV*( FP(I,J-1) - FP(I,J+1) ) )
 |          S      UV     = 0.5*BQU*RAV
 |          S      IF( QS.LE.QCRIT )  GO TO 50
 |                 "
 |          S      KK     = KK + 1
 |          S      CMQS   = CS - QS
 |          S      FQ     = 1. / QS
 |          S      AUU    = US*FQ
 |          S      BUU    = RS(J)*AUU
 |          S      BVV    = VS*FQ
 |          S      AVV    = RS(J)*BVV
 |          S      BUV    = UV*FQ
 |          S      AUV    = BQU*ABS(RAV)*FQ*TE
 |          S      PHINN  = BVV*PHIXX - BUV*PHIXY + BUU*PHIYY
 |          S      B(J)   = CS*BUU
 |          S      PHIXT  = PHIXT - CMQS*( AUU + AUU - AUV ) + CS*BVV
 |          S      PHIYT  = PHIYT - CMQS*( AVV + AVV - AUV )
 |          S      C(J)   = B(J) + PHIYT
 |          S      PHIXXM = RPP(J)
 |          S      IF( V.LT.0 )  GO TO 45
 |                 "
 |          S      PHIYYM = PHI(I,J+2) - PHI(I,J+1) - PHIYP
 |                 "
 |          S      PHIXYM = PHIYP + PHI(IM,J) - PHI(IM,J+1)
 |          S      GO TO 46
 |                 "
 |         S45     PHIXYM = PHI(IM,J) - PHI(IM,J-1) - PHIYM
 |          S      BQ     = B(J)
 |          S      B(J)   = C(J)
 |          S      C(J)   = BQ
 |         S46     PHISS  = AUU*PHIXXM + AUV*PHIXYM + AVV*PHIYYM
 |          S      A(J)   = - ( B(J) + C(J) + PHIXT )
 |          S      D(J)   = D(J) + CMQS*PHISS + CS*PHINN - E(J)*PHIXT
 |          S      GO TO 60
 |                 "
 |          S 50   C(J)   = RS(J)*CMVS
 |          S      B(J)   = C(J) + PHIYT
 |          S      PHIXT  = PHIXT + CMUS
 |          S      A(J)   = XA*CMUS - B(J) - C(J) - PHIXT
 |          S      D(J)   = D(J) + CMUS*PHIXX - UV*PHIXY + C(J)*PHIYY - PHIXT*E(J)
 |          S      IF( V.LT.0. )  GO TO 60
 |                 "
 |          S      B (J)  = C(J)
 |          S      C (J)  = C(J) + PHIYT
 V----------  60   RPP(J)  = PHIXX
```

⇩

```
                DIMENSION PHIXP(0:31),PHIYY(31),PHIYP(31)

                PHIYP(1)  = PHI(1,2) - PHI(1,1)
                PHIYY(1)  = PHIYP(1) + PHI0 - PHI(1,1)
                PHIXX     = PHI(1+1,1) + PHI(1-1,1) - PHI(1,1) - PHI(1,1)
                PHIXP(0)  = PHI(1+1,1) - PHI(1-1,1)
                PHIXP(1)  = PHI(1+1,2) - PHI(1-1,2)

              DO 60 J = 2, N
V-----------> |  PHIXX  = PHI(I+1,J) + PHI(I-1,J) - PHI(I,J) - PHI(I,J)
|             |
|             |  PHIXP(J) = PHI(I+1,J+1) - PHI(I-1,J+1)
|             |  PHIXY  = PHIXP(J) - PHIXP(J-2) + ( E(J+1) - E(J-1) )*FAC
|             |
|             |  DU     = PHIXP(J-1)*DELTH
|             |
|             |  PHIYP(J)  = PHI(I,J+1) - PHI(I,J)
|             |  PHIYY(J)  = PHIYP(J) - PHIYP(J-1)
|             |  PHIYYM    = PHIYY(J-1)
|             |  U      = R(J)*DU - SI(I)
|             |  DV     = R(J)*( PHI(I,J+1) - PHI(I,J-1) )*DELR
|             |  V      = DV*R(J) - CO(I)
|             |  RAV    = R(J)*RA*V
|             |  BQ     = 1./FP(I,J)
|             |  BQU    = BQ*U
|             |  US     = BQU*U
|             |  UV     = ( BQU+BQU )*V
|             |  VS     = BQ*V*V
|             |  QS     = US + VS
|             |  CS     = C1 - C2*QS
|             |  CMVS   = CS - VS
|             |  CMUS   = CS - US
|             |  PHIXT  = BETA*ABS( U )
|             |  PHIYT  = BETA*ABS(RAV)
|             |
|             |  D(J)   = RA4*(  ( CMVS + US - VS )*DV - UV*DU  )
|             |         + RI(J)*QS*BQ*(    U *( FP(I-1,J) - FP(I+1,J) )
|            1|                          + RAV*( FP(I,J-1) - FP(I,J+1) )  )
|             |  UV     = 0.5*BQU*RAV
|             |  IF( QS.LE.QCRIT )  GO TO 50
|             |
|             |  KK     = KK + 1
|             |  CMQS   = CS - QS
|             |  FQ     = 1. / QS
|             |  AUU    = US*FQ
|             |  BUU    = RS(J)*AUU
|             |  BVV    = VS*FQ
|             |  AVV    = RS(J)*BVV
|             |  BUV    = UV*FQ
|             |  AUV    = BQU*ABS(RAV)*FQ*TE
|             |  PHINN  = BVV*PHIXX - BUV*PHIXY + BUU*PHIYY(J)
|             |  B(J)   = CS*BUU
|             |  PHIXT  = PHIXT - CMQS*( AUU + AUU - AUV ) + CS*BVV
|             |  PHIYT  = PHIYT - CMQS*( AVV + AVV - AUV )
|             |  C(J)   = B(J) + PHIYT
|             |  PHIXXM = RPP(J)
|             |
|             |  IF( V.LT.0 )  GO TO 45
|             |
|             |  PHIYYM = PHI(I,J+2) - PHI(I,J+1) - PHIYP(J)
|             |  PHIXYM = PHIYP(J) + PHI(IM,J) - PHI(IM,J+1)
|             |  GO TO 46
|             |
|          45 |  PHIXYM = PHI(IM,J) - PHI(IM,J-1) - PHIYP(J-1)
|             |  BQ     = B(J)
|             |  B(J)   = C(J)
|             |  C(J)   = BQ
|          46 |  PHISS  = AUU*PHIXXM + AUV*PHIXYM + AVV*PHIYYM
|             |  A(J)   = -( C(J) + B(J) + PHIXT )
|             |  D(J)   = D(J) + CMQS*PHISS + CS*PHINN - E(J)*PHIXT
|             |  GO TO 60
|             |
|          50 |  C(J)   = RS(J)*CMVS
|             |  B(J)   = C(J) + PHIYT
|             |  PHIXT  = PHIXT + CMUS
|             |  A(J)   = XA*CMUS - B(J) - C(J) - PHIXT
|             |  D(J)   = D(J) + CMUS*PHIXX - UV*PHIXY + C(J)*PHIYY(J) - PHIXT*E(J)
|             |  IF( V.LT.0. )  GO TO 60
|             |
|             |  B  (J) = C(J)
|             |  C  (J) = C(J) + PHIYT
V-------------    60 |  RPP(J) = PHIXX
```

B 2 スカラ変数の配列化

MURMA 1 サブルーチン

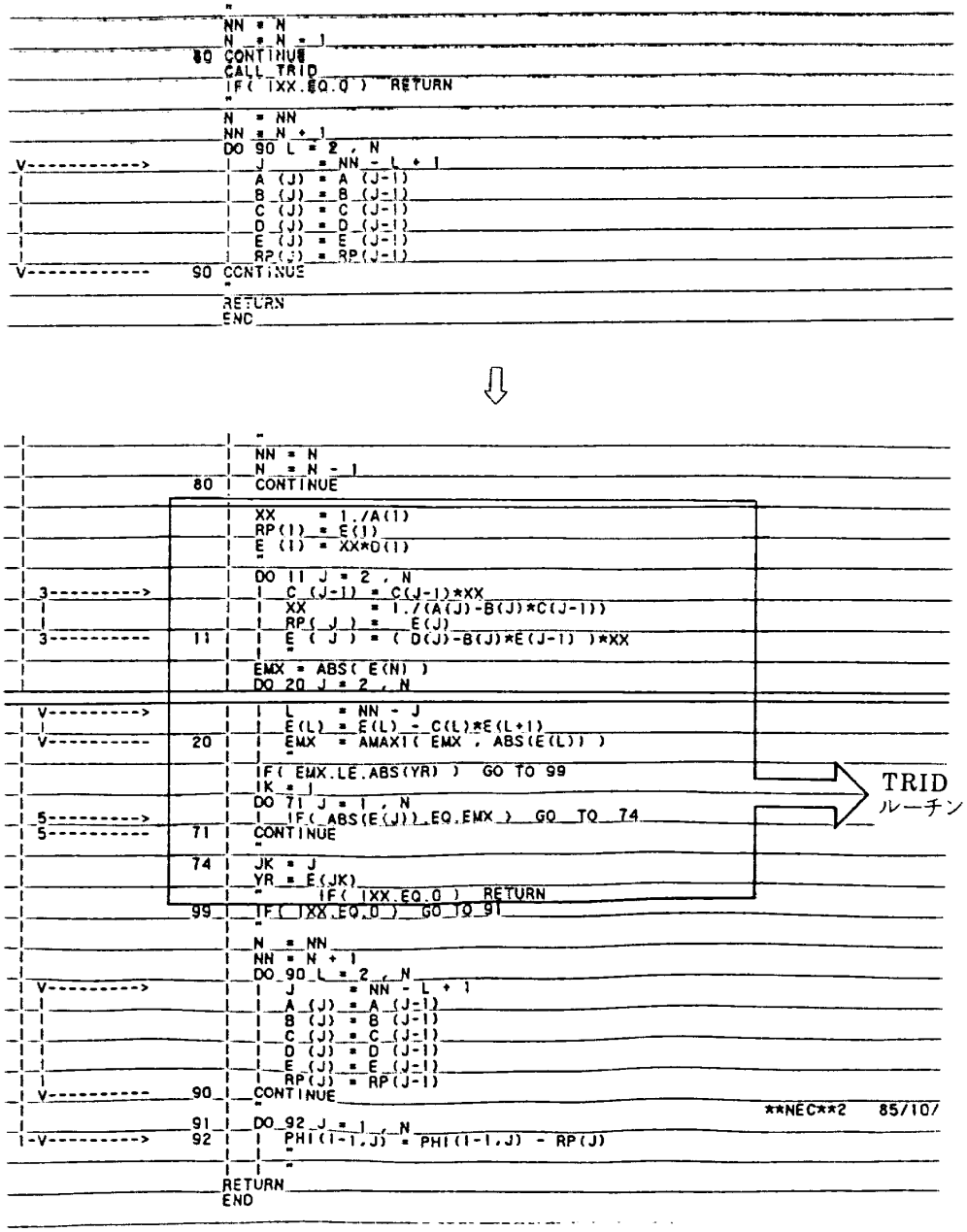
```
                      PHIO  = PHI(I,2) - 2.*DR*CO(I)
                      PHIYP = PHI(I,2) - PHI(I,1)
                      PHIYY = PHIYP + PHIO - PHI(I,1)
                      PHIXX = PHI(I+1,1) + PHI(I-1,1) - PHI(I,1) - PHI(I,1)
                      PHIXM = PHI(I+1,1) - PHI(I-1,1)
                      PHIXP = PHI(I+1,2) - PHI(I-1,2)

                      DO 60 J = 2, N
V-------------->      |  PHIXX = PHI(I+1,J) + PHI(I-1,J) - PHI(I,J) - PHI(I,J)
|             S       |  DU    = PHIXP
|             S       |  PHIXP = PHI(I+1,J+1) - PHI(I-1,J+1)
|             S       |  PHIXY = PHIXP - PHIXM
|             S       |  PHIXM = DU
|                     |  DU    = DU*DELTH
|             S       |  PHIYM = PHIYP
|             S       |  PHIYP = PHI(I,J+1) - PHI(I,J)
|             S       |  PHIYY = PHIYP - PHIYM
|                     |  U     = R(J)*DU - SI(I)
|                     |  DV    = R(J)*( PHI(I,J+1) - PHI(I,J-1) )*DELR
|                     |  V     = DV*R(J) - CO(I)
|                     |  RAV   = R(J)*RA*V
|                     |  BQ    = 1./FP(I,J)
|                     |  BQU   = BQ*U
|                     |  US    = BQU*U
|                     |  UV    = ( BQU+BQU )*V
|                     |  VS    = BQ*V*V
|                     |  QS    = US + VS
|                     |  IF( QS.LE.QCRIT )  GO  TO  50
|                     |  D(J)  = 0.
|                     |  GO  TO  60
|                     |
|           50        |  CS    = C1 - C2*QS
|                     |  CMVS  = CS - VS
|                     |  CMUS  = CS - US
|                     |  UV1   = .5*BQU*RAV
|                     |  C(J)  = RS(J)*CMVS
|                     |  D(J)  = RA4*(  ( CMVS + US - VS )*DV - UV*DU  )
|                     |          + R1(J)*QS*BQ*(  U *( FP(I-1, J ) - FP(I+1, J ) )
|                     ||                        + RAV*( FP( I ,J-1) - FP( I ,J+1) )  )
|                     ||         + CMUS*PHIXX - UV1*PHIXY + C(J)*PHIYY
|                     |  D(J)  = - D(J)/CS
|                     |
V-------------  60  CONTINUE
                      RETURN
                      END
```

⇩

```
                      DIMENSION PHIXP(0:31),PHIYP(31)

                      PHIO  = PHI(I,2) - 2.*DR*CO(I)

                      PHIYP(1) = PHI(I,2) - PHI(I,1)

                      PHIYY = PHIYP(1) + PHIO - PHI(I,1)
                      PHIXX = PHI(I+1,1) + PHI(I-1,1) - PHI(I,1) - PHI(I,1)

                      PHIXP(0) = PHI(I+1,1) - PHI(I-1,1)

                      PHIXP(1) = PHI(I+1,2) - PHI(I-1,2)

                      DO 60 J = 2, N

V-------------->      |  PHIXX = PHI(I+1,J) + PHI(I-1,J) - PHI(I,J) - PHI(I,J)
|                     |
|                     |  PHIXP(J) = PHI(I+1,J+1) - PHI(I-1,J+1)
|                     |
|                     |  PHIXY = PHIXP(J) - PHIXP(J-2)
|                     |
|                     |  DU    = PHIXP(J-1) * DELTH
|                     |
|                     |  PHIYP(J) = PHI(I,J+1) - PHI(I,J)
|                     |
|                     |  PHIYY = PHIYP(J) - PHIYP(J-1)
|                     |  U     = R(J)*DU - SI(I)
|                     |  DV    = R(J)*( PHI(I,J+1) - PHI(I,J-1) )*DELR
|                     |  V     = DV*R(J) - CO(I)
|                     |  RAV   = R(J)*RA*V
|                     |  BQ    = 1./FP(I,J)
|                     |  BQU   = BQ*U
|                     |  US    = BQU*U
|                     |  UV    = ( BQU+BQU )*V
|                     |  VS    = BQ*V*V
|                     |  QS    = US + VS
|                     |  IF( QS.LE.QCRIT )  GO  TO  50
|                     |  D(J)  = 0.
|                     |  GO  TO  60
|                     |
|           50        |  CS    = C1 - C2*QS
|                     |  CMVS  = CS - VS
|                     |  CMUS  = CS - US
|                     |  UV1   = .5*BQU*RAV
|                     |  C(J)  = RS(J)*CMVS
|                     |  D(J)  = RA4*(  ( CMVS + US - VS )*DV - UV*DU  )
|                     |          + R1(J)*QS*BQ*(  U *( FP(I-1, J ) - FP(I+1, J ) )
|                     ||                        + RAV*( FP( I ,J-1) - FP( I ,J+1) )  )
|                     ||         + CMUS*PHIXX - UV1*PHIXY + C(J)*PHIYY
|                     |  D(J)  = - D(J)/CS
|                     |
V-------------  60  CONTINUE
                      RETURN
                      END
```

B 3　サブルーチンのインライン化

TRIDをMURMANに埋め込む。



B 4 サブルーチン呼び出し回数の減少

SWEEPサブルーチン

```
                    LLP = LL+1
                    DO 30 I = LLP , M
  |---------->      |  IF( IQC.EQ.1 ) GO TO 25
  |                 |  CALL MURMAN
  |                 |  GO TO 26
  |              25 |  CALL MURMQC
  |              26 |  CONTINUE
  |                 |  DO 30 J = 1 , N
  |-V---------->  30 |  PHI(I-1,J) = PHI(I-1,J) - RP(J)
```

⇩

```
                    IF( IQC.NE.1 ) THEN
                    |  CALL MURMAN
                    ELSE
                    |  CALL MURMQC
                    END IF
```

MURMAN 1ルーチンを以下に示す。

```
                    LLP = LL+1
                    DO 92 I = LLP , M
  |---------->      |
  |                 |  +-------------+
  |                 |  | MURMANルーチン |
  |                 |  +-------------+
  |                 |
  |              91 |  DO 92 J = 1 , N
  |-V---------->  92 |  PHI(I-1,J) = PHI(I-1,J) - RP(J)
                    |
                    RETURN
                    END
```

B 5　サブルーチン呼び出し回数の減少

SWEEPサブルーチン

```
                    I = LL
2---------->    80  I = I-1
|                   IF( IQC.EQ.1 )  GO TO 85
|                   CALL MURMAN
|                   GO TO 86
|               85  CALL MURMQC
|               86  CONTINUE
|                   DO 60 J = 1, N
| V---------->  60  PHI(I+1,J) = PHI(I+1,J) - RP(J)
2-----------        IF( I.GT.2 ) GO TO 80
                    DO 70 J = 1, N
```

⇩

```
                    IF( IQC.NE.1 ) THEN
                    |  CALL MURMAN2
                    ELSE
                    |  CALL MURMQC
                    END IF
```

MURMAN 2を以下に示す。

```
                    I = LL
|---------->    81  I = J - 1
|
|                   MURMANルーチン
|
|               91  DO 92 J = 1, N
| V---------->  92  PHI(I+1,J) = PHI(I+1,J) - RP(J)
|-----------        IF( I.GT.2 ) GO TO 81
                    RETURN
                    END
```

B 6 サブルーチン呼び出し回数の減少

INVRSEサブルーチン

```
                     |  |  DO 1340 I = LL , MM
 31------>           |  |  |  PHI00 = PHI0(I)
 |                   |  |  |  IF( IQC.EQ.1 ) GO TO 1336
 |                   |  |  |  "
 |                   |  |  |  CALL MURMAN
 |                   |  |  |  IF( IPHIR.EQ.4 )  CALL TRDPHC
 |                   |  |  |  GO  TO  1337
 |                   |  |  |  "
 |             1336  |  |  |  CALL MURMQC
 |                   |  |  |  IF( IPHIR.EQ.4 )  CALL TRDPQC
 |             1337  |  |  |  CONTINUE
 |                   |  |  |  DO 1340 J = 2 , N
 31V----->     1340  |  |  |  |  PHI(I-1,J) = PHI(I-1,J)-RP(J)
```

⇩

```
 | |                 |  |
 | |                 |  |  IF( IQC.NE.1 ) THEN
 | |                 |  |  |  CALL MURMAN3
 | |                 |  |  ELSE
 | |                 |  |  |  CALL MURMQC
 | |                 |  |  END IF
```

MURMAN 3を以下に示す。

```
                        DO 1340 I=LL,MM
 |---------->           |   PHI00=PHI0(I)
 |                      |
 |                      |    +----------------------+
 |                      |    |                      |
 |                      |    |   MURMANルーチン      |
 |                      |    |                      |
 |                   10 |    |                      |
 |                      |    +----------------------+
 |                      |
 |                   91 |   IF( IPHIR.EQ.4 ) CALL TRDPHC
 |                      |   DO 1340 J=2,N
 L-V--------->     1340 |   |  PHI(I-1,J)=PHI(I-1,J)-RP(J)
                        RETURN
                        END
```

B 7　サブルーチン呼び出し回数の減少

INVRSEサブルーチン

```
                    |  |  |      = LL
33------>  1380     |  |  |      = I-1
                    |  |  PHI00 = PHI0(I)
                    |  |  IF( IQC.EQ.1 )  GO  TO  1385
                    |  |  CALL MURMAN
                    |  |  IF( IPHIR.EQ.4 )  CALL TRDPHC
                    |  |  GO  TO  1386
                    |  |  "
            1385    |  |  CALL MURMQC
                    |  |  IF( IPHIR.EQ.4 )  CALL TRDPQC
            1386    |  |  CONTINUE
                    |  |  DO 1400 J = 2 , N
 V------>   1400    |  |  |  PHI(I+1,J) = PHI(I+1,J)-RP(J)
33--------          |  |  IF( I.GT.2 )  GO  TO  1380
                    |  |  DO 1420 J = 2 , N
```

⇩

```
                    |  |  IF( IQC.NE.1 ) THEN
                    |  |  |  CALL MURMAN4
                    |  |  ELSE
                    |  |  |  CALL MURMQC
                    |  |  END IF
```

MURMAN 4 を以下に示す。

```
                       I=LL
------------>  1380    I=I-1
                       PHI00=PHI0(I)
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                       |
                    10 |
                       |
                       |
                       |
                       |
                       |
                    91 IF( IPHIR.EQ.4 ) CALL TRDPHC
                       DO 1400 J=2,N
 V---------->   1400  |  PHI(I+1,J)=PHI(I+1,J)-RP(J)
------------           IF( I.GT.2 ) GOTO 1380
                       RETURN
                       END
```

MURMANルーチン

航空宇宙技術研究所報告909号

昭和61年7月発行

発行所　航空宇宙技術研究所
東京都調布市深大寺東町7－44－1
電話武蔵野三鷹(0422)47-5911(大代表)〒182

印刷所　株式会社　共進
東京都杉並区久我山5－6－17

Printed in Japan