

航空宇宙技術研究所報告

TECHNICAL REPORT OF NATIONAL AEROSPACE LABORATORY

TR-986

航空宇宙技術研究所汎用プログラムの高速化

中 村 絹 代 ・ 吉 田 正 廣 ・ 福 田 正 大

1988年7月

航空宇宙技術研究所
NATIONAL AEROSPACE LABORATORY

航空宇宙技術研究所汎用プログラムの高速化*

中 村 絹 代** 吉 田 正 廣** 福 田 正 大**

High Speed Implementation of NAL Application Software Packages

Kinuyo NAKAMURA, Masahiro YOSHIDA
and Masahiro FUKUDA

ABSTRACT

The Numerical Simulator system (NS system), installed in the National Aerospace Laboratory (NAL) in February 1987, is composed of FACOM VP400, VP200, and M780 computers. The first two are vector-processor-type supercomputers and the last is a scalar computer.

To extract full performance from supercomputers, it is necessary to give consideration to solution algorithms, programs and data structure, program coding, etc. to match individual supercomputers.

NAL has various application software programmed for a scalar computer. They are not necessarily matched for the VP400 and VP200, therefore we tuned them up to facilitate high speed processing on those supercomputers.

This paper reports the relative performance of the VP400, VP200 and M780, by comparing computing time of the original and the tuned programs on each computer. Also, methods of effective FORTRAN programming to increase vectorization rate and high speed computation with FORTRAN programs on the machines are described.

1. ま え が き

昭和62年2月に航空宇宙技術研究所(以下、航技研という)でFACOM M780, VP200およびVP400より成る数値シミュレータシステムが稼動開始となった。M780は汎用計算機, VP200およびVP400はベクトル計算機と呼ばれるスーパーコンピュータである。

航技研に我が国で初めてのベクトル計算機FACOM 230-75APシステムが導入された昭和53年に比較して、現在では多くのスーパーコンピュ

ータが開発され、普及している。

汎用計算機と違って科学技術計算専用機であるスーパーコンピュータは、それを活かした利用法を行わなければ大きな効果を挙げることが出来ないが、『汎用』スーパーコンピュータとして普及した理由の一つとして利用の容易さが格段に改善されたことが挙げられる。

例えば、コンパイラによる自動ベクトル化機能がある。これはFORTRAN言語で書かれたプログラムの高速処理可能な部分をコンパイラが自動的に抽出し、高速処理実行命令を用いたオブジェクトを作成する機能である。この機能によりFORTRAN言語で書かれたプログラムをそのま

* 昭和63年4月7日受付

** 数理解析部

まスーパーコンピュータで実行することが出来る。

一世代前のスーパーコンピュータと比較すると、230-75AP では AP-FORTRAN という擬似 FORTRAN 言語でプログラムを書いたため、若干の利用上の不便さがあり、また FORTRAN プログラムの互換性を保つことが出来なかった。一方 230-75AP と同時期の STAR-100 や Cyber 203/205 等の計算機では自動ベクトル化機能を持っていたが、現在のものと比較して機能・性能ともに不十分であった（注 1）。

しかし、どのように利用してもスーパーコンピュータを用いれば汎用計算機の何十倍もの処理速度が得られると考えるのは早計である。プログラム / データ構造、コーディング方法、さらには計算スキーム / アルゴリズム等によりこの自動ベクトル化機能が十分に活用されない場合も多々あり、汎用計算機の処理速度と同程度にしかないこともある。この場合には、スーパーコンピュータの性能を引き出すようにプログラム変更、計算スキーム / アルゴリズムの変更を行うことが必要になる。このうち主として前者の変更をプログラムチューニングという。

プログラムチューニングを行うことによって何十倍にも高速化されたプログラムの実行は大変効率が良いので、様々な条件下で数値シミュレーションを実行する時に得られる計算時間の短縮は測り知れないものとなる。高速化の必要性はこの点にあり、数値シミュレーションそのものおよびそれを利用した研究・開発の生産性向上に役立つ。

本報告は航技研数理解析部に登録されている汎用プログラム 12 本のプログラムチューニングを行い、これら汎用プログラムの各種計算機での実行時間の比較及び検討、高速化手法について述べたものである。

以下 2 章では汎用計算機とベクトル計算機の相違点について簡単に説明し、3 章ではチューニングプログラムの各種計算機での実行時間についての比較及び検討、4 章では高速化手法について述

べる。

2. 数値シミュレータシステムの概要

航技研数理解析部に設置されている数値シミュレータシステムは、FACOM M780 をフロントエンド計算機とし、スーパーコンピュータである FACOM VP200 および VP400 を計算エンジンとするものである。そのハードウェア構成を図 2.1 に示す。また表 2.1 に M780, VP200/400 および M380（注 2）の性能を示す。

VP200/400 はそれが採用している高速演算処理方式からベクトル計算機といわれることもある。またこれに対応して汎用計算機はスカラ計算機といわれる（ベクトル、スカラの意味について次に述べる）。VP200/400 はベクトル計算機であるが、これをスカラ計算機として使用する場合には M380 と同等である。表 2.1 において倍率 1 および倍率 2 は、性能 2 を基に算出したものである。

2.1 ベクトル計算機の概要

計算機がデータに施す演算の仕方として、i) 単一データ単一演算、ii) 多数データ単一演算、iii) 多数データ異種演算が考えられる。このうち i) の処理を行うのがスカラユニット、スカラプロセッサ、スカラ計算機である。ii) の処理を行うのがベクトルユニットであり、これを有している計算機がベクトル計算機あるいはベクトルプロセッサである¹⁾。当然のことではあるがベクトル計算機といえどもスカラ処理をする必要があるのでスカラユニットは備えている。VP200/400 をスカラ計算機として使用するということは、それが持っているベクトルユニットを全く使用せず、スカラユニットのみを使用するということである。

多数のデータに単一の演算を施す方式も種々考えられるが、VP200/400 が採用しているのはパイプライン方式である。この方式でベクトル処理命令を実行する場合、パイプライン 1 段当たり処理時間（1 ピッチ）はマシンサイクルに相当し、パイプラインの中が n 重化されていればこのマシン

注 1) たとえば、IF 文やリストベクトルは FORTRAN の自動ベクトル化の対象ではなかった。

注 2) M380 の性能は参考として示す。

NAL Numerical Simulator System

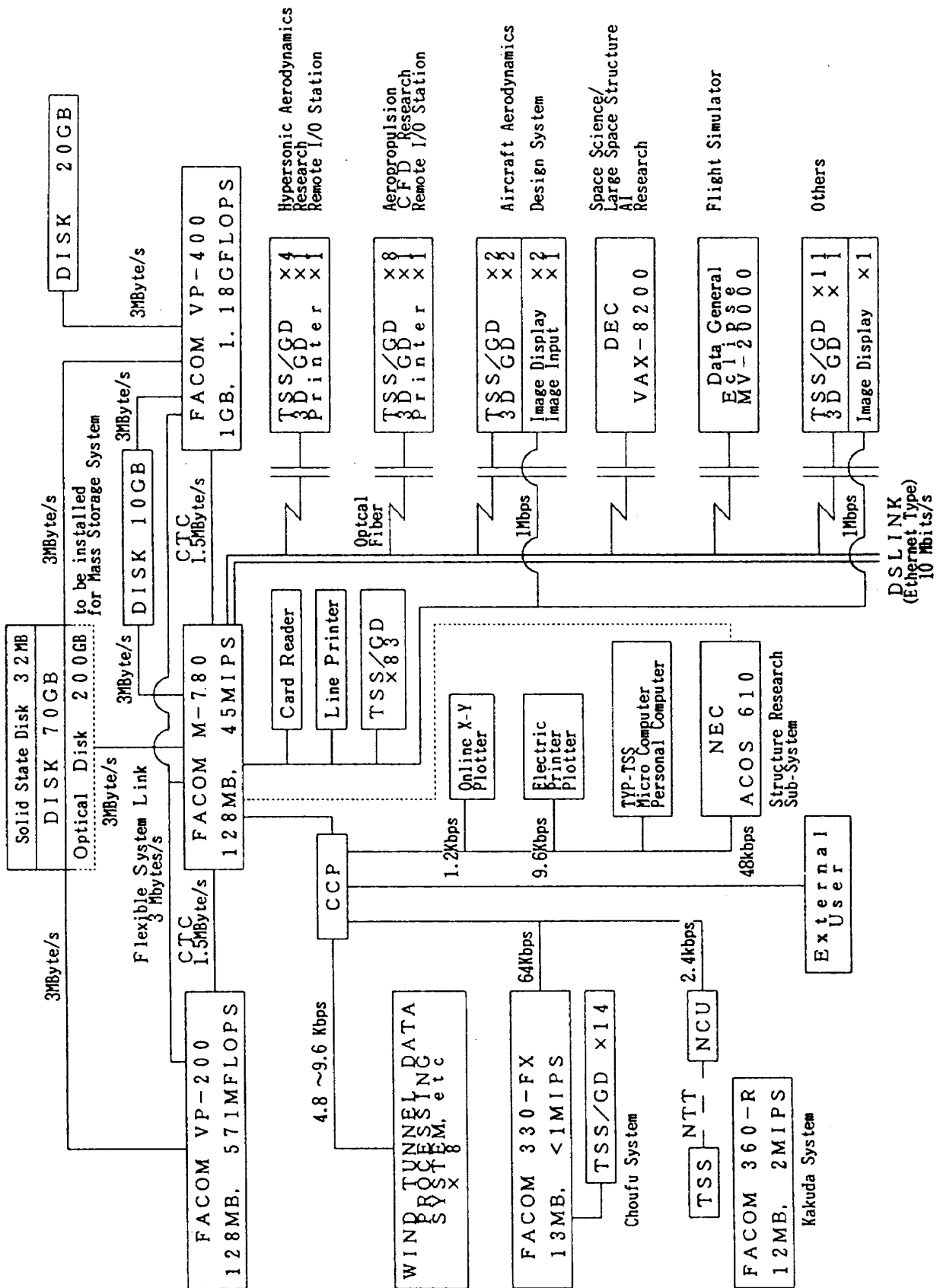


図 2.1 航技研数値シミュレーションシステム

表 2.1 計算機性能比較

| システム | 性 能 1 | 性 能 2 | 倍 率 1 | 倍 率 2 |
|-------|---------|-------------|-------|-------|
| M380 | 20 MIPS | 8 MFLOPS | 1 | — |
| M780 | 45 MIPS | 18 MFLOPS | 2.25 | 1 |
| VP200 | — | 571 MFLOPS | 71.4 | 31.7 |
| VP400 | — | 1185 MFLOPS | 148.1 | 65.8 |

MIPS = million instructions per second

MFLOPS = mega floating operations per second

サイクル毎に n 個の演算結果が得られる²⁾。また最近のパイプライン方式のスーパーコンピュータは異なる演算パイプラインが同時に動作するようになっており、これは部分的に前述のⅢ)の処理方式を実現していると見ることも可能である。このような観点からパイプライン方式のスーパーコンピュータを分類すれば、

- (1) 多重パイプライン単一プロセッサ方式—パイプラインを多重化する。

例, FACOM VP²⁾

- (2) 多重プロセッサ方式—パイプラインプロセッサを共通主記憶上で多重化する。

例, CRAY XMP³⁾

- (3) 多重パイプライン多重プロセッサ方式—(1)および(2)の方式を併用する。

例, ETA ETA10⁴⁾

となる。

以下に FACOM VP の概要を示す。図 2.2 は VP のハードウェア構成を、表 2.2 は性能諸元を示す。VP の本体系装置はベクトル処理装置、主記憶装置、チャンネル装置およびサービスプロセッサから成る。ベクトル処理装置はベクトルユニットおよびスカラユニットより成る。前者には、

- (1) 加算 / 論理演算パイプライン (1 本)
- (2) 乗算パイプライン (1 本)
- (3) 除算パイプライン (1 本)
- (4) マスクパイプライン (1 本)

- (5) ロード / ストアパイプライン (2 本)

の 5 種類 6 本のパイプラインが備えられている。これらパイプラインのうち、(1)~(3)の 3 本中の 2 本、(4)の 1 本、(5)の 2 本の合計 5 本のパイプラインが並列動作可能である。このうち表 2.1 のピーク演算処理速度に関係するのは(1)~(3)のパイプラインだけである。

以上のことから VP 200 のピーク演算処理速度を求めると、ベクトルユニットのマシンサイクル 7 ns、1 つのパイプラインの中は 2 重化、並列動作可能な演算パイプラインが 2 本なので、

$$1/(7 \times 10^{-9} \text{s} / 2 \text{FLOP}) \times 2 = 571 \text{MFLOPS}$$

となる。

一方 VP 400 は各々のパイプラインの中が 4 重化されており、パイプライン 1 本の処理能力は VP 200 の 2 倍になっている。また並列動作可能な演算パイプラインも 2 本のままであるが、航技研に設置されている VP 400 のベクトルユニットのマシンサイクルは 6.75 ns に強化されている。この結果、航技研 VP 400 のピーク演算処理速度は、

$$1/(6.75 \times 10^{-9} \text{s} / 4 \text{FLOP}) \times 2 = 1185 \text{MFLOPS}$$

となる。

ここで注意すべきは VP 400 は VP 200 の 2 倍の性能のロード / ストアパイプラインを 1 本しか持っていないことである。従って、ピークロード / ストア性能のデータ供給能力は VP 400 は VP 200 と同等である。計算機で扱われるベクトルデータ

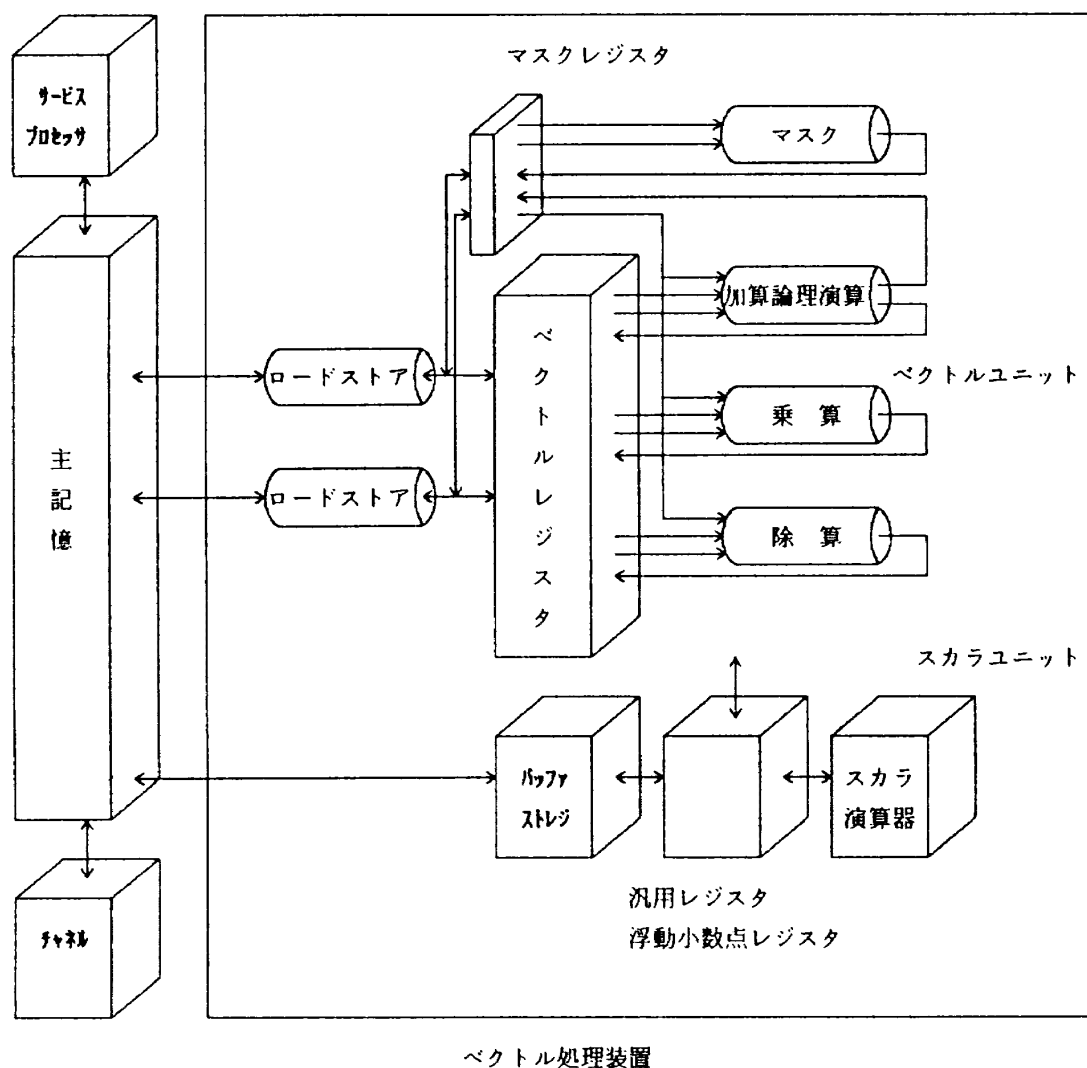


図 2.2 FACOM VP システムのハードウェア構成図

は連続ベクトルデータ、ストライド付きベクトルデータおよびリストベクトルデータの三つに分類できる⁵⁾。これらのうち、連続ベクトルデータの供給が最高速で処理される。ストライド付きベクトルデータの場合、距離間隔 2（1 個飛びデータ）のデータの供給能力は連続ベクトルデータの供給能力の 1/2 であり、距離間隔 4 のデータの場合は 1/4 であり、距離間隔 8 のデータの場合は 1/8 となる（倍精度データ、メモリ・バンクの競合は考えないとする）。これは VP 400 も VP 200 も同じである。リストベクトルデータの場合、VP 400 のデータ供給能力は VP 200 より劣る。

表 2.2 の演算パイプラインの項から明らかなように VP 200 のロード / ストアパイプラインは他の演算パイプラインの 2 倍の量である。VP 400 の

ロード / ストアパイプラインの量は他の演算パイプラインと同量である。このため VP 400 は VP 200 に比して相対的に演算処理能力に対するデータ供給能力が劣る。従って、VP 400 を有効に使うためには、ベクトル処理される DO ループ内の演算量をロード / ストア（主記憶装置とベクトルレジスタ間のデータ転送）に比べて多くする、即ち、不必要なロード / ストアは避けることが重要である。例えば、

```
DO 10 I=1, N
  A(I)=B(I)*C(I)
  D(I)=A(I)+E(I)
```

```
10 CONTINUE
```

のプログラムにおいて、配列 A(I) は後で使用されないデータであれば、

表 2.2 ハードウェアの主要諸元

| 項 目 | | 機 種 | VP-200 | VP-400 |
|-----------|-------------------|--------------------------------------|---|----------|
| ベクトル処理装置 | 命令数 | | 83 ベクトル命令 195 スカラ命令 | |
| | 汎用レジスタ | | 16 (32ビット) | |
| | 浮動小数点レジスタ | | 8 (64ビット) | |
| | 制御レジスタ | | 16 (32ビット) | |
| | ベクトルレジスタ | | 64 Kバイト | 128 Kバイト |
| | マスクレジスタ | | 1 Kバイト | 2 Kバイト |
| | データ形式 | ベクトル命令 論理 固定小数点 浮動小数点 | 1, 64ビット 32ビット 32, 64ビット | |
| | | スカラ命令 論理 固定小数点 浮動小数点 10進 | 8, 32, 64ビット 16, 32ビット 32, 64, 128ビット 8ビット | |
| | ベクトル浮動小数点 | | 64ビット | |
| | 演算パイプライン | | | |
| | 加算/論理 | | 2×1 | 4×1 |
| | 乗算 | | 2×1 | 4×1 |
| | 除算 | | 2×1 | 4×1 |
| | マスク | | 2×1 | 4×1 |
| | ロード/ストア | | 2×2 | 4×1 |
| | 並列動作可能 パイプライン数 | | 5 | 4 |
| 主記憶装置 | 容量 | | 64, 128, 256 Mバイト | |
| | インタリーブ数 | | 128, 256, 256 | |
| チャンネル処理装置 | CHP接続台数 | | 1 | |
| | チャンネル接続台数 | | 最大32 | |

```
DO 10 I=1, N
```

```
A=B(I) * C(I)
```

```
D(I)=A+E(I)
```

```
10 CONTINUE
```

のように配列 A(I) を WORK 変数 A として用いる方がよい。

この他にベクトル処理時間に影響を及ぼすものとしてベクトル演算の要素数（ベクトル長）がある。これは、1マシンサイクル毎に n 個の結果が得られるようになるまでには前処理時間（立ち上がり）、後処理時間が必要となり、これはベクトル長に依存しない。従ってベクトル長が長ければ1要素当たりのこれらの時間が短くなることによる。しかし現今のスーパーコンピュータでは演算チェイニングと称される処理を行っているので必ずしも単純な議論で済まないこともある。

図 2.3 は次の DO ループで書かれたプログラム

```
DO 10 I=1, N
```

```
:
```

```
:
```

```
D(I)=A(I) * B(I)+K * C(I)
```

```
:
```

```
:
```

```
10 CONTINUE
```

が VP 200 および VP 400 のベクトルユニットで処理される様子を視覚的に示したものである。ある時刻 t の断面でロードパイプライン、乗算パイプライン、加算パイプラインが並列に動作していることがわかる。また、VP 200 と VP 400 ではロード/ストアパイプラインの数が違うので配列のロード方法が異っている。

2.2 相対処理速度

ベクトルプロセッサとスカラプロセッサでの実行時間の比である相対処理速度 P （アムダールの式）は次の式で表される⁶⁾。

$$P = 1 / [(1 - V) + V / \alpha]$$

V ; ベクトル化率

プログラム全体の汎用計算機における計算時間のうちベクトル命令で実行可能な計算時間の割合

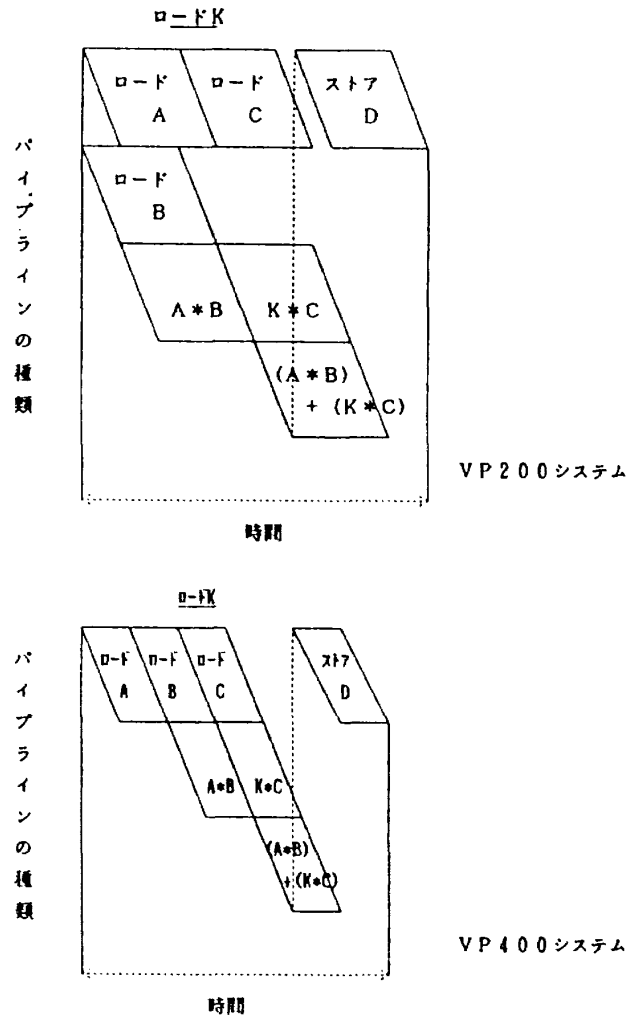


図 2.3 ベクトル命令の実行

α ; ベクトル / スカラ速度比

ベクトル命令の導入可能な部分における、汎用計算機の計算速度に対するベクトル命令を利用した場合の相対速度のプログラム内平均値

P を大きくするには、 V および α を大きくすればよい。

V を大きくするということは、ベクトルユニットで処理される部分を多くすることであり、ベクトル命令とスカラ命令で処理される速度が格段に違うことから当然である。これはまたプログラム全体を高速処理しているかどうかの量的判断基準といえる。図 2.4 に相対処理速度 P とベクトル化率 V の関係を示す。ここで注意すべきは、 α を無限大即ち無限大の処理速度をもつベクトル計算機を仮定しても、ベクトル化率が 50% であれば 2 倍の、90% であっても 10 倍の処理速度向上が得

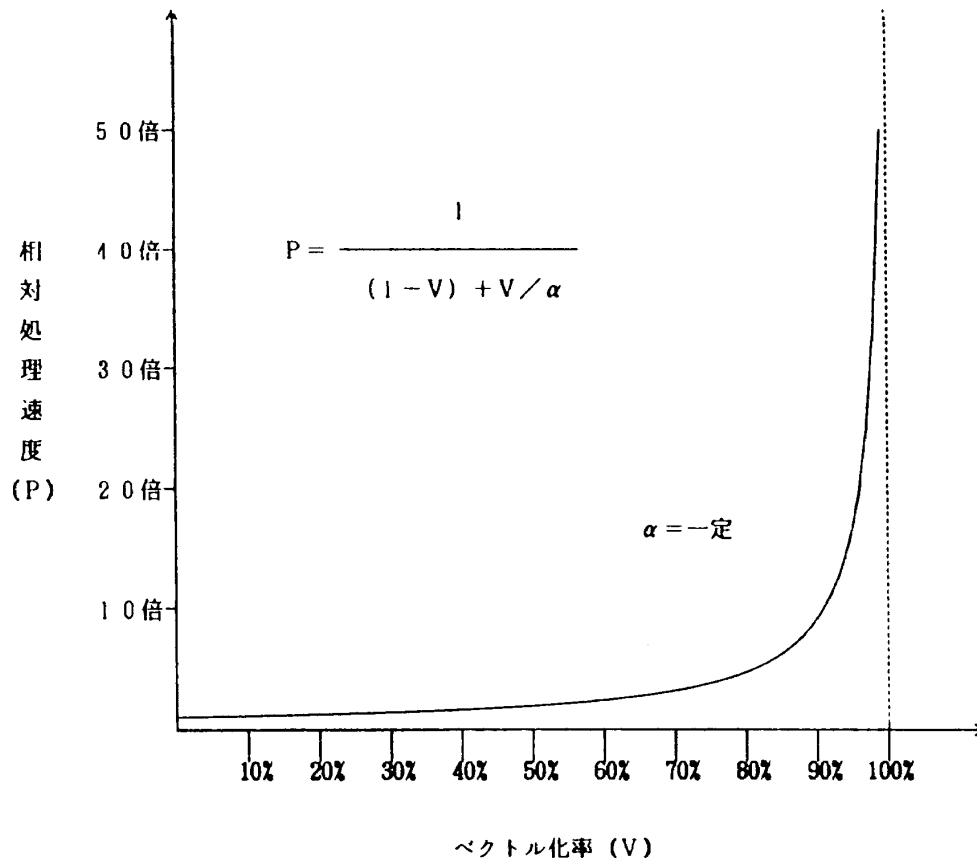


図 2.4 相対処理速度とベクトル化率

られるに過ぎないことである。従ってベクトル化率を可能な限り大きくすることが重要であり、99%以上を目指すべきである。

一方 α を大きくするということは、ベクトル化された部分のベクトル処理速度を大きくすることであり、これはまたベクトル化の質を測るものといえる。例えば100%のベクトル化率が達成されたとしても、 α が1であれば即ちベクトル/スカラ速度比が1であれば処理速度の向上は得られない。また極端な例としてベクトル計算のためのオーバーヘッドが大きいことのみがきいてくる条件の場合では100%のベクトル化率であってもベクトル計算機での実行時間の方が遅くなることもある。たとえば、DOループが完全にベクトル化されたとしてもベクトル長が5とか6など大変小さい値であれば、そのDOループをベクトル化するための準備に実行時間が費やされるので、ベクトル化のためのオーバーヘッドがかかることになる。 α を大きくするためには、並列動作可能なパイプラインの数を増やすこと、ベクトル長を

長くすること、ロード/ストアを出来るだけ減少させ効率の良いロード/ストアを行うこと等である。これらの実際的な面については4章で述べる。

3. 汎用プログラム的高速化

3.1 高速化対象汎用プログラムの概要

表3.1に高速化対象とした汎用プログラムの内容、次元、方程式、手法および計算法を示す。A～Fは2次元流を対象としたプログラム、G～Lは3次元流を対象としたプログラムである。

表3.1に示したプログラムを表3.2に示す条件で高速化し、実行時間の計測を行った。これら計算条件の中には、実際の計算条件のものもあれば、長時間かかるプログラムについては回数制限で短時間化したものもある。実行時間については次節で比較および検討を行う。

尚、汎用プログラムのチューニング作業は、短いもので1カ月、長いもので3～4カ月の期間を要した。

表 3.1 高速化汎用コードの概要

| コ ー ド | 目 的 | 次元、方程式名 | 手 法 | 計算法 |
|-------|------------------------|--------------|-------|------|
| A | 二次元遷音速流解析 | 二次元、フルポテンシャル | 差分法 | SLOR |
| B | 二次元遷音速流解析 | 二次元、フルポテンシャル | 差分法 | SLOR |
| C | 二次元遷音速翼設計 | 二次元、フルポテンシャル | 差分法 | SLOR |
| D | 二次元遷音速微小擾乱解析 | 二次元、TSD | 差分法 | SLOR |
| E | 遷音速翼型解析のための格子形成 | 二次元、ポアソン | 差分法 | SLOR |
| F | 二次元遷音速翼型解析 | 二次元、ナビエストークス | 差分法 | IAF |
| G | 三次元遷音速翼解析 | 三次元、フルポテンシャル | 差分法 | SLOR |
| H | 翼胴結合体適合格子形成 | 三次元 | 等角写像 | |
| I | 翼胴結合体まわり完全ポテンシャル流の数値解析 | 三次元、フルポテンシャル | 差分法 | SLOR |
| J | 全機形態のまわりの流れ解析 (対称) | 三次元、ラプラス | 境界要素法 | 直接法 |
| K | 全機形態のまわりの流れ解析 (非対称) | 三次元、ラプラス | 境界要素法 | 直接法 |
| L | ピッチング振動する三次元翼まわりのNS解析 | 三次元、ナビエストークス | 差分法 | ADI |

TSD : Transonic Small Disturbance 法
 SLOR: Succesive Line Over Relaxation 法
 IAF : Implicit Approximate Factorization 法
 ADI : Alternating Direction Implicit 法

3.2 汎用プログラムの各計算機での実行時間の比較

表 3.3(a)および表 3.3(b)は高速化対象となった汎用プログラムのオリジナルコードとチューニングコード(プログラムチューニングした汎用プログラム)を M780, VP200, VP400で実行した場合の実行時間(GOジョブステップの実行時間)の一覧表である。

表 3.3(c)および表 3.3(d)は、表 3.3(a)および表 3.3(b)のオリジナルコードの M780での実行時間を 1とした場合の各計算機でのオリジナルコードおよびチューニングコードの比率の一覧表である。

表 3.3(c)および表 3.3(d)において 1より小さい値は M780でのオリジナルコードの実行時間より多くの実行時間を必要としたことを意味し、また 1

より大きい値は M780でのオリジナルコードの実行時間より少ない実行時間で済むことを意味している。

オリジナルコードで比較すると E, Gおよび Lを除く 9本のプログラムで、汎用計算機 M780の方が実行時間が短くて済んでいる。ベクトル計算機での実行時間の方が短くなっている E, Gおよび Lにあってもその処理速度向上は、大きい場合でも 4倍弱である。この値は表 2.1のピーク演算処理速度の倍率, 65倍に比べて貧弱であり、ベクトル計算機の性能を活用しているとは言い難い。即ちプログラムチューニングが必要であると言える。

チューニングコードの VP とオリジナルコードの M780での実行時間の比は, 0.56~19.32 と 幅

表 3.2 高速化汎用コード実行条件

| コ ー ド | 格 子 | 反 復 回 数 | 備 考 |
|-------|-----------------|---|-------------------------------|
| A | COARSE 80×15 | ポアソンソルバ SLOR 1 } 20回 3 } | |
| | FINE 160×30 | ポアソンソルバ SLOR 1 } 10回 3 } | |
| B | COARSE 80×15 | ポアソンソルバ SLOR 1 } 20回 3 } | |
| | FINE 160×30 | ポアソンソルバ SLOR 1 } 10回 3 } | |
| C | 160×30 | 直接ステップ ポアソンスイープ 1 } SLORスイープ 6 } 70回 | 直接ステップ } 逆ステップ } 3回繰り返す |
| | | 逆ステップ SLORスイープ 6 } 20回 ポアソンスイープ 1 } SLORスイープ 6 } 70回 | |
| D | COARSE 20×14 | 125 | |
| | MEDIUM 39×28 | 130 | |
| | FINE 77×56 | 260 | |
| E | 103×51 | ラプラス 876 ポアソン 449 | |
| F | 125×51 | 1000 | |
| G | COARSE 48× 6× 8 | 100 | |
| | MEDIUM 96×12×16 | 100 | |
| | FINE 192×24×32 | 100 | |
| H | 176×24×32 | | |
| I | COARSE 44× 6× 8 | 200 | |
| | MEDIUM 88×12×16 | 200 | |
| | FINE 176×24×32 | 100 | |
| J | 560パネル | | |
| K | 560パネル | | |
| L | 161×29×35 | 100 | |

SLOR: Succesive Line Over Relaxation 法

表 3.3(b) 高速化汎用コード(3次元)の各システムの
GO ジョブステップ実行時間

| コ ー ド | | VP400 (秒) | VP200 (秒) | M780 (秒) |
|-------|---|-----------|-----------|----------|
| | | | | |
| G | O | 51.82 | 61.80 | 182.14 |
| | T | 41.14 | 50.10 | 154.34 |
| H | O | 231.66 | 241.49 | 178.40 |
| | T | 215.00 | 223.69 | 172.01 |
| I | O | 1239.85 | 1360.19 | 660.82 |
| | T | 329.72 | 342.94 | 600.91 |
| J | O | 30.40 | 31.73 | 17.64 |
| | T | 1.25 | 1.85 | 14.08 |
| K | O | 71.82 | 74.75 | 49.84 |
| | T | 2.58 | 3.73 | 54.55 |
| L | O | 625.14 | 677.39 | 2428.88 |
| | T | 193.05 | 242.09 | 2332.00 |

O:オリジナルコード

T:チューニングコード

表 3.3(a) 高速化汎用コード(2次元)の各システムの
GO ジョブステップ実行時間

| コ ー ド | | VP400 (秒) | VP200 (秒) | M780 (秒) |
|-------|---|-----------|-----------|----------|
| | | | | |
| A | O | 4.35 | 4.62 | 2.12 |
| | T | 3.44 | 3.78 | 2.14 |
| B | O | 3.98 | 4.24 | 2.20 |
| | T | 3.50 | 3.80 | 2.22 |
| C | O | 49.39 | 52.11 | 27.73 |
| | T | 20.50 | 22.46 | 31.60 |
| D | O | 4.57 | 4.83 | 3.99 |
| | T | 2.09 | 2.27 | 4.58 |
| E | O | 16.59 | 17.85 | 29.16 |
| | T | 4.48 | 4.91 | 19.89 |
| F | O | 811.52 | 859.79 | 726.76 |
| | T | 99.26 | 113.57 | 923.70 |

O:オリジナルコード

T:チューニングコード

表 3.3(c) 高速化汎用コード (2次元) の各システムの
GO ジョブステップ実行時間の比率

| コ ー ド | | VP400 | VP200 | M780 |
|-------|---|-------|-------|------|
| | | | | |
| A | O | 0.49 | 0.46 | 1 |
| | T | 0.62 | 0.56 | 0.99 |
| B | O | 0.55 | 0.52 | 1 |
| | T | 0.63 | 0.58 | 0.99 |
| C | O | 0.56 | 0.53 | 1 |
| | T | 1.35 | 1.23 | 0.88 |
| D | O | 0.87 | 0.83 | 1 |
| | T | 1.91 | 1.76 | 0.87 |
| E | O | 1.76 | 1.63 | 1 |
| | T | 6.51 | 5.94 | 1.47 |
| F | O | 0.90 | 0.85 | 1 |
| | T | 7.32 | 6.40 | 0.79 |

O:オリジナルコード
T:チューニングコード

表 3.3(d) 高速化汎用コード (3次元) の各システムの
GO ジョブステップ実行時間の比率

| コ ー ド | | VP400 | VP200 | M780 |
|-------|---|-------|-------|------|
| | | | | |
| G | O | 3.51 | 2.95 | 1 |
| | T | 4.43 | 3.64 | 1.18 |
| H | O | 0.77 | 0.74 | 1 |
| | T | 0.83 | 0.80 | 1.04 |
| I | O | 0.53 | 0.49 | 1 |
| | T | 2.00 | 1.93 | 1.10 |
| J | O | 0.58 | 0.56 | 1 |
| | T | 14.11 | 9.54 | 1.25 |
| K | O | 0.69 | 0.67 | 1 |
| | T | 19.32 | 13.36 | 0.91 |
| L | O | 3.89 | 3.59 | 1 |
| | T | 12.58 | 10.03 | 1.04 |

O:オリジナルコード
T:チューニングコード

表 3.4(a) 高速化汎用コード(2次元)の各システムの
GO ジョブステップ実行時間

| コード | VP400 (秒) | | | | | VP200 (秒) | | | | |
|-----|-----------|--------|--------|--------|-------|-----------|--------|--------|---|-------|
| | V | U | S | U | C P U | V | U | S | U | C P U |
| A | O | 0.97 | 3.38 | 4.35 | | 1.01 | 3.61 | 4.62 | | |
| | T | 0.78 | 2.66 | 3.44 | | 0.92 | 2.86 | 3.78 | | |
| B | O | 0.87 | 3.11 | 3.98 | | 0.91 | 3.33 | 4.24 | | |
| | T | 0.82 | 2.68 | 3.50 | | 0.92 | 2.88 | 3.80 | | |
| C | O | 13.34 | 36.05 | 49.39 | | 13.73 | 38.38 | 52.11 | | |
| | T | 5.67 | 14.83 | 20.50 | | 6.43 | 16.03 | 22.46 | | |
| D | O | 0.47 | 4.10 | 4.57 | | 0.51 | 4.32 | 4.83 | | |
| | T | 0.62 | 1.47 | 2.09 | | 0.69 | 1.58 | 2.27 | | |
| E | O | 1.63 | 14.96 | 16.59 | | 2.23 | 15.62 | 17.85 | | |
| | T | 2.45 | 2.03 | 4.48 | | 2.78 | 2.13 | 4.91 | | |
| F | O | 288.90 | 522.62 | 811.52 | | 310.31 | 549.48 | 859.79 | | |
| | T | 76.05 | 23.21 | 99.26 | | 87.52 | 26.05 | 113.57 | | |

O: オリジナルコード
T: チューニングコード
VU: ベクトルユニット実行時間
SU: スカラーユニット実行時間
CPU: 実行時間

表 3.4(b) 高速化汎用コード(3次元)の各システムの
GO ジョブステップ実行時間

| コード | VP400 (秒) | | | | | VP200 (秒) | | | | |
|-----|-----------|--------|--------|---------|-------|-----------|---------|---------|---|-------|
| | V | U | S | U | C P U | V | U | S | U | C P U |
| G | O | 10.09 | 41.73 | 51.82 | | 18.28 | 43.52 | 61.80 | | |
| | T | 17.31 | 23.83 | 41.14 | | 24.52 | 25.58 | 50.10 | | |
| H | O | 0.13 | 231.53 | 231.66 | | 0.16 | 241.33 | 241.49 | | |
| | T | 6.01 | 208.99 | 215.00 | | 6.31 | 217.38 | 223.69 | | |
| I | O | 253.80 | 986.05 | 1239.85 | | 321.71 | 1038.48 | 1360.19 | | |
| | T | 131.58 | 198.14 | 329.72 | | 139.25 | 203.69 | 342.94 | | |
| J | O | 2.91 | 27.49 | 30.40 | | 3.04 | 28.69 | 31.73 | | |
| | T | 1.09 | 0.16 | 1.25 | | 1.54 | 0.31 | 1.85 | | |
| K | O | 9.64 | 62.18 | 71.82 | | 10.06 | 64.69 | 74.75 | | |
| | T | 2.31 | 0.27 | 2.58 | | 3.28 | 0.45 | 3.73 | | |
| L | O | 224.53 | 400.61 | 625.14 | | 260.02 | 417.37 | 677.39 | | |
| | T | 172.96 | 20.09 | 193.05 | | 220.13 | 21.96 | 242.09 | | |

O: オリジナルコード
T: チューニングコード
VU: ベクトルユニット実行時間
SU: スカラーユニット実行時間
CPU: 実行時間

広い高速化倍率の分布を示している。A, BおよびHでは M780より遅くなっており, これらのプログラムがベクトル計算機向きでないことを意味している(これらを高速処理するには計算スキーム, アルゴリズムに戻って考える必要がある)。残りのプログラムについては改善が見られ, M780より遅かったものは M780より速くなり, もともと M780より速かったものもさらに速くなっている。

オリジナルコードとチューニングコードの M780での実行時間を比べてみると 1 を境にして 2 種類に分けることができる。1 より大きい E, G, H, I, J および L は, チューニングコードの方がオリジナルコードよりも速くなっていることを意味し, プログラムチューニングが汎用計算機にとっても有効な内容を含むことを示している。一方 1 より小さい A, B, C, D, F および K は, チューニングコードの方がオリジナルコードよりも遅くなっていることを意味し, 汎用計算機での計算時間が増えるようなプログラムチューニングであってもベクトル計算機での実行時間が短くなる場合が

あることを示している。

表 3.4 はオリジナルコードとチューニングコードの VP200 および VP400 での実行時間 (cpu), ベクトルユニット使用時間 (vu), スカラユニット使用時間 ($su = cpu - vu$) を示したものである。但し, su はベクトルユニット走行中のスカラユニット走行時間を含まない。

表 3.3 および表 3.4 のデータを基に相対実行時間を視覚化したのが図 3.1 (1)~(3) である。ここでは, オリジナルコードの M780 での実行時間を 100 とした時のオリジナルコードおよびチューニングコードの VP400 での実行時間を示す。図中, s は su を v は vu を表す。M780 と M380 の性能比は 2.25 (表 2.1) であるのでオリジナルコードを M380 で実行した場合の相対実行時間を最右端の縦線で示す。これはオリジナルコードを VP400 でスカラ実行した場合の実行時間と考えてよい。

図 3.1 の見方を F を例に説明するため, これを図 3.2 として再掲する。オリジナルコードの M380 での実行時間を a , M780 での実行時間を b , VP400 での実行時間を c , チューニングコードの VP

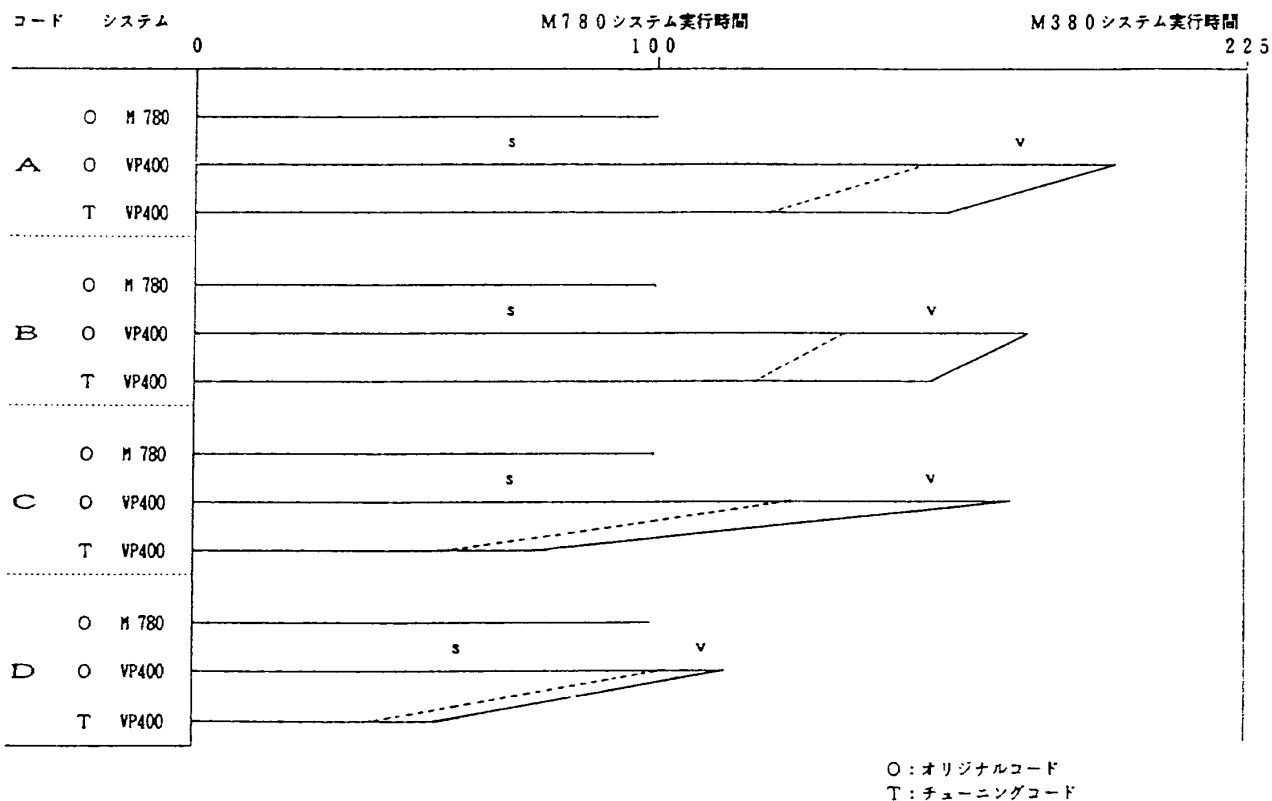


図 3.1 汎用プログラムの各種システムの実行時間(1)

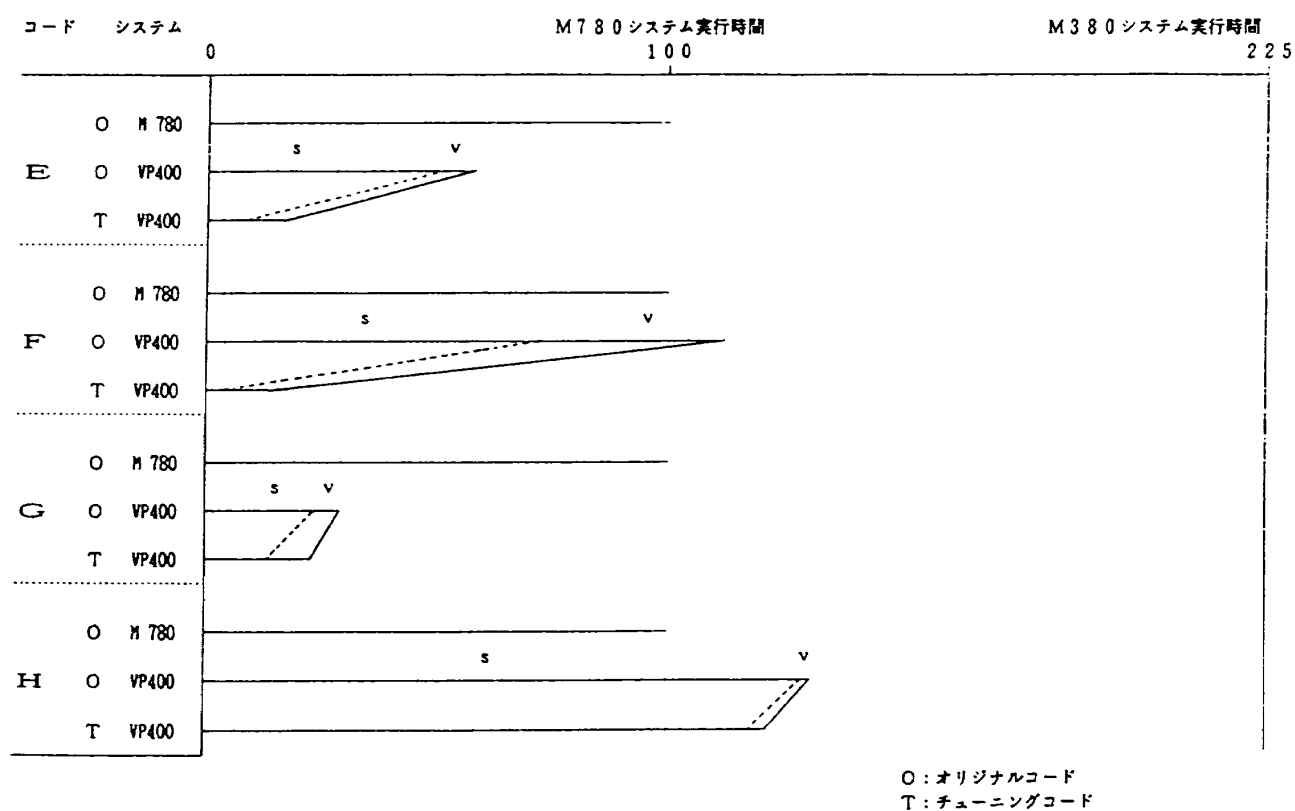


図 3.1 汎用プログラムの各種システムの実行時間(2)

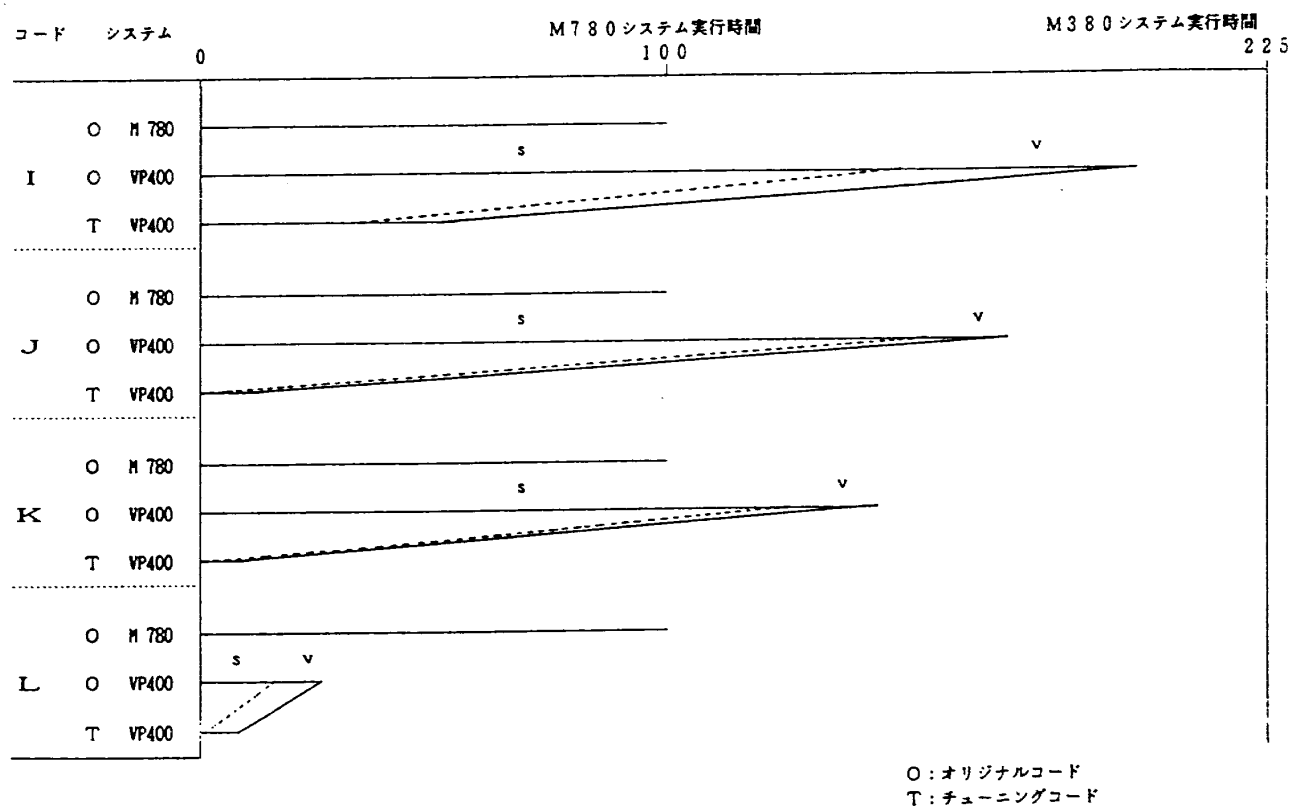


図 3.1 汎用プログラムの各種システムの実行時間(3)

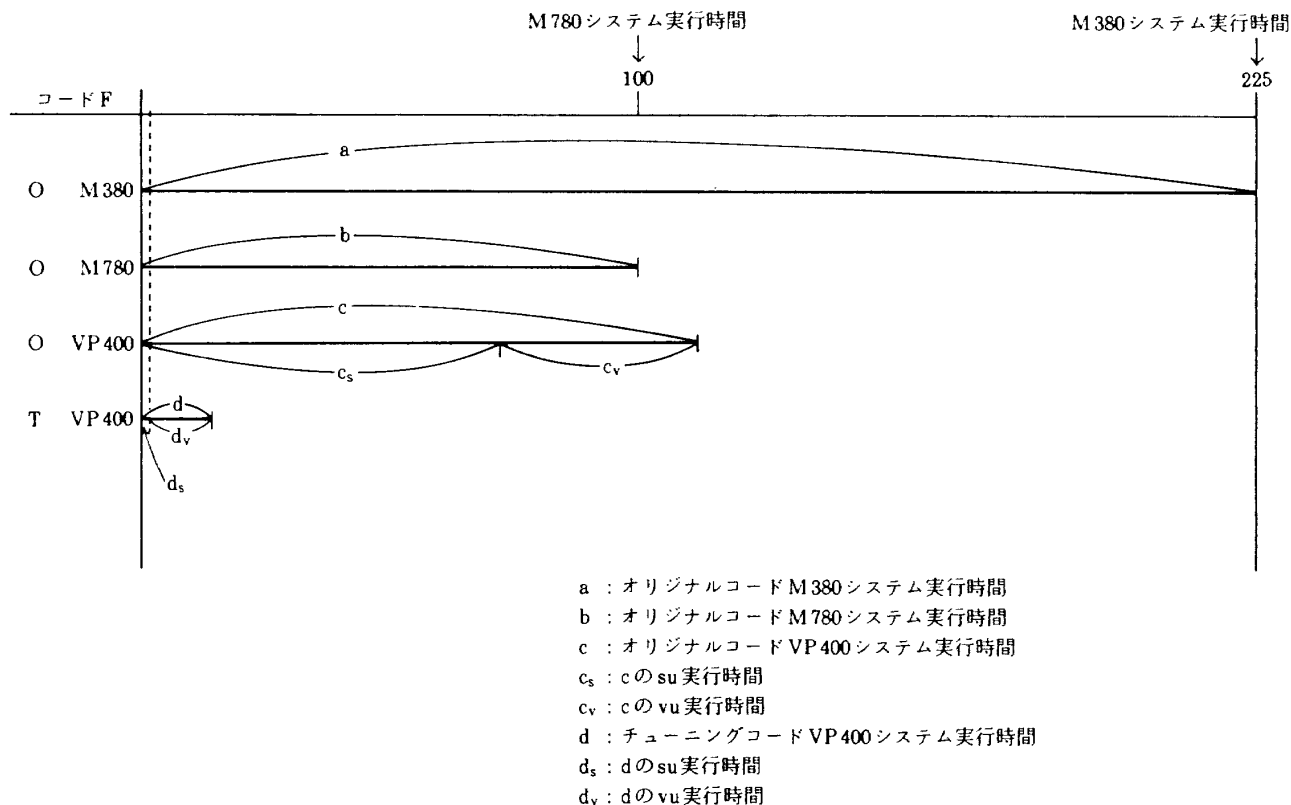


図 3.2 プログラム F の例

400での実行時間をdとする。またcおよびdのsu, vuをそれぞれ c_s , d_s , c_v および d_v で表わす。

$a > c > b$ であることより、オリジナルコードに何も手を加えなくてもVPをスカラ計算機として使用するよりは速いが、汎用計算機M780よりは遅いことがわかる。 $d < c$ によりプログラムチューニングすることにより高速化されたことがわかる。このように定性的なことがらはこの図からすぐに読み取れる。

定量的な評価は2.2で定義したVと α を

$$V = (a - d_v) / a$$

$$\alpha = (a - d_s) / d_v$$

によって計算すればよい。この場合には $V = 0.986$, $\alpha = 21.2$ となり、Fではプログラムの大部分がベクトルユニットで処理され、しかもスカラ速度比は21倍にもなっている（この場合、スカラ速度比はM380に対するものであるので表2.1のピーク演算処理速度の倍率は148倍に対応するものである）。

表3.5は各プログラムのVおよび α の値の一覧表である。高速化倍率の悪いプログラムA, Bお

よびHのうちAとBはV, α 共に値が小さくこれの高速化のためには多方面からの検討が必要と考えられる。1ケースの計算は数秒で終了するが、境界層補正を行うこともできるので、その場合には約10倍の計算時間が必要である。それが頻繁に使用されるのであれば高速化の必要性がある。しかし、境界層補正を行う計算が少なければ高速化倍率が低くてもさほど影響はないので高速化の必要性はないと考えられる。一方プログラムHについては、 α が大きいのでVを大きくできるかどうかの検討が必要になる。このプログラムは3次元翼胴結合体廻りに計算格子を生成するもので、対象を変えて何度も計算する場合には高速化を図る必要があるが、一度格子を生成すればそれをデータとして何度も流れ場を計算するというのであれば比較的高速化の必要性は小さくなる。このプログラムの高速化には、形状操作の部分等プログラム構成を変更したりあるいは等角写像という方法を変更する必要があると考えられる。

一方高速化倍率の高いプログラムJ, KおよびLにあっては、ベクトル化率Vは99%を越えてお

表 3.5 高速化汎用コードのベクトル化率およびベクトル/スカラ速度比

| コード | ベクトル化率 ($V \times 100\%$) | ベクトル/スカラ速度比 (α 倍) | 高速化倍率* |
|-----|-----------------------------|---------------------------|--------|
| A | 44.2% | 2.7倍 | 0.62 |
| B | 45.9% | 2.8倍 | 0.63 |
| C | 76.2% | 8.4倍 | 1.35 |
| D | 83.6% | 12.1倍 | 1.91 |
| E | 96.9% | 26.0倍 | 6.51 |
| F | 98.6% | 21.2倍 | 7.32 |
| G | 94.2% | 22.3倍 | 4.43 |
| H | 47.9% | 32.0倍 | 0.83 |
| I | 86.7% | 9.8倍 | 2.00 |
| J | 99.6% | 36.3倍 | 14.11 |
| K | 99.8% | 48.4倍 | 19.32 |
| L | 99.6% | 31.5倍 | 12.58 |

*高速化倍率 = M780 システム実行時間 / VP400 システム実行時間

り、その高速化倍率の相違は α の大小によっているのがわかる。この表の結果から大雑把にいて、高速化倍率を上げるには『まずベクトル化率 V を大きくする。しかる後ベクトル/スカラ比 α を大きくすることを考える』という手順になる。

次にチューニングコードのVP200とVP400での実行時間の比較を行う。図3.3にチューニングコードのVP200での実行時間を1とした場合のVP400での実行時間を示す。図中 s は su 、 v は vu を表し、各計算機での実行時間の先端を実線で、 su を点線で結んである。また表2.1のピーク演算処理速度からVP400はVP200の約2倍の性能があることになるが、現実的な処理条件下では1.7倍程度が一つの努力目標値と思われる。そのため図中ほぼ中央の縦線でVP200での実行時間を1と

して、VP400がその1.7倍の性能が出た場合の実行時間を示してある。

図3.3では各コードのグラフの先端を結んだ角度が鋭い程、また中央縦線に近い程VP200とVP400での実行時間に差があることを示し、VP400の特徴を活かしたプログラムチューニングになっているといえる。これらをかなりの程度満足しているのは3次元を対象としているプログラムJおよびKだけであり、同じく3次元のプログラムGおよびLがこれに続く。同じ3次元のプログラムでありながらHおよびIでは数%の短縮にしかない。2次元のプログラム(A~F)ではプログラムの区別なく10%前後の短縮となっている。

また点線で結ばれた su をVP200とVP400で比

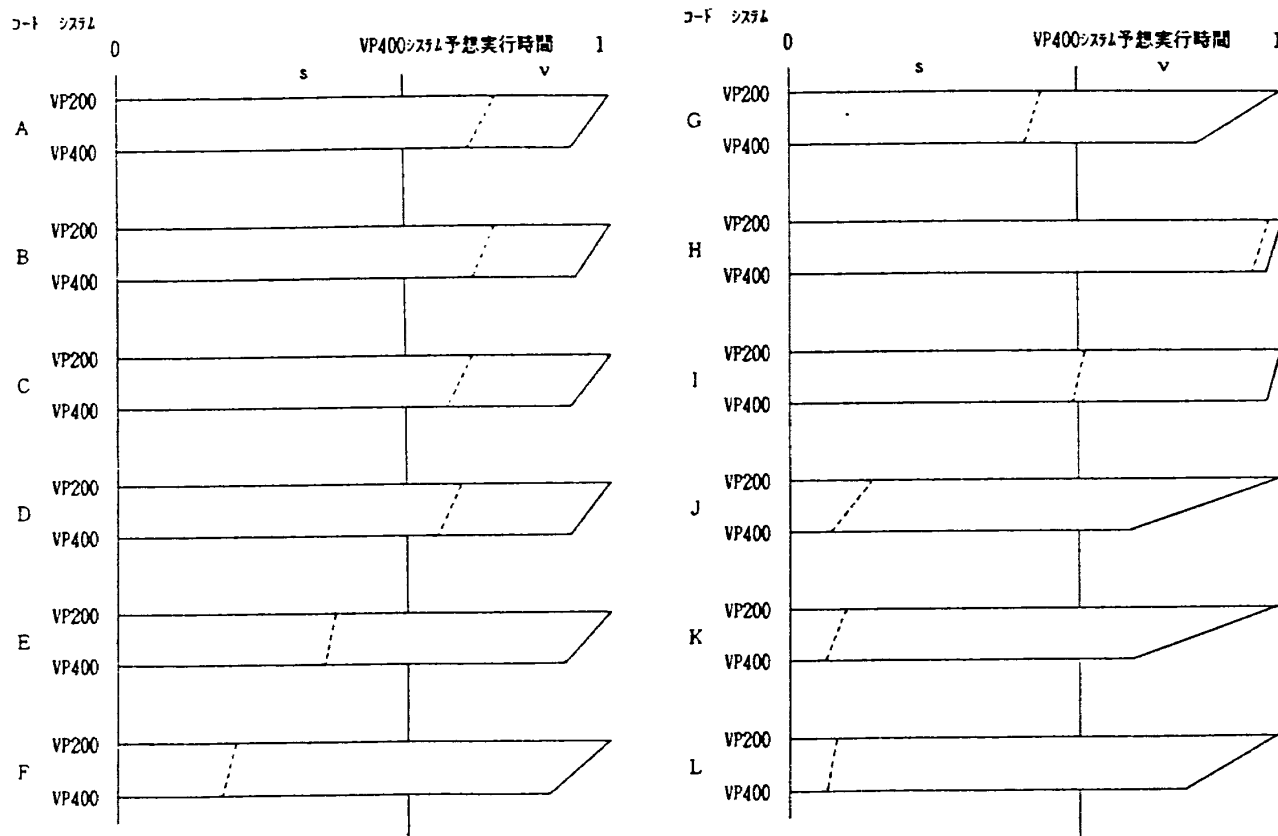


図 3.3 チューニングコードの VP 200 システムおよび VP 400 システム実行時間

較するとすべて VP 400 の方が小さい。これは一部には VP 200 と VP 400 のマシンサイクルの違いもあるが、コンパイラが生成するオブジェクトの違いが大きいと考えられる。

これまでの議論は高速化の相対的数値に基づいてきたが、多くの計算時間を必要とするプログラムにあっては、たとえ高速化倍率が小さくても(1よりは大きくて)得られる絶対的時間短縮は大きいものとなる。即ち実行時間の長いプログラムほど努力の必要性がある。

4. 高速化手法^{5), 6)}

航技研の汎用プログラムをスーパーコンピュータ向きに高速化するための作業は大きく次の三つに分類される。

- 1) FORTRAN 77 レベルへの移行
- 2) プログラム構造の整理
- 3) 高速化のためのチューニング

1) は ENCODE 文の修正等、2) はプログラムの計算部と入出力部の分離や配列の大きさを可変に

するという内容であり、すべての汎用プログラムについて行われたものでないことと高速化とは直接関係がないことから本章では、1) および 2) については記述せず、3) について述べる。

全汎用プログラムについて高速化手法をまとめ手法別に述べるのが最も理解しやすい形式になると思えたが、実際に分類を行うとベクトル化されない理由が様々であることや高速化の一方法が様々な手法を含むことなどの理由により分類が困難である。また、無理に分類をして手法説明を行うと整いすぎて実際のコーディングと多少異なり、マニュアル説明と同様のものになったり全チューニングを載せられなかったりする。また、整理されたコーディングは実際にユーザが書いたコーディングと離れてしまって理解しにくくなる可能性がある。

これらの理由によりチューニング説明はプログラム別に行うこととした。しかし、同一プログラムにおいて同一のチューニングが何度も適用されている場合には、繁雑さを避けるため最初に一度

説明することとした。また、同様のチューニングが他のプログラムでも施されている場合にはチューニング説明は除き、どのチューニングが施されたかを述べるにとどめる。説明方法は、まずチューニング内容の説明を行い、それを行う理由や効果について述べる。プログラムを示す際にはオリジナルコードの部分プログラム、または理解しやすい例を用いて説明する。

本章に載せた手法は50を越える。これらは、プログラムコーディング、ベクトル化不可の原因、チューニング法およびチューニング目的の組み合わせにより多くの数となった。簡単なチューニングもあればプログラムの大改造を必要とするチューニングもある。また、同じテクニックを用いてプログラム全体にわたり修正したものもある。したがって、チューニングに要する時間と効果にそれほど相関はない。

チューニングは回帰演算や配列の定義・参照の矛盾によりベクトル化されない DO ループをベクトル化されるように修正したもの、すなわち、ベクトル化の促進を目的にしたものが多い。そして、その過程で効率的なベクトル化や省力化を行っている。

高速化手法の効果についてはほとんど触れなかった。それは、その手法が適用されるプログラムのステートメントの実行回数や演算命令の種類および数、ベクトル長等に左右されるからである。したがって、同一のチューニング法でもプログラム X には効果があり、プログラム Y には効果がない場合があり得る。なお、プログラムの高速化に最も寄与したチューニング内容のものがある場合には各節の最後に述べることにする。

また、図が多いので煩雑さを避けるため、4章の図のみ巻末に載せる。図の表題は、たとえばプログラム A のチューニング 1 の変更前のプログラムは、

A-1 変更前のプログラム

というように、汎用プログラムの記号およびチューニング番号を示すものとする。

4.1 プログラム A の高速化

(1) チューニング 1

部分ベクトル化されている DO ループ内の単変数を配列にして DO ループを完全ベクトル化した。理解を容易にするため図 4.1.1 に例を示す。

図 4.1.1(a)において DO ループ中の変数 X 1 および X 2 は前回の反復における値を参照しているので、変数の定義・参照が逆になることから回帰演算となる。そのために DO ループは完全にベクトル化されない。変数を配列にすることにより回帰的關係が解除されるので DO ループは完全にベクトル化される。チューニング後のコーディングを図 4.1.1(b)に示す。

プログラムの高速化のためには VP システムのベクトルユニットで走行する部分をできるだけ多くすることが基本である。すなわち、FORTRAN プログラムにおいてベクトル化される部分ができるだけ多い方がよい。ベクトル化の対象は DO ループで書かれる部分であるので DO ループのベクトル化をできるだけ図り、ベクトル処理部を拡大する必要がある。

DO ループがスカラ処理される場合（ベクトル化されない場合）、原則としてベクトルユニットは使用されない。また、部分ベクトル化された DO ループはスカラ処理部分とベクトル処理部分の混在によるオーバーヘッドがかかる。完全にベクトル化された DO ループにはそれがないので効率よく実行される。

本チューニングはプログラムのベクトル処理部の拡大を目的とするものである。

(2) チューニング 2

頻繁に呼び出されるサブルーチンを呼び出し元に展開した。理解を容易にするため図 4.1.2 に例を示す。

サブルーチンの 1 回の呼び出しに要するオーバーヘッドは小さいが 100 万回も呼び出される場合、そのオーバーヘッドは秒のオーダーとなる（本プログラムで最も呼び出し回数の多いサブルーチンは約 1 万回である）。引数が多い程オーバーヘッドも大きい。M380 システムにおいて引数なし、呼び出し回数 100 万回実行の場合、サブルーチン呼び出

しに対するオーバーヘッドは約1.5秒である。また、引数がある場合その引数の数に応じて（1個当たり0.1～0.2秒程度）CPU時間は増加する。

サブルーチンを呼び出し元に展開することにより呼び出しのためのオーバーヘッドがなくなり実行時間は短縮される。

CALL 文はベクトル化非対象の FORTRAN 文である。従って、DO ループ内に CALL 文が存在するとその DO ループはベクトル化されない。DO ループのベクトル化されない理由が CALL 文のみにある場合にはこれを解除することにより DO ループがベクトル化される可能性があり、その場合にはベクトル化の促進が行える。本プログラムにおけるチューニングでは展開により DO ループのベクトル化までには至らなかった。

4.2 プログラム B の高速化

4.1 のチューニング 1 およびチューニング 2 と同様の修正を行った。

4.3 プログラム C の高速化

(1) チューニング 1

ベクトル化されない DO ループ内での要素位置（インデックス）計算を DO ループ外で予め行い、DO ループを二つに分割した。また、ベクトル長の長いインデックスで DO ループを作成した。オリジナルおよびチューニング後のコーディングを図 4.3.1 に示す。

図 4.3.1(a)において DO ループ中に現れる配列のインデックスを DO ループ内で求めてその位置の配列要素を参照している。そして規則的間隔で積算している。そのため IQ が参照・定義という順になり回帰的になっていることなど、複雑なコーディングとなっていることによりベクトル化されない。DO ループがベクトル化されるように、要素位置を予め、別の DO ループで計算して求めておき、積算を行う時にはそのインデックスを用いて、しかも、NS*ND というより長いベクトル長の DO ループを作成した。

本チューニングによりベクトル化の促進およびベクトル長の拡大による高速化効果が得られる。

本チューニングを施したサブルーチンのベクトル化率は 8.7% から 88.0% に拡大した（チューニングツール FORTUNE による）。

(2) チューニング 2

DO ループ内へのサブルーチン展開により生じた不要の LOAD/STORE 命令を削除した。オリジナルおよびチューニング後のコーディングを図 4.3.2 に示す。

図 4.3.2(a)では変数 IXX の値により配列 A, B, C, D, E, RP の位置をずらしている。また、N の値を N-1 に置き換えている。三項方程式を解くサブルーチン実行後、配列 A から RP の位置を元に戻し、N の値は N に戻している。これにより、図 4.3.2(a)の線で囲まれた部分の DO ループをすべて削除でき、ステートメント一行について 100 万回実行されている LOAD/STORE 命令がすべて行われなくなる。

(3) チューニング 3

4.1(1)と同様のチューニングを行った。

(4) チューニング 4

4.1(2)と同様のチューニングを行った。

プログラム C の高速化に寄与したチューニングはチューニング 1 および 3 である。

4.4 プログラム D の高速化

(1) チューニング 1

DO ループ内にあらわれる関数で求められる値を予め計算しておき、DO ループ内では関数を呼ぶのではなく配列参照の形式にして、DO ループをベクトル化した。図 4.4.1 にオリジナルおよびチューニング後のコーディングを示す。

図 4.4.1(a)の下線のついた PX は関数でありベクトル化非対象であるため DO ループはベクトル化されない。関数 PX は I の値により PX の計算方法を変えている。I が変化する度に関数 PX が呼ばれ、IF 判定が行われ、PX が計算される。この DO ループをベクトル化するために新しくサブルーチン PXCOM を作成し、その中で PX をまとめて計算し、DO ループ内では計算された値を参照するようにした。これにより、DO ループはベクトル化された。サブルーチン PXCOM では

PX が配列 PXC にストアされる。また、I が端点の場合のみ PX の計算式が異なるので、端点の判別を行うための IF 判定を除いた DO ループ計算となっている。

これらの手法を施すことにより、ベクトル化の促進、IF 判定削除および関数呼び出しによるオーバーヘッドの減少による高速化効果がある。

(2) チューニング 2

部分ベクトル化されている DO ループ内の IF 判定を減少させ、DO ループを分割し、それぞれの DO ループをベクトル化した。図 4.4.2 にオリジナルおよびチューニング後のコーディングを示す。

図 4.4.2(a)の㉑および㉒の部分は形式が同じであるので図 4.4.2(b)の㉓の部分のようにまとめることができる。また、図 4.4.2(a)の㉔および㉕のステートメントは同じ変数 IEM を異なる IF 判定で使用しているためにこの部分は部分ベクトル化となっている。この IF 判定は配列 EMIL および EMIU に格納されている値の中で 1.3 を超えるものが 1 つでもあれば IEM=1 とする内容であるので、図 4.4.2(b)の㉖のようにコーディング変更を行った。

本チューニングはベクトル化の促進を目的とする。

(3) チューニング 3

DO ループ内の呼び出し元へ関数を展開し、IF-GO TO 文を IF-THEN-ENDIF 文にして DO ループをベクトル化した。さらに、プログラム構造を単純にした。図 4.4.3 にオリジナルおよびチューニング後のコーディングを示す。

図 4.4.3(a)の㉗および㉘の IF 判定は最大値を求めるものであるが、図 4.4.3(b)の㉙および㉚のように変更した。最大値を求めるコーディングはこの形式でベクトル化される。また、関数 PX では I の値により PX の計算式が異なるため IF 判定が行われており、オーバーヘッドがかかるので I の値によって、図 4.4.3(b)の㉛または㉜を選択する形式に変更した。

これらの手法により、ベクトル化の促進、プログラム構造の単純化によるオーバーヘッドの削減による高速化効果が得られる。

(4) チューニング 4

IF-GO TO 文によりベクトル化されない DO ループを IF-THEN-ENDIF 文に変更してベクトル化した。

図 4.4.4(a)は DO ループ内で ABS (RHS(J)) の最大値およびその位置を求めている。これを図 4.4.4(b)のように IF-THEN-ENDIF 文でコーディングしなおすことにより DO ループはベクトル化され、ベクトル化の促進が行える。

(5) チューニング 5

複数の DO ループを合併して 1 つの DO ループにした。図 4.4.5 にオリジナルおよびチューニング後のコーディングを示す。

図 4.4.5(a)の DO 30, DO 40, DO 50 ループの繰返し範囲が JBOT から JTOP まですべて同一であるので合併することができ、図 4.4.5(b)の㉝のように変更できる。DO 50 ループの配列 RHS は最終的には不要の配列であるので変数に変えた。

DO ループの合併による立ち上がり時間の削減、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。また、合併により不要となった配列はテンポラリ変数とすることにより不要の LOAD/STORE 命令をなくすることができる。

(6) チューニング 6

部分ベクトル化のオーバーヘッドが多いため、ベクトル化されない DO ループを分割してそれぞれの DO ループをベクトル化した。

図 4.4.6(a)の㉞の部分を図 4.4.6(b)の㉟および㊱のように変更して、IF 判定のない形式にして、DO ループをベクトル化した。これにより、ベクトル化の促進が行えた。

(7) チューニング 7

DO ループの合併、配列要素の拡大等を行い、プログラム構造を単純化した。図 4.4.7 にオリジナルおよびチューニング後のコーディングを示す。

図 4.4.7(a)の㊲および㊳の DO ループを合併し、D の部分では配列 P の列要素数を前後に広げて J が JBOT および JTOP の場合も DO ループ内で計算可能とした。ただし、これは広げた配列の値が 0 になっていることが条件である。これらの修正

により図 4.4.7(a)の⑥の部分は削除され、図 4.4.7(b)のように構造を変えた。

DO ループの合併についてはチューニング 5 に述べた効果がある。また、構造の単純化による効果は DO 21 ループを削除できたことや図 4.4.7(a)の④の部分の IF 文の削除などの余分な処理の削除をあげることができる。

(8) チューニング 8

ベクトル化されない DO ループ内の変数を配列化することにより DO ループをベクトル化した。図 4.4.8 にオリジナルおよびチューニング後のコーディングを示す。

図 4.4.8(a)の変数 WPX1, WPX2, WPX3 および WPX4 の参照が定義に先行する可能性があるので DO ループはベクトル化されない。DO ループをベクトル化するために変数の配列化を行って、図 4.4.8(b)のように変更した。この結果、DO ループはベクトル化され、ベクトル化の促進が行えた。

(9) チューニング 9

サブルーチン SYOR では配列 P(J, I) について新しい値を求めている。P を求める計算は J および I の両方向とも回帰演算となるのでベクトル化は不可能であるが、ベクトル化可能部分を抽出して最大限のベクトル化を図った。

改造前の処理フローおよび改造後の処理フローを図 4.4.9 に示す。図中、DO(i) は i 方向の DO ループであることを示す。また、図 4.4.9(a)において→⑧は図 4.4.9(b)の⑧の部分に移ったことを示す。図をみるとわかるように構造がかなり変更されたことがわかる。たとえば図 4.4.9(a)の③は図 4.4.9(b)の⑧、⑨、⑩の部分に分散した。これは③では P の新しい値および古い値の両方を使用するので、新しい値を使用する計算部は図 4.4.9(b)の点線より下の部分で計算し、P の古い値を使用する計算部は点線より上の部分で計算するように変更した。また、サブルーチン BCEND の境界条件の計算においても、P の新しい値を用いる計算部は図 4.4.9(b)の点線以下に、P の古い値を用いる計算部は新サブルーチン BCEND2 で行う。本チューニングの概要は以上であるが、サブルーチンを載せても何頁にもわたり、複雑になってい

るので理解しにくいと思われる。そこで具体的にどのような手法が効果を得たかを以下に説明する。

(イ) 配列を多次元化する。

図 4.4.10(a)はオリジナルプログラムの図 4.4.9(a)の⑥の一部である。DO ループは I でまわっているため⑥の部分はスカラー演算となる。そこで一次元配列 SUP(J)等を二次元配列 SUP(J, I)として I でまわる DO ループをつくることによりベクトル化した。その結果、図 4.4.10(b)に示すコーディングとなり、ベクトル化の促進が行われた。

(ロ) IF 文を別の形の判定に置き換えて削除した。

1) 図 4.4.11(a)の下線部は $EMU(J, I1) \geq 0$ ならば 0 を $VC(J) < 0$ ならば $VC(J)$ をストアすることを目的としている。これを図 4.4.11(b)の下線部のようにコーディングすることにより、 $EMU(J, I)$ に 0 または $VC(J, I)$ の値が入るので IF 文を削除できる。

DO ループ内の IF 文の実行は他の演算命令の実行に比較して時間がかかる。IF 文を使わずにコーディングできる場合には使わない方がよい。

2) 図 4.4.12(a)において、FCR が真と偽であることの違いは下線のあるステートメントのみである。これにより二重下線のついた配列 EMU の値が異なるわけであるが、これは FCR の真偽により古い値を用いるか新しい値を用いるかを決定している。そのため図 4.4.12(b)のコーディングとなり FCR の IF 判定を削除できる。効果は 1) で述べたことと同様である。

(ハ) DO ループの分割による高速化処理を行った。

図 4.4.11(b)において DO 1110 ループおよび DO 1210 ループは元は同一 DO ループであったがベクトル化されないため、二つの DO ループに分割してそれぞれの DO ループをベクトル化した。ベクトル化されない理由は $EMU(J, I)$ の値が同一ループ内では保証されないからである。本チューニングによりベクトル化の促進が行えた。

(ニ) サブルーチン内でのスカラー処理を配列処理に変更した。

図 4.4.13(a)の構造を図 4.4.13(b)のように変

更してサブルーチンの中では呼び出される度に1つずつ処理されていたのをサブルーチンの中で複数の処理を行えるようにした。その結果、サブルーチン内での処理がDOループで行えるようになりベクトル化されたのでベクトル化の促進が行えた。また、これにともなうサブルーチンの呼び出し回数の減少によりサブルーチン呼び出しのオーバーヘッドが削除できた。

(10) チューニング10

ベクトル化されないDOループ内に現れる関数をDOループ内に展開してDOループをベクトル化した。4.1(2)のチューニングと同様である。

(11) チューニング11

4.3(1)と同様のチューニングを行った。

プログラムDの高速化に寄与したチューニングはチューニング8および11である。

4.5 プログラムEの高速化

(1) チューニング1

方程式解法サブルーチンにおいて回帰計算となるためベクトル化されないDOループを、計算格子と反復計算をミックスしたDOループに構造変更し、ベクトル処理部を拡大した。

本サブルーチンは $K \times J$ の二次元格子配列について繰り返し計算を行っている。計算の順序はJ方向について三項方程式を解き、値を入れかえて、次のKに進み、全格子が終了したら、収束判定を行い、収束しない場合は繰り返し計算をする。これを図4.5.1に示す。図より明らかであるが

$$P_{J,K}^T = f(P_{J\pm 1,K\pm 1}^{T-1})$$

となり、J、K、Tとも回帰演算となる。

Jについては戻り計算があるのでこの回帰計算は回避できないが、KおよびTについては一方向性のため、KとTに斜めの方向は同時計算可能である。図4.5.2はKとTの関係を示したものである。図中の数字は計算の順序と同時計算可能なKとTの位置を示したものである。たとえば最初の繰り返し計算($T=1$ のとき)の $K=1$ の時の1、 $K=2$ の時の2の順に計算され、 $K=3$ の時の3となり、この時 $T=2$ 、 $K=1$ の時のPの値が計算可能

となる。このように同番号のTとTの時のPは計算可能となるので、この番号をパラメータ(T')として繰り返し計算を行うことによりK方向をベクトル化可能とできる。Kと T' の関係を図4.5.3に示す。図中、 T_n はオリジナルプログラムにおける繰り返し回数を示す。

このように処理構造を変更することにより本サブルーチンの処理フローは図4.5.4(a)から図4.5.4(b)のように変更された。この変更により収束判定が多少複雑になるのでそれについて説明する。図4.5.5は収束判定説明図である。収束判定に用いている全格子の残差和および最大値は同じ T_n について求め、 K_{max} まで行った時点で T_n の収束判定が可能である。図において、ある T_n 時の値は線で囲われている部分のようになり、 T_9 で収束した場合、Kによっては T_{11} 、 T_{10} の時の値となっている。そこで線で囲われている値を保存しておき、 T_7 より T_9 まで再計算を行う。ただし、 T_{10} 、 T_{11} は計算しない。

(2) チューニング2

定数の不要計算を削除した。図4.5.6(a)の定数演算を図4.5.6(b)のように変更して演算回数の省略を図った。

(3) チューニング3

4.4(5)同様DOループの合併を行った。

プログラムEの高速化に寄与したチューニングはチューニング1である。

4.6 プログラムFの高速化

(1) チューニング1

DOループの一次元化を行った。 4×4 の二重ループをインデックスを用いてベクトル長16の一重ループにした。

多重DOループは自動ベクトル化の対象となる。すなわち、最内DOループだけでなく外側のDOループも条件さえ満たせばベクトル化される。しかし、外側のDOループまでベクトル化されるためには、配列が密であることなど、きつい条件があるため外側のDOループのベクトル化は難しい。したがって、通常は最内DOループのみベクトル化される例が多い。

本オリジナルプログラムにおいて、最内 DO ループのみベクトル化されている多重 DO ループがある。最内 DO ループはベクトル化されているが、ベクトル長が 4 である。DO ループの立ち上がり時間はそのベクトル長に関係なく、ほぼ一定であるのでベクトル長は長い方が効率がよい。そのため、ベクトル長の拡大を図るため、インデックスを用いる事により二次元配列の実行上の一次元化を行った。

図 4.6.1(a)に変更前のプログラムを、図 4.6.1(b)に変更後のプログラムを示す。

(2) チューニング 2

ベクトル長が短いためベクトル化のオーバーヘッドがかかるという理由により、ベクトル化されない DO ループをベクトル化されるようにした。これによりベクトル化の促進が行えた。

図 4.6.2(a)に変更前のプログラムを、図 4.6.2(b)に変更後のプログラムを示す。

(3) チューニング 3

DO ループ内で配列の位置を求める計算が毎回行われているので、これを予め一度計算しておき、その後は参照する形式にして、不要演算を削減した。

図 4.6.3(a)に変更前のプログラムを、図 4.6.3(b)に変更後のプログラムを示す。

(4) チューニング 4

同一 DO ループ内で同配列に計算結果をストアしているためベクトル化されない DO ループを二つに分割してベクトル化されるようにした。これによりベクトル化の促進が行えた。

図 4.6.4(a)に変更前のプログラムを、図 4.6.4(b)に変更後のプログラムを示す。

(5) チューニング 5

多重 DO ループにおいて外側の DO ループ内の処理もベクトル化されるように DO ループを分割して、それぞれの DO ループがベクトル化されるようにした。ベクトル化の促進を目的とする。

図 4.6.5(a)に変更前のプログラムを、図 4.6.5(b)に変更後のプログラムを示す。

(6) チューニング 6

DO ループ内の変数の定義・参照が回帰的な

っているためベクトル化されない DO ループをアルゴリズムを変更することによりベクトル化した。ベクトル化の促進を目的とする。

図 4.6.6(a)に変更前のプログラムを、図 4.6.6(b)に変更後のプログラムを示す。

(7) チューニング 7

EQUIVALENCE 文により宣言された変数が不正境界となり、その変数を用いる DO ループが完全ベクトル化されないで、EQUIVALENCE 文を用いないで同じ値の格納された他の変数名を代入して DO ループをベクトル化し、ベクトル化の促進を図った。同時に、構造を変更して DO ループの合併を行った。

ベクトル化対象のデータの型は、

(i) 4 バイトの整数型

(ii) 実数型

(iii) 倍精度実数型

(iv) 複素数型

(v) 倍精度複素数型

(vi) 4 バイトの論理型

であり、ベクトル化されるデータは正しい境界に割り当てられていなければならない。これらのうち、(i), (ii), (vi)の正しい境界は 4 の倍数の番地であり、(iii), (v)のそれは 8 の倍数の番地である。また、(iv)の場合は 4 または 8 の倍数の番地が正しい境界である。各種の配列はその配列の割り付け時に 4 バイト整数型や倍精度実数型の混在により正しい境界に割り付けられない場合がある。これを不正境界に割り付けられるという。これは、本チューニングのようにプログラム上で修正を行っても、ジョブ制御文において境界合わせのパラメータ指定を行う方法によっても回避できる。

図 4.6.7(a)に変更前のプログラムを、図 4.6.7(b)に変更後のプログラムを示す。

(8) チューニング 8

多重 DO ループにおいて DO ループを入れ換えてベクトル長の最も長いループでベクトル化されるようにして、ベクトル長の拡大を図った。

図 4.6.8(a)に変更前のプログラムを、図 4.6.8(b)に変更後のプログラムを示す。

(9) チューニング9

二重 DO ループにおいて、ベクトル長の短い内部 DO ループを展開して、外側の大きい DO ループでベクトル化した。ベクトル長の拡大、パイプラインの同時走行種類の拡大、ベクトル化の促進が図れる。

図 4.6.9(a)に変更前のプログラムを、図 4.6.9(b)に変更後のプログラムを示す。

(10) チューニング10

回帰計算を回避してベクトル処理部の拡大を図った。

図 4.6.10に示すサブルーチン FILTERX は方程式解法処理を行うサブルーチンであり、三項方程式解法ルーチン BTRI を CALL して処理を行っている。BTRI ルーチンは一方向一列を解くサブルーチンであり、その構造は回帰的でベクトル化できない。BTRI ルーチンを二次元化し、もう一方向の DO ループを設定してベクトル化した。

図 4.6.11(a)にオリジナルの処理フローを、図 4.6.11(b)にチューニング後の処理フローを示す。

(11) チューニング11

最適化制御行のそう入によりベクトル化されない DO ループをベクトル化した。ベクトル化の促進を目的とする。

図 4.6.12(a)は $JJ=J$ の場合には配列 Q および S の値が保証されない。 $JJ \neq J$ が予め判っている場合には保証できるのでこの DO ループを強制的にベクトル化することが可能である。

(12) チューニング12

演算結果は同じであるがプログラム構造上、毎回計算が行われる場合がある。

計算格子について

```
RR  =1.0/Q ( )
U ( )=Q ( )*RR
V ( )=Q ( )*RR
  ⋮
  ⋮
```

のパターンの計算が数ヶ所で繰り返されているので新サブルーチンの作成により予め計算しておいて参照する形式に変更した。目的は不要計算削除である。図 4.6.13に新しく作成されたサブルーチ

ンを示す。図 4.6.14(a)はオリジナル、図 4.6.14(b)はチューニング後の新サブルーチンで求められた配列を用いたプログラムである。

(13) チューニング13

4.4(5)同様、DO ループの合併を行った。

プログラム F の高速化に寄与したチューニングはチューニング10である。

4.7 プログラム G の高速化

(1) チューニング 1

方程式解法サブルーチンにおいて反復の度に計算される係数部の計算結果は反復が異なっても変わらない。そのため、解法サブルーチンの実行前に予め一度計算し、反復時にはそれを参照する形式にして不要演算の削除、計算の省略化を図った。拡大利用可能となったメモリの活用でもある。本チューニングは 4.6(12)と同様にメモリを活用して計算の省力化を行うものであるが、後者は反復毎に再計算されるのに対し、前者は一度の計算で求められる値を反復毎に使用する。

(2) チューニング 2

最適化制御行を用いてベクトル化されない DO ループをベクトル化して、ベクトル化の促進を図った。

図 4.7.1(a)の DO ループの直前に

```
* VOCL LOOP, KTE2. LT. N, KTE2.
  LT. K2
```

をそう入することにより DO 32 ループをベクトル化した。この DO ループは N と KTE2, KTE2 と K2 の大小関係がわからないためベクトル化されなかったもので、コンパイラにそれを知らせることによりベクトル化された。

(3) チューニング 3

ベクトル化されない DO ループをベクトル化される部分とされない部分に分割した。

図 4.7.2(a)の DO 22 ループは配列 LINE が CHARACTER 宣言されていること、K が回帰的参照関係をとることおよびベクトル化非対象文 WRITE 文を含むことによりベクトル化されない。そのため、ベクトル化非対象文を含むステートメントとそうでないステートメントを分けて DO ルー

ブを分割した。後者の DO ループはさらに、変数 K を配列にすることにより DO ループをベクトル化した。

DO ループ中にベクトル化非対象の FORTRAN 文を含んでいるため DO ループがベクトル化されない場合、ベクトル化非対象文を含むステートメントと他のステートメントを分けて DO ループを作成することにより、一方はベクトル化される DO ループとすることができる。このように、ベクトル化されない DO ループ、または、部分ベクトル化される DO ループは、スカラ部分とベクトル部分を分けて別々の DO ループにする方が、DO ループ内のスカラ処理部分とベクトル処理部分の混在によるオーバーヘッドをなくすることができる。

プログラム G の高速化に最も寄与したチューニングはチューニング 1 である。

4.8 プログラム H の高速化

(1) チューニング 1

無条件に呼び出されるサブルーチンを条件付きで呼び出すようにして呼び出し回数を減少させた。

図 4.8.1(a) にオリジナル、図 4.8.1(b) にチューニング後のコーディングを示す。

(2) チューニング 2

定数除算を予めデータ文で与えて乗算に変更した。

除算に要する時間は乗算に要する時間より多く、命令実行時間で約 8 倍遅い。従って、除算はできるだけ削減する方がよい。予め、除算を行い、後からは乗算で実行可能にできるものはその形式にした方がよい。

(3) チューニング 3

頻繁にあらわれる定数計算を予め計算しておき、以後、参照の形式にした。

図 4.8.2(a) において、 $-PTA-PTB$ および $PTB-PTA$ は頻繁にあらわれる計算式であるので、それらの値を予め計算しておいて、 $-PTAPT B$ および $PTBPTA$ をつくり置き換えた。目的は計算の省力化である。

(4) チューニング 4

4.4(4)と同様に、IF-GO TO 文を含む DO ルー

プをベクトル化した。

(5) チューニング 5

4.7(1)と同様のチューニングを行い、不要演算を削除した。

(6) チューニング 6

4.7(3)と同様に、DO ループの分割による高速化を行った。

4.9 プログラム I の高速化

(1) チューニング 1

ベクトル化非対象文 (WRITE 文, CALL 文) があること、WRITE 文中に配列があることおよび回帰的参照形式をとる変数があることによりベクトル化されない DO ループを、ベクトル化非対象文を除き、DO ループを分割することにより IF 文を削除した。

図 4.9.1(a) はオリジナルのループ構造図である。CALL 文はチェック用のものなので点線で囲われた部分を取り除いた。また、J は

$$J=1, 2, 3, \dots, J1, J1+1, J1+2, \dots, N$$

について処理し、 $J \leq J1$, $J = J1+1$, $J \geq J1+2$ の三つの場合に分けられる。DO ループ内で行われているこの判定を DO ループ外に排出して、DO ループ処理を行うようにした。その結果、DO ループ構造図は図 4.9.1(b) のようになる。

これにより二つの DO ループができるがそれぞれの DO ループがベクトル化されるのでベクトル化の促進が行える。また、DO ループ内の IF 判定を削除することにより効率的に処理されるようになった。

(2) チューニング 2

配列の位置を DO ループ中で求めるためベクトル化されないステートメントのある DO ループを二つに分割して原因を取り除き、二つの DO ループをベクトル化した。その結果、ベクトル化の促進および IF 文の一部削除がなされた。

図 4.9.2 にオリジナルの DO ループ構造を示す。L が DO ループ内で求められているので、L を含むステートメントはベクトル化されない。これを、チューニング 1 と同様に、J は

$$J=1, 2, 3, \dots, J1-1, J1, J1+1, \dots, N$$

となっているので、 $J \leq J1-1$, $J=J1$, $J \geq J1+1$ の三つの場合に分けて DO ループを作成し、ベクトル化した。

(3) チューニング 3

配列の位置の決定が DO ループ内で行われていることおよび変数の参照がその定義に先行しているとの理由でベクトル化されない DO ループを分割してベクトル化されるようにした。目的は DO ループのベクトル化の促進および IF 文の削減である。

図 4.9.3(a)にオリジナルプログラムを、図 4.9.3(b)にチューニングプログラムを示す。

(4) チューニング 4

部分ベクトル化されている DO ループの構造変更を行ってベクトル化されるようにし、DO ループ中で判定する必要のない IF 文を削除した。

図 4.9.4(a)はオリジナルの DO ループである。KY が配列 PD の回帰性の原因となっていることおよび JTPP と KY の関係が不明のため DO ループは部分ベクトル化となっている。また、IF 文は DO ループ外で判定可能であるので DO ループ外に排出して、その条件により配列 PD の計算を行う DO ループを選択する構造に変更した。これによりベクトル化の促進、IF 文の削除によるオーバーヘッドの削減が行われた。

(5) チューニング 5

4.7(2)と同様に、最適化制御行 (NOVREC) の利用により DO ループのベクトル化を図った。

(6) チューニング 6

4.8(2)と同様に、定数除算を乗算に変更した。

4.10 プログラム J の高速化

(1) チューニング 1

ベクトル化非対象文 (DO WHILE 文) があるためにベクトル化されない DO ループを、DO WHILE 文を含む DO ループと含まない DO ループに分割して後者をベクトル化した。これにともない、両方の DO ループで使われる変数を配列にした。目的はベクトル化の促進である。

図 4.10.1(a)にオリジナルプログラムを、図 4.10.1(b)にチューニングプログラムを示す。

(2) チューニング 2

配列の添字の順序を変えてメモリアクセスの高速化を図った。

図 4.10.1(a)の下線のある配列 RE は $RE(1, 1, N)$ となっているが、DO ループは N でベクトル化されているので飛びのある配列参照となっている。それを $RE(N, 1, 1)$ と順序を入れ換えることにより、 N で連続的に参照されるので最も効率の良いデータアクセスが行える。

(3) チューニング 3

サブルーチンのプログラム構造を組みかえてモジュール化をはかり、ベクトル化の促進と冗長計算部分の除去を行った。

図 4.10.2(a)はオリジナルのプログラム構造である。サブルーチン SBCH はほとんどスカラ計算でありベクトル化率が低く、DO ループでベクトル化されている場合もベクトル長 3～4 と大変短いものである。また、サブルーチン SBCH は二種類のデータを計算する機能をもつために、データ A を求めるために必要な計算、データ B を求めるために必要な計算およびデータ A と B を求めるために必要な共通計算の三つに分けられる。したがって、A だけを求めたい場合にも A の計算には不要な B の計算を行うプログラム構造となっている。ベクトル長 NCS2 の二重ループになっているため、不要計算部が多い場合には余分な計算に時間を費すことになる (本ルーチンでは共通計算がステートメントにして半分以上占めている)。これらの処理を目的毎に一括配列処理を行う構造にかえて、ベクトル化の促進および不要計算のできるだけ除去を図った。

(4) チューニング 4

配列データの定義・引用が回帰的参照であるためベクトル化されない DO ループを原因となる配列を単変数におきかえることによりベクトル化した。

図 4.10.3(a)のオリジナルプログラムの下線部の配列を図 4.10.3(b)の下線部の単変数にした。目的はベクトル化の促進である。

(5) チューニング 5

配列の大きさを設定しなおしてメモリアクセス

を効率の良いものにした。目的はバンク競合の回避である。

たとえば配列 A (600, 30) であれば

A : 単精度 \Rightarrow A (602, 30)

A : 倍精度 \Rightarrow A (601, 30)

とすればよい。

(6) チューニング 6

4.1 (2)と同様に DO ループ内で呼び出されるサブルーチンを呼び出し元に展開した。

(7) チューニング 7

4.6 (9)と同様にベクトル長の短い DO ループを展開して外側の長いベクトル長でベクトル化されるようにした (この場合は配列の次元の拡大をともなった)。

(8) チューニング 8

4.8 (2)と同様に、定数除算を DO ループ外で予め計算して DO ループ内では乗算に変更した。

プログラム J の高速化に最も寄与したチューニングはチューニング 3 である。

4.11 プログラム K の高速化

(1) チューニング 1

DO ループ内のサブルーチンから SSL II ライブラリ DLAX (連立一次方程式解法ルーチン) が呼ばれていたが、これを用いずにその場で解法計算を行うことにより DO ループをベクトル化した。

DO ループ内にベクトル非対象文 (CALL 文) が存在すると DO ループはベクトル化されない。通常は SSL II ライブラリを展開することは考えられない。ここでは三元三列の連立一次方程式の解法であるのでクラメル公式を用いて DO ループ内で直接求めるようにした。オリジナルの場合には約 7 万 5 千回、DLAX が呼ばれていたが、本修正により CALL 文の削除、DO ループのベクトル化の促進が行えた。

図 4.11.1 にオリジナルおよびチューニング後の簡単な DO ループ構成を示す。

(2) チューニング 2

4.1 (2)と同様に、DO ループ内で呼び出されるサブルーチンを呼び出し元に展開した。

(3) チューニング 3

4.6 (9)と同様に、ベクトル長の短い DO ループを展開して外側の長いベクトル長でベクトル化されるようにした。

(4) チューニング 4

4.10 (2)と同様に、配列の添字の順序を変えてメモリアクセスの高速化を図った。

(5) チューニング 5

4.10 (3)と同様に、サブルーチンの構造を変更してベクトル化を図った。

(6) チューニング 6

4.10 (8)と同様に、定数除算を DO ループ外で予め計算して DO ループ内では乗算に変更した。

プログラム K の高速化に最も寄与したチューニングはチューニング 5 である。

4.12 プログラム L の高速化

(1) チューニング 1

変数の参照がその定義に先行しているためベクトル化されない DO ループをコーディング変更してベクトル化した。

図 4.12.1 (a) の RESMX の引用が定義に先行しているためベクトル化されないのを図 4.12.1 (b) のコーディングにして DO ループをベクトル化した。この結果 4 つの IF 判定を 1 つに減らすことができより効率的に処理されることになった。

(2) チューニング 2

二重 DO ループの内側の DO ループのベクトル長がその DO ループ内で使用される配列の大きさと等しいことをコンパイラに認識させて、外側の DO ループのベクトル化まで図った。これはコーディング変更による DO ループの一重化と同じであるが、変更が簡単なわりに効果の大きい高速化方法である。三重 DO ループの場合も同様の方法で一重化可能である。これによりベクトル長が拡大できる。図 4.12.2 にオリジナルプログラムおよびチューニングプログラムを示す。

(3) チューニング 3

二重 DO ループの内側のベクトル長の短い DO ループでベクトル化されている DO ループを、外側のベクトル長の長い DO ループでベクトル化さ

れるようにした。

これにより、ベクトル長の拡大が行われ、さらにJ方向でベクトル化されるために配列の参照が連続的に行われるようになりメモリアクセスも効率的に行われるようになる。図4.12.3にオリジナルおよびチューニング後のプログラムを示す。

(4) チューニング 4

最外 DO ループのベクトル長は短くベクトル化されていないので、それを内側の DO ループに展開してベクトル化される演算を増やし、スカラ処理部分を減少させた。

これにより、DO ループ内のベクトル演算の増大およびオーバーヘッドの削減が行われた。図4.12.4にオリジナルおよびチューニング後のコーディングを示す。

(5) チューニング 5

4.4 (5)と同様に、DO ループの統合を行った。

(6) チューニング 6

4.7 (2)と同様に、最適化制御行 (NOVREC) を用いてベクトル化の促進を行った。

(7) チューニング 7

4.10(4)と同様に、DO ループ内で回帰的定義参照関係となっている配列を単変数にして DO ループをベクトル化した。

プログラム L の高速化に寄与したチューニングはチューニング 2 である。

4.13 ま と め

4.1～4.12節のチューニングの種類が多いので、5倍以上の高速化倍率を得たプログラムを対象に最も寄与したチューニング内容および効果を表4.1にまとめる。

表 4.1 各プログラムの高速化に寄与したチューニング内容

| コード | チューニング内容 | 目的、効果 | 高速化倍率* |
|-----|--|----------------------|--------|
| E | 方程式解法サブルーチンにおいて回帰計算となるためベクトル化されない DO ループを、計算格子と反復計算をミックスした DO ループに構造変更した。(チューニング 1) | 回帰計算の回避によるベクトル処理部の拡大 | 6.51 |
| F | 方程式解法処理を行うサブルーチンにおいて呼び出される三項方程式解法ルーチンは一方向一列を解くサブルーチンであり、その構造は回帰的でベクトル化できないので、三項方程式解法ルーチンを二次元化し、もう一方の DO ループを設定した。(チューニング 10) | 回帰計算の回避によるベクトル処理部の拡大 | 7.32 |
| J | オリジナルのプログラム構造を組み換えてモジュール化を図った。(チューニング 3) | ベクトル化の促進および冗長計算部の除去 | 14.11 |
| K | オリジナルのプログラム構造を組み換えてモジュール化を図った。(チューニング 5) | ベクトル化の促進および冗長計算部の除去 | 19.32 |
| L | 多重 DO ループの内側の DO ループのベクトル長がその DO ループ内で使用される配列の大きさと等しいことをコンパイラに認識させ、外側の DO ループのベクトル化まで図り、DO ループの一直化を行った。(チューニング 2) | ベクトル長の拡大 | 12.58 |

*高速化倍率=M780システム実行時間/V P400システム実行時間

5. む す び

航技研の汎用プログラム12本の高速化を図り、その結果、以下のことが得られた。

オリジナルコードの VP 200/400 での実行において M780 での実行時間より短縮されたプログラムは 3 本であり、他の 9 本のプログラムでは逆に M780 での実行時間より長くなっている。実行時間の短縮された 3 本のプログラムについてもその処理速度向上は、大きい場合でも 4 倍弱であり、ピーク演算処理速度の倍率、65 倍に比較して、ベクトル計算機を活用している倍率とは言い難く、プログラムチューニングを行い実行時間の短縮を図ることの必要性が確認された。

プログラムチューニングを行った結果、9 本のプログラムについては M780 での実行時間の 1.35 ～ 19.32 倍の高速化を得た。チューニングを行っても M780 での実行時間より短くならなかったプログラムは 3 本であった。

高速化倍率の特に高い 3 本のプログラムのベクトル化率は 99% 以上であり、ベクトル / スカラ速度比は 31 ～ 48 倍である。これらの倍率はピーク演算処理速度の倍率、148 倍に対応するものであり、最高でも 48 倍にしか達していないという感を持たれるかもしれない。しかし、148 倍というのはメモリ競合などが起こらない理想の環境でのピーク演算処理速度であり、実際の環境ではメモリバンク競合やパイプラインの並列走行がのぞめない場合が多々あり、このような環境で 48 倍の高速化倍率を得ることはかなりの高速化が達成されたと考えてよい。

一方、プログラムチューニングを行っても高速化倍率が 1 以下の改善されなかったプログラムのうち、2 本は V も α も共に小さく、もう 1 本は V が小さいが α は大きい。

これらのプログラムの有効利用のためには、多方面からの検討が必要であると考えられる。しかし、実行時間が数秒で終了する小規模ジョブであったり、数分程度の実行時間を必要とするが、実行される度合いが少ない場合など、他に影響を及ぼす可能性の少ない場合には、それほど高速化の必

要性はないと考えられる。しかし、長時間ジョブの場合には高速化倍率が低くても絶対時間の短縮は大きい。このように、高速化の必要性は個々のプログラムの使われ方にも依存するが、大規模ジョブを何ケースも実行する場合が多い現在、プログラムの高速化は避けては通れない。

一般的に、プログラムの処理内容は前処理、数値計算および後処理の 3 つの部分に分けられるが、数値計算の処理に最も実行時間が費やされる。前後処理は入出力は多いが、実行時間は数値計算の処理に比べれば小さい。これらはプログラム構造を単純にする上からも分けた方がよい。本報告において高速化倍率の高いプログラムは前後処理は切り離されており、これからも、この分離傾向は強くなり、前後処理は TSS 処理となってゆくであろう。

また、汎用プログラムはどのような条件の場合でも計算可能にするため入力パラメータが多く、プログラム上ではそのための IF 判定が多くなり、プログラム構造が複雑になる。複雑なプログラムは高速化のためのチューニングにおいても手を入れにくいので、プログラム構造はできるだけ単純である方がよい。また、IF 文は他の演算命令に比較して実行時間がかかるので、DO ループ中での使用はできるだけ避けるべきである。このように、汎用プログラムであるからといって一つのプログラムで IF 文を多用した過度の汎用化は避けて数種のプログラムを作成して、IF 文を減らすことを考えることも必要である。

高速化のためのチューニング内容は回帰演算や配列の定義・参照の矛盾によりベクトル化されない部分をベクトル化されるようにチューニングしたものが多く、高速化の基本であるベクトル化の促進が行われている。そして、ベクトル化の促進のチューニング過程で効率的なベクトル化や省力化も合わせて配慮されている。また、汎用計算機にも有効な内容のチューニングもある。

VP 200/400 の実行時間の差が大きいベクトル化率の高い 3 次元のプログラム 2 本が、VP 400 の特徴を活かしたチューニングとなっている。

以上に述べたように、プログラムの高速実行は

既成の自動ベクトル化機能等を用いるだけでは得られない。ベクトル計算機を活かすようにプログラム設計やプログラム作成に工夫を必要とする。

既に作成されている汎用プログラムは手を入れて高速化するより方法がなく、その高速化倍率には本報告によって得られた率のように限度がある。したがって、これから作成されるプログラムは開発の段階でスーパーコンピュータを活用する設計および作成法を行うようにすべきである。本報告がそれに対して参考になれば幸いである。

最後に、汎用プログラムのチューニング作業を担当されたシステムズ・デザイン株式会社の山本秀秋氏、大興電子通信株式会社の出口良二氏、株式会社システム総合研究所の関光彦氏に感謝の意を表する。

参 考 文 献

- 1) 渡辺 貢；スーパーコンピュータの高速化技術，第3回航空機計算空気力学シンポジウム論文集，1985年11月
- 2) 平栗俊男，田畑 晃，槌本隆光，田口尚三；マシンサイクル7.5nsを達成した並列処理パイプライン処理方式のスーパーコンピュータFACOM VP，日経エレクトロニクス，1983. 4. 11
- 3) The CRAY X-MP Series of Computer Systems, 日本クレイ株式会社パンフレット，1987年10月
- 4) 8CPU 構成で10GFLOPSの最大性能をもつスーパーコンピュータETA 10, 日経データプロ・EDP速報版，1987年7月
- 5) 中村絹代，吉田正廣，峯尾真一；プログラム高速化技術とスーパーコンピュータSXシステムによる検証，航空宇宙技術研究所報告TR-909，1986年7月
- 6) 吉田正廣，中村絹代，内田啓一郎，棚倉由行；スーパーコンピュータVPシステムによるプログラムの高速化技術，航空宇宙技術研究所報告TR-915，1986年10月

```

      :
      X1=B (I+1, 1) -B (I-1, 1)
      X2=B (I+1, 2) -B (I-1, 2)
      DO 100 J=2, N
      A1=X1
      X1=B (I+1, J+1) -B (I-1, J+1)
      C (J) =X1-X2+D (J+1) -D (J-1)
      X2=A1
100  CONTINUE
      :

```

図 4.1.1(a) A-1 変更前のプログラム

```

      DIMENSION X1 (N), X2 (N)
      :
      X1 (1) =B (I+1, 1) -B (I-1, 1)
      X2 (1) =B (I+1, 2) -B (I-1, 2)
      DO 100 J=2, N
      A1 (J) =X1 (J-1)
      X1 (J) =B (I+1, J+1) -B (I-1, J+1)
      C (J) =X1 (J) -X2 (J-1) +D (J+1) -D (J-1)
      X2 (J) =A1 (J)
100  CONTINUE
      :

```

図 4.1.1(b) A-1 変更後のプログラム

```

      DO 100 I=1, N
      :
      CALL SUB (A, B)
      :
100  CONTINUE
      :

      SUBROUTINE SUB (A, B)
      A=B**2
      RETURN
      END

      ↓

      DO 100 I=1, N
      :
      A=B**2
      :
100  COUTINUE

```

図 4.1.2 A-2 変更例


```

21 ND=NQ/K
   NS=NS*K
   NR=K
   IQ=0
   ID=0
   DO 22 I=1, NS
   DO 24 J=1, ND
   L=IQ+J
   LP=L+ND
   M=ID
   W=F(L)+F(LP)*CMPLX(CN(M+1), SN(M+1))
   IF(NR.EQ.2) GO TO 24
   L=LP
   DO 26 K=3, NR
   L=L+ND
   M=M+ID
   IF(M.GE.N) M=M-N
26 W=W+F(L)*CMPLX(CN(M+1), SN(M+1))
24 X(ID+J)=W
   ID=ID+ND
   IQ=IQ+NQ
   IF(IQ.GE.N) IQ=IQ-N
22 CONTINUE

```

図 4.3.1(a) C-1 変更前のプログラム

| | | |
|---|---|--------------|
| <pre> DO 22 I = 1, NS MMS = ND * (I - 1) LLS = MOD(NQ * (I - 1) , N) II = ND * (I - 1) DO 22 J = 1, ND MM(II + J) = MMS LL(II + J) = LLS + J 22 CONTINUE </pre> | } | インデックスの計算 |
| <pre> C DO 23 I = 1, NS * ND X(I) = F(LL(I)) 23 CONTINUE </pre> | } | インデックス化したループ |
| <pre> C DO 24 K = 2, NR KND = ND * (K - 1) DO 24 I = 1, NS*ND MMS = MOD(MM(I)*(K-1), N) + 1 X(I) = X(I) + F(LL(I)+KND) * CMPLX(CN(MMS), SN(MMS)) 24 CONTINUE </pre> | } | インデックス化したループ |

図 4.3.1(b) C-1 変更後のプログラム

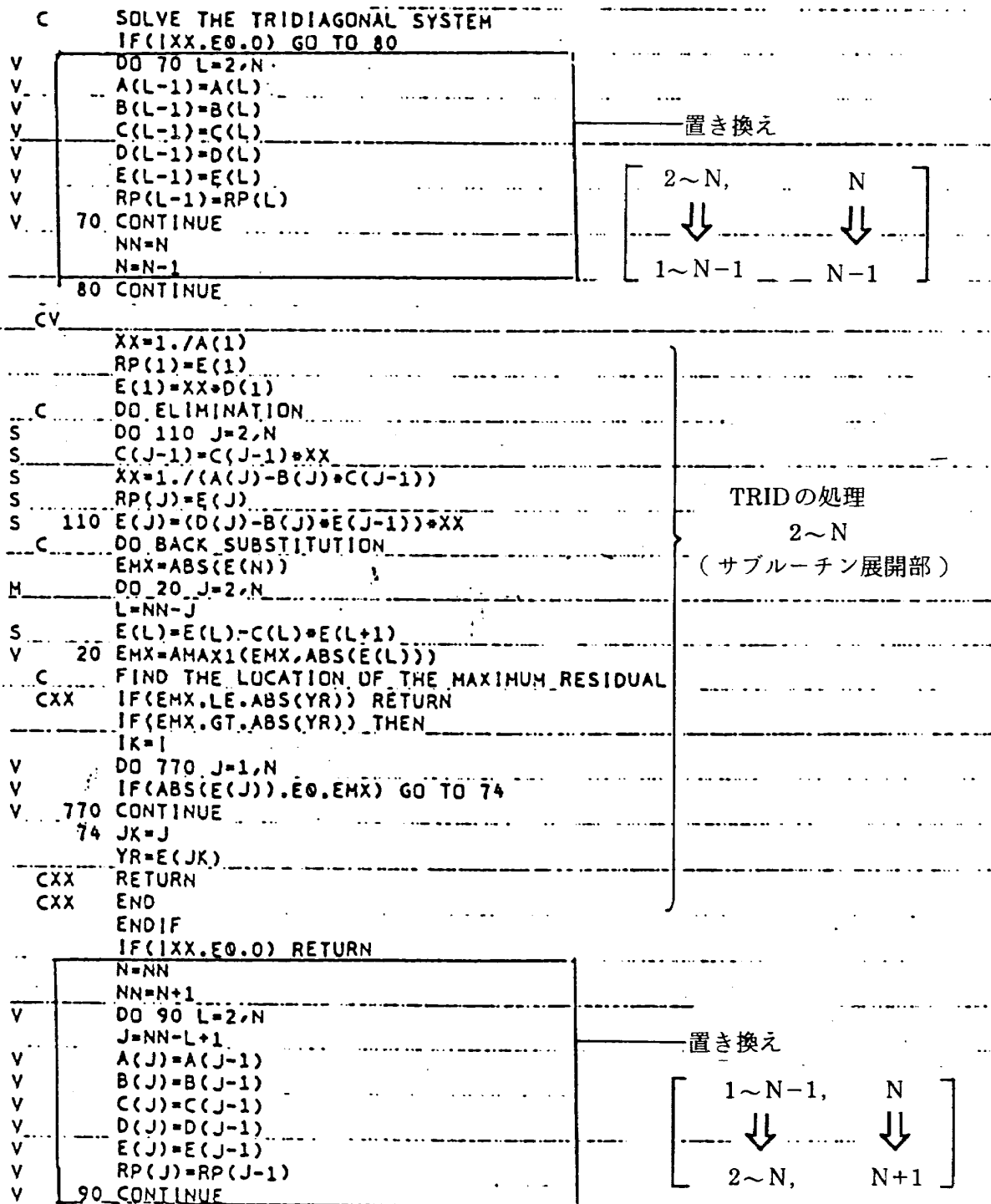


図 4.3.2(a) C-2 変更前のプログラム

```

CXX  SOLVE N DIMENSIONAL TRIDIAGONAL SYSTEM OF EQUATIONS.
CV
    IF( IXX .EQ. 0 ) THEN
        CV
            XX=1./A(1)
            RP(1)=E(1)
            E(1)=XX*D(1)
        C
            DO ELIMINATION
        S
            DO 110 J=2,N
        S
            C(J-1)=C(J-1)*XX
        S
            XX=1./(A(J)-B(J)*C(J-1))
        S
            RP(J)=E(J)
        S
            110 E(J)=(D(J)-B(J)*E(J-1))*XX
        C
            DO BACK SUBSTITUTION
            EMX=ABS(E(N))
        M
            DO 20 J=2,N
            L=NN-J
        S
            E(L)=E(L)-C(L)*E(L+1)
        V
            20 EMX=AMAX1(EMX,ABS(E(L)))
        C
            FIND THE LOCATION OF THE MAXIMUM RESIDUAL
        CXX
            IF(EMX.LE.ABS(YR)) RETURN
            IF(EMX.GT.ABS(YR)) THEN
                IK=I
        V
            DO 770 J=1,N
        V
            IF(ABS(E(J)).EQ.EMX) GO TO 74
        V
            770 CONTINUE
            74 JK=J
            YR=E(JK)
        CXX
            RETURN
        CXX
            END
        ENDIF
    CV
    ELSE
        CV
            XX=1./A(2)
            RP(2)=E(2)
            E(2)=XX*D(2)
        C
            DO ELIMINATION
        S
            DO 111 J=3,N
        S
            C(J-1)=C(J-1)*XX
        S
            XX=1./(A(J)-B(J)*C(J-1))
        S
            RP(J)=E(J)
        S
            111 E(J)=(D(J)-B(J)*E(J-1))*XX
        C
            DO BACK SUBSTITUTION
            EMX=ABS(E(N-1))
        M
            DO 21 J=2,N-1
            L=NN-J
        S
            E(L)=E(L)-C(L)*E(L+1)
        V
            21 EMX=AMAX1(EMX,ABS(E(L)))
        C
            FIND THE LOCATION OF THE MAXIMUM RESIDUAL
        CXX
            IF(EMX.LE.ABS(YR)) RETURN
            IF(EMX.GT.ABS(YR)) THEN
                IK=I
        V
            DO 771 J=2,N
        V
            IF(ABS(E(J)).EQ.EMX) GO TO 75
        V
            771 CONTINUE
            75 JK=J-1
            YR=E(J)
        CXX
            RETURN
        CXX
            END
        ENDIF
    CX
    ENDIF
    CX

```

} 'TRID' の処理
2~N

} 'TRID' の処理
3~N

図 4. 3. 2(b) C-2 変更後のプログラム

```

CJ02 = Y(JUP) / (Y(JUP) - Y(JLOW))
DO 10 I = IMIN , IMAX
  WPX1 = PX(I , JLOW)
  WPX2 = PX(I , JLOW-1)
  WPX3 = PX(I , JUP)
  WPX4 = PX(I , JUP+1)
  UL   = CJLOW * WPX1 - CJLOW1 * WPX2

```

図 4.4.1(a) D-1 変更前のプログラム

```

      CJ02 = Y(JUP) / (Y(JUP) - Y(JLOW))
*VNCL LOOP,NOVREC
      DO 10 I = IMIN , IMAX
V      WPX1 = PX(JLOW , I)
V      WPX2 = PX(JLOW-1 , I)
V      WPX3 = PX(JUP , I)
V      WPX4 = PX(JUP+1 , I)
V      UL   = CJLOW * WPX1 - CJLOW1 * WPX2

```

図 4.4.1(b) D-1 変更後のプログラム

```

      REAL FUNCTION PX * 4 (I,J)
C
C *** REAL FUNCTION PX * 8 (I,J)
C
C      FUNCTION PX COMPUTES U = DP/DX AT POINT I,J
C      CALLED BY - CDCOLE, DRAG, FINDSK, MACHMP, M1LINE,
C      NEWISK, PRINT1, PRTFLD, PRTWAL , PLWALL
C
CV      GENERIC
COMMON      P(102,101) , X(100) , Y(100)
COMMON / COM1/ IMIN , IMAX , IUP , IDOWN , ILE ,
1      ITE , JMIN , JMAX , JUP , JLOW ,
2      JTOP , JBOT
COMMON / COM5/ XDIFF(100) , YDIFF(100)
C
C      TEST TO LOCATE END POINTS
C      IF(I .EQ. IMIN) GO TO 10
C      IF(I .EQ. IMAX) GO TO 20
C
C      INTERIOR MESH POINT
C      PJI = P(J,I)
C      PX = .5 * (XDIFF(I+1) * (P(J , I+1) - PJI)
C      *      + XDIFF(I) * (PJI - P(J , I-1)))
C      RETURN
10  CONTINUE
C      UPSTREAM BOUNDARY
C      PX = 1.5 * XDIFF(I+1) * (P(J , I+1) - P(J , I)) -
1      0.5 * XDIFF(I+2) * (P(J , I+2) - P(J , I+1))
C      RETURN
20  CONTINUE
C      DOWNSTREAM BOUNDARY
C      PX = 1.5 * XDIFF(I) * (P(J , I) - P(J , I-1))
1      -0.5 * XDIFF(I-1) * (P(J , I-1) - P(J , I-2))
C      RETURN
END

```

図 4.4.1(c) D-1 変更前の関数 PX

```

SUBROUTINE    PXCOM
CV
CV
CV
COMMON        P(102,101) , X(100) , Y(100)
COMMON / COM1/ IMIN      , IMAX      , IUP      , IDOWN      , ILE      ,
1             ITE       , JMIN      , JMAX      , JUP       , JLOW      ,
2             JTOP      , JBOT
COMMON / COM5/ XDIFF(100) , YDIFF(100)
COMMON / VPPX/ PXC(102,101)

C
C
S          DO 10 I = IMIN+1 , IMAX-1          IMIN < I < IMAX    POINT
V          DO 20 J = 1 , 102
V          PXC(J,I) = .5 * ( XDIFF(I+1) * ( P(J,I+1) - P(J,I) )
*              + XDIFF(I) * ( P(J,I) - P(J,I-1) ) )
V 20      CONTINUE
S 10      CONTINUE
C
C          I = IMIN    POINT
V          DO 30 J = 1 , 102
V          PXC(J,I) = 1.5 * XDIFF(I+1) * ( P(J,I+1) - P(J,I) )
*              - 0.5 * XDIFF(I+2) * ( P(J,I+2) - P(J,I+1) )
V 30      CONTINUE
C
C          I = IMAX    POINT
V          DO 40 J = 1 , 102
V          PXC(J,I) = 1.5 * XDIFF(I) * ( P(J,I) - P(J,I-1) )
*              - 0.5 * XDIFF(I-1) * ( P(J,I-1) - P(J,I-2) )
V 40      CONTINUE
          RETURN
          END

```

図 4. 4. 1(d) D-1 変更後の新サブルーチン PXCOM

```

*VOCL LOOP,NOVREC
M DO 10 I = IMIN, IMAX
V WPX1 = PXC(JLOW, I)
V WPX2 = PXC(JLOW-1, I)
V WPX3 = PXC(JUP, I)
V WPX4 = PXC(JUP+1, I)
V UL = CJLOW * WPX1 - CJLOW1 * WPX2
V IF(I.GT.ITE) UL=CJ01 * WPX3 + CJ02 * WPX1
V IF(I.LT.ILE) UL=CJ01 * WPX3 + CJ02 * WPX1
V CPL(I) = -2.0 * UL * CPFACT
CV EM1L(I) = EMACH1(UL)
V AK1 = AK - GAM1 * UL
V IF(.NOT.PHYS) EM1L(I) = AK1
V IF(PHYS) THEN
V ARG = DELRT2 * AK1
V IF(SIMDEF.E0.2) ARG = ARG * EMROOT * EMROOT
V IF(SIMDEF.E0.3) ARG = ARG * EMACH
V ARG = 1. -ARG
V EM1L(I) = 0.
V IF(ARG.GT.0.) EM1L(I) = SQRT(ARG)
ENDIF
C
M IF (EM1L(I).GT.1.3) IEM = 1
V UU = CJUP*WPX3 - CJUP1*WPX4
V IF (I.GT.ITE) UU = UL
V IF (I.LT.ILE) UU = UL
V CPU(I) = -2.0 * UU * CPFACT
CV EM1U(I) = EMACH1(UU)
V AK1 = AK - GAM1 * UU
V IF(.NOT.PHYS) EM1U(I) = AK1
V IF(PHYS) THEN
V ARG = DELRT2 * AK1
V IF(SIMDEF.E0.2) ARG = ARG * EMROOT * EMROOT
V IF(SIMDEF.E0.3) ARG = ARG * EMACH
V ARG = 1. -ARG
V EM1U(I) = 0.
V IF(ARG.GT.0.) EM1U(I) = SQRT(ARG)
ENDIF
C
M IF (EM1U(I).GT.1.3) IEM = 1
V CPMAX = MAX(CPMAX, CPU(I), CPL(I))
V CPMIN = MIN(CPMIN, CPU(I), CPL(I))
V 10 CONTINUE
CPLARG = MAX(CPMAX, ABS(CPMIN))
UNPCOL = CPLARG / 29.

```

(a)

(b)

(c)

(d)

図 4.4.2(a) D-2 変更前のプログラム

```

*VDCI LOOP,NOVREC
V      DO 10 I = IMIN , IMAX
V      WPX1 = PXC(JLOW , I)
V      WPX2 = PXC(JLOW-1 , I)
V      WPX3 = PXC(JUP , I)
V      WPX4 = PXC(JUP+1 , I)
V      UL = CJLOW * WPX1 - CJLOW1 * WPX2
V      UU = CJUP * WPX3 - CJUP1 * WPX4
V      IF (I .GT. ITE .OR. I .LT. ILE) THEN
V          UL = CJ01 * WPX3 + CJ02 * WPX1
V          UU = UL
V      ENDIF
V      CPL(I) = -2.0 * UL * CPEACT
V      CPU(I) = -2.0 * UU * CPEACT
CV      EM1L(I) = EMACH1(UL)
CV      EM1U(I) = EMACH1(UU)
V      AK1UL = AK - GAM1 * UL
V      AK1UU = AK - GAM1 * UU
V      IF (.NOT. PHYS) THEN
V          EM1L(I) = AK1UL
V          EM1U(I) = AK1UU
V      ENDIF
V      IF (PHYS) THEN
V          ARGUL = DELRT2 * AK1UL
V          ARGUU = DELRT2 * AK1UU
V          IF (SIMDEF .EQ. 2) THEN
V              ARGUL = ARGUL * EMRUOT * EMRUOT
V              ARGUU = ARGUU * EMRUOT * EMRUOT
V          ENDIF
V          IF (SIMDEF .EQ. 3) THEN
V              ARGUL = ARGUL * EMACH
V              ARGUU = ARGUU * EMACH
V          ENDIF
V          ARGUL = 1. - ARGUL
V          ARGUU = 1. - ARGUU
V          EM1L(I) = 0.
V          EM1U(I) = 0.
V          IF (ARGUL .GT. 0.) EM1L(I) = SQRT(ARGUL)
V          IF (ARGUU .GT. 0.) EM1U(I) = SQRT(ARGUU)
V      ENDIF
C
V      CPMAX = MAX(CPMAX , CPU(I) , CPL(I))
V      CPMIN = MIN(CPMIN , CPU(I) , CPL(I))
V      10 CONTINUE
CV
V      IEMX = 0
V      DO 20 I = IMIN , IMAX
V          IF (EM1L(I) .GT. 1.3) IEMX = IEMX + 1
V      20 CONTINUE
V      DO 21 I = IMIN , IMAX
V          IF (EM1U(I) .GT. 1.3) IEMX = IEMX + 1
V      21 CONTINUE
V      IF (IEMX .GT. 0) IEM = 1
C
CPLARG = MAX(CPMAX , ABS(CPMIN))
UNPCOL = CPLARG / 29.

```

図4. 4. 2(b) D-2 変更後のプログラム

```

DO 14 I = ILE , ITE
  WPX1 = PX(I , JLOW)
  WPX2 = PX(I , JLOW-1)
  UL = CJLOW * WPX1 - CJLOW1 * WPX2
  CPOLD = CPL(I)
  CPL(I) = -2.0 * UL * CPFACT
  CPERR = ABS(CPL(I) - CPOLD)
  IF (CPERR .LE. CPMAXL) GO TO 12
  CPMAXL = CPERR
  IERRL = I
12 WPX3 = PX(I , JUP)
  WPX4 = PX(I , JUP+1)
  UU = CJUP * WPX3 - CJUP1 * WPX4
  CPOLD = CPU(I)
  CPU(I) = -2.0 * UU * CPFACT
  CPERR = ABS(CPU(I) - CPOLD)
  IF (CPERR .LE. CPMAXU) GO TO 14
  CPMAXU = CPERR
  IERRU = I
14 CONTINUE

```

(a)

(b)

図 4.4.3(a) D-3 変更前のプログラム

```

CV      IF (ILE.EQ.IMIN .OR. ITE.EQ.IMAX) GO TO 20
CV
V      DO 14 I = ILE , ITE
CV      WPX1 = PX(I , JLOW)
CV      WPX2 = PX(I , JLOW-1)
CV      WPX3 = PX(I , JUP)
CV      WPX4 = PX(I , JUP+1)
V      WPX1 = .5 * ( XDIFF(I+1) * (P(JLOW,I+1) - P(JLOW,I))
      * XDIFF(I) * (P(JLOW,I) - P(JLOW,I-1)) )
V      WPX2 = .5 * ( XDIFF(I+1) * (P(JLOW-1,I+1) - P(JLOW-1,I))
      * XDIFF(I) * (P(JLOW-1,I) - P(JLOW-1,I-1)) )
V      WPX3 = .5 * ( XDIFF(I+1) * (P(JUP,I+1) - P(JUP,I))
      * XDIFF(I) * (P(JUP,I) - P(JUP,I-1)) )
V      WPX4 = .5 * ( XDIFF(I+1) * (P(JUP+1,I+1) - P(JUP+1,I))
      * XDIFF(I) * (P(JUP+1,I) - P(JUP+1,I-1)) )
C
V      UL = CJLOW * WPX1 - CJLOW1 * WPX2
V      CPOLD = CPL(I)
V      CPL(I) = -2.0 * UL * CPFACT
V      CPERR = ABS(CPL(I) - CPOLD)
CV
V      IF (CPERR .GT. CPMAXL) THEN
V      CPMAXL = CPERR
V      IERRL = I
V      ENDIF
C
V      UU = CJUP * WPX3 - CJUP1 * WPX4
V      CPOLD = CPU(I)
V      CPU(I) = -2.0 * UU * CPFACT
V      CPERR = ABS(CPU(I) - CPOLD)
CV
V      IF (CPERR .GT. CPMAXU) THEN
V      CPMAXU = CPERR
V      IERRU = I
V      ENDIF
V      14 CONTINUE
      GO TO 21
CV
CV      20 CONTINUE

```

P Xを展開

(c)

(d)

図 4.4.3(b)-1 D-3 変更後のプログラム


```

M      DO 15 I = ILE , ITE
CV      WPX1 = PX(I , JLOW)
CV      WPX2 = PX(I , JLOW-1)
CV      WPX3 = PX(I , JUP)
CV      WPX4 = PX(I , JUP+1)
V      IF (I.NE.IMIN .AND. I.NE.IMAX) THEN
M          WPX1 = .5 * ( XDIFF(I+1) * (P(JLOW,I+1) - P(JLOW,I))
*              + XDIFF(I) * (P(JLOW,I) - P(JLOW,I-1)) )
M          WPX2 = .5 * ( XDIFF(I+1) * (P(JLOW-1,I+1) - P(JLOW-1,I))
*              + XDIFF(I) * (P(JLOW-1,I) - P(JLOW-1,I-1)) )
M          WPX3 = .5 * ( XDIFF(I+1) * (P(JUP,I+1) - P(JUP,I))
*              + XDIFF(I) * (P(JUP,I) - P(JUP,I-1)) )
M          WPX4 = .5 * ( XDIFF(I+1) * (P(JUP+1,I+1) - P(JUP+1,I))
*              + XDIFF(I) * (P(JUP+1,I) - P(JUP+1,I-1)) )
V          ENDIF
V          IF (I.EQ.IMIN) THEN
M              WPX1 = 1.5 * XDIFF(I+1) * (P(JLOW,I+1) - P(JLOW,I))
*              - 0.5 * XDIFF(I+2) * (P(JLOW,I+2) - P(JLOW,I+1))
M              WPX2 = 1.5 * XDIFF(I+1) * (P(JLOW-1,I+1) - P(JLOW-1,I))
*              - 0.5 * XDIFF(I+2) * (P(JLOW-1,I+2) - P(JLOW-1,I+1))
M              WPX3 = 1.5 * XDIFF(I+1) * (P(JUP,I+1) - P(JUP,I))
*              - 0.5 * XDIFF(I+2) * (P(JUP,I+2) - P(JUP,I+1))
M              WPX4 = 1.5 * XDIFF(I+1) * (P(JUP+1,I+1) - P(JUP+1,I))
*              - 0.5 * XDIFF(I+2) * (P(JUP+1,I+2) - P(JUP+1,I+1))
V              ENDIF
V              IF (I.EQ.IMAX) THEN
M                  WPX1 = 1.5 * XDIFF(I) * (P(JLOW,I) - P(JLOW,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JLOW,I-1) - P(JLOW,I-2))
M                  WPX2 = 1.5 * XDIFF(I) * (P(JLOW-1,I) - P(JLOW-1,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JLOW-1,I-1) - P(JLOW-1,I-2))
M                  WPX3 = 1.5 * XDIFF(I) * (P(JUP,I) - P(JUP,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JUP,I-1) - P(JUP,I-2))
M                  WPX4 = 1.5 * XDIFF(I) * (P(JUP+1,I) - P(JUP+1,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JUP+1,I-1) - P(JUP+1,I-2))
V                  ENDIF
C
M          UL = CJLOW * WPX1 - CJLOW1 * WPX2
V          CPOLD = CPL(I)
V          CPL(I) = -2.0 * UL * CPFACT
V          CPERR = ABS(CPL(I) - CPOLD)
CV
V          IF (CPERR .GT. CPMAXL) THEN
V              CPMAXL = CPERR
V              IERPL = I
V          ENDIF
C
M          UU = CJUP * WPX3 - CJUP1 * WPX4
V          CPULD = CPU(I)
V          CPU(I) = -2.0 * UU * CPFACT
V          CPERR = ABS(CPU(I) - CPULD)
CV
V          IF (CPERR .GT. CPMAXU) THEN
V              CPMAXU = CPERR
V              IERRU = I
V          ENDIF
V      15 CONTINUE
CV
V      21 CONTINUE
C

```

P X を展開

①

図 4. 4. 3(b)-2 D-3 変更後のプログラム

```

S      IF ( .NOT. OUTERR) GO TO 110
S      DO 100 J=JBOT,JTOP
S      ARHS = ABS(RHS(J))
S      IF (ARHS .GT. BIGRL) GO TO 90
S      GO TO 100
S      90 CONTINUE
S      BIGRL = ARHS
S      IRL = I
S      JRL = J
S      100 CONTINUE
S      110 CONTINUE

```

図 4.4.4(a) D-4 変更前のプログラム

```

S      IF ( .NOT. OUTERR) GO TO 110
S      CV
S      DO 100 J=JBOT,JTOP
S      ARHS = ABS(RHS(J))
S      IF (ARHS .GT. BIGRL) THEN
S      BIGRL = ARHS
S      IRL = I
S      JRL = J
S      ENDIF
S      100 CONTINUE
S      110 CONTINUE

```

図 4.4.4(b) D-4 変更後のプログラム

```

S      DO 10 J = JBOT , JTOP
S      VC(J) = C1(I) - (CXL(I) * POLD(J , I2) + CXC(I) * P(J , I)
S      1      + CXR(I) * P(J , I+1))
S      EMU(J , I1) = 0.0
S      POLD(J , I1) = P(J , I)
S      10 CONTINUE
S      DO 20 J = JBOT , JTOP
S      IF (VC(J) .LT. 0.0) EMU(J , I1) = VC(J)
S      20 CONTINUE
S      IF ( FCR ) GO TO 22
S      DO 21 J = JBOT , JTOP
S      EMU(J , I2) = EMU(J , I1)
S      21 CONTINUE
S      22 CONTINUE

C
C      COMPUTE ELEMENTS OF MATRIX
C
S      DO 30 J = JBOT , JTOP
S      DIAG(J) = (EMU(J , I1) - VC(J)) * CXXC(I) * WI
S      1      + EMU(J , I2) * CXXR(I-1) - CYYC(J)
S      SUP(J) = CYYD(J)
S      SUB(J) = CYYU(J)
S      30 CONTINUE

C
C      COMPUTE RESIDUAL
C
S      DO 40 J = JBOT , JTOP
S      RHS(J) = -(VC(J) - EMU(J , I1)) *
S      1      (CXXL(I)*P(J,I-1) - CXXC(I)*P(J,I) + CXXR(I)*P(J,I+1))
S      40 CONTINUE
M      DO 50 J = JBOT , JTOP
M      RHS(J) = RHS(J) - (EMU(J , I2) * (CXXL(I-1) * P(J , I2)
M      1      - CXXC(I-1)*P(J,I-1) + CXXR(I-1)*P(J,I)))
S      50 CONTINUE
S      JA = JBOT + 1
S      JB = JTOP - 1
S      DO 60 J = JA , JB
S      RHS(J) = RHS(J) - (CYYD(J) * P(J-1 , I) - CYYC(J) * P(J , I)
S      1      + CYYU(J) * P(J+1 , I))
S      60 CONTINUE
S      RHS(JBOT) = RHS(JBOT) - (-CYYC(JBOT) * P(JBOT , I)
S      1      + CYYU(JBOT) * P(JBOT+1 , I))
S      IF (JBOT .EQ. JMIN) GO TO 61
S      RHS(JBOT) = RHS(JBOT) - CYYD(JBOT) * P(JBOT-1 , I)
S      61 CONTINUE
S      RHS(JTOP) = RHS(JTOP) - (CYYD(JTOP) * P(JTOP-1 , I)
S      1      - CYYC(JTOP) * P(JTOP , I))
S      IF (JTOP .EQ. JMAX) GO TO 62
S      RHS(JTOP) = RHS(JTOP) - CYYU(JTOP) * P(JTOP+1 , I)
S      62 CONTINUE

```

図 4.4.5(a) D-5 変更前のプログラム

```

S      DO 10 J = JBOT , JTOP
S      VC(J) = C1(I) - (CXL(I) * POLD(J , I2) + CXC(I) * P(J , I)
1      + CXR(I) * P(J , I+1))
S      EMU(J , I1) = 0.0
S      POLD(J , I1) = P(J , I)
S      IF (VC(J) .LT. 0.0) EMU(J , I1) = VC(J)
S      10 CONTINUE
C
S      IF (FCR) GO TO 22
S      DO 21 J = JBOT , JTOP
S      EMU(J , I2) = EMU(J , I1)
S      21 CONTINUE
S      22 CONTINUE
C
C      COMPUTE ELEMENTS OF MATRIX
C
C      COMPUTE RESIDUAL
CV
S      DO 30 J = JBOT , JTOP
S      DIAG(J) = (EMU(J , I1) - VC(J)) * CXXC(I) * W1
1      + EMU(J , I2) * CXXR(I-1) - CYYC(J)
S      SUP(J) = CYYD(J)
S      SUB(J) = CYYU(J)
S      RHSXJX = -(VC(J) - EMU(J , I1)) *
1      (CXXL(I)*P(J,I-1) - CXXC(I)*P(J,I) + CXXR(I)*P(J,I+1))
S      RHS(J) = RHSXJX - (EMU(J , I2) * (CXXL(I-1) * P(J , I2)
1      - CXXC(I-1)*P(J,I-1) + CXXR(I-1)*P(J,I)))
S      30 CONTINUE
C
S      JA = JBOT + 1
S      JB = JTOP - 1
V      DO 60 J = JA , JB
V      RHS(J) = RHS(J) - (CYYD(J) * P(J-1 , I) - CYYC(J) * P(J , I)
1      + CYYU(J) * P(J+1 , I))
V      60 CONTINUE
S      RHS(JBOT) = RHS(JBOT) - (-CYYC(JBOT) * P(JBOT , I)
1      + CYYU(JBOT) * P(JBOT+1 , I))
S      IF (JBOT .EQ. JMIN) GO TO 61
S      RHS(JBOT) = RHS(JBOT) - CYYD(JBOT) * P(JBOT-1 , I)
S      61 CONTINUE
S      RHS(JTOP) = RHS(JTOP) - (CYYD(JTOP) * P(JTOP-1 , I)
1      - CYYC(JTOP) * P(JTOP , I))
S      IF (JTOP .EQ. JMAX) GO TO 62
S      RHS(JTOP) = RHS(JTOP) - CYYU(JTOP) * P(JTOP+1 , I)
S      62 CONTINUE

```

図 4. 4. 5(b) D-5 変更後のプログラム

```

      K = JMIN - KSTEP
      DO 10 J = JMIN , JMAX
      K = K + KSTEP
      IF (J .EQ. JUP) K = K + KSTEP - 1
      P(J , IMIN) = CIRCFF * VUP(K) + DUB * DUP(K)
      P(J , IMAX) = CIRCFF * VDOWN(K) + DUB * DDOWN(K)
10  CONTINUE
      IF(BCTYPE .NE. 1) GO TO 25
C      UPDATE BOUNDARY CONDITIONS ON TOP AND BOTTOM
      K = IMIN - KSTEP
      DO 20 I = IMIN , IMAX
      K = K + KSTEP
      P(JMIN , I) = CIRCFF * VBOT(K) + DUB * DBOT(K)
      P(JMAX , I) = CIRCFF * VTOP(K) + DUB * DTOP(K)
20  CONTINUE
25  RETURN
      END

```

図 4.4.6(a) D-6 変更前のプログラム

```

      K = JMIN - KSTEP
CV  DO 10 J = JMIN , JUP-1
      K = K + KSTEP
      P(J , IMIN) = CIRCFF * VUP(K) + DUB * DUP(K)
      P(J , IMAX) = CIRCFF * VDOWN(K) + DUB * DDOWN(K)
10  CONTINUE
      K = K + KSTEP - 1
      DO 15 J = JUP , JMAX
      K = K + KSTEP
      P(J , IMIN) = CIRCFF * VUP(K) + DUB * DUP(K)
      P(J , IMAX) = CIRCFF * VDOWN(K) + DUB * DDOWN(K)
15  CONTINUE
C      IF(BCTYPE .NE. 1) GO TO 25
C      UPDATE BOUNDARY CONDITIONS ON TOP AND BOTTOM
      K = IMIN - KSTEP
      DO 20 I = IMIN , IMAX
      K = K + KSTEP
      P(JMIN , I) = CIRCFF * VBOT(K) + DUB * DBOT(K)
      P(JMAX , I) = CIRCFF * VTOP(K) + DUB * DTOP(K)
20  CONTINUE
25  RETURN
      END

```

図 4.4.6(b) D-6 変更後のプログラム

```

SUBROUTINE SYOR
C
C
C
C
CV
  COMMON / CUM1/ P(102,101), X(100), Y(100)
  1  COMMON / CUM1/ IMIN, IMAX, IUP, IDOWN, ILE,
  2  JTE, JMIN, JMAX, JUP, JLOW,
  JTOP, JBOT

```

```

CV
S  DO 10 J = JBOT, JTOP
S  VC(J) = C1(I) - (CXL(I) * POLD(J, I2) + CXC(I) * P(J, I)
1  + CXR(I) * P(J, I+1))
S  EMU(J, I1) = 0.0
S  POLD(J, I1) = P(J, I)
S  IF (VC(J) .LT. 0.0) EMU(J, I1) = VC(J)
S  10 CONTINUE
C
S  IF (FCR) GO TO 22
S  DO 21 J = JBOT, JTOP
S  EMU(J, I2) = EMU(J, I1)
S  21 CONTINUE
S  22 CONTINUE
C
C
C
C
CV
S  DO 30 J = JBOT, JTOP
S  DIAG(J) = (EMU(J, I1) - VC(J)) * CXXC(I) * WJ
1  + EMU(J, I2) * CXXR(I-1) - CYYC(J)
S  SUP(J) = CYYD(J)
S  SUB(J) = CYYU(J)
S  RHSXJX = -(VC(J) - EMU(J, I1)) *
1  (CXXL(I)*P(J, I-1) - CXXC(I)*P(J, I) + CXXR(I)*P(J, I+1))
S  RHS(J) = RHSXJX - (EMU(J, I2) * (CXXL(J-1) * P(J, I2)
1  - CXXC(I-1)*P(J, I-1) + CXXR(I-1)*P(J, I)))
S  30 CONTINUE
C
S  JA = JBOT + 1
S  JB = JTOP - 1
V  DO 60 J = JA, JB
V  RHS(J) = RHS(J) - (CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
1  + CYYU(J) * P(J+1, I))
V  60 CONTINUE
S  RHS(JBOT) = RHS(JBOT) - (-CYYC(JBOT) * P(JBOT, I)
1  + CYYU(JBOT) * P(JBOT+1, I))
S  IF (JBOT .EQ. JMIN) GO TO 61
S  RHS(JBOT) = RHS(JBOT) - CYYD(JBOT) * P(JBOT-1, I)
S  61 CONTINUE
S  RHS(JTOP) = RHS(JTOP) - (CYYD(JTOP) * P(JTOP-1, I)
1  - CYYC(JTOP) * P(JTOP, I))
S  IF (JTOP .EQ. JMAX) GO TO 62
S  RHS(JTOP) = RHS(JTOP) - CYYU(JTOP) * P(JTOP+1, I)
S  62 CONTINUE

```

図 4.4.7(a) D-7 変更前のプログラム

```

SUBROUTINE SYOR
C
C
C
CV  GENERIC
COMMON / CUM1/ P(0:103,101), X(100), Y(100)
1  IMIN, IMAX, IUP, IDOWN, ILE,
2  ITE, JMIN, JMAX, JUP, JLOW,
JTUP, JBOT

```

```

S  IF ( FCR ) THEN
CV
S  DO 10 J = JBOT, JTUP
S  VC(J) = C1(I) - (CXAL(I) * POLD(J, I2) + CXCL(I) * P(J, I)
1  + CXR(I) * P(J, I+1))
S  EMU(J, I1) = 0.0
S  POLD(J, I1) = P(J, I)
S  IF (VC(J) .LT. 0.0) EMU(J, I1) = VC(J)
S  DIAG(J) = (EMU(J, I1) - VC(J)) * WIV
1  + EMU(J, I2) * CXXR(I-1) - CYYC(J)
S  SUP(J) = CYYD(J)
S  SUB(J) = CYYU(J)
S  RHSXJX = -(VC(J) - EMU(J, I1)) *
1  (CXAL(I)*P(J,I-1) - CXXC(I)*P(J,I) + CXXR(I)*P(J,I+1))
S  RHSXJX = RHSXJX - (EMU(J, I2) * (CXAL(I-1) * P(J, I2)
1  - CXXC(I-1)*P(J,I-1) + CXXR(I-1)*P(J,I)))
S  RHS(J) = RHSXJX - (CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
1  + CYYU(J) * P(J+1, I))
10 CONTINUE
CV
S  ELSE
CV
S  DO 20 J = JBOT, JTUP
S  VC(J) = C1(I) - (CXAL(I) * POLD(J, I2) + CXCL(I) * P(J, I)
1  + CXR(I) * P(J, I+1))
S  EMU(J, I1) = 0.0
S  POLD(J, I1) = P(J, I)
S  IF (VC(J) .LT. 0.0) EMU(J, I1) = VC(J)
C
S  EMU(J, I2) = EMU(J, I1)
C
S  DIAG(J) = (EMU(J, I1) - VC(J)) * WIV
1  + EMU(J, I2) * CXXR(I-1) - CYYC(J)
S  SUP(J) = CYYD(J)
S  SUB(J) = CYYU(J)
S  RHSXJX = -(VC(J) - EMU(J, I1)) *
1  (CXAL(I)*P(J,I-1) - CXXC(I)*P(J,I) + CXXR(I)*P(J,I+1))
S  RHSXJX = RHSXJX - (EMU(J, I2) * (CXAL(I-1) * P(J, I2)
1  - CXXC(I-1)*P(J,I-1) + CXXR(I-1)*P(J,I)))
S  RHS(J) = RHSXJX - (CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
1  + CYYU(J) * P(J+1, I))
20 CONTINUE
CV
ENDIF

```

図 4.4.7(b) D-7 変更後のプログラム

```

M      DO 15 I = ILE , ITE
CV      WPX1 = PX(I , JLOW)
CV      WPX2 = PX(I , JLOW-1)
CV      WPX3 = PX(I , JUP)
CV      WPX4 = PX(I , JUP+1)
V      IF (I.NE.IMIN .AND. I.NE.IMAX) THEN
M          WPX1 = .5 * ( XDIFF(I+1) * (P(JLOW,I+1) - P(JLOW,I))
*              + XDIFF(I) * (P(JLOW,I) - P(JLOW,I-1)) )
M          WPX2 = .5 * ( XDIFF(I+1) * (P(JLOW-1,I+1) - P(JLOW-1,I))
*              + XDIFF(I) * (P(JLOW-1,I) - P(JLOW-1,I-1)) )
M          WPX3 = .5 * ( XDIFF(I+1) * (P(JUP,I+1) - P(JUP,I))
*              + XDIFF(I) * (P(JUP,I) - P(JUP,I-1)) )
M          WPX4 = .5 * ( XDIFF(I+1) * (P(JUP+1,I+1) - P(JUP+1,I))
*              + XDIFF(I) * (P(JUP+1,I) - P(JUP+1,I-1)) )
*
V      ENDIF
V      IF (I.EQ.IMIN) THEN
M          WPX1 = 1.5 * XDIFF(I+1) * (P(JLOW,I+1) - P(JLOW,I))
*              - 0.5 * XDIFF(I+2) * (P(JLOW,I+2) - P(JLOW,I+1))
M          WPX2 = 1.5 * XDIFF(I+1) * (P(JLOW-1,I+1) - P(JLOW-1,I))
*              - 0.5 * XDIFF(I+2) * (P(JLOW-1,I+2) - P(JLOW-1,I+1))
M          WPX3 = 1.5 * XDIFF(I+1) * (P(JUP,I+1) - P(JUP,I))
*              - 0.5 * XDIFF(I+2) * (P(JUP,I+2) - P(JUP,I+1))
M          WPX4 = 1.5 * XDIFF(I+1) * (P(JUP+1,I+1) - P(JUP+1,I))
*              - 0.5 * XDIFF(I+2) * (P(JUP+1,I+2) - P(JUP+1,I+1))
*
V      ENDIF
V      IF (I.EQ.IMAX) THEN
M          WPX1 = 1.5 * XDIFF(I) * (P(JLOW,I) - P(JLOW,I-1))
*              - 0.5 * XDIFF(I-1) * (P(JLOW,I-1) - P(JLOW,I-2))
M          WPX2 = 1.5 * XDIFF(I) * (P(JLOW-1,I) - P(JLOW-1,I-1))
*              - 0.5 * XDIFF(I-1) * (P(JLOW-1,I-1) - P(JLOW-1,I-2))
M          WPX3 = 1.5 * XDIFF(I) * (P(JUP,I) - P(JUP,I-1))
*              - 0.5 * XDIFF(I-1) * (P(JUP,I-1) - P(JUP,I-2))
M          WPX4 = 1.5 * XDIFF(I) * (P(JUP+1,I) - P(JUP+1,I-1))
*              - 0.5 * XDIFF(I-1) * (P(JUP+1,I-1) - P(JUP+1,I-2))
*
V      ENDIF
C
M      UL      = CJLOW * WPX1 - CJLOW1 * WPX2
V      CPOLD   = CPL(I)
V      CPL(I)  = -2.0 * UL * CPFACT
V      CPERR   = ABS(CPL(I) - CPOLD)
CV
V      IF (CPERR .GT. CPMAXL) THEN
V          CPMAXL = CPERR
V          IERRL  = I
V      ENDIF
C
M      UU      = CJUP * WPX3 - CJUP1 * WPX4
V      CPOLD   = CPU(I)
V      CPU(I)  = -2.0 * UU * CPFACT
V      CPERR   = ABS(CPU(I) - CPOLD)
CV
V      IF (CPERR .GT. CPMAXU) THEN
V          CPMAXU = CPERR
V          IERRU  = I
V      ENDIF
V      15 CONTINUE

```

図 4. 4. 8(a) D-8 変更前のプログラム

```

V      DO 15 I = ILE , ITE
CV      WPX(I,1) = PX(I , JLOW)
CV      WPX(I,2) = PX(I , JLOW-1)
CV      WPX(I,3) = PX(I , JUP)
CV      WPX(I,4) = PX(I , JUP+1)
V      IF (I.NE.IMIN .AND. I.NE.IMAX) THEN
V          WPX(I,1) = .5 * ( XDIFF(I+1) * (P(JLOW,I+1) - P(JLOW,I))
*              + XDIFF(I) * (P(JLOW,I) - P(JLOW,I-1)) )
V          WPX(I,2) = .5 * ( XDIFF(I+1) * (P(JLOW-1,I+1) - P(JLOW-1,I))
*              + XDIFF(I) * (P(JLOW-1,I) - P(JLOW-1,I-1)) )
V          WPX(I,3) = .5 * ( XDIFF(I+1) * (P(JUP,I+1) - P(JUP,I))
*              + XDIFF(I) * (P(JUP,I) - P(JUP,I-1)) )
V          WPX(I,4) = .5 * ( XDIFF(I+1) * (P(JUP+1,I+1) - P(JUP+1,I))
*              + XDIFF(I) * (P(JUP+1,I) - P(JUP+1,I-1)) )
V          ENDIF
V          IF (I.EQ.IMIN) THEN
V              WPX(I,1) = 1.5 * XDIFF(I+1) * (P(JLOW,I+1) - P(JLOW,I))
*                  - 0.5 * XDIFF(I+2) * (P(JLOW,I+2) - P(JLOW,I+1))
V              WPX(I,2) = 1.5 * XDIFF(I+1) * (P(JLOW-1,I+1) - P(JLOW-1,I))
*                  - 0.5 * XDIFF(I+2) * (P(JLOW-1,I+2) - P(JLOW-1,I+1))
V              WPX(I,3) = 1.5 * XDIFF(I+1) * (P(JUP,I+1) - P(JUP,I))
*                  - 0.5 * XDIFF(I+2) * (P(JUP,I+2) - P(JUP,I+1))
V              WPX(I,4) = 1.5 * XDIFF(I+1) * (P(JUP+1,I+1) - P(JUP+1,I))
*                  - 0.5 * XDIFF(I+2) * (P(JUP+1,I+2) - P(JUP+1,I+1))
V              ENDIF
V          IF (I.EQ.IMAX) THEN
V              WPX(I,1) = 1.5 * XDIFF(I) * (P(JLOW,I) - P(JLOW,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JLOW,I-1) - P(JLOW,I-2))
V              WPX(I,2) = 1.5 * XDIFF(I) * (P(JLOW-1,I) - P(JLOW-1,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JLOW-1,I-1) - P(JLOW-1,I-2))
V              WPX(I,3) = 1.5 * XDIFF(I) * (P(JUP,I) - P(JUP,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JUP,I-1) - P(JUP,I-2))
V              WPX(I,4) = 1.5 * XDIFF(I) * (P(JUP+1,I) - P(JUP+1,I-1))
*                  - 0.5 * XDIFF(I-1) * (P(JUP+1,I-1) - P(JUP+1,I-2))
V              ENDIF
C
V          UL      = CJLOW * WPX(I,1) - CJLOW1 * WPX(I,2)
V          CPOLD   = CPL(I)
V          CPL(I)  = -2.0 * UL * CPFACT
V          CPERR   = ABS(CPL(I) - CPOLD)
CV
V          IF (CPERR .GT. CPMAXL) THEN
V              CPMAXL = CPERR
V              IERRL  = I
V          ENDIF
C
V          UU      = CJUP * WPX(I,3) - CJUP1 * WPX(I,4)
V          CPOLD   = CPU(I)
V          CPU(I)  = -2.0 * UU * CPFACT
V          CPERR   = ABS(CPU(I) - CPOLD)
CV
V          IF (CPERR .GT. CPMAXU) THEN
V              CPMAXU = CPERR
V              IERRU  = I
V          ENDIF
V      15 CONTINUE

```

図 4. 4. 8(b) D-8 変更後のプログラム

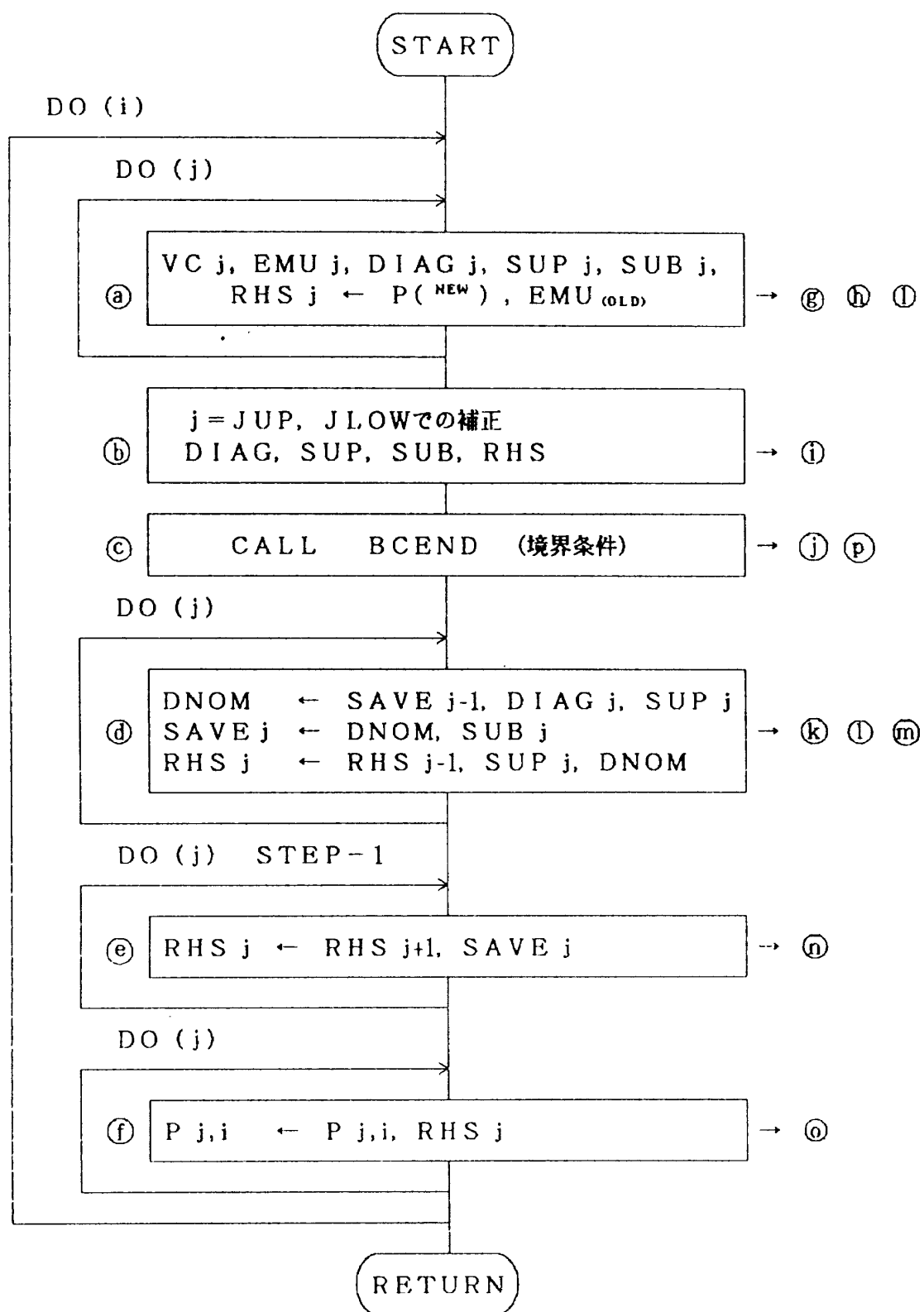


図 4.4.9(a) D-9 改造前の処理フロー

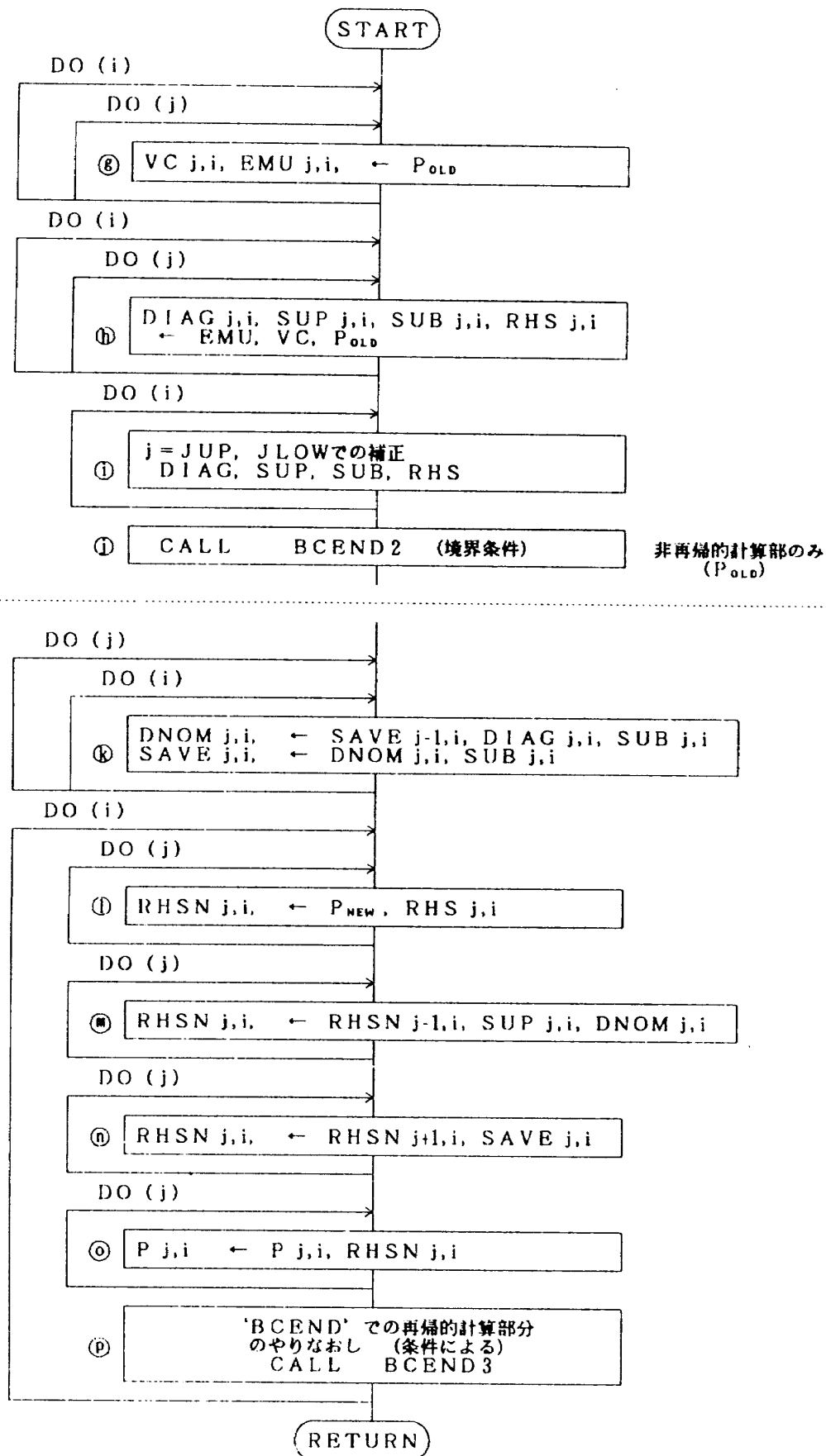


図 4.4.9(b) D-9 改造後の処理フロー

```

C
  J      = JUP
S  DIAG(J) = DIAG(J) + CYYC(J) - CYYBUC
S  SUP(J)  = 0.0
S  SUB(J)  = CYYBUU
S  RHS(J)  = RHS(J) + CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
1      + CYYU(J) * P(J+1, I)
2      - (-CYYBUC * P(J, I) + CYYBUU * P(J+1, I) + FXUBC(I))
  J      = JLOW
S  DIAG(J) = DIAG(J) + CYYC(J) - CYYBLC
S  SUP(J)  = CYYBLD
S  SUB(J)  = 0.0
S  RHS(J)  = RHS(J) + CYYD(J) * P(J-1, I)
1      - CYYC(J) * P(J, I) + CYYU(J) * P(J+1, I)
2      - (-CYYBLC * P(J, I) + CYYBLD * P(J-1, I) + FXLBC(I))
S  GO TO 80
C

```

図 4. 4. 10(a) D-9(イ) 変更前のプログラム

```

C
V  DO 1400 I = ILE, ITE
C
  J      = JUP
V  DIAG( J, I ) = DIAG(J, I) + CYYC(J) - CYYBUC
V  SUP( J, I )  = 0.0
V  SUB( J, I )  = CYYBUU
V  RHS( J, I )  = RHS(J, I) + CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
1      + CYYU(J) * P(J+1, I)
2      - ( -CYYBUC * P(J, I) + CYYBUU * P(J+1, I) + FXUBC(I) )
  J      = JLOW
V  DIAG( J, I ) = DIAG( J, I ) + CYYC(J) - CYYBLC
V  SUP( J, I )  = CYYBLD
V  SUB( J, I )  = 0.0
V  RHS( J, I )  = RHS(J, I) + CYYD(J) * P(J-1, I)
1      - CYYC(J) * P(J, I) + CYYU(J) * P(J+1, I)
2      - ( -CYYBLC * P(J, I) + CYYBLD * P(J-1, I) + FXLBC(I) )
C
V 1400 CONTINUE
C

```

図 4. 4. 10(b) D-9(イ) 変更後のプログラム

```

C
C
H 200 I = IUP, IDOWN
C
C      COMPUTE VC = 1 - H**2
C      COMPUTE ELEMENTS OF MATRIX
C      COMPUTE RESIDUAL
CV
S      WIV = CXXC(I) * WI
C
CV
S      IF ( FCR ) THEN
CV
S      DO 10 J = JBOT, JTOP
S      VC(J) = C1(I) - (CXL(I) * POLD(J, I2) + CXC(I) * P(J, I)
1      + CXR(I) * P(J, I+1))
S      EMU(J, I1) = 0.0
S      POLD(J, I1) = P(J, I)
S      IF (VC(J) .LT. 0.0) EMU(J, I1) = VC(J)
S      DIAG(J) = (EMU(J, I1) - VC(J)) * WIV
1      + EMU(J, I2) * CXXR(I-1) - CYYC(J)
S      SUP(J) = CYYD(J)
S      SUB(J) = CYYU(J)
S      RHSXJX = -(VC(J) - EMU(J, I1)) *
1      (CXXL(I)*P(J,I-1) - CXXC(I)*P(J,I) + CXXR(I)*P(J,I+1))
S      RHSXJX = RHSXJX - (EMU(J, I2) * (CXXL(I-1) * P(J, I2)
1      - CXXC(I-1)*P(J,I-1) + CXXR(I-1)*P(J,I)))
S      RHS(J) = RHSXJX - (CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
1      + CYYU(J) * P(J+1, I))
10 CONTINUE
CV
S      ELSE
CV
S      DO 20 J = JBOT, JTOP
S      VC(J) = C1(I) - (CXL(I) * POLD(J, I2) + CXC(I) * P(J, I)
1      + CXR(I) * P(J, I+1))
S      EMU(J, I1) = 0.0
S      POLD(J, I1) = P(J, I)
S      IF (VC(J) .LT. 0.0) EMU(J, I1) = VC(J)
C
S      EMU(J, I2) = EMU(J, I1)
C
S      DIAG(J) = (EMU(J, I1) - VC(J)) * WIV
1      + EMU(J, I2) * CXXR(I-1) - CYYC(J)
S      SUP(J) = CYYD(J)
S      SUB(J) = CYYU(J)
S      RHSXJX = -(VC(J) - EMU(J, I1)) *
1      (CXXL(I)*P(J,I-1) - CXXC(I)*P(J,I) + CXXR(I)*P(J,I+1))
S      RHSXJX = RHSXJX - (EMU(J, I2) * (CXXL(I-1) * P(J, I2)
1      - CXXC(I-1)*P(J,I-1) + CXXR(I-1)*P(J,I)))
S      RHS(J) = RHSXJX - (CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
1      + CYYU(J) * P(J+1, I))
20 CONTINUE
CV
S      ENDIF
CV

```

図 4.4.11(a) D-9 (ロ) 変更前のプログラム

```

C
S      DO 1100 I = IUP, IDOWN
C
V      DO 1110 J = JBOT, JTOP
C
V      VC(J, I) = C1(I) - ( CXL(I) * P(J, I-1) + CXC(I) * P(J, I)
+      + CXR(I) * P(J, I+1) )
C
V      EMU(J, I) = AMIN1( VC(J, I), 0.0 )
C
V 1110 CONTINUE
S 1100 CONTINUE
C
S      DO 1200 I = IUP, IDOWN
C
S      CXXCWI = CXXC(I) * WI
C
V      DO 1210 J = JBOT, JTOP
C
V      DIAG(J, I) = ( EMU(J, I) - VC(J, I) ) * CXXCWI
+      + EMU(J, IDEMU(I)) * CXXR(I-1) - CYYC(J)
+      - EPSX(I)
V      SUP(J, I) = CYYD(J)
V      SUB(J, I) = CYYU(J)
C
V      RHS(J, I) = - ( CYYD(J) * P(J-1, I) - CYYC(J) * P(J, I)
+      + CYYU(J) * P(J+1, I) )
V 1210 CONTINUE
S 1200 CONTINUE

```

図 4.4.11(b) D-9(ロ) 変更後のプログラム

```

IF (FCR) THEN
  DO 10 J=JBOT,JTOP
    :
    EMU(J,I1)=0.0
    :
    IF (VC(J).LT.0.0) EMU(J,I1)=VC(J)
    DIAG(J)= (EMU(J,I1) . . . . . ) * WIV
1    + EMU(J,I2) * . . . . .
    :
10  CONTINUE
ELSE
  DO 20 J=JBOT,JTOP
    :
    EMU(J,I1)=0.0
    IF (VC(J).LT.0.0) EMU(J,I1)=VC(J)
    EMU(J,I2)=EMU(J,I1)
    DIAG(J)= (EMU(J,I1)-VC(J)) * WIV
1    + EMU(J,I2) * . . . . .
    :
20  CONTINUE
ENDIF

```

```

:
IFCR=0
IF (FCR) IFCR=1
DO 1000 I=IUP,IDOWN
:
IDEMU(I)=1-IFCR
:
1000 CONTINUE
:

```

以下に図 4. 4. 10 (b) に示すステートメントが続く

図 4. 4. 12(b) D-9(ロ) 変更後のプログラム

図 4. 4. 12(a) D-9 (ロ) 変更前のプログラム

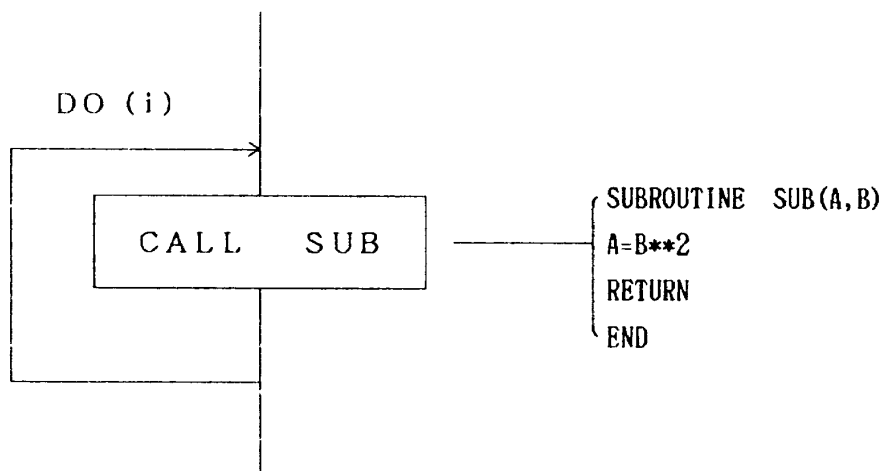


図 4. 4. 13(a) D-9 (ニ) 変更前のプログラム構造

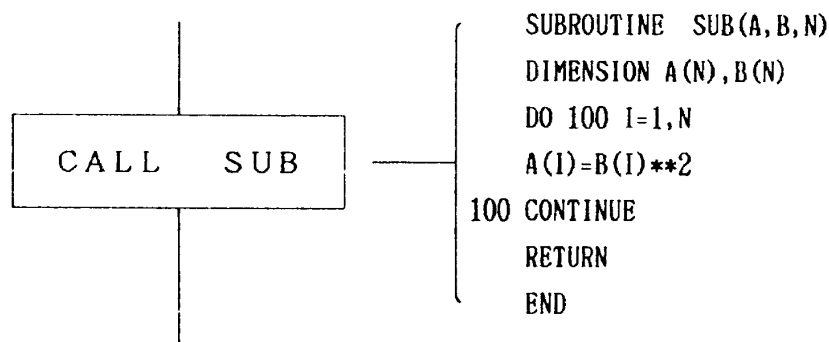


図 4. 4. 13(b) D-9(ニ) 変更後のプログラム構造

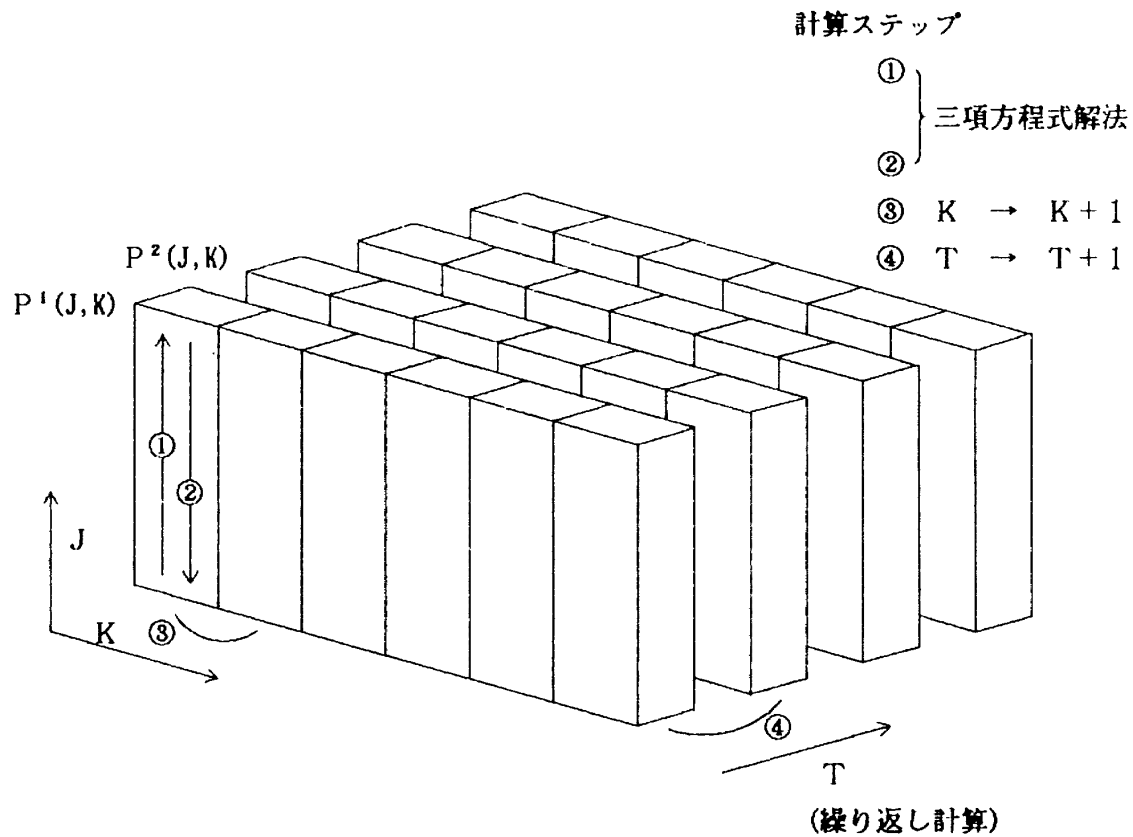


図 4.5.1 E-1 オリジナルプログラムの計算構造

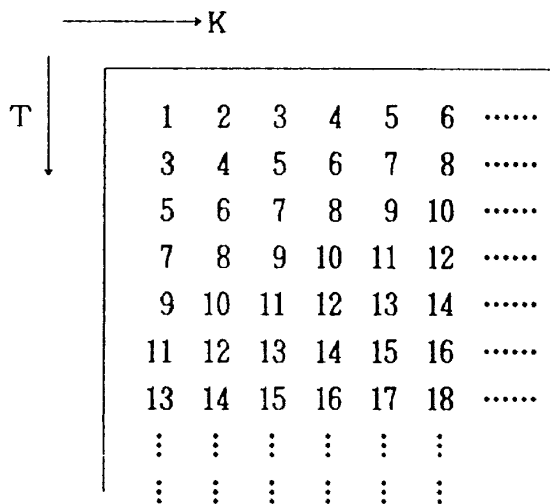


図 4.5.2 E-1 KとTの関係

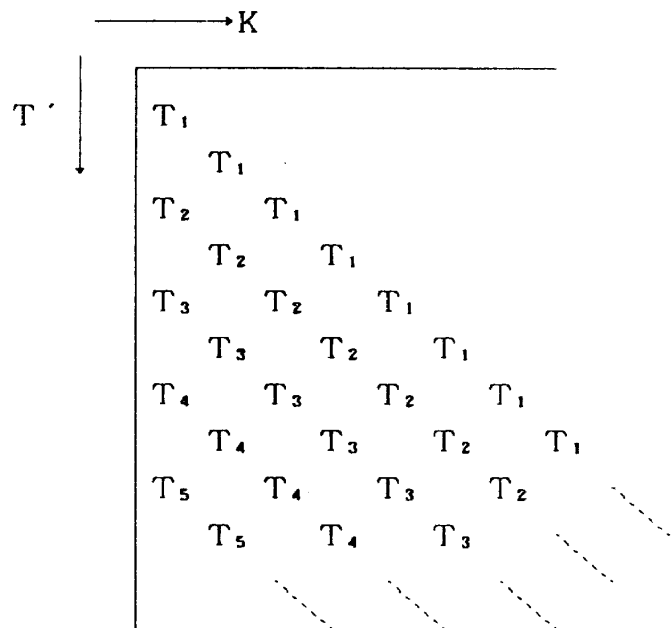


図 4.5.3 E-1 KとT'の関係

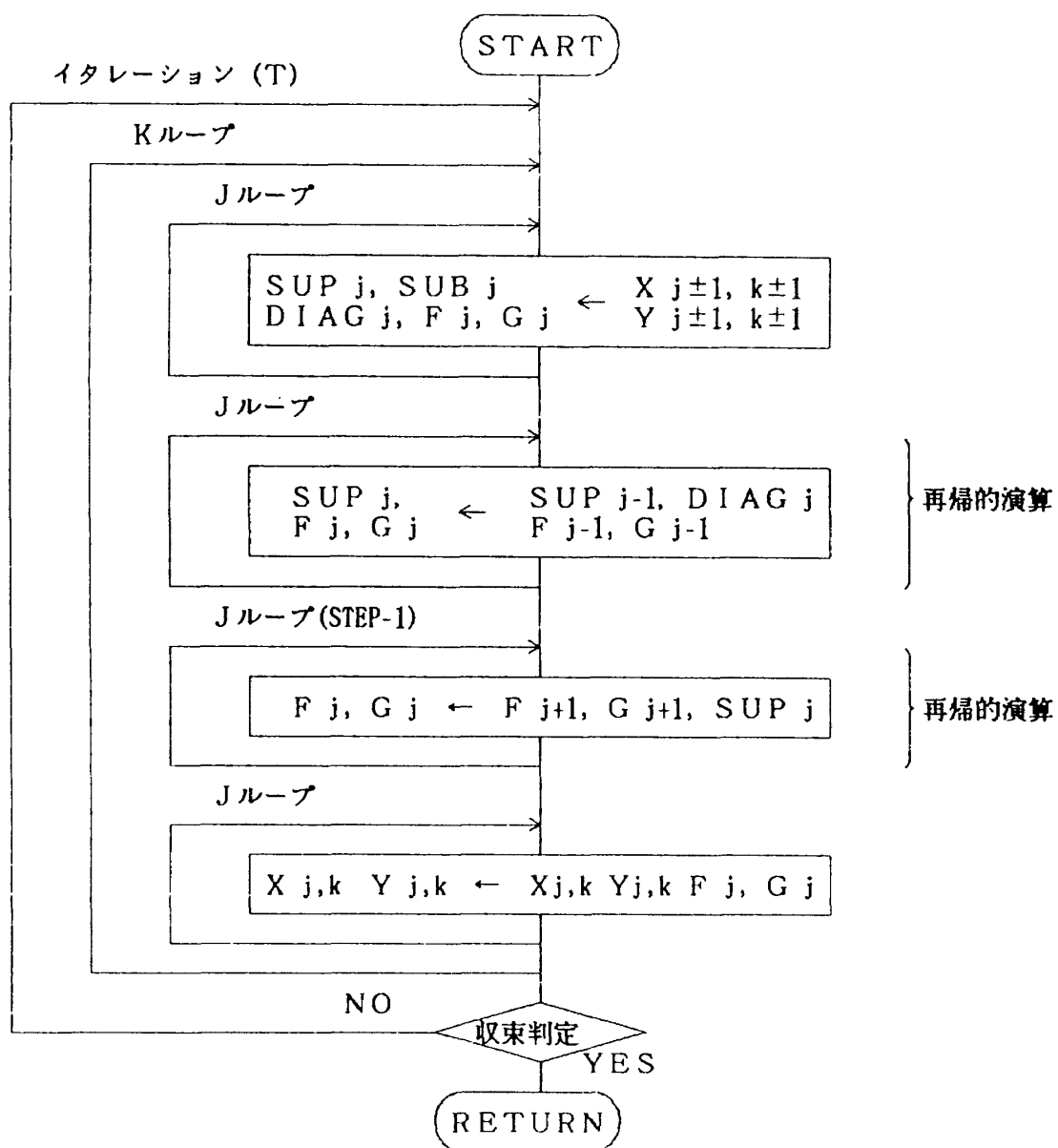


図 4. 5. 4(a) E-1 改造前の処理フロー

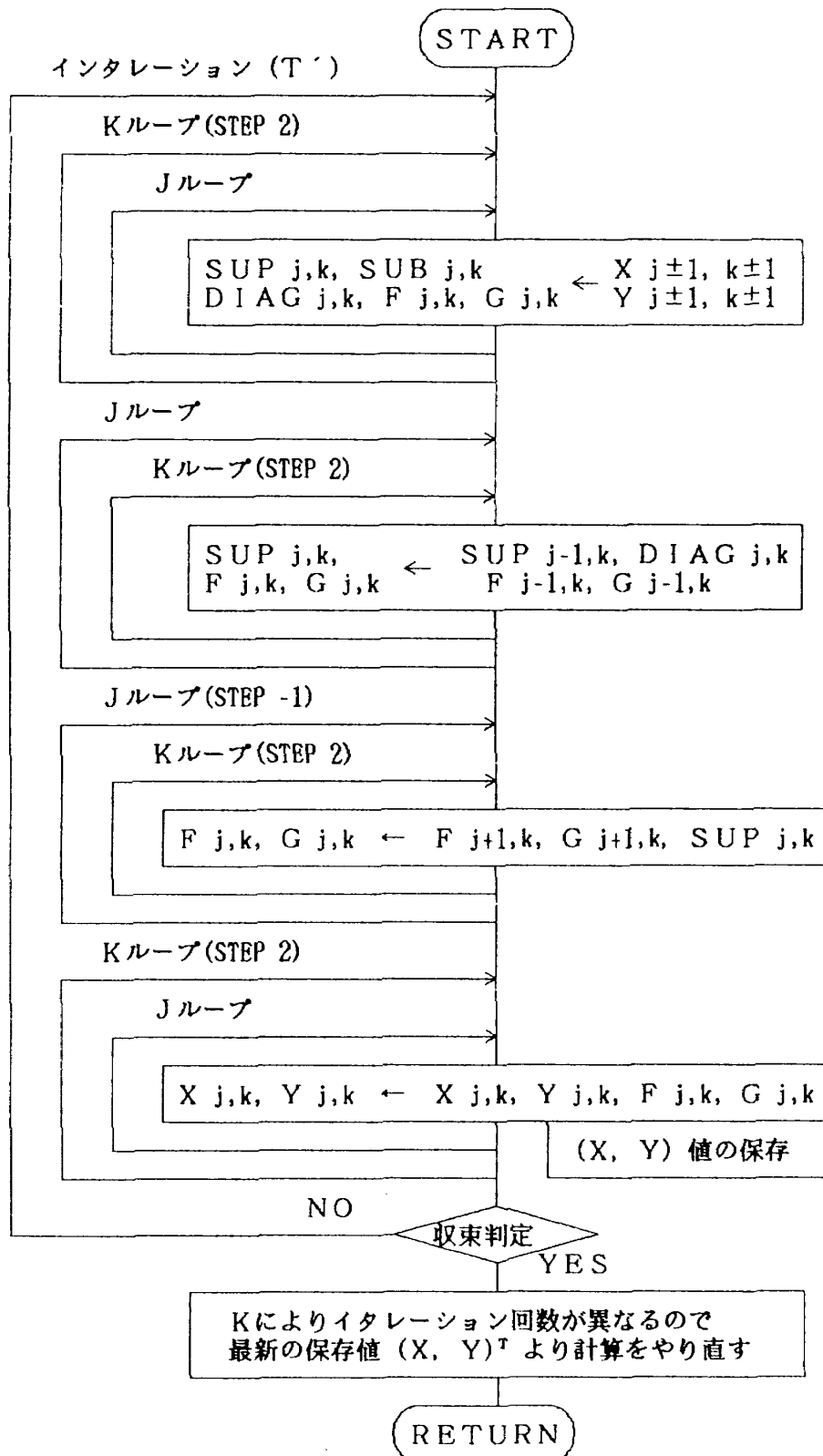


図 4.5.4(b) E-1 改造後の処理フロー

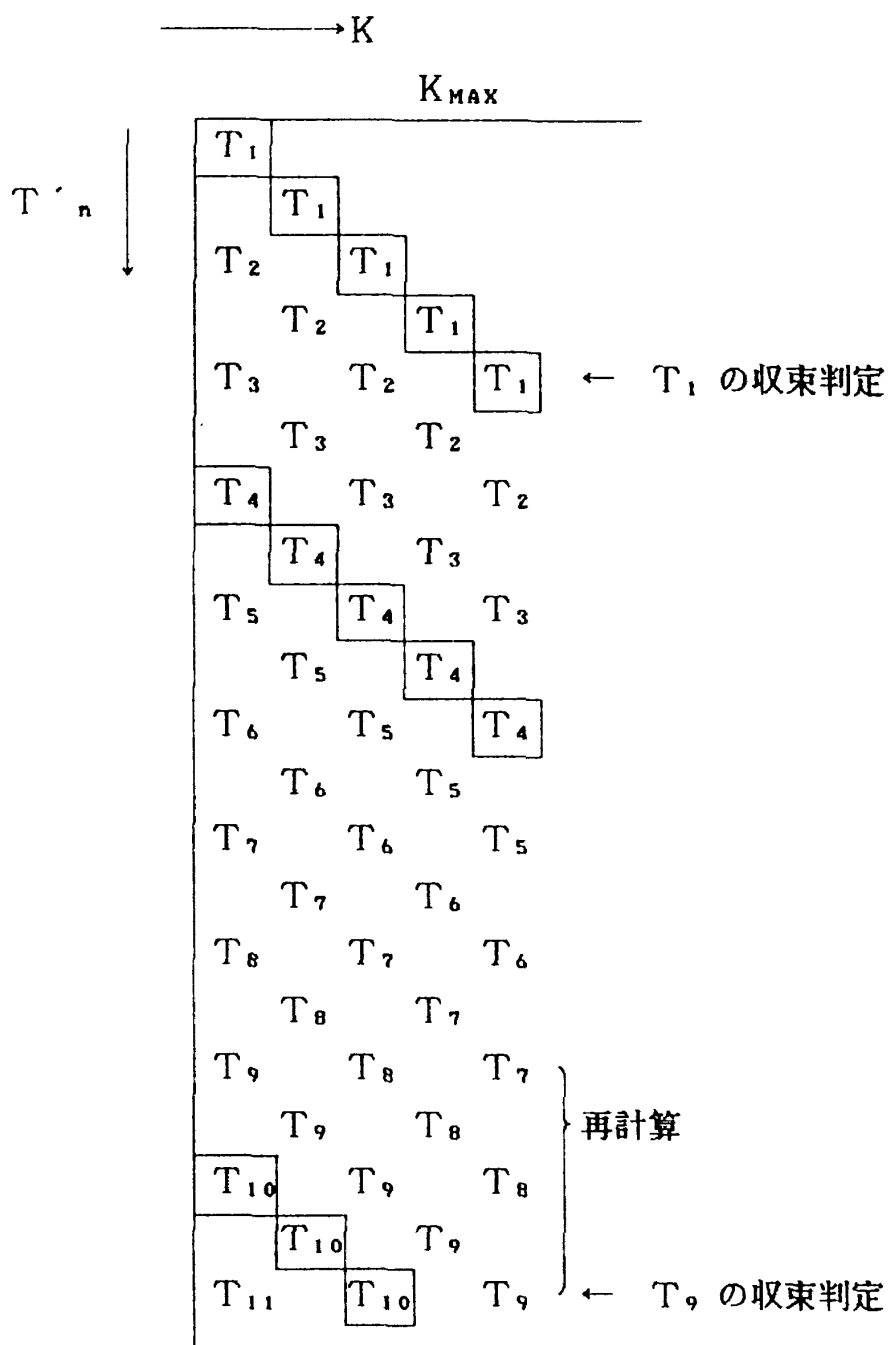


図 4.5.5 E-1 収束判定の方法

```

DO 11950 IDXJ = JS , JE
WXXI      = 0.5 * ( X (IDXJ+1,K ) - X (IDXJ-1,K ) )
WYXI      = 0.5 * ( Y (IDXJ+1,K ) - Y (IDXJ-1,K ) )
WXETA     = 0.5 * ( X (IDXJ ,K+1) - X (IDXJ ,K-1) )
WYETA     = 0.5 * ( Y (IDXJ ,K+1) - Y (IDXJ ,K-1) )

WSUP      = WXETA * WXETA + WYETA * WYETA
SUP (IDXJ,K) = WSUP
SUB (IDXJ,K) = WSUP
WGD       = WXXI * WXXI + WYXI * WYXI
DIAG (IDXJ,K) = (-2.0) * (WSUP + WGD)
WBD       = 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) ) )

F (IDXJ,K) = WBD * 0.25 * ( X (IDXJ+1,K+1) - X (IDXJ+1,K-1)
1          - X (IDXJ-1,K+1) + X (IDXJ-1,K-1) )
2          - WGD * ( X (IDXJ ,K+1) + X (IDXJ ,K-1) )
G (IDXJ,K) = WBD * 0.25 * ( Y (IDXJ+1,K+1) - Y (IDXJ+1,K-1)
1          - Y (IDXJ-1,K+1) + Y (IDXJ-1,K-1) )
2          - WGD * ( Y (IDXJ ,K+1) + Y (IDXJ ,K-1) )
11950 CONTINUE

```

図 4.5.6(a) E-2 変更前のプログラム

```

DO 11950 IDXJ = JS , JE
WXETA     = X (IDXJ ,K+1) - X (IDXJ ,K-1)
WYETA     = Y (IDXJ ,K+1) - Y (IDXJ ,K-1)
WSUP      = 0.25 * ( WXETA * WXETA + WYETA * WYETA )
SUP (IDXJ,K) = WSUP
SUB (IDXJ,K) = WSUP
WXXI      = X (IDXJ+1,K ) - X (IDXJ-1,K )
WYXI      = Y (IDXJ+1,K ) - Y (IDXJ-1,K )
WGD       = 0.25 * ( WXXI * WXXI + WYXI * WYXI )
DIAG (IDXJ,K) = (-2.0) * (WSUP + WGD)
WBD       = 0.125 * ( WXXI * WXETA + WYXI * WYETA )

F (IDXJ,K) = WBD * ( X (IDXJ+1,K+1) - X (IDXJ+1,K-1)
1          - X (IDXJ-1,K+1) + X (IDXJ-1,K-1) )
2          - WGD * ( X (IDXJ ,K+1) + X (IDXJ ,K-1) )
G (IDXJ,K) = WBD * ( Y (IDXJ+1,K+1) - Y (IDXJ+1,K-1)
1          - Y (IDXJ-1,K+1) + Y (IDXJ-1,K-1) )
2          - WGD * ( Y (IDXJ ,K+1) + Y (IDXJ ,K-1) )
11950 CONTINUE

```

図 4.5.6(b) E-2 変更後のプログラム

```

C
S      DO 13 I = IL , IU
S      IF (I .EQ. IL) THEN
V      DO 11 K = 1 , 4
V      DO 11 M = 1 , 4
V      G(K,M) = B(I,K,M)
V      11 CONTINUE
S      ELSE
V      IR = I - 1
V      DO 12 K = 1 , 4
V      F(I,K) = F(I,K) - A(I,K,1) * F(IR,1) - A(I,K,2) * F(IR,2)
V      1      - A(I,K,3) * F(IR,3) - A(I,K,4) * F(IR,4)
V      12 CONTINUE
S      DO 14 M = 1 , 4
V      DO 14 K = 1 , 4
V      G(K,M) = B(I,K,M) - A(I,K,1) * B(IR,1,M) - A(I,K,2) * B(IR,2,M)
V      1      - A(I,K,3) * B(IR,3,M) - A(I,K,4) * B(IR,4,M)
V      14 CONTINUE
S      ENDIF
C

```

図 4.6.1(a) F-1 変更前のプログラム

```

      DIMENSION      IX(16), JX(16), SUM(4)
      DATA IX / 1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4 /
      DATA JX / 1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4 /

```

```

C
S      DO 13 I = IL , IU
S      IF (I .EQ. IL) THEN
V      DO 11 L = 1 , 16
V      K = IX(L)
V      M = JX(L)
V      G(K,M) = B(I,K,M)
V      11 CONTINUE
S      ELSE
V      IR = I - 1
V      DO 12 K = 1 , 4
V      F(I,K) = F(I,K) - A(I,K,1) * F(IR,1) - A(I,K,2) * F(IR,2)
V      +      - A(I,K,3) * F(IR,3) - A(I,K,4) * F(IR,4)
V      12 CONTINUE
S      DO 14 L = 1 , 16
V      M = JX(L)
V      K = IX(L)
V      G(K,M) = B(I,K,M) - A(I,K,1) * B(IR,1,M) - A(I,K,2) * B(IR,2,M)
V      1      - A(I,K,3) * B(IR,3,M) - A(I,K,4) * B(IR,4,M)
V      14 CONTINUE
S      ENDIF
C

```

図 4.6.1(b) F-1 変更後のプログラム

```

C
  IT = IL + IU
S   DO 21 II = IS, IU
    I = IT - II
    IP = I + 1
S   DO 17 M = 1, 4
S   F(I,M) = F(I,M) - B(I,M,1) * F(IP,1) - B(I,M,2) * F(IP,2)
    1      - B(I,M,3) * F(IP,3) - B(I,M,4) * F(IP,4)
S   17 CONTINUE
S   21 CONTINUE
C

```

図 4.6.2(a) F-2 変更前のプログラム

```

C
V   DO 210 M = 1, 4
V   SUM(M) = B(IU-1,M,1) * F(IU,1) + B(IU-1,M,2) * F(IU,2)
    +      B(IU-1,M,3) * F(IU,3) + B(IU-1,M,4) * F(IU,4)
V   210 CONTINUE
C
  IT = IL + IU
S   DO 21 II = IS, IU
    I = IT - II
    DO 17 M = 1, 4
V   F(I,M) = F(I,M) - SUM(M)
V   17 CONTINUE
V   DO 170 M = 1, 4
V   SUM(M) = B(I-1,M,1) * F(I,1) + B(I-1,M,2) * F(I,2)
    +      B(I-1,M,3) * F(I,3) + B(I-1,M,4) * F(I,4)
V   170 CONTINUE
S   21 CONTINUE
C

```

図 4.6.2(b) F-2 変更後のプログラム

```

S   DO 40 M = 1, 4
S   DO 40 N = 1, 4
S   A(KMAX,N,M) = D(2,N,M)
V   DO 40 K = 2, KMAX
    I = KMAX + 1 - K
V   A(I,N,M) = D(K+1,N,M)
V   C(I,N,M) = -D(K-1,N,M)
V   40 CONTINUE

```

図 4.6.3(a) F-3 変更前のプログラム

```

V   DO 100 K = 2, KMAX
V   IXX(K) = KMAX + 1 - K
V   100 CONTINUE
C
S   DO 40 M = 1, 4
S   DO 40 N = 1, 4
S   A(KMAX,N,M) = D(2,N,M)
V   DO 40 K = 2, KMAX
V   I = IXX(K)
V   A(I,N,M) = D(K+1,N,M)
V   C(I,N,M) = -D(K-1,N,M)
V   40 CONTINUE

```

← インデックス

図 4.6.3(b) F-3 変更後のプログラム

```

S      DO 47 K = 2 , KMAX
S      Z(K+KM) = P(JJ,K)
        I      = KMAX + 1 - K
S      Z(I) = P(J,K)
S      47 CONTINUE

```

図 4.6.4(a) F-4 変更前のプログラム

```

V      DO 47 K = 2 , KMAX
V      Z(K+KM) = P(JJ,K)
V      47 CONTINUE
V      DO 48 K = 2 , KMAX
V      I      = IXX(K)
V      Z(I) = P(J,K)
V      48 CONTINUE

```

図 4.6.4(b) F-4 変更後のプログラム

```

S      DO 20 J = 2 , JM
S      Z(J)      = SMUIM / P(J,K)
S      EA(J)     = 1.0 + EB(J-1) - EF(J)
S      EC(J)     = 1.0 - EB(J) + EF(J+1)
S      DO 20 M = 1 , 4
S      DO 20 N = 1 , 4
V      A(J,N,M)  = -EA(J) * D(J-1,N,M)
V      B(J,N,M)  = (EA(J) - EC(J)) * D(J,N,M)
V      C(J,N,M)  = EC(J) * D(J+1,N,M)
V      20 CONTINUE

```

図 4.6.5(a) F-5 変更前のプログラム

```

M      DO 20 J = 2 , JM
S      Z(J)      = SMUIM / P(J,K)
V      EA(J)     = 1.0 + EB(J-1) - EF(J)
V      EC(J)     = 1.0 - EB(J) + EF(J+1)
V      20 CONTINUE
S      DO 21 M = 1 , 4
S      DO 21 N = 1 , 4
V      DU 21 J = 2 , JM
V      A(J,N,M)  = -EA(J) * D(J-1,N,M)
V      B(J,N,M)  = (EA(J) - EC(J)) * D(J,N,M)
V      C(J,N,M)  = EC(J) * D(J+1,N,M)
V      21 CONTINUE

```

図 4.6.5(b) F-5 変更後のプログラム

```

C
S      IFLG = 0
S      DO 1200 J = 2 , JM
        I = JMAX1 - J2
S      IF (EB(I) .LT. 0.1) THEN
S      IFLG = 0
S      ELSE
S      IFLG = IFLG + 1
S      IF (IFLG .GT. 3) EB(I) = 0.0
S      ENDIF
S      1200 CONTINUE
C
S      IFLG = 0
S      DO 1300 J = 2 , JM
S      IF (EF(J) .LT. 0.1) THEN
S      IFLG = 0
S      ELSE
S      IFLG = IFLG + 1
S      IF (IFLG .GT. 3) EF(J) = 0.0
S      ENDIF
S      1300 CONTINUE

```

図 4.6.6(a) F-6 変更前のプログラム

```

C
C..... VP TUNNING
V      DO 1200 J = JMAX1 - JM, JMAX1 - 2
V      IF (EB(J) .LT. 0.1) THEN
V      IFVP(J) = 0
V      ELSE
V      IFVP(J) = 1
V      ENDIF
V      1200 CONTINUE
V      DO 1201 J = JMAX1 - JM, JMAX1 - 5

```

FORTRAN 77/VP V10L20 SHCKOP DATE 87.04.22 TIME 17.12.05

```

V      IF( IFVP(J).GE.1 .AND. IFVP(J+1).GE.1
+      .AND. IFVP(J+2).GE.1 .AND. IFVP(J+3).GE.1 ) EB(J) = 0.0
V      1201 CONTINUE
C..... VP TUNNING
V      DO 1300 J = 2 , JM
V      IF (EF(J) .LT. 0.1) THEN
V      IFVP(J) = 0
V      ELSE
V      IFVP(J) = 1
V      ENDIF
V      1300 CONTINUE
V      DO 1301 J = 5 , JM
V      IF( IFVP(J).GE.1 .AND. IFVP(J-1).GE.1
+      .AND. IFVP(J-2).GE.1 .AND. IFVP(J-3).GE.1 ) EF(J) = 0.0
V      1301 CONTINUE
C

```

図 4.6.6(b) F-6 変更後のプログラム

```

COMMON / VARS/ Q(145,55,4) , R(145,55,4) , P(145,55)
COMMON /LARGE/ X(145,55) , Y(145,55) , S(145,55,4)
COMMON /BTRID/ ABC(145,48) , D(145,4,4) ,
1 DU(145,4,4) , F(145,4)
1 DIMENSION RR(1) , R1(1) , R2(1) , U(1) , V(1) ,
1 C1(1) , C2(1) , GR1(1) , GR2(1) ,
2 UR1(145) , VR2(145) , QS(145) , CO(145) , GQS(145)
EQUIVALENCE (D(1,1,1),RR(1)) , (D(1,1,2),R1(1)) ,
1 (D(1,1,3),R2(1)) ,
2 (F(1,1),U(1)) , (F(1,2),V(1)) ,
3 (F(1,3),C1(1)) , (F(1,4),C2(1)) ,
4 (D(1,2,4),GR1(1)) , (D(1,3,4),GR2(1))
-----
C
C IF(MODE) 10 , 10 , 20
C
10 CONTINUE
K = LL
M DO 11 IDX = 11 , 12
S R1(IDX) = R(IDX,K,1) * HD
S R2(IDX) = R(IDX,K,2) * HD
V RR(IDX) = 1.0 / Q(IDX,K,1)
V U(IDX) = Q(IDX,K,2) * RR(IDX)
V V(IDX) = Q(IDX,K,3) * RR(IDX)
M C2(IDX) = Q(IDX,K,4) * RR(IDX) * GAMMA
V 11 CONTINUE
GO TO 30
C
20 CONTINUE
J = LL
M DO 21 IDX = 11 , 12
S R1(IDX) = R(J,IDX,3) * HD
S R2(IDX) = R(J,IDX,4) * HD
V RR(IDX) = 1.0 / Q(J,IDX,1)
V U(IDX) = Q(J,IDX,2) * RR(IDX)
V V(IDX) = Q(J,IDX,3) * RR(IDX)
M C2(IDX) = Q(J,IDX,4) * RR(IDX) * GAMMA
V 21 CONTINUE
C
30 CONTINUE
GAMH = 0.5 * GAM1
GAM3 = 3.0 - GAMMA
C
M DO 31 IDX = 11 , 12
V C1(IDX) = GAMH * (U(IDX) ** 2 + V(IDX) ** 2)
S D(IDX,1,1) = 0.0
S D(IDX,1,4) = 0.0
S UR1(IDX) = U(IDX) * R1(IDX)
S VR2(IDX) = V(IDX) * R2(IDX)
S QS(IDX) = UR1(IDX) + VR2(IDX)
S GR1(IDX) = GAMH * R1(IDX)
S GR2(IDX) = GAMH * R2(IDX)
S D(IDX,2,1) = C1(IDX) * R1(IDX) - U(IDX) * QS(IDX)
S D(IDX,2,2) = GAM3 * UR1(IDX) + VR2(IDX)
S D(IDX,2,3) = -GR1(IDX) * V(IDX) + U(IDX) * R2(IDX)
S D(IDX,3,1) = C1(IDX) * R2(IDX) - V(IDX) * QS(IDX)
S D(IDX,3,2) = V(IDX) * R1(IDX) - GR2(IDX) * U(IDX)
S D(IDX,3,3) = UR1(IDX) + GAM3 * VR2(IDX)
V CO(IDX) = C2(IDX) - C1(IDX)
S GQS(IDX) = GAMH * QS(IDX)
M D(IDX,4,1) = (C1(IDX) - CO(IDX)) * QS(IDX)
S D(IDX,4,2) = CO(IDX) * R1(IDX) - U(IDX) * GQS(IDX)
S D(IDX,4,3) = CO(IDX) * R2(IDX) - V(IDX) * GQS(IDX)
S D(IDX,4,4) = GAMMA * QS(IDX)
V 31 CONTINUE
C
RETURN
END

```

図 4.6.7(a) F-7 変更前のプログラム

```

      GAMH      = 0.5 * GAM1
      GAM3      = 3.0 * GAMMA
C
      IF(MODE) 10 , 10 , 20
C
10  CONTINUE
C
      K = LL
C
      DO 11 IDX = 11 , 12
C
      D(IDX,1,2) = R(IDX,K,1) * HD
      D(IDX,1,3) = R(IDX,K,2) * HD
      DIV       = 1.0 / Q(IDX,K,1)
      F(IDX,1)  = Q(IDX,K,2) * DIV
      F(IDX,2)  = Q(IDX,K,3) * DIV
      F(IDX,4)  = Q(IDX,K,4) * DIV * GAMMA
C
      F(IDX,3)  = GAMH * (F(IDX,1)*F(IDX,1) + F(IDX,2)*F(IDX,2))
      D(IDX,1,1) = 0.0
      D(IDX,1,4) = 0.0
      UR1       = F(IDX,1) * D(IDX,1,2)
      VR2       = F(IDX,2) * D(IDX,1,3)
      QS        = UR1 + VR2
      D(IDX,2,4) = GAM1 * D(IDX,1,2)
      D(IDX,3,4) = GAM1 * D(IDX,1,3)
      D(IDX,2,1) = F(IDX,3) * D(IDX,1,2) - F(IDX,1) * QS
      D(IDX,2,2) = GAM3 * UR1 + VR2
      D(IDX,2,3) = -D(IDX,2,4) * F(IDX,2) + F(IDX,1) * D(IDX,1,3)
      D(IDX,3,1) = F(IDX,3) * D(IDX,1,3) - F(IDX,2) * QS
      D(IDX,3,2) = F(IDX,2) * D(IDX,1,2) - D(IDX,3,4) * F(IDX,1)
      D(IDX,3,3) = UR1 + GAM3 * VR2
      CO        = F(IDX,4) - F(IDX,3)
      GOS       = GAM1 * QS
      D(IDX,4,1) = (F(IDX,3) - CO) * QS
      D(IDX,4,2) = CO * D(IDX,1,2) - F(IDX,1) * GOS
      D(IDX,4,3) = CO * D(IDX,1,3) - F(IDX,2) * GOS
      D(IDX,4,4) = GAMMA * QS
C
11  CONTINUE
C
      GO TO 30
C
20  CONTINUE
      J = LL
      DO 21 IDX = 11 , 12
C
      D(IDX,1,2) = R(J,IDX,3) * HD
      D(IDX,1,3) = R(J,IDX,4) * HD
      DIV       = 1.0 / Q(J,IDX,1)
      F(IDX,1)  = Q(J,IDX,2) * DIV
      F(IDX,2)  = Q(J,IDX,3) * DIV
      F(IDX,4)  = Q(J,IDX,4) * DIV * GAMMA
C
      F(IDX,3)  = GAMH * (F(IDX,1)*F(IDX,1) + F(IDX,2)*F(IDX,2))
      D(IDX,1,1) = 0.0
      D(IDX,1,4) = 0.0
      UR1       = F(IDX,1) * D(IDX,1,2)
      VR2       = F(IDX,2) * D(IDX,1,3)
      QS        = UR1 + VR2
      D(IDX,2,4) = GAM1 * D(IDX,1,2)
      D(IDX,3,4) = GAM1 * D(IDX,1,3)
      D(IDX,2,1) = F(IDX,3) * D(IDX,1,2) - F(IDX,1) * QS
      D(IDX,2,2) = GAM3 * UR1 + VR2
      D(IDX,2,3) = -D(IDX,2,4) * F(IDX,2) + F(IDX,1) * D(IDX,1,3)
      D(IDX,3,1) = F(IDX,3) * D(IDX,1,3) - F(IDX,2) * QS
      D(IDX,3,2) = F(IDX,2) * D(IDX,1,2) - D(IDX,3,4) * F(IDX,1)
      D(IDX,3,3) = UR1 + GAM3 * VR2
      CO        = F(IDX,4) - F(IDX,3)
      GOS       = GAM1 * QS
      D(IDX,4,1) = (F(IDX,3) - CO) * QS
      D(IDX,4,2) = CO * D(IDX,1,2) - F(IDX,1) * GOS
      D(IDX,4,3) = CO * D(IDX,1,3) - F(IDX,2) * GOS
      D(IDX,4,4) = GAMMA * QS
C
21  CONTINUE
C
30  CONTINUE
      RETURN
      END

```

図 4.6.7(b) F-7 変更後のプログラム

```

S      DO 1000 K = 1 , KMAX
S      DO 1000 J = 1 , JMAX
V      DO 1000 M = 1 , 4
V      S(J,K,M) = F00
V      Q(J,K,M) = Q(J,K,M) / P(J,K)
V 1000 CONTINUE

```

図 4. 6. 8(a) F-8 変更前のプログラム

```

S      DO 1000 K = 1 , KMAX
S      DO 1000 M = 1 , 4
V      DO 1000 J = 1 , JMAX
V      S(J,K,M) = F00
V      Q(J,K,M) = Q(J,K,M) / P(J,K)
V 1000 CONTINUE

```

図 4. 6. 8(b) F-8 変更後のプログラム

```

S      DO 62 IDS = 1 , 2
S      DO 61 K = 1 , KM
S      ROH1 = Q(J1,K,1) * P(J1,K)
S      ROH2 = Q(J2,K,1) * P(J2,K)
S      Q(J,K,1) = ROH1 + ROH1 - ROH2
S      RR = 1.0 / Q(J1,K,1)
S      U1 = Q(J1,K,2) * RR
S      V1 = Q(J1,K,3) * RR
S      RR = 1.0 / Q(J2,K,1)
S      U2 = Q(J2,K,2) * RR
S      V2 = Q(J2,K,3) * RR
S      U0 = U1 + U1 - U2
S      V0 = V1 + V1 - V2
S      Q(J,K,2) = Q(J,K,1) * U0
S      Q(J,K,3) = Q(J,K,1) * V0
S      Q(J,K,4) = EW + 0.5 * (Q(J,K,2) * U0 + Q(J,K,3) * V0)
C
C      SCALE ALL VARIABLES
C
S      ZZZ = 1.0 / P(J,K)
V      DO 61 N = 1 , 4
V      Q(J,K,N) = Q(J,K,N) * ZZZ
V 61 CONTINUE
J      = JMAX
J1     = JM
J2     = JMAX - 2
62 CONTINUE

```

図 4. 6. 9(a) F-9 変更前のプログラム

```

V      DO 62 IDS = 1 , 2
V      DO 61 K = 1 , KM
V      ROH1 = Q(J1,K,1) * P(J1,K)
V      ROH2 = Q(J2,K,1) * P(J2,K)
V      Q(J,K,1) = ROH1 + ROH1 - ROH2
V      RR = 1.0 / Q(J1,K,1)
V      U1 = Q(J1,K,2) * RR
V      V1 = Q(J1,K,3) * RR
V      RR = 1.0 / Q(J2,K,1)
V      U2 = Q(J2,K,2) * RR
V      V2 = Q(J2,K,3) * RR
V      U0 = U1 + U1 - U2
V      V0 = V1 + V1 - V2
V      Q(J,K,2) = Q(J,K,1) * U0
V      Q(J,K,3) = Q(J,K,1) * V0
V      Q(J,K,4) = EW + 0.5 * (Q(J,K,2) * U0 + Q(J,K,3) * V0)
C
C      SCALE ALL VARIABLES
C
V      ZZZ = 1.0 / P(J,K)
C
CX     DO 61 N = 1 , 4
CX     Q(J,K,N) = Q(J,K,N) * ZZZ
V      Q(J,K,1) = Q(J,K,1) * ZZZ
V      Q(J,K,2) = Q(J,K,2) * ZZZ
V      Q(J,K,3) = Q(J,K,3) * ZZZ
V      Q(J,K,4) = Q(J,K,4) * ZZZ
V 61 CONTINUE
J      = JMAX
J1     = JM
J2     = JMAX - 2
62 CONTINUE

```

図 4. 6. 9(b) F-9 変更後のプログラム

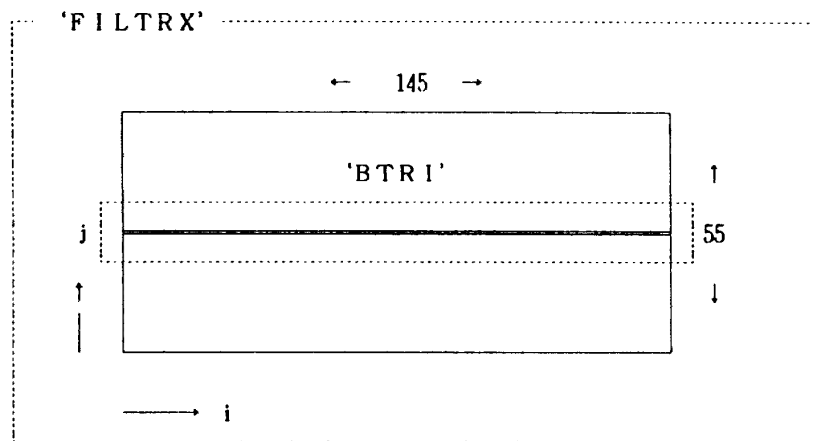


図 4. 6. 10 F-10 方程式解法の計算構造

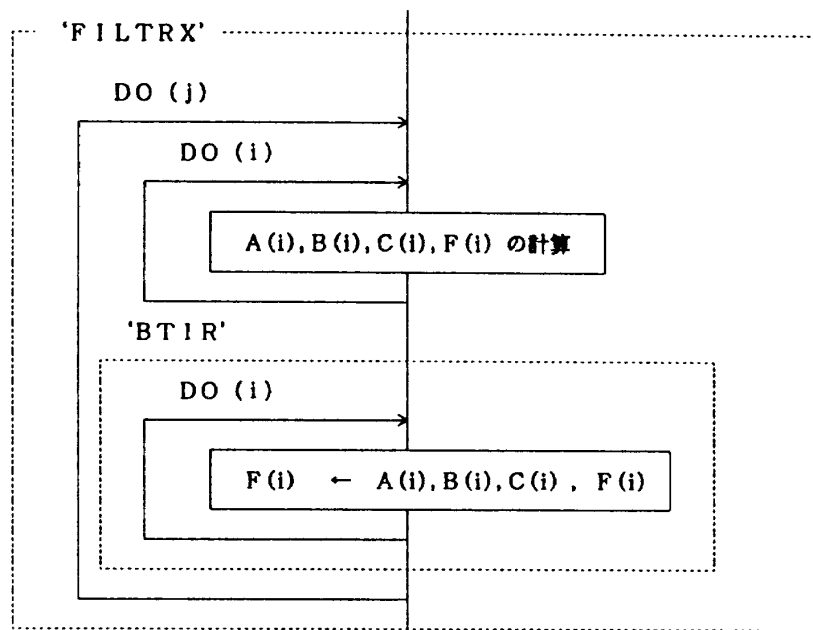


図 4. 6. 11(a) F-10 改造前の処理フロー

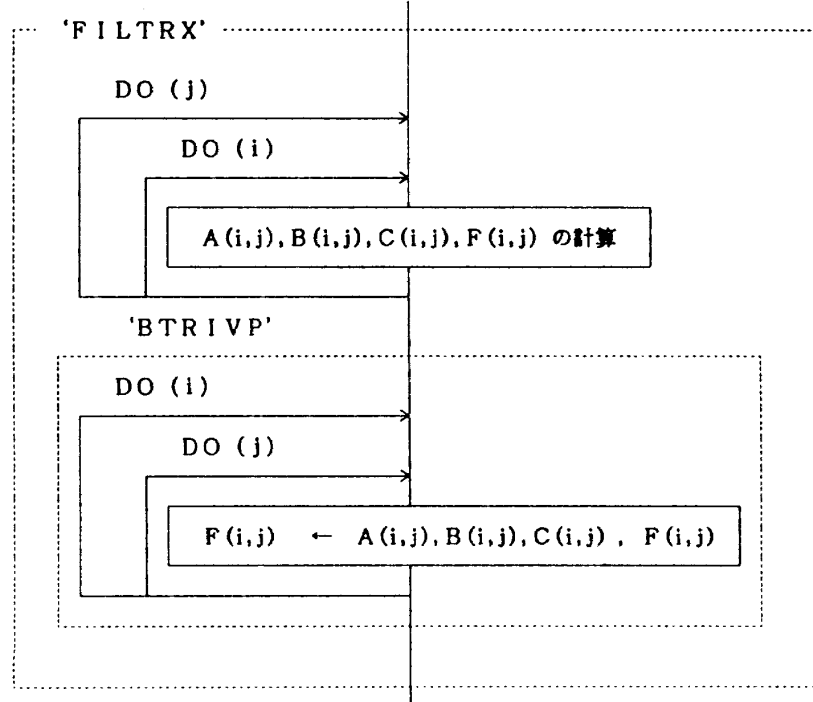


図 4. 6. 11(b) F-10 改造後の処理フロー

```

S          DO 3200 J = 2, JTLR1
C
S          JJ = JXX( J )
C
S          Q(JJ,K,N) = Q(JJ,K,N) + FF(J,N,K+KM)
S          S(JJ,K,N) = FF(J,N,K+KM)
S          Q(J,K,N) = Q(J,K,N) + FF(J,N,I)
S          S(J,K,N) = FF(J,N,I)
C
S 3200     CONTINUE

```

図 4. 6. 12(a) F-11 変更前のプログラム

```

*VOCL LOOP, NOVREC
V          DO 3200 J = 2, JTLR1
C
V          JJ = JXX( J )
C
V          Q(JJ,K,N) = Q(JJ,K,N) + FF(J,N,K+KM)
V          S(JJ,K,N) = FF(J,N,K+KM)
V          Q(J,K,N) = Q(J,K,N) + FF(J,N,I)
V          S(J,K,N) = FF(J,N,I)
C
V 3200     CONTINUE

```

図 4. 6. 12(b) F-11 変更後のプログラム

```

-----
SUBROUTINE CALUV
C..... VEC TUNNING V01
C
C  CALCULATE U, V FROM Q
C
COMMON / VARS / Q(145,55,4), R(145,55,4), P(145,55)
COMMON /UVSTCK/ VP1U(145,55), VP1V(145,55),
+               VP1W(145,55), VP1S(145,55)
C
C
V          DO 1000 K = 1, 55
C
V          DO 1000 J = 1, 145
C
V          IF( Q(J,K,1) .NE. 0.0 ) THEN
C
V              DIV = 1.0 / Q(J,K,1)
V              VP1U( J, K ) = Q(J,K,2) * DIV
V              VP1V( J, K ) = Q(J,K,3) * DIV
V              VP1W( J, K ) = Q(J,K,4) * DIV
V              VP1S( J, K ) = 0.5 * ( VP1U(J,K) * VP1U(J,K)
+                                     + VP1V(J,K) * VP1V(J,K) )
C
V          ENDIF
C
V 1000     CONTINUE
          RETURN
          END
-----

```

図 4. 6. 13 F-12 新サブルーチン CALUV

```

C
V DO 11 IDX = 11, 12
C
V D(IDX,1,2) = R(IDX,K,1) * HD
V D(IDX,1,3) = R(IDX,K,2) * HD
V DIV = 1.0 / Q(IDX,K,1)
V F(IDX,1) = Q(IDX,K,2) * DIV
V F(IDX,2) = Q(IDX,K,3) * DIV
V F(IDX,4) = Q(IDX,K,4) * DIV * GAMMA
C
V F(IDX,3) = GAMH * (F(IDX,1)*F(IDX,1) + F(IDX,2)*F(IDX,2))
V D(IDX,1,1) = 0.0
V D(IDX,1,4) = 0.0
V UR1 = F(IDX,1) * D(IDX,1,2)
V VR2 = F(IDX,2) * D(IDX,1,3)

```

図 4.6.14(a) F-12 変更前のプログラム

```

C
V DO 11 IDX = 11, 12
C
V D(IDX,1,2) = R(IDX,K,1) * HD
V D(IDX,1,3) = R(IDX,K,2) * HD
V F(IDX,1) = VP1U(IDX,K)
V F(IDX,2) = VP1V(IDX,K)
V F(IDX,4) = VP1W(IDX,K) * GAMMA
C
V F(IDX,3) = GAMH * VP1S(IDX,K)
V D(IDX,1,1) = 0.0
V D(IDX,1,4) = 0.0
V UR1 = F(IDX,1) * D(IDX,1,2)
V VR2 = F(IDX,2) * D(IDX,1,3)

```

図 4.6.14(b) F-12 変更後のプログラム

```

C
31 N = KTE2 + 1
M DO 32 K = N, K2
V ZZ = ( Z(K) - Z(KTE2) )/B
V A = EXP( -ZZ )
M XC(K) = XC(KTE2) + S*ZZ + ( S2-S )*( 1.-A )
M YC(K) = YC(KTE2) + T*ZZ + ( T2-T )*( 1.-A )
V XZ(K) = ( S + ( S2-S )*A )/B
V YZ(K) = ( T + ( T2-T )*A )/B
V XZZ(K) = -( S2-S )*A/( B*B )
V YZZ(K) = -( T2-T )*A/( B*B )
V 32 CONTINUE
V RETURN
V END

```

図 4.7.1(a) G-2 変更前のプログラム

```

CV
*VOCL LOOP,KTE2,LT,N,KTE2,LT,K2
V DO 32 K = N, K2
V ZZ = ( Z(K) - Z(KTE2) )/B
V A = EXP( -ZZ )
V XC(K) = XC(KTE2) + S*ZZ + ( S2-S )*( 1.-A )
V YC(K) = YC(KTE2) + T*ZZ + ( T2-T )*( 1.-A )
V XZ(K) = ( S + ( S2-S )*A )/B
V YZ(K) = ( T + ( T2-T )*A )/B
V XZZ(K) = -( S2-S )*A/( B*B )
V YZZ(K) = -( T2-T )*A/( B*B )
V 32 CONTINUE
V RETURN
V END

```

図 4.7.1(b) G-2 変更後のプログラム

```

SUBROUTINE CPLOT( I1,I2,FMACH,X,Y,CP )
CHARACTER*4 KODE,LINE
DIMENSION KODE(2),LINE(100),X(1),Y(1),CP(1)
DATA KODE/1H ,1H+/
WRITE(6,2)
2 FORMAT(50HOPLOT OF CP AT EQUAL INTERVALS IN THE MAPPED PLANE /
1 10H0 X ,10H Y ,10H CP )
CP0 = ( ( 1.0+0.2*FMACH**2 )**3.5 - 1.0 )/( 0.7*FMACH**2 )
DO 12 I=1,100
LINE(I) = KODE(1)
12 CONTINUE
C
DO 22 I=I1,I2
K = 30.0*( CP0 - CP(I) ) + 4.5
K = MIN0( 100 , K )
LINE(K) = KODE(2)
WRITE(6,610) X(I),Y(I),CP(I),LINE
LINE(K) = KODE(1)
22 CONTINUE
C
610 FORMAT(3F10.4,100A1)
RETURN
END

```

図 4. 7. 2(a) G-3 変更前のプログラム

```

SUBROUTINE CPLOT( I1,I2,FMACH,X,Y,CP )
CHARACTER*4 KODE,LINE
DIMENSION KODE(2),LINE(100),X(1),Y(1),CP(1)
CV
DIMENSION K(193)
DATA KODE/1H ,1H+/
WRITE(6,2)
2 FORMAT(50HOPLOT OF CP AT EQUAL INTERVALS IN THE MAPPED PLANE /
1 10H0 X ,10H Y ,10H CP )
CP0 = ( ( 1.0+0.2*FMACH**2 )**3.5 - 1.0 )/( 0.7*FMACH**2 )
DO 12 I=1,100
LINE(I) = KODE(1)
12 CONTINUE
C
CV
DO 22 I=I1,I2
K(I) = 30.0*( CP0 - CP(I) ) + 4.5
K(I) = MIN0( 100 , K(I) )
22 CONTINUE
DO 32 I=I1,I2
LINE(K(I)) = KODE(2)
WRITE(6,610) X(I),Y(I),CP(I),LINE
LINE(K(I)) = KODE(1)
32 CONTINUE
CV
C
610 FORMAT(3F10.4,100A1)
RETURN
END

```

図 4. 7. 2(b) G-3 変更後のプログラム

```

DO 100 I=1,N
:
CALL SUB1(A,B,ICON)
CALL SUB2(ICON)
:
100 CONTINUE

```

```

SUB2(ICON)
:

```

```

IF(ICON.NE.0) THEN
:

```

```

ENDIF
:

```

```

RETURN

```

```

END

```

```

DO 100 I=1,N
:

```

```

CALL SUB1(A,B,ICON)

```

```

IF(ICON.NE.0) CALL SUB2(ICON)
:

```

```

100 CONTINUE

```

図 4. 8. 1(a) H-1 変更前のプログラム

図 4. 8. 1(b) H-1 変更後のプログラム

```

IF( (( XX .LT. PTB ).OR.( RDT .LT. 0.0 )) .AND.
1  (( XX .GT. PTA ).OR.( RDL .LT. 0.0 )) ) THEN
  XX=AMAX1(XX,PTA)
  XX=AMIN1(XX,PTB)
  PFT = ASIN((2.0*XX-PTA-PTB)/(PTB-PTA))
  :
ENDIF

```

図 4. 8. 2(a) H-3 変更前のプログラム

```

PTAPT B =PTA+PTB      } 変数定義
PTBPTA =PTB-PTA
IF( (( XX .LT. PTB ).OR.( RDT .LT. 0.0 )) .AND.
1  (( XX .GT. PTA ).OR.( RDL .LT. 0.0 )) ) THEN
  XX=AMAX1(XX,PTA)
  XX=AMIN1(XX,PTB)
  PFT = ASIN((2.0*XX-PTA-PTB)/(PTB-PTA))
  PPT = ASIN((2.0*XX-PTAPT B)/(PTBPTA))
  :
ENDIF

```

図 4. 8. 2(b) H-3 変更後のプログラム

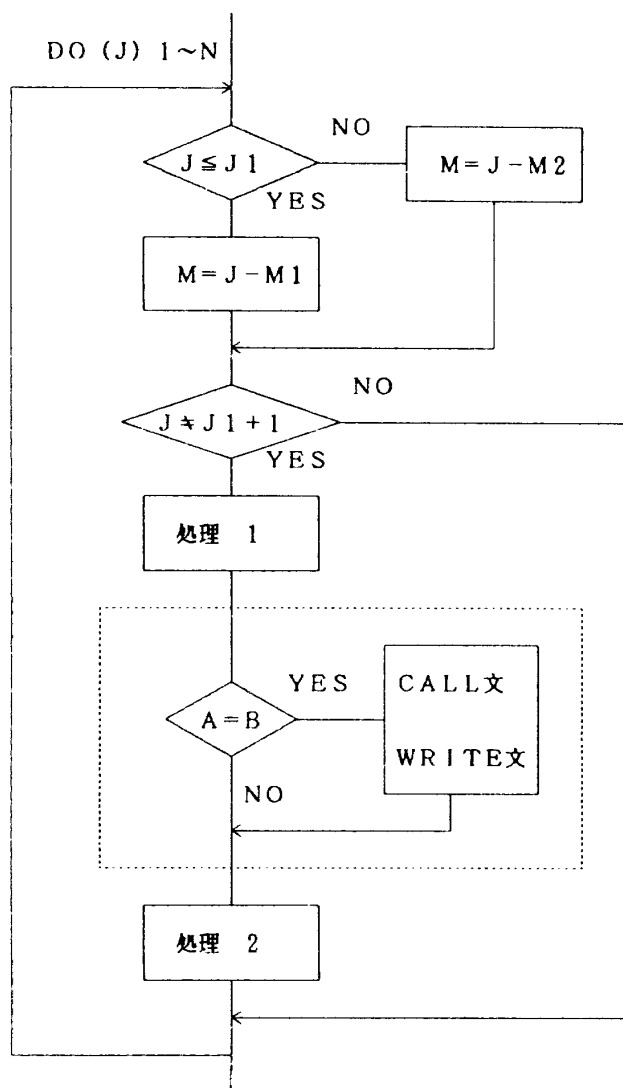


図 4. 9. 1(a) I-1 改造前の DO ループ構造

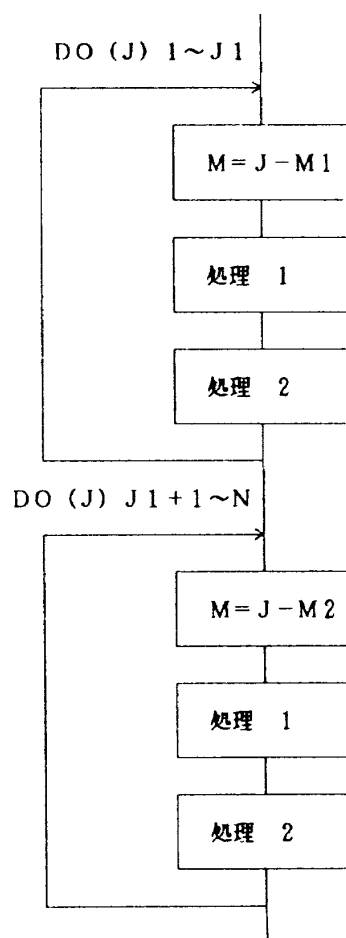


図 4. 9. 1(b) I-1 改造後の DO ループ構造

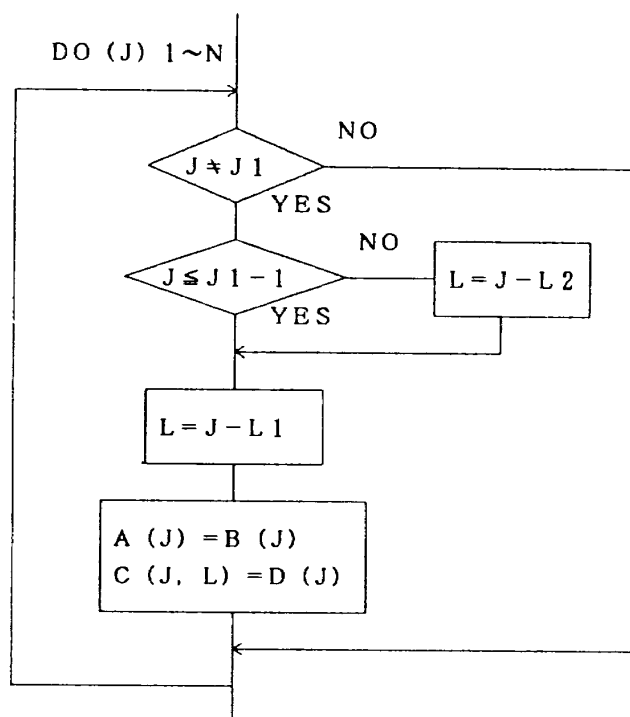


図 4.9.2 I-2 改造前の DO ループ構造

```

M      DO 74 J=1,JE
V      IF( J .LE. KY ) THEN
V      M=J
V      L=I
V      ELSE
V      M=2*KY-J
V      L=KX-I+1
V      ENDIF
V      COUNTF=COUNTF+1.0
V      SCF=SCF+ABS(CF(J))
S      IF( ABS(CF(J)) .GT. ABS(CFMAX) ) THEN
S      CFMAX=CF(J)
V      IFMAX=L
V      JFMAX=M
V      KFMAX=K
V      ENDIF
C**
M      F(L,M,K)=F(L,M,K)+CF(J)
C**
V      COUNTD=COUNTD+1.0
V      SD2=SD2+ABS(CS(J))
S      IF( ABS(CS(J)) .GT. ABS(D2MAX) ) THEN
S      D2MAX=CS(J)
V      IDMAX=L
V      JDMAX=M
V      KDMAX=K
V      ENDIF
V      74 CONTINUE
  
```

図 4.9.3(a) I-3 改造前のプログラム

```

V      DO 74 J=1,JE
V      IF( J .LE. KY ) THEN
V      M=J
V      L=I
V      ELSE
V      M=2*KY-J
V      L=KX-I+1
V      ENDIF
V      COUNTF=COUNTF+1.0
V      SCF=SCF+ABS(CF(J))
C-----
V      CFABS(J)=ABS(CF(J))
C-----
V      COUNTD=COUNTD+1.0
V      SD2=SD2+ABS(CS(J))
C-----
V      CSABS(J)=ABS(CS(J))
C-----
V      74 CONTINUE
C-----
V      JEKY=MIN(JE,KY)
V      L=I
V      DO 1074 J=1,JEKY
V      M=J
V      F(L,M,K)=F(L,M,K)+CF(J)
V 1074 CONTINUE
V      IF(JE.LE.KY) GOTO 1200
V      L=KX-I+1
V      DO 1174 J=KY+1,JE
V      M=2*KY-J
V      F(L,M,K)=F(L,M,K)+CF(J)
V 1174 CONTINUE
1200 CONTINUE
C
CFMAXP= 0.
CSMAXP= 0.
  
```

図 4.9.3(b)-1 I-3 変更後のプログラム

```

C
V      DO 1274 J = 1, JE
V      IF( CFABS(J) .GT. CFMAXP) THEN
V      CFMAXP = CFABS(J)
V      JMAXCF = J
V      ENDIF
V      IF( CSABS(J) .GT. CFMAXP) THEN
V      CSMAXP = CSABS(J)
V      JMAXCS = J
V      ENDIF
V 1274 CONTINUE
C
      IF(ABS(CFMAX).LT.CFMAXP) THEN
      CFMAX=CF(JMAXCF)
      IF( JMAXCF .LE. KY ) THEN
      M=JMAXCF
      L=I
      ELSE
      M=2*KY-JMAXCF
      L=KX-I+1
      ENDIF
      IFMAX=L
      JFMAX=M
      KFMAX=K
      ENDIF
      CFMAX,
      IFMAX, JFMAX
      KFMAX のセット部
C
      IF(ABS(D2MAX).LT.CSMAXP) THEN
      D2MAX=CS(JMAXCS)
      IF( JMAXCS .LE. KY ) THEN
      M=JMAXCS
      L=I
      ELSE
      M=2*KY-JMAXCS
      L=KX-I+1
      ENDIF
      IDMAX=L
      JDMAX=M
      KDMAX=K
      ENDIF
      D2MAX,
      IDMAX, JDMAX
      KDMAX のセット部

```

図 4.9.3(b)-2 1-3 変更後のプログラム

```

M      DO 80 J=1,JTPP
V      Q2D( J)=Q2U( J)
V      RHD( J)=RHU( J)
V      RHUD( J)=RHUU( J)
V      RHVD( J)=RHVU( J)
V      RHWD( J)=RHWU( J)
V      AXD( J)=AXU( J)
V      AYD( J)=AYU( J)
V      AZD( J)=AZU( J)
V      QXYD( J)=QXYU( J)
V      QYZD( J)=QYZU( J)
V      QZXD( J)=QZXU( J)
V      QXYZD(J)=QXYZU(J)
V      IF((( IDR .EQ. 0 ).AND.( I .EQ. IC-1 )).OR.
1      ( I .EQ. IC
S      PD(J)=0.0
V      ELSEIF( I .EQ. IT2 ) THEN
M      PD(J+KY+2)=P(J)
V      ELSEIF(( IDR .NE. 0 ).OR.( I .NE. IT1-1 )) THEN
M      PD(J)=P(J)
V      ENDIF
V      80 CONTINUE

```

図 4.9.4 (a) I-4 変更前のプログラム

```

V      DO 80 J=1,JTPP
V      Q2D( J)=Q2U( J)
V      RHD( J)=RHU( J)
V      RHUD( J)=RHUU( J)
V      RHVD( J)=RHVU( J)
V      RHWD( J)=RHWU( J)
V      AXD( J)=AXU( J)
V      AYD( J)=AYU( J)
V      AZD( J)=AZU( J)
V      QXYD( J)=QXYU( J)
V      QYZD( J)=QYZU( J)
V      QZXD( J)=QZXU( J)
V      QXYZD(J)=QXYZU(J)
V      80 CONTINUE
      IF((( IDR .EQ. 0 ).AND.( I .EQ. IC-1 )).OR.
1      ( I .EQ. IC
V      DO 1080 J=1,JTPP
V      PD(J)=0.0
V 1080      CONTINUE
      ELSEIF( I .EQ. IT2 ) THEN
V      DO 1180 J=1,JTPP
V      PD(J+KY+2)=P(J)
V 1180      CONTINUE
      ELSEIF(( IDR .NE. 0 ).OR.( I .NE. IT1-1 )) THEN
V      DO 1280 J=1,JTPP
V      PD(J)=P(J)
V 1280      CONTINUE
      ENDIF

```

PD (j) のセット部

図 4.9.4 (b) I-4 変更後のプログラム


```

DO 700 N = 1, NCS2
  AA = SQRT( ( RE(1,1,N)-RE(2,1,N) )**2
  .      + ( RE(1,2,N)-RE(2,2,N) )**2
  .      + ( RE(1,3,N)-RE(2,3,N) )**2 )
  BB = SQRT( ( RE(1,1,N)-RE(3,1,N) )**2
  .      + ( RE(1,2,N)-RE(3,2,N) )**2
  .      + ( RE(1,3,N)-RE(3,3,N) )**2 )
  CC = SQRT( ( RE(2,1,N)-RE(4,1,N) )**2
  .      + ( RE(2,2,N)-RE(4,2,N) )**2
  .      + ( RE(2,3,N)-RE(4,3,N) )**2 )
  DD = SQRT( ( RE(3,1,N)-RE(4,1,N) )**2
  .      + ( RE(3,2,N)-RE(4,2,N) )**2
  .      + ( RE(3,3,N)-RE(4,3,N) )**2 )
  EE = SQRT( ( RE(2,1,N)-RE(3,1,N) )**2
  .      + ( RE(2,2,N)-RE(3,2,N) )**2
  .      + ( RE(2,3,N)-RE(3,3,N) )**2 )

  S1 = 0.5*( AA + BB + EE )
  S2 = 0.5*( CC + DD + EE )
  SS1 = SQRT( S1*( S1 - AA )*( S1 - BB )*( S1 - EE ) )
  SS2 = SQRT( S2*( S2 - CC )*( S2 - DD )*( S2 - EE ) )
  SS = SS1 + SS2
  CL( IRI ) = CL( IRI ) + CP(N)*EN(3,N)*SS
  CS( IRI ) = CS( IRI ) + CP(N)*EN(2,N)*SS
  CD( IRI ) = CD( IRI ) + CP(N)*EN(1,N)*SS
  CX( IRI ) = CX( IRI ) + CP(N)*
  .      ( EN(1,N)*TRRR(1) + EN(2,N)*TRRR(2) + EN(3,N)*TRRR(3) )
CCCC .      *SS
  CY( IRI ) = CY( IRI ) + CP(N)*
  .      ( EN(1,N)*TRPP(1) + EN(2,N)*TRPP(2) + EN(3,N)*TRPP(3) )
CCCC .      *SS
  CZ( IRI ) = CZ( IRI ) + CP(N)*
  .      ( EN(1,N)*TRQQ(1) + EN(2,N)*TRQQ(2) + EN(3,N)*TRQQ(3) )
CCCC .      *SS
  CMP( IRI ) = CMP( IRI ) + CP(N)*
  .      ( ( ( REO(2,N)-RRR(2) )*EN(3,N) -
  .      ( REO(3,N)-RRR(3) )*EN(2,N) )*TRPP(1)
  .      + ( ( REO(3,N)-RRR(3) )*EN(1,N) -
  .      ( REO(1,N)-RRR(1) )*EN(3,N) )*TRPP(2)
  .      + ( ( REO(1,N)-RRR(1) )*EN(2,N) -
  .      ( REO(2,N)-RRR(2) )*EN(1,N) )*TRPP(3) )*SS
  CMR( IRI ) = CMR( IRI ) + CP(N)*
  .      ( ( ( REO(2,N)-RRR(2) )*EN(3,N) -
  .      ( REO(3,N)-RRR(3) )*EN(2,N) )*TRRR(1)
  .      + ( ( REO(3,N)-RRR(3) )*EN(1,N) -
  .      ( REO(1,N)-RRR(1) )*EN(3,N) )*TRRR(2)
  .      + ( ( REO(1,N)-RRR(1) )*EN(2,N) -
  .      ( REO(2,N)-RRR(2) )*EN(1,N) )*TRRR(3) )*SS
  CMQ( IRI ) = CMQ( IRI ) + CP(N)*
  .      ( ( ( REO(2,N)-RRR(2) )*EN(3,N) -
  .      ( REO(3,N)-RRR(3) )*EN(2,N) )*TRQQ(1)
  .      + ( ( REO(3,N)-RRR(3) )*EN(1,N) -
  .      ( REO(1,N)-RRR(1) )*EN(3,N) )*TRQQ(2)
  .      + ( ( REO(1,N)-RRR(1) )*EN(2,N) -
  .      ( REO(2,N)-RRR(2) )*EN(1,N) )*TRQQ(3) )*SS
  K = 1
  DO 710 WHILE ( IND(N).NE.CHAK )
    K = K + 1
710  CONTINUE

  CCL( IRI,K,J ) = CCL( IRI,K,J ) + CP(N)*EN(J,N)*SS
700  CONTINUE

```

図 4.10.1 (a) J-1 変更前のプログラム

```

V      DO 700 N = 1, NCS2
V      AA      = SORT( (RE(N,1,1)-RE(N,2,1))*(RE(N,1,1)-RE(N,2,1))
      .          + (RE(N,1,2)-RE(N,2,2))*(RE(N,1,2)-RE(N,2,2))
      .          + (RE(N,1,3)-RE(N,2,3))*(RE(N,1,3)-RE(N,2,3)) )
V      BB      = SORT( (RE(N,1,1)-RE(N,3,1))*(RE(N,1,1)-RE(N,3,1))
      .          + (RE(N,1,2)-RE(N,3,2))*(RE(N,1,2)-RE(N,3,2))
      .          + (RE(N,1,3)-RE(N,3,3))*(RE(N,1,3)-RE(N,3,3)) )
V      CC      = SORT( (RE(N,2,1)-RE(N,4,1))*(RE(N,2,1)-RE(N,4,1))
      .          + (RE(N,2,2)-RE(N,4,2))*(RE(N,2,2)-RE(N,4,2))
      .          + (RE(N,2,3)-RE(N,4,3))*(RE(N,2,3)-RE(N,4,3)) )
V      DD      = SORT( (RE(N,3,1)-RE(N,4,1))*(RE(N,3,1)-RE(N,4,1))
      .          + (RE(N,3,2)-RE(N,4,2))*(RE(N,3,2)-RE(N,4,2))
      .          + (RE(N,3,3)-RE(N,4,3))*(RE(N,3,3)-RE(N,4,3)) )
V      EE      = SORT( (RE(N,2,1)-RE(N,3,1))*(RE(N,2,1)-RE(N,3,1))
      .          + (RE(N,2,2)-RE(N,3,2))*(RE(N,2,2)-RE(N,3,2))
      .          + (RE(N,2,3)-RE(N,3,3))*(RE(N,2,3)-RE(N,3,3)) )

V      S1      = 0.5*( AA + BB + EE )
V      S2      = 0.5*( CC + DD + EE )
V      SS1     = SORT( S1*( S1 - AA )*( S1 - BB )*( S1 - EE ) )
V      SS2     = SORT( S2*( S2 - CC )*( S2 - DD )*( S2 - EE ) )
V      SS(N)   = SS1 + SS2
V      CL( IRI ) = CL( IRI ) + CP(N)*EN(N,3)*SS(N)
V      CS( IRI ) = CS( IRI ) + CP(N)*EN(N,2)*SS(N)
V      CD( IRI ) = CD( IRI ) + CP(N)*EN(N,1)*SS(N)
V      CX( IRI ) = CX( IRI ) + CP(N)*
      .          ( EN(N,1)*TRRR(1) + EN(N,2)*TRRR(2) + EN(N,3)*TRRR(3) )
V      CCCC    *SS
V      CY( IRI ) = CY( IRI ) + CP(N)*
      .          ( EN(N,1)*TRPP(1) + EN(N,2)*TRPP(2) + EN(N,3)*TRPP(3) )
V      CCCC    *SS
V      CZ( IRI ) = CZ( IRI ) + CP(N)*
      .          ( EN(N,1)*TR00(1) + EN(N,2)*TR00(2) + EN(N,3)*TR00(3) )
V      CCCC    *SS
V      CMP( IRI ) = CMP( IRI ) + CP(N)*
      .          ( ( (RE0(2,N)-RRR(2))*EN(N,3) -
      .          (RE0(3,N)-RRR(3))*EN(N,2))*TRPP(1)
      .          + ( (RE0(3,N)-RRR(3))*EN(N,1) -
      .          (RE0(1,N)-RRR(1))*EN(N,3))*TRPP(2)
      .          + ( (RE0(1,N)-RRR(1))*EN(N,2) -
      .          (RE0(2,N)-RRR(2))*EN(N,1))*TRPP(3) ) *SS(N)
V      CMR( IRI ) = CMR( IRI ) + CP(N)*
      .          ( ( (RE0(2,N)-RRR(2))*EN(N,3) -
      .          (RE0(3,N)-RRR(3))*EN(N,2))*TRRR(1)
      .          + ( (RE0(3,N)-RRR(3))*EN(N,1) -
      .          (RE0(1,N)-RRR(1))*EN(N,3))*TRRR(2)
      .          + ( (RE0(1,N)-RRR(1))*EN(N,2) -
      .          (RE0(2,N)-RRR(2))*EN(N,1))*TRRR(3) ) *SS(N)
V      CM0( IRI ) = CM0( IRI ) + CP(N)*
      .          ( ( (RE0(2,N)-RRR(2))*EN(N,3) -
      .          (RE0(3,N)-RRR(3))*EN(N,2))*TR00(1)
      .          + ( (RE0(3,N)-RRR(3))*EN(N,1) -
      .          (RE0(1,N)-RRR(1))*EN(N,3))*TR00(2)
      .          + ( (RE0(1,N)-RRR(1))*EN(N,2) -
      .          (RE0(2,N)-RRR(2))*EN(N,1))*TR00(3) ) *SS(N)

CCC      K = 1
C      DO 710 WHILE ( IND(N).NE.CHA(K))
C      K = K + 1
C 710 CONTINUE

C      CCL( IRI,K,J ) = CCL( IRI,K,J ) + CP(N)*EN(J,N)*SS(N)
V 700 CONTINUE
      DO 720 N=1,NCS2
      DO 710 K=1,50
      IF ( IND(N).EQ.CHA(K)) THEN
      CCL( IRI,K,1 ) = CCL( IRI,K,1 ) + CP(N)*EN(N,1)*SS(N)
      CCL( IRI,K,2 ) = CCL( IRI,K,2 ) + CP(N)*EN(N,2)*SS(N)
      CCL( IRI,K,3 ) = CCL( IRI,K,3 ) + CP(N)*EN(N,3)*SS(N)
      GOTO 720
      ENDIF
      710 CONTINUE
      720 CONTINUE

```

図 4.10.1 (b) J-1 変更後のプログラム

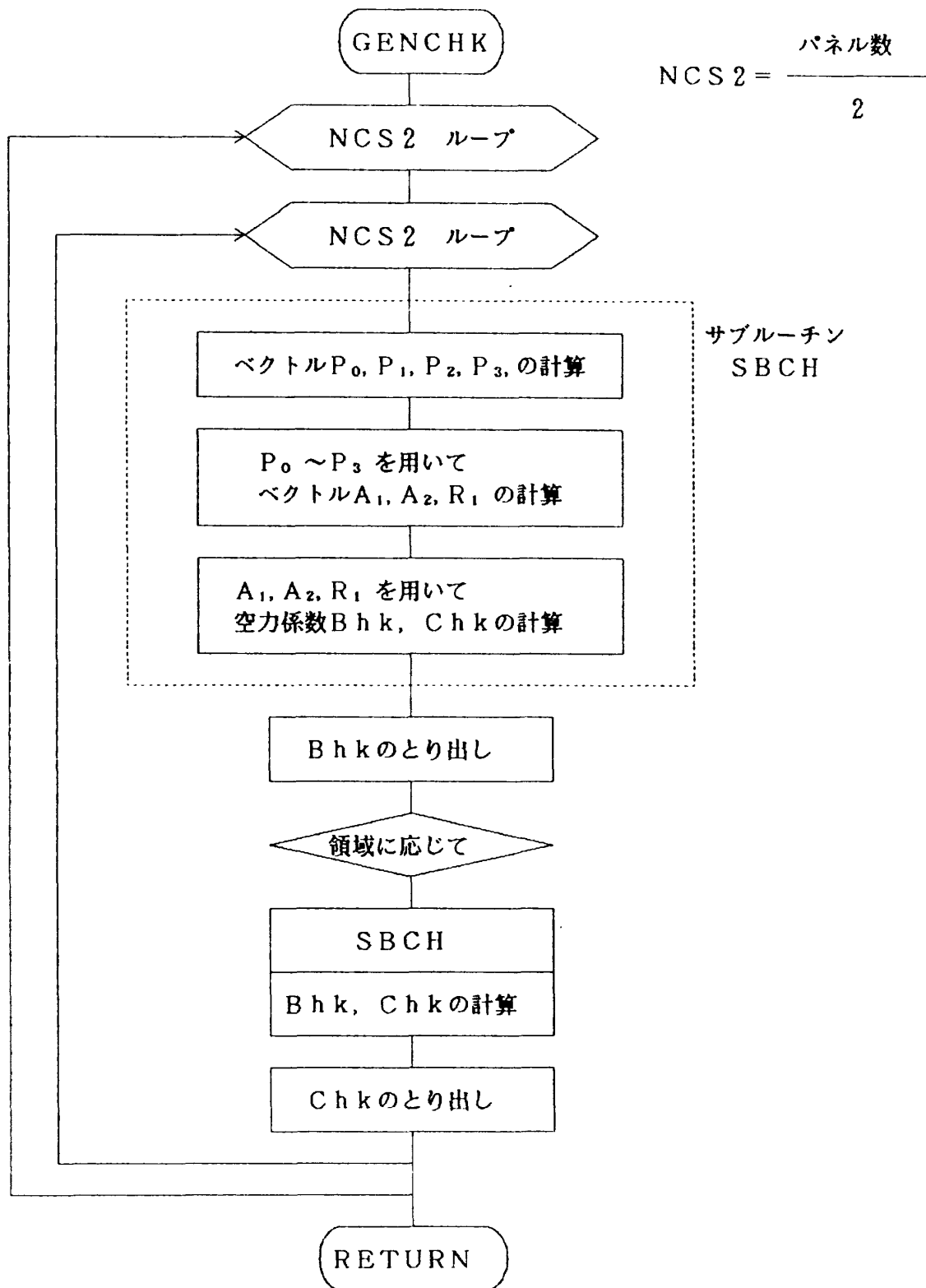


図 4. 10. 2 (a) J-3 改造前の処理フロー

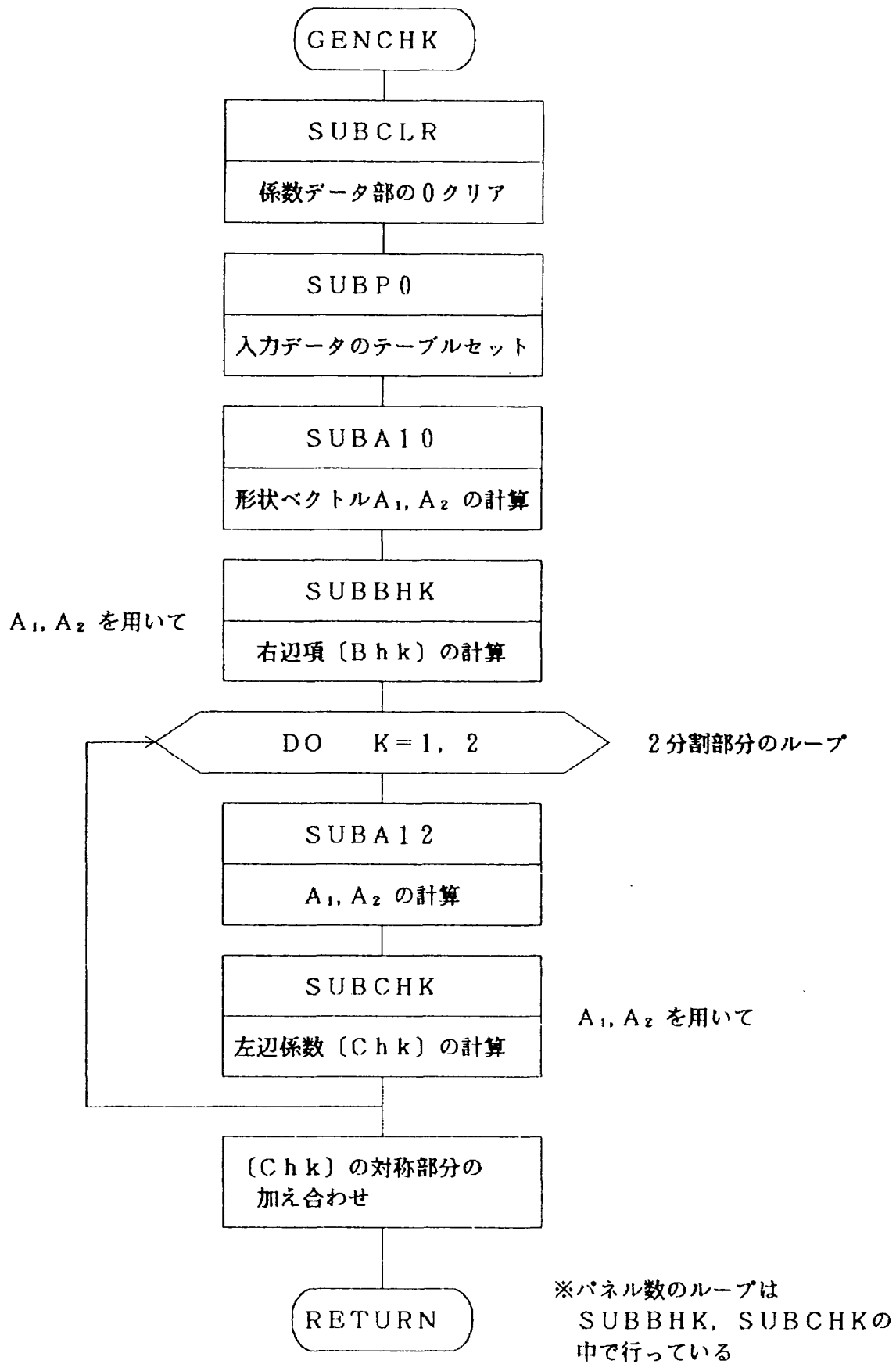


図 4.10.2 (b) J-3 改造後の処理フロー


```

C
C C
C XI EIGENVALUES
S
V DO 203 K = KLOW,KUP
V DO 204 J = 1, JMAX
V XDT = XT(J,K,L)
V RR = 1./Q(J,K,L,1)
V UU(J,K) = XDT + ( XX(J,K,L)*Q(J,K,L,2) +
* XY(J,K,L)*Q(J,K,L,3) + XZ(J,K,L)*Q(J,K,L,4) )*RR
V PP = GAMI*(Q(J,K,L,5) -
* .5*(Q(J,K,L,2)**2+Q(J,K,L,3)**2+Q(J,K,L,4)**2)*RR)
V CC(J,K) = SQRT( GAMMA*PP*RR *
* (XX(J,K,L)**2+XY(J,K,L)**2+XZ(J,K,L)**2) )
V 204 CONTINUE
S 203 CONTINUE
C

```

図 4. 12. 2 (a) L-2 変更前のプログラム

```

C
C XI EIGENVALUES
V
V DO 203 K = KLOW,KUP
V DO 204 J = 1, JDIM
V XDT = XT(J,K,L)
V RR = 1./Q(J,K,L,1)
V UU(J,K) = XDT + ( XX(J,K,L)*Q(J,K,L,2) +
* XY(J,K,L)*Q(J,K,L,3) + XZ(J,K,L)*Q(J,K,L,4) )*RR
V PP = GAMI*(Q(J,K,L,5) -
* .5*(Q(J,K,L,2)**2+Q(J,K,L,3)**2+Q(J,K,L,4)**2)*RR)
V CC(J,K) = SQRT( GAMMA*PP*RR *
* (XX(J,K,L)**2 + XY(J,K,L)**2 + XZ(J,K,L)**2) )
V 204 CONTINUE
V 203 CONTINUE

```

JDIMは配列宣言数である

図 4. 12. 2 (b) L-2 変更後のプログラム

| | | | | |
|-------|--|--|-------------|------------|
| C | | DO 210 J = 3, JM-1 | 297003 | 99003 |
| | | JP1 = J+1 | 559348003 | 15543003 |
| | | JM1 = J-1 | | |
| | | JP2 = J+2 | | |
| | | JM2 = J-2 | | |
| V | | DO 212 K = KLOW, KUP | | |
| C | | ARTIFICIAL DISSIPATION MODEL AT INTERIOR | | |
| V | | DJ = SPECT(J, K, L) | 2517966003 | 419661003 |
| V | | DJP = SPECT(JP1, K, L) | 419661003 | 419661003 |
| V | | DJM = SPECT(JM1, K, L) | 419661003 | 419661003 |
| V | | EPJM = DJ + DJM | 419661003 | 419661003 |
| V | | EPJ = DJP + DJ | 419661003 | 419661003 |
| V | | COEF2M = EPS2*COEF(JM1, K, L)*EPJM | 419661003 | 419661003 |
| V | | COEF4M = EPS4*EPJM | 419661003 | 419661003 |
| V | | COEF4M = COEF4M - AMIN1(COEF4M, COEF2M) | 1258980E023 | 419661003 |
| V | | COEF2 = EPS2*COEF(J, K, L)*EPJ | 1258980E023 | 419661003 |
| V | | COEF4 = EPS4*EPJ | 1678640E023 | 419661003 |
| | | | 1258980E023 | 419661003 |
| C---- | | STATEMENT (STPF3D) -----3-----4-----5-----6-----7-- | S-COST | EXECUTIONS |
| V | | COEF4 = COEF4 - AMIN1(COEF4, COEF2) | 1258980E023 | 419661003 |
| V | | AR(J, K) = Q(JM2, K, L, 6)*COEF4M | 1478640E023 | 419661003 |
| V | | BR(J, K) = -(COEF2M + 3.*COEF4M + COEF4)*Q(JM1, K, L, 6) | 8393219E013 | 419661003 |
| V | | CR(J, K) = Q(J, K, L, 6)*(COEF2M+3.*COEF4M+COEF2+3.*COEF4) | 2517969E023 | 419661003 |
| V | | DR(J, K) = -(COEF2 + 3.*COEF4 + COEF4M)*Q(JP1, K, L, 6) | 2098299E023 | 419661003 |
| V | | ER(J, K) = Q(JP2, K, L, 6)*COEF4 | 1478640E023 | 419661003 |
| C | | ADD IN JACOBIAN DIFFERENCING 4TH ORDER INTERIOR | | |
| V | | AR(J, K) = AR(J, K) + COM2*(UU(JM2, K)*SN*CC(JM2, K))*HD4 | 8393219E013 | 419661003 |
| V | | BR(J, K) = BR(J, K) + COM1*(UU(JM1, K)*SN*CC(JM1, K))*HD4 | 2517969E023 | 419661003 |
| V | | DR(J, K) = DR(J, K) + COP1*(UU(JP1, K)*SN*CC(JP1, K))*HD4 | 2517969E023 | 419661003 |
| V | | ER(J, K) = ER(J, K) + COP2*(UU(JP2, K)*SN*CC(JP2, K))*HD4 | 2517969E023 | 419661003 |
| C | | CONTINUE | | |
| S 210 | | CONTINUE | 2424708E023 | 419661003 |
| | | | 3118500E003 | 15543003 |

オリジナル DO210ループのループ長 $\frac{15543}{99} = 157$

オリジナル DO212ループのループ長 $\frac{419661}{15543} = 27$

図 4. 12. 3 (a) L-3 変更前のプログラム

| | | | | |
|-------|--|--|-------------|------------|
| C | | DO 212 K = KLOW, KUP | 198003 | 99003 |
| V | | DO 210 J = 3, JM-1 | 155034003 | 2673003 |
| | | JP1 = J+1 | 2937627003 | 419661003 |
| | | JM1 = J-1 | | |
| | | JP2 = J+2 | | |
| | | JM2 = J-2 | | |
| C | | ARTIFICIAL DISSIPATION MODEL AT INTERIOR | | |
| V | | DJ = SPECT(J, K, L) | 419661003 | 419661003 |
| V | | DJP = SPECT(JP1, K, L) | 839322003 | 419661003 |
| V | | DJM = SPECT(JM1, K, L) | 419661003 | 419661003 |
| V | | EPJM = DJ + DJM | 419661003 | 419661003 |
| V | | EPJ = DJP + DJ | 419661003 | 419661003 |
| V | | COEF2M = EPS2*COEF(JM1, K, L)*EPJM | 419661003 | 419661003 |
| V | | COEF4M = EPS4*EPJM | 419661003 | 419661003 |
| V | | COEF4M = COEF4M - AMIN1(COEF4M, COEF2M) | 1258980E023 | 419661003 |
| | | | 1258980E023 | 419661003 |
| C---- | | STATEMENT (STPF3D) -----3-----4-----5-----6-----7-- | S-COST | EXECUTIONS |
| V | | COEF2 = EPS2*COEF(J, K, L)*EPJ | 1678640E023 | 419661003 |
| V | | COEF4 = EPS4*EPJ | 1258980E023 | 419661003 |
| V | | COEF4 = COEF4 - AMIN1(COEF4, COEF2) | 1258980E023 | 419661003 |
| V | | AR(J, K) = Q(JM2, K, L, 6)*COEF4M | 1478640E023 | 419661003 |
| V | | BR(J, K) = -(COEF2M + 3.*COEF4M + COEF4)*Q(JM1, K, L, 6) | 8393219E013 | 419661003 |
| V | | CR(J, K) = Q(J, K, L, 6)*(COEF2M+3.*COEF4M+COEF2+3.*COEF4) | 2517969E023 | 419661003 |
| V | | DR(J, K) = -(COEF2 + 3.*COEF4 + COEF4M)*Q(JP1, K, L, 6) | 2098299E023 | 419661003 |
| V | | ER(J, K) = Q(JP2, K, L, 6)*COEF4 | 1478640E023 | 419661003 |
| C | | ADD IN JACOBIAN DIFFERENCING 4TH ORDER INTERIOR | | |
| V | | AR(J, K) = AR(J, K) + COM2*(UU(JM2, K)*SN*CC(JM2, K))*HD4 | 8393219E013 | 419661003 |
| V | | BR(J, K) = BR(J, K) + COM1*(UU(JM1, K)*SN*CC(JM1, K))*HD4 | 2098299E023 | 419661003 |
| V | | DR(J, K) = DR(J, K) + COP1*(UU(JP1, K)*SN*CC(JP1, K))*HD4 | 2517969E023 | 419661003 |
| V | | ER(J, K) = ER(J, K) + COP2*(UU(JP2, K)*SN*CC(JP2, K))*HD4 | 2517969E023 | 419661003 |
| C | | CONTINUE | | |
| S 210 | | CONTINUE | 2501929E023 | 419661003 |
| S 212 | | CONTINUE | 344500E003 | 2673003 |

Ver.2 DO 212ループのループ長 $\frac{2673}{99} = 27$

Ver.2 DO 210ループのループ長 $\frac{419661}{2673} = 157$

図 4. 12. 3 (b) L-3 変更後のプログラム

```

C      PRESSURE CALCULATION
C
S      DO 31 L = 1,3
S      DO 31 K = 1,KMAX
V      DO 31 J = JAM,JBP
V      31 FR(J,K,L) = GAMI*(Q(J,K,L,5)-.5*(Q(J,K,L,2)**2+Q(J,K,L,3)**2
      *                +Q(J,K,L,4)**2)/Q(J,K,L,1)) *Q(J,K,L,6)

```

図 4.12.4 (a) L-4 変更前のプログラム

```

C      PRESSURE CALCULATION
C
C      CXXXX DO 31 L = 1,3
S      DO 31 K = 1,KDIM
V      DO 31 J = JAM,JBP
V      FR(J,K,1) = GAMI*(Q(J,K,1,5)-.5*(Q(J,K,1,2)**2+Q(J,K,1,3)**2
      *                + Q(J,K,1,4)**2)/Q(J,K,1,1)) *Q(J,K,1,6)
V      FR(J,K,2) = GAMI*(Q(J,K,2,5)-.5*(Q(J,K,2,2)**2+Q(J,K,2,3)**2
      *                + Q(J,K,2,4)**2)/Q(J,K,2,1)) *Q(J,K,2,6)
V      FR(J,K,3) = GAMI*(Q(J,K,3,5)-.5*(Q(J,K,3,2)**2+Q(J,K,3,3)**2
      *                + Q(J,K,3,4)**2)/Q(J,K,3,1)) *Q(J,K,3,6)
C      CXXXX FR(J,K,L) = GAMI*(Q(J,K,L,5)-.5*(Q(J,K,L,2)**2+Q(J,K,L,3)**2
V      CXXXX*      + Q(J,K,L,4)**2)/Q(J,K,L,1)) *Q(J,K,L,6)
V      31 CONTINUE

```

最外のLのループをJのループの中に展開した

図 4.12.4 (b) L-4 変更後のプログラム

航空宇宙技術研究所報告 986号

昭和63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丁目44番地1
電話三鷹 (0422) 47-5911 (大代表) 〒182

印刷所 株式会社三興印刷
東京都新宿区西早稲田 2-1-18

Printed in Japan