JAERI-Data/Code
97-051

# 原子力コードのVPP500におけるベクトル化，並列化及び移植（ベクトル化編）
## —平成8年度作業報告書—

1997年12月

根本俊行*・川井　渉*・川崎信夫※・渡辺秀雄*・田辺豪信※
鈴木信太郎*・原田裕夫・庄司　誠・久米悦雄・藤井　実

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

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

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.



編集兼発行　日本原子力研究所
印　　刷　㈱原子力資料サービス

JAERI－Data/Code 97－051

# 原子力コードのVPP500におけるベクトル化，並列化及び移植（ベクトル化編）
## ― 平成８年度作業報告書 ―

日本原子力研究所計算科学技術推進センター
根本　俊行*・川井　　渉*・川崎　信夫※
渡辺　秀雄*・田辺　豪信※・鈴木信太郎*
原田　裕夫　・庄司　　誠　・久米　悦雄
藤井　　実

（1997年11月28日受理）

本報告書は、平成8年度に計算科学技術推進センター情報システム管理課で行った原子力コードのVPP500における高速化及び移植作業のうち、ベクトル化作業部分について記述したものである。

原子力コードのVPP500における高速化及び移植作業は、平成8年度に11件行われた。これらの作業内容は、同種の作業を行うユーザに有益な情報を提供することを意図して、「並列化編」、「ベクトル化編」、「移植編」の3冊にまとめられている。

本報告書の「ベクトル化編」では、中性子・光子輸送計算コードDORT-TORT、気体流動解析コードFLOWGR、相対論的ボルツマン・ウェーリング・ウーレンベック法によるシミュレーションコードRBUUに対して行ったベクトル化作業について記述されている。別冊の「並列化編」では、2次元相対論的電磁粒子コードEM2D、円筒座標系直接数値シミュレーションコードCYLDNS、ダイヤモンド型結晶を扱う分子動力学コードDGRに対して行ったベクトル並列化作業について記述されている。また、別冊の「移植編」では、軽水炉安全解析コードRELAP5/MOD3.2及びRELAP5/MOD3.2.1.2、原子核データ処理システムNJOY、2次元多群ディスクリート・オーディネーツ輸送コードTWOTRAN-IIに対して行った移植作業と汎用図形処理解析システムIPLOTに対して行ったUNIX OSへの移行調査作業について記述されている。

日本原子力研究所（東海駐在）：〒319-1195　茨城県那珂郡東海村白方白根2-4
※ 外来研究員：富士通株式会社
＊ 富士通株式会社

JAERI－Data/Code 97－051

# Vectorization, Parallelization and Porting of Nuclear Codes on the VPP500 System (Vectorization) — Progress Report Fiscal 1996 —

Toshiyuki NEMOTO*, Wataru KAWAI*, Nobuo KAWASAKI※,
Hideo WATANABE*, Hidenobu TANABE※, Shintaro SUZUKI*,
Hiroo HARADA, Makoto SHOJI, Etsuo KUME
and Minoru FUJII

Center for Promotion of Computational Science and Engineering
(Tokai site)
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received November 28, 1997)

Several computer codes in the nuclear field have been vectorized, parallelized and transported on the FUJITSU VPP500 system at Center for Promotion of Computational Science and Engineering in Japan Atomic Energy Research Institute. These results are reported in 3 parts, i.e., the vectorization part, the parallelization part and the porting part. In this report, we describe the vectorization.

In this vectorization part, the vectorization of two and three dimensional discrete ordinates simulation code DORT-TORT, gas dynamics analysis code FLOWGR and relativistic Boltzmann-Uehling-Uhlenbeck simulation code RBUU are described. In the parallelization part, the parallelization of 2-Dimensional relativistic electromagnetic particle code EM2D, Cylindrical Direct Numerical Simulation code CYLDNS and molecular dynamics code for simulating radiation damages in diamond crystals DGR are described. And then, in the porting part, the porting of reactor safety analysis code RELAP5/MOD3.2 and RELAP5/MOD3.2.1.2, nuclear data processing system NJOY and 2-D multigroup discrete ordinate transport code TWOTRAN- II

※ On leave from FUJITSU, Ltd
* FUJITSU, Ltd

are described. And also, a survey for the porting of command-driven interactive data analysis plotting program IPLOT are described.

Keywords: DORT-TORT, FLOWGR, RBUU, Vectorization, VPP500, Nuclear Codes

# 目　　次

# Contents

# 1. はじめに

計算科学技術推進センター情報システム管理課では，原子力コードをセンターが保有する各種のスーパーコンピュータ上に整備し，それぞれのスーパーコンピュータに最適な高速化を施す作業を実施している．この作業は，コンピュータ資源の効率的利用を図るとともにユーザの計算待ち時間の短縮を通じてユーザの仕事の効率化を図っている．

原子力コードのVPP500における高速化及び移植作業は，平成8年度に11件行われた．これらの作業内容は，同種の作業を行うユーザに有益な情報を提供することを意図して，「並列化編」，「ベクトル化編」，「移植編」の3冊にまとめられている．

本報告書の「ベクトル化編」では，中性子・光子輸送計算コードDORT-TORT，気体流動解析コードFLOWGR，相対論的ボルツマン・ウェーリング・ウーレンベック法によるシミュレーションコードRBUUに対して行ったベクトル化作業について記述されている．別冊の「並列化編」では，2次元相対論的電磁粒子コードEM2D，円筒座標系直接数値シミュレーションコードCYLDNS，ダイヤモンド型結晶を扱う分子動力学コードDGRに対して行ったベクトル並列化作業について記述されている．また，別冊の「移植編」では，軽水炉安全解析コードRELAP5/MOD3.2及び，RELAP5/MOD3.2.1.2，原子核データ処理システムNJOY，2次元多群ディスクリート・オーディネーツ輸送コードTWOTRAN-IIについてのVPP500への移植作業と汎用図形処理解析システムIPLOTのUNIX OSへの移行調査作業について記述されている．

第2章「DORT-TORTコードのベクトル化」では，中性子・光子輸送計算コードDORT-TORTコードに対して行った高速化作業について述べる．本コードは，配布されたファイルから使用する計算機に適合する部分を抽出してコードを作成するようになっている．これまでに使用されてきたコードは，SUN版を抽出していたものであるが，これは，ベクトル処理が考慮されていないため，VPP500上では高速性能が得られていなかった．そこで，ベクトル処理が考慮されているCRAY版を抽出し，VPP500向けのベクトル化を施した．この結果，オリジナル版のベクトル実行に比較して，約2倍の速度性能の向上が得られた．

第3章「FLOWGRのコードベクトル化」では，気体流動解析コードFLOWGRに対して行った高速化作業について述べる．本コードは，富士通製大型汎用計算機M780上で利用されていたもので，1ケースの実行につき30回のリスタートを行い，計30時間もの実行時間を要していた．このことから，VPP500上でのベクトル化を行った．この結果，VPP500上で，1PEスカラ実行に比較して，約4.3倍の速度性能の向上が得られた．

第4章「RBUUコードのベクトル化」では，相対論的ボルツマン・ウェーリング・ウーレンベック法によるシミュレーションコードRBUUに対して行った高速化作業について述べる．本コードは，IBM社製計算機向けに，ベクトル化及び並列化されていたが，VPP500上にインストールしたところベクトル化効率が低下し，速度低下を招いていた．このため，VPP500向けベクトル化チューニングを施した．この結果，VPP500上での1PEスカラ実行に比較して，約8.4～13.4倍の速度性能の向上を得た．

尚，本報告書の第2章の作業は根本が，第3章の作業は川井が，第4章の作業は川崎がそれぞれ担当した．

# 2. DORT-TORT コードのベクトル化

DORT-TORT(ver.2.8.14) コード[1]のベクトル化を行った．このコードはすでに富士通(株)製分散メモリ型ベクトル並列計算機 VPP500/42(以下 VPP500)上にインストールされていたが，ベクトル化が施されていないために高速な処理ができなかった．このコードは米国・ORNL で開発され各機関に配布されている．オリジナルのコードには，何種類かの計算機に適応するコーディングが含まれており，ユーザが使用する計算機に合致するコーディング部分を抽出することでコードが完成する．既に原研で使用されているバージョンは，VPP500 の FORTRAN 仕様にもっとも近い Sun 版を抽出して使用している．しかし，これはベクトル処理が考慮されていないため，前述の通り VPP500 では高速性能がでない．そこで，今回のベクトル化にあたり，ベクトル化が考慮されている CRAY 版を抽出してベクトル化を図ることにした．ここでは CRAY 版の抽出，VPP500 へのインストール作業，および VPP500 用のベクトル化について報告する．

## 2.1 インストール前の準備作業

DORT-TORT コードは，いくつかのコード群から成り立っている．前処理用の gip，grtuncl，中間処理用の torsed，visa，2 次元解析用の dort，3 次元解析用の tort である．また，gip，grtuncl，torsed，visa には専用のライブラリを使用する．これらのオリジナルコードはシェルスクリプト（正確にはいくつかのシェルスクリプトを含むファイル）に含まれている．このシェルスクリプトにはオリジナルコード以外に，ツール類，サンプル入力データ等も含まれている．また実行することによって，コード等の抽出と同時に，コンパイル・リンクも行われロードモジュールが作成される．

### 2.1.1 ソース抽出

CRAY 版のソースを抽出するために，このシェルスクリプトを東京・計算科学技術推進センターに設置してある CRAY-T94 に移し，ここで実行した．この実行は ORNL より送られてきたシェルスクリプト 'd2p8p14.id'，および 't2p8p14.id' の 2 つを使用する．これらに含まれるソースを Table 2.1 に示す．

これらのシェルスクリプトは，実行中にコンパイル・リンクも行う．DORT-TORT コードは CRAY 用のコーディングを含んでいるが，その仕様は，CRAY 計算機がいくつか採用しているコンパイラのひとつである C90 モードで書かれている．CRAY-T94 のコンパイラの仕様は C90 モードではなく T90 モードであるため，このままでは正常にコンパイル・リンクはできない．よって，C90 モードにするために，Fig. 2.1 の矢印のように，環境変数 TARGET に 'cray-c90' を定義する必要がある [2].

シェルスクリプト 'd2p8p14.id' には，dort コードの他に gip，grtuncl，torsed，visa コードが含まれている．これらのうち，dort，gip，grtuncl のコード抽出は Fig. 2.1 に示した一連のコマンド，およびシェルスクリプトの実行によってでき，しかも gip，および dort のサンプル問題，各種ツール類も抽出される．また，実行ロードモジュールも作成される．torsed，お

よびvisaコードの抽出方法をFig. 2.2に示す．この実行は，dortのコードを抽出したディレクトリ上で実行する必要がある．これは，dortのコード抽出の際にできたライブラリ・ツール等を使用するためである．この実行によって，コード以外にサンプル問題，CRAY用実行シェルスクリプト等が抽出される．

tortコードの抽出方法を Fig. 2.3に示す．これには't2p8p14.id'を使用する．ただし，CRAY-T94のコンパイラの障害か，仕様の違いか不明であるが，このシェルスクリプトを使用すると，正常に終了しない．現象として，tortコードのコンパイル処理が異常終了してしまう．詳しく調査してみたところ，tortのコンパイルに指定しているコンパイルオプション'-e mxz'のうち'z'をはずすと正常に動作することがわかった．'z'はデバッグツール等に利用されるデバッグ表の生成オプションであり，実行には影響が無いのではずしても問題はない．この修正した部分をFig. 2.4に示す．

これらの実行で個々のコードを抽出することができた．コードには，複数のソースから成り立つものがある．これらをTable 2.2に示す．

実際の取り出し方として，dortコードの抽出(Fig.2.1)が終わった後，Table 2.2に示してあるソースを別のディレクトリに複写する．同様にインクルードファイル，サンプル入力データも複写する．これは，torsed，visaの抽出，およびtortの抽出の際，それぞれにおいて同名のインクルードファイル，サンプル問題等が作成されるためである．このうち抽出されたサンプル問題をTable 2.3に示す．

### 2.1.2 CRAY-T94上での実行確認

VPP500での計算結果と比較するために，CRAY上で実行を行った．2.1.1で述べたように'd2p8p14.id'，および't2p8p14.id'を実行したことで，dort，tort等の実行ロードモジュールは作成されている．また，実行シェルスクリプトも作成されている．この実行シェルスクリプトはTSSでの実行を想定している．しかし，原研のシステムでは原則的にジョブはNQSによるバッチ方式を採用しているため，バッチ用のシェルを作成した．この例をFig. 2.5に示す．

バッチ用の実行シェルを使いサンプル問題を実行してみた．結果の確認は，ORNLよりDORT-TORTコードに添付されてきた計算結果と比較することで行った．しかし，この計算結果は今回配布されたDORT-TORTコードとは異なるバージョンで実行したものであったため，正確な比較ができなかった．このため，おおよその比較を行った．その結果をTable 2.4，および2.5に示す．

DORT-TORTコードの実行は，最初にgipによって前処理(断面積データライブラリの作成)を行い，その後dort，またはtortによってgipの結果を用いて本計算を行う．サンプル問題の結果は，dortに関してはほぼ一致しているが，2ケースにおいて，gipが異常終了したため，dortの実行ができなかった．これはコードのエラーか，入力データのエラーか判断不明であり対策を行わなかった．一方，tortに関しては，gip，tortと連続して実行できるケースが1組しかなかったが正常に動作した．中間処理コードであるtorsed，visaに関しては，5問のサンプル問題すべてにおいてエラーが発生している．これは，コードの内部を調査してみたところ，バージョンがdort，tortのものと異なっていることがわかった．このため，今回のバージョンでの導入を諦めた．gip，grtuncl，dort，tortについては，サンプル問題によるテスト実行は

よびvisaコードの抽出方法をFig. 2.2に示す．この実行は，dortのコードを抽出したディレクトリ上で実行する必要がある．これは，dortのコード抽出の際にできたライブラリ・ツール等を使用するためである．この実行によって，コード以外にサンプル問題，CRAY用実行シェルスクリプト等が抽出される．

tortコードの抽出方法を Fig. 2.3に示す．これには't2p8p14.id'を使用する．ただし，CRAY-T94のコンパイラの障害か，仕様の違いか不明であるが，このシェルスクリプトを使用すると，正常に終了しない．現象として，tortコードのコンパイル処理が異常終了してしまう．詳しく調査してみたところ，tortのコンパイルに指定しているコンパイルオプション'-e mxz'のうち'z'をはずすと正常に動作することがわかった．'z'はデバッグツール等に利用されるデバッグ表の生成オプションであり，実行には影響が無いのではずしても問題はない．この修正した部分をFig. 2.4に示す．

これらの実行で個々のコードを抽出することができた．コードには，複数のソースから成り立つものがある．これらをTable 2.2に示す．

実際の取り出し方として，dortコードの抽出(Fig.2.1)が終わった後，Table 2.2に示してあるソースを別のディレクトリに複写する．同様にインクルードファイル，サンプル入力データも複写する．これは，torsed，visaの抽出，およびtortの抽出の際，それぞれにおいて同名のインクルードファイル，サンプル問題等が作成されるためである．このうち抽出されたサンプル問題をTable 2.3に示す．

### 2.1.2 CRAY-T94上での実行確認

VPP500での計算結果と比較するために，CRAY上で実行を行った．2.1.1で述べたように'd2p8p14.id'，および't2p8p14.id'を実行したことで，dort，tort等の実行ロードモジュールは作成されている．また，実行シェルスクリプトも作成されている．この実行シェルスクリプトはTSSでの実行を想定している．しかし，原研のシステムでは原則的にジョブはNQSによるバッチ方式を採用しているため，バッチ用のシェルを作成した．この例をFig. 2.5に示す．

バッチ用の実行シェルを使いサンプル問題を実行してみた．結果の確認は，ORNLよりDORT-TORTコードに添付されてきた計算結果と比較することで行った．しかし，この計算結果は今回配布されたDORT-TORTコードとは異なるバージョンで実行したものであったため，正確な比較ができなかった．このため，おおよその比較を行った．その結果をTable 2.4，および2.5に示す．

DORT-TORTコードの実行は，最初にgipによって前処理(断面積データライブラリの作成)を行い，その後dort，またはtortによってgipの結果を用いて本計算を行う．サンプル問題の結果は，dortに関してはほぼ一致しているが，2ケースにおいて，gipが異常終了したため，dortの実行ができなかった．これはコードのエラーか，入力データのエラーか判断不明であり対策を行わなかった．一方，tortに関しては，gip，tortと連続して実行できるケースが1組しかなかったが正常に動作した．中間処理コードであるtorsed，visaに関しては，5問のサンプル問題すべてにおいてエラーが発生している．これは，コードの内部を調査してみたところ，バージョンがdort，tortのものと異なっていることがわかった．このため，今回のバージョンでの導入を諦めた．gip，grtuncl，dort，tortについては，サンプル問題によるテスト実行は

完全とは言えないが，実データによる実行でなんらかの障害が発生した場合は，その都度対応を行うことにして，作業を進めることにした．

## 2.2 VPP500 へのインストール

2.1で述べたようにCRAY上においてソースの抽出ができた．これをVPP500に転送してインストール（変換）作業を行った．作業的には富士通製のプログラム変換支援パッケージSTREAM77[3]と手作業による変換を行った．

### 2.2.1 STREAM77の実行

CRAY計算機とVPP500のFORTRAN仕様の大きな違いは，1ワードの長さが異なることである．CRAYは整数・実数ともに単精度が64ビットであるのに対し，VPP500は32ビットである．今回のVPP500へのインストールの目的のひとつが，倍精度で計算することであるので，すべての実数型定数・実数型変数に対し，倍精度化する必要がある．DORT-TORTコードは，実数型変数・配列が整数型変数・配列とEQUIVALENCE，または副プログラムとの引数による共有をしているものがある．このため，これらの整数には，次元をひとつ上げて実数型変数・配列との整合性をとる必要がある．このイメージをFig. 2.6に示す．STREAM77の機能はこの倍精度化を含め次のような機能がある．

- CRAY特有の命令の富士通FORTRANの命令への置き換え
- 実数変数・配列の倍精度化
- 組込み関数の倍精度化

(1) STREAM77を実行するための準備

STREAM77は，コンパイラがFORTRAN77(V10)時代のツールであるため，M780上にしかない．このため，ソース群を東海研のM780に転送した．またSTREAM77において上で述べた変換を行うものはC-エンジンという部品である．これは，入力ソースが大文字であることを前提としている．そのために，DORT-TORTコードを構成するすべてのソースに対し大文字化した．

さらに，STREAM77がなるべく完全な変換を行うことができるように，入力ソースにSTREAM77の変換制御行を挿入した．これは，次元上げすべき整数配列を，STREAM77は宣言文等から判断するが，STREAM77は完全に正しく判断することはできない．そのため，情報をSTREAM77に与える機能が用意されている．具体的には，(STREAM77が判断できない)次元上げすべき配列を入力ソース中に変換制御行を用いて宣言する．変換制御行には2種類あり，全副プログラムに対して有効な整数配列名と，各副プログラムのみに有効な整数配列名を指定するものがある．挿入したこれらの指示行をFig. 2.7～2.11に示す．これらのなかで，'STREAM77G'で始まるものは全副プログラムに対して有効なもので，'STREAM77L'で始まるものは宣言した副プログラムのみで有効なものである．

(2) STREAM77の実行

STREAM77はM780上ではJCLにて実行する．このJCLの例をFig. 2.12に示す．

### 2.2.2 手作業による変換作業

STREAM77 にて変換を行ったが，これで完全に変換が終わったわけではない．ルーチン間の関係等は STREAM77 では見ることができないため，これらに伴う変換は手作業にて行わなければならない．

(1) 領域合わせ

整数配列・変数が羅列している COMMON ブロックにおいて，先頭の配列を引数として副プログラムに渡し，副プログラムでは実数配列で受け，COMMON に含まれる配列・変数に値を代入するものがある．この例を Fig. 2.13 に示す．実数の倍精度化に伴い，COMMON の配列・変数とずれが生じるため，領域を合わせるようにした．合わせ方は，EQUIVALENCE 文を使い Fig. 2.14 に示すようにした．また，この例では，配列 locl を整数配列でも受け取っている．locl の次元を上げたことで，これら受け側の整数配列に関しても次元を上げる必要がある．このような例は，何箇所も存在する．この修正を行った整数配列を COMMON 毎に Table 2.6 に示す．

(2) 宣言の変更

サブルーチン wot3(bb.lib.f, dort) では，親ルーチン tape(gip) の実引数で直接文字を指定されたものを，仮引数では実数型変数で受け取っている．この変数はサブルーチン edit(bb.lib.f, dort) に引数で渡し，edit の中では，4 バイトの整数型変数で受け，実際の使用時には文字型として使用している．wot3 での実数変数を 4 バイトの文字型に変更し，edit で使用している変数も同様に 4 バイトの文字型に変更した．これについて，オリジナルを Fig. 2.15，倍精度化したものを Fig. 2.16 に示す．

これと同様なものが，wot4(bb.lib.f)，edit 間であった．このため同様な処理をした．

(3) 整数の引き渡し

親ルーチンの実引数に直接整数，または整数型変数を指定し，子ルーチンの仮引数では実数型変数として指定しているものがいくつかあった．これは，倍精度化された実数の前半部に整数値が入ることになる．今回の変換では，整数と実数変数が共有する場合は，実数の後半部に整数を代入することにしているため，修正した．例としてオリジナルのサブルーチン tape，seqio を Fig. 2.17，修正したものを Fig. 2.18 に示す．ここに示すように，実数変数 ww とこれと共有する大きさ 2 の整数配列 iww を設け，iww の添字が 2 のエリアに渡すべき整数の値・変数を代入し，ww を実引数に指定した．これと同様なものが何箇所かある．これを Table 2.7 に示す．

また，実引数・仮引数は共に整数型であるが仮引数が次元上げの対象となっている整数配列の場合がある．このような場合は，親ルーチンで大きさ 2 の整数配列 iww を作成し，この整数配列の添字 2 のエリアに，実引数で指定してあった定数，または変数を代入して，iww を実引数に指定した．また '0' を実引数に指定している場合は，DATA 文でゼロクリアした大きさ 2 の整数配列 izero を作成し，これを実引数に指定した．Fig. 2.19 にオリジナルのサブルーチン vario, blkio を，Fig. 2.20 にこの修正例を示す．これらのようなところは何箇所か存在する．これらの gip，grtuncl，dort，tort のそれぞれ修正したものを Table 2.8 ～ 2.11 に示す．

(4) 組込み関数

サブルーチン timez(bb.lib.f，dort，tort) で，CPU 時間，および現在の時刻を取得するために CRAY 特有のサービスルーチンが使われていた．これを VPP500 に用意されているサブ

ルーチンに置き換えた．これを Fig. 2.21 に示す．

(5) 動的領域確保

CRAY 版の DORT-TORT コードは，コードが実行時に使用する領域を動的に確保する仕様になっている．VPP500 では動的領域確保の機能が無いため，静的に領域を確保しなくてはいけない．CRAY 版は動的領域確保をサブルーチン memorex，およびこのルーチンから呼ばれるいくつかのサービスルーチンで行っている．一方，Sun 版の DORT-TORT コードは，静的に領域を確保する仕様，すなわち，あらかじめ実際に使用する領域の大きさに係わらず，大きく領域を確保している．実際のコードの使用に影響がないので，Sun 版の (静的領域確保を行う)memorex を採用する．Sun 版のメインルーチンは単精度でコーディングされているので，倍精度化し VPP500 版のコードに組み込んだ．これは，gip，grtuncl，dort，tort のすべてで行った．

(6) 文字の取扱い

整数型の配列に対し，文字・整数の両方で使用するものがある．文字として使用する場合，正しい表示ができない．そのために，文字型の配列を設け，これを整数型配列と共有 (EQUIVALENCE) させ，使い方によってこれらを選んで使うようにした．この処置をしたのはサブルーチン fidas，mapr(共に gip，dort，tort，unc) である．例としてオリジナルの mapr を Fig. 2.22，修正版を Fig. 2.23 に示す．

また，STREAM77 によって，(オリジナルでは)DATA 文にて文字が定義されていた整数型変数が，実数型に変更させられたものがある．この (実数型に変えられた) 変数を使って，別の整数変数を定義してる箇所があった．VPP500 では実数以外のものを格納した実数型変数を使った計算式を実行すると，無効な演算を実行したというエラーが発生する．そのため，実数型の宣言文を正しい型である文字型に変更すると同時に，これと (EQUIVALENCE 文による) 共有する整数型変数を設け，計算式の中では整数型変数を使用するようにした．この修正を行ったのは，サブルーチン loco(dort，tort) である．オリジナルを Fig. 2.24，修正版を Fig. 2.25 に示す．

(7) アセンブラの取扱い

CRAY 版の DORT-TORT コードは，一部のサブルーチンでアセンブラで記述されたものがある．これは，rowd と rowi というルーチンで，これらはサブルーチン rowc，および rowv から呼ばれている．(アセンブラルーチンも含めて) これらは，CRAY 用に作成されたベクトル化済のサブルーチンである．しかし，CRAY のアセンブラについては変換ができない．プログラム中のコメント，および DORT-TORT のマニュアルには，ベクトル化版を採用しない場合は，これらのかわりに rowt というサブルーチンを使用することができる旨が記述されていた．実際，Sun 版では rowt が呼ばれている．ここで，ベクトル版，およびスカラ版のツリー構造図を Fig. 2.26 に示す．

サブルーチン row はコントロールフラックスを計算する親ルーチンである．row は解析内容によって，解法手法別にいくつか用意されているサブルーチンを選択して，実行する．これら解法手法別サブルーチンに中に rowv，および rowc が含まれている．

具体的には row では解析対象によって，変数 irow，および imode の値が決定される．irow の値は 1 から 7 までの値をとり，それぞれ解法手法別サブルーチンが対応している，imode

は，これ自身の値よって，1，もしくは2種類の irow の値をとることになっており，この値に基づいて呼び出す解法手法別サブルーチンが決定する．CRAY 版，および Sun 版におけるこの組合せを Table 2.12 に示す．これによると，imode=3, または 4 の時に CRAY 版では rowc, rowv が呼ばれているのに対し，Sun 版は rowt が呼ばれている．しかし，imode=5，または 9 の時は Sun 版も rowc を呼んでいる．これは，Sun 版用に CRAY 版とは異なる rowc が用意されているためであり，このルーチンからはアセンブラルーチンは呼んでいない．

これらのことより，VPP500 版では，Sun 版同様に imode=3，または 4 の時に rowt を呼ぶようにし，imode=5，または 9 の時に呼び出す rowc は，Sun 版のものを倍精度化して採用することにした．オリジナルの row の一部を Fig. 2.27，rowv，rowc の変わりに rowt を呼ぶように変更した row の一部を Fig. 2.28 に示す．

(8) 断面積ライブラリの読出部分の変更

DORT-TORT コードの grtuncl，dort，および tort の実行時に断面積ライブラリを読み込む．(原研で作成した) 断面積ライブラリは現在，単精度版のコードで作成したものしかないため，この読み込み部分を単精度対応にしなくてはいけない．読み込み部分はサブルーチン reed で行っている．断面積は grtuncl では論理機番 4 番，dort では同じく 8 番で読んでいる．論理機番は変数 lun を用いているので，lun が 4 (dort の場合は 8 番) の時だけ単精度のワーク配列 (sarray) を使って読み込み，その後，倍精度化して本来の読み込みに使用する配列 array に代入するようにした．修正例として grtuncl のオリジナルの reed を Fig. 2.29，修正後の reed を Fig. 2.30 に示す．

### 2.2.3 VPP500 版の実行，および計算結果

STREAM77，および手作業によって変換を行った VPP500 版を実際に実行してみた．これを，すでにインストールされている Sun 版 (単精度版) と比較してみた．計算結果の最終的な値は Sun 版と近い値であったが，異なる箇所もあった．これは，単精度計算と倍精度計算の違いによる要素が大きく影響している．dort，tort はいくつかの収束計算があり，その計算途中の値を出力する箇所があり，倍精度計算と単精度計算の違いがここで現れている．収束した値は同様な値となるが，その (収束) 過程での値が異なるため出力結果が異なるのである．コード内部では，差分式の値がしきい値より小さくなった場合に，収束計算を終了する箇所がある．IEEE 形式の場合はM形式と比べて，単精度より倍精度が顕著に小さい値を表すことができるので，この違いが現れる．

## 2.3 ベクトル化について

STREAM77，および手作業による変換を行った後，ベクトル化を行った．DORT-TORT コードの中でベクトル化の対象としたのは dort，および tort である．

### 2.3.1 動的挙動解析

dort において入力データ 'j175.jffnj' の規模を小さくし，動的挙動解析ツール ANALYZER[4] を使用してコスト分布を計測した．ANALYZER は VPP500 システムの GSP で動作する．数

値形式がM形式でありIEEE形式であるVPP500とは異なるために，挙動がVPP500とは若干異なるが，コスト解析上ではあまり問題がない．このANALYZERの解析結果をTable 2.13に示す．

これによると一番コストが高いのがrowtで84.8%，次がwsolが9.0%である．よってこれらを中心にベクトル化を行っていくことにした．

### 2.3.2 ベクトル化のための変更

ここでは，ベクトル化を施した副プログラムについて，変更内容を述べる．

(1) サブルーチンrowt(dort)

サブルーチンrowtはコントロールフラックスを計算する一部である．このサブルーチンのANALYZERによるループリスト(コスト分布)をTable 2.14に示す．これによると，一番コストが高いのはtotalの79.2%である．ループリストでtotalが高いということは，DOループ以外の部分のコストが高いことを意味している．ベクトル処理の対象となるのはDOループであり，このままではtotal以外のDOループをいくらベクトル化しても高速化できない．そのためにDOループの無いところにDOループを作成する必要がある．

rowtのDOループの無い部分の構造図をFig. 2.31に示す．この構造図によるとGO TO文によるループがあることがわかる．これは，i1に関するループである．このループの中で割当て型GO TO文によるループの外への分岐があるが，分岐後の処理を行って再びループ内に戻ってくる．割当て型GO TO文に指定される変数initmlは，このループの中では不変である．ループはi1が変数imsblkによって定義され，ループ1回につき1ずつ値が引かれてゆき，0になったらループの後の処理に進むことになっている．これらのことより，割当て型GO TO文で飛ぶ先の処理について，それぞれ分割して独立させたDOループを構成することができる．この構成をFig. 2.32に示す．これらのDOループをベクトル化の対象とする．

これらのDOループをベクトル化するにあたり，ベクトル化非対象のものを外す必要がある．Fig. 2.31のオリジナルの構成において，i1のループからwrite文の処理に飛び(if(nou.gt.0) go to 9900)，再び戻ってくる(go to 3900)箇所がある．本来なら分割したDOループにもこのwrite文の処理を展開しなければならない．しかし，write文はベクトル化できない．このwrite文はループ内で計算された値を書き出すものだが，write文の目的はデバッグ用のものであると思われる．この理由は，if文でwrite文を実行するかどうか決めていることと，実際にwrite文を実行すると，(実行回数が多いために)極めて莫大な出力量になり通常のディスクに書くことが困難になるからである．このため，このwrite文を採用することをやめた．

ここで，i1のループのベクトル化について説明する．このオリジナルのループの一部をFig. 2.33に示す．また，割当て型GO TO文の飛び先の処理のひとつである6030 continue以降の処理ブロックをFig. 2.34に示す．これらによると，変数i1は，単にループの回転の数のカウンタとしてのみ使用されている．

ここで変数iに着目する．iは初期値が変数istrによって定義され，ループ1回ごとに増分値incが足しこまれていく．増分値incはループ実行中は不変である．また，ループの回転数はi1の初期値を定義しているimsblkを使ってループ実行前に確定することができるので，終

値が決まる．iはループ中，および割当てGO TO文の飛び先の処理ブロック内でも配列の添字として使用されている．よって，ベクトル化対象ループのDO変数としてiを採用することにした．ループの形はFig. 2.35のようになる．このループの中に，旧i1ループ内の処理，および割当てGO TO文の飛び先の処理が入る．

このループに対してベクトル化を行う．旧i1のループ(Fig. 2.33)と旧6030 continueの処理ブロックを例に説明する．これらをひとつのループに納めた場合，そのままではベクトル化できない．これは配列fjo，およびfmoは参照が先に，定義が後に現れることと，変数fiom，および配列fixが回帰計算を行っているためである．

前者の場合，コンパイラは「定義の前に参照がある」との理由でベクトル化を行わない．fjo，fmoを定義する際，(旧)fjo，fmoの値を使っていないので回帰計算にはなっていない．そこでベクトル化を行うために，ワーク配列を新規に設けて，このループ実行前にfjo，fmoの内容を複写しておく．DOループ内では，参照部分に関してこのワーク配列を使用することにする．これだとfjo，fmoは定義にのみ使われることになるので，ベクトル化の条件を満たすことになる．このワーク配列の複写部分をFig. 2.36に示す．

一方後者の場合，fiom，fixについての回帰計算は回避することはできない．しかし，fiomとは関係のない処理もあるので，DOループをfiom，fixに関する計算と，これ以外のベクトル計算の対象となる処理に分割した．なお，処理の都合上，ベクトル化fiom，fixの計算の前後にベクトル化対象となる処理を配置した．この時点のこの構造について，Fig. 2.37に示す．このDOループ分割をしたため，DOループをまたいで，定義・参照されるいくつかの変数については配列化した．このコーディング部分の例のとして旧6030 continueブロックのものをFig. 2.38に示す．

Fig. 2.38に旧6030 continueブロックの処理内容を示したが，他のブロックについても同様な処置を行った．ここで，Fig. 2.36に示したfjo，fixのワーク配列への複写はどのブロックでも必要な処理である．そこで，一連の処理群の先頭に，このワーク配列への複写部分を移動した．これで，rowtのベクトル化が完成した．

(2) サブルーチンrowb(tort)

tortコードにおいて，dortコードのrowtに相当するサブルーチンがrowbである．今回のtortに関する実入力データは無かったが，このサブルーチンのコストは高くなると予想される．このため，ベクトル化を行った．

tortのrowbは，rowtほど複雑ではないが，構造的によく似ているため同様なベクトル化手法をとった．rowbの構造図をFig. 2.39に示す．これによると割当て型GO TO文による分岐先は2か所である．ここでは，変数initmxの値によって文番号3600，または3700に分岐する．文番号3700以降の処理は，文番号3600の処理の後に必ず実行されるので，rowtのように分岐先ごとに処理ブロックを作る必要はない．initmxの値の定義はinitmwの値によって決定する．すなわち，initmwの値によって文番号3600の処理ブロックを実行するか否か決定がされる．割当て型GO TO文はベクトル化の対象となっていない．このために，割当て型GO TO文を廃止して，IF文を使用してinitmwの値によって，文番号3600の処理の実行を制御するようにした．これにより，rowtのように分岐先ごとに処理ブロック群を作る必要はなくなった．

また，Fig. 2.39 に示した構造図より，i1 に関するループが存在する．このループは rowt 同様 i によるスカラ，およびベクトルの DO ループに分割することができる．オリジナルの rowb の i1 のループの一部を Fig. 2.40 に示す．これを rowt 同様，i のループに変更しベクトル化を施す．ベクトル化の弊害となるのは，変数 nvl の値によって変数が定義される部分である．nvl の値によって，変数 fjomi，atxifw が定義される場合がある（されない場合もある）．この部分の後に，これら変数の参照があるため「定義の前の参照がある」ということでベクトル化できない．このために，nvl の IF 文を無くすことにする．nvl の値が 0 以外の時は，fjomi は配列 fjo，atxia は計算式 dzxiz*at によって定義され，nvl の値が 0 の時は定義されない．fjomi，および atxia は，i のループに入る前にそれぞれ 0.0 という値で初期化されている．また，nvl は i のループ実行中に値が変わることはない．このため，この IF 文による定義部分を i ループから独立させ，i ループの実行前に値を定義することにする．ただし，fjomi に関しては，(nvl が 0 以外の時) 配列 fjo によって定義される．fjo は，rowt 同様，i ループの後半で定義される．よって，rowt と同様な手法を採用した．すなわち，fjo 用のワーク配列 (wfjo) を用意し，nvl が 0 の時は 0.0，0 以外のときは fjo の値を代入するようにした．wfjo は i ループ中で fjomi の定義に使用する．また fjo 同様，参照・定義の順序で現れる配列 fko がある．これもワーク配列 wfko を用意して，このループで代入する．このループを Fig. 2.41 に示す．

次に，i ループをベクトル化するが，これも変数 fiom に関して回帰計算が発生する．このため，fiom に関係する部分と関係しない部分に分割し，関係しない部分はベクトル処理，関係する部分はスカラ処理を行うようにした．分割方法は rowt と同様である．ただし，rowt は割当て GO TO 文の分岐先ごとに処理ブロックを作成したが，rowb の場合は，1 つの処理ブロック内で分岐先ごとの処理をできることは前述した．initmw の値によって割当て型 GO TO 文による分岐先が決まっていたので，処理ブロック内で initmw の値によって処理の制御を行うようにした．具体的には，IF 文を用いて，旧 3600 の処理の制御を行った．ベクトル化を施した i ループの一部を Fig. 2.42 に示す．

(3) サブルーチン wsol(dort)

5 重対角行列の解法の一部を行うサブルーチンである．5 重対角行列を変形して 3 重対角行列にし，ガウスの消去法で解いている．このサブルーチンは ANALYZER によると，rowt に続いて 2 番目にコストの高いものである．このサブルーチンのループリストを Table 2.15 に示す．

このサブルーチンはすでにベクトル化されている．コストの高い上位 3 つのループのコスト比率は全部で 78.5% であり，これをベクトル化の対象とする．これらのループは，スカラ処理されているが，これは，コンパイラがこのループ内の計算が独立である（回帰計算がない）と判断できないため，ベクトル化処理していない．これらのループは CRAY 版の独立を意味する最適化指示行 (cdir$ ivdep) が挿入されている．よって，これに相当する VPP500 での最適化指示行 [5]*vocl loop,novrec を挿入することでベクトル化することができた．例として，DO 270 のループのオリジナルを Fig. 2.43，ベクトル化したこのループを Fig. 2.44 に示す．

(4) その他 (dort, tort)

dort，および tort はベクトル化されたサブルーチンがいくつか含まれている．今回のベク

トル化作業で使用した入力データでは実行されないサブルーチンにもいくつかベクトル化版があった．これにはCRAY版の最適化指示行(cdir$ ivdep)等が挿入されている．VPP500のFORTRANコンパイラでこれらのDOループにおいてベクトル化がされていないものには，*vocl loop,novrecを挿入したり，ごく簡単な修正をしてベクトル化を行った．これらのサブルーチンをTable 2.16に示す．

## 2.4 ベクトル化効果

ここでは，ベクトル化版の計算結果，および効果について述べる．

### 2.4.1 計算結果

作成したベクトル化版を用いて計算を行った．オリジナル版のスカラ実行の計算結果とベクトル化版のスカラ実行の計算結果は完全に一致した．また，オリジナル版のスカラ実行に対し，ベクトル化版のベクトル化実行の結果は若干異なるが総じて結果は合っていた．この違いは，ベクトル処理とスカラ処理は数値の丸めが異なることと，最適化の違いによるものである．

### 2.4.2 ベクトル化の効果

ベクトル化版に対しANALYZERを用いてコスト分布を調査した．その結果をTable 2.17に示す．この表のv-effectとTable 2.13に示したオリジナルのコスト分布表のv-effectを比較すると，ベクトル化を施したサブルーチンは倍率が上昇していることがわかる．また，入力データ'j42.jffnj'，および'j175.jffnj'について，オリジナル版のベクトル実行時間（CPU），およびベクトル化版のベクトル実行時間（同）の比較を行った(最適化モードは共にOPT(E))．この結果をTable 2.18に示す．これによると，約2倍の高速化が達成されている．高倍率が達成されなかった理由として，コストの高いサブルーチンにおいて回帰計算があったためベクトル化率が高くならなかったことと，出力ファイルの書き出し量が多く(全部で500MB 以上)，I/Oに時間がかかったことが原因と思われる．

## 2.5 まとめ

今回の作業はCRAY版の変換(倍精度化を含む)からベクトル化まで行ったため，作業量は多くなったが，CRAY上でベクトル化してあるサブルーチンは，簡単にVPP500上でベクトル化ができたので，この分作業は比較的スムーズに進めることができた．ただし，コストの高いルーチンで回帰計算があったために高倍率は達成できなかったが，オリジナルが長時間ジョブのため，ベクトル化の効果はあると思われる．

また，コードの性質上書き出すファイルの量が多いため，ディスクを圧迫することが多く，作業を進めていく上で多少の手間がかかってしまった．CPUが高速になるにつれ，このようなことはこれから多くなっていくと思われる．よって，これからはジョブの実行環境について，総合的に考慮していく必要があると思う．

Tabel 2.1 DORT-TORT's shell-scripts

| ファイル名 | 内　　容 |
|---|---|
| d2p8p14.id | gip, grtuncl, dort, torsed, visaのソース、<br>サンプル入力データ、実行シェル、ツール等 |
| t2p8p14.id | tortのソース、サンプル入力データ、実行シェル |

Table 2.2 Composition of DORT-TORT code

| コード名 | 構成するソース群 |
|---|---|
| dort | bb.dor.f, bb.ccc.c(C言語), bb.cal.f(アセンブラ) |
| tort | bb.tor.f, bb.ccc.c(C言語), bb.cal.f(アセンブラ) |
| gip | bb.gip.f, bb.lib.f, bb.ccc.c(C言語) |
| grtuncl | bb.unc.f, bb.lib.f, bb.ccc.c(C言語) |
| torsed | bb.sed.f, bb.lib.f, bb.ccc.c(C言語) |
| visa | bb.vis.f, bb.lib.f, bb.ccc.c(C言語) |

Table 2.3 Sample problems

| コード名 | 入力データ |
|---|---|
| gip (dort 用) | probg1.F probg2.F probg3.F probg4.F probg5.F<br>probg6.F probg7.F probg8.F probg9.F probga.F<br>probgb.F probgc.F probgd.F |
| dort | probp1.F probp2.F probp3.F probp4.F probp5.F<br>probp6.F probp7.F probp8.F probp9.F probpa.F<br>probpb.F probpc.F probpd.F |
| gip (tort 用) | probg8.F probg20.F |
| tort | probp1.F probp2.F probp3.F probp4.F probp5.F<br>probp6.F probp7.F probp8.F |
| torsed, visa | yseddog.F yseddog2.F yseddog8.F yseddoga.F<br>yseddogc.F |

Table 2.4 Executed results of sample problems (gip, dort)

| gip(dort用) | | dort | |
|---|---|---|---|
| 入力データ | 結　果 | 入力データ | 結　果 |
| probg1.F | 一致 | probp1.F | 一致 |
| probg2.F | 一致 | probp2.F | 一致 |
| probg3.F | 一致 | probp3.F | 一致 |
| probg4.F | 一致 | ppobp4.F | 一致 |
| probg5.F | 一致 | probp5.F | 一致 |
| probg6.F | 一致 | probp6.F | 一致 |
| probg7.F | 一致 | probp7.F | 一致 |
| probg8.F | 一致 | probp8.F | 一致 |
| probg9.F | 一致 | probp9.F | 一致 |
| probga.F | エラー発生 | probpa.F | 未実行 |
| probgb.F | エラー発生 | probpb.F | 未実行 |
| probgc.F | 一致 | probpc.F | 一致 |
| probgd.F | 一致 | probpd.F | 一致 |

Table 2.5 Executed results of sample problems (gip, tort)

| gip(tort用) | | tort | |
|---|---|---|---|
| 入力データ | 結　果 | 入力データ | 結　果 |
| | | probp1.F | 未実行 |
| | | probp2.F | 未実行 |
| | | probp3.F | 未実行 |
| | | ppobp4.F | 未実行 |
| | | probp5.F | 未実行 |
| | | probp6.F | 未実行 |
| | | probp7.F | 未実行 |
| probg8.F | 一致 | probp8.F | 一致 |
| probg20.F | 正常終了 | | |

Table 2.6 Improved integer arrays

| コード名 | ｲﾝｸﾙｰﾄﾞ名 | COMMON名 | 対象となる整数配列 | 補 足 |
|---|---|---|---|---|
| gip | comdot | cmpar | locp, loce | |
| | | comdot | loci | |
| | comflx | cmflx | locf | |
| | comin | comin | mult (imult) | 名前を変更 |
| | comio | iorec | iocm (iiocm) | 名前を変更 |
| | comloc | comloc | locl | |
| | vppbul | bulkbu | locl | 新規にｲﾝｸﾙｰﾄﾞﾌｧｲﾙを作成 |
| grtuncl | comdot | cmpar | locp, loce | |
| | | comdot | loci | |
| | comflx | comflx | locf | |
| | comin | comin | mult (imult) | 名前を変更 |
| | comio | iorec | iocm (iiocm) | 名前を変更 |
| | comloc | comloc | locl | |
| dort | comdot | cmpar | locp, loce | |
| | | comdot | loci | |
| | comflx | comflx | locf | |
| | comin | comin | mult (imult) | 名前を変更 |
| | comio | iorec | iocm (iiocm) | 名前を変更 |
| | comloc | comloc | locl | |
| | comrow | comrow | nbfri | |
| tort | comdot | cmpar | locp, loce | |
| | | comdot | loci | |
| | comflx | comflx | locf | |
| | comin | comin | mult (imult) | 名前を変更 |
| | comio | iorec | iocm (iiocm) | 名前を変更 |
| | comloc | comloc | locl | |
| | comrow | comrow | lpointv, 100 | |

Table 2.7 Improvements of argument (actual argument is integer constant or integer variable, and dummy argument is real variable)

| コード名 | ｻﾌﾞﾙｰﾁﾝ名 | CALL文 | |
|---|---|---|---|
| gip | tape | 修正前 | call seqio(2, nt4, 777, 1,0) |
| | | 修正後 | call seqio(2, nt4, ww , 1,0) |
| | wandr | 修正前 | call seqio(0, nunix, nunit, 0,0) |
| | | 修正後 | call seqio(0, nunix, ww , 0,0) |
| grtuncl | wandr | 修正前 | call seqio(0, nunix, nunit, 0,0) |
| | | 修正後 | call seqio(0, nunix, ww , 0,0) |
| tort | wandr | 修正前 | call seqio(0, nunix, nunit, 0,0) |
| | | 修正後 | call seqio(0, nunix, ww , 0,0) |

Table 2.8 Improvements of argument in gip (actual argument is integer constant or integer variable, and dummy argument is integer array)

| コード名 | ｻﾌﾞﾙｰﾁﾝ名 | CALL文 | |
|---|---|---|---|
| gip | seqio | 修正前 | call dopc(-1,nt,0,jerr,0,0,0,0) |
| | | 修正後 | call dopc(-1,nt,0,jerr,izero,0,0,0) |
| | | 修正前 | call dopc(3,nt,nt,jerr,0,0,0,0) |
| | | 修正後 | call dopc(3,nt,nt,jerr,izero,0,0,0) |
| | tape | 修正前 | call wot3(cx(1,i04),0,i05,4hpos.,4hmat.,・・) |
| | | 修正後 | call wot3(cx(1,i04),izero,i05,4hpos.,4hmat.,・・) |
| | vario | 修正前 | call blkio(ii,lun,iary, ixx,irec,ld,nrecmx, ・・) |
| | | 修正後 | call blkio(ii,lun, iww, ixx,irec,ld,nrecmx, ・・) |
| | | 修正前 | call blkio(ii,lun,0,0,0,ld,0,0,0) |
| | | 修正後 | call blkio(ii,lun,izero,0,0,ld,0,0,0) |
| | | 修正前 | call blkio(ii,lun,0,0,0,ld,0,0,0) |
| | | 修正後 | call blkio(ii,lun,izero,0,0,ld,0,0,0) |
| | | 修正前 | call blkio(ii,lun,0,0,0,ld,nrecmx,nwblkm,0) |
| | | 修正後 | call blkio(ii,lun,izero,0,0,ld,nrecmx,nwblkm,0) |
| | | 修正前 | call blkio(ii,lun,ibuf,nwds,irec, ・・) |
| | | 修正後 | call blkio(ii,lun,iww ,nwds,irec, ・・) |

Table 2.9 Improvements of argument in grtuncl (actual argument is integer constant or integer variable, and dummy argument is integer array)

| コード名 | ｻﾌﾞﾙｰﾁﾝ名 | CALL文 | |
|---|---|---|---|
| grtuncl | seqio | 修正前 | call dopc( -1, nt, 0, jerr, 0,0,0,0) |
| | | 修正後 | call dopc( -1, nt, 0, jerr, izero,0,0,0) |
| | | 修正前 | call dopc( 3, nt, nt, jerr, 0,0,0,0) |
| | | 修正後 | call dopc( 3, nt, nt, jerr, izero,0,0,0) |
| | tpface | 修正前 | call wrdso(im,iset,0.d0,0,d,igm,0,jm,lm,im, ・・) |
| | | 修正後 | call wrdso(iww,iset,0.d0,0,d,igm,0,jm,lm,im,・・) |
| | unclf | 修正前 | call blkio(102,nbuf2,ixx,ijmn,0,k,migmi,0,0) |
| | | 修正後 | call blkio(102,nbuf2,iww,ijmn,0,k,migmi,0,0) |
| | vario | 修正前 | call blkio (ii,lun, iary, ixx, irec, ld,・・) |
| | | 修正後 | call blkio (ii,lun, iww , ixx, irec, ld,・・) |
| | | 修正前 | call blkio( ii, lun, 0,0,0, ld, 0,0,0 ) |
| | | 修正後 | call blkio( ii, lun, izero,0,0, ld, 0,0,0 ) |
| | | 修正前 | call blkio( ii, lun, 0,0,0, ld, nrecmx, ・・) |
| | | 修正後 | call blkio( ii, lun, izero,0,0, ld, nrecmx, ・・) |
| | | 修正前 | call blkio( ii, lun,ibuf , nwds, irec,・・) |
| | | 修正後 | call blkio( ii, lun, iww , nwds, irec,・・) |
| | | 修正前 | call blkio( ii, lun,ibuf, nwds, irec, ・・) |
| | | 修正後 | call blkio( ii, lun, iww, nwds, irec, ・・) |
| | wott | 修正前 | call wot3( x, 0, ncol, -ltbl, top1,・・) |
| | | 修正後 | call wot3( x, izero, ncol, -ltbl, top1,・・) |

Table 2.10 Improvements of argument in dort (actual argument is integer constant or integer variable, and dummy argument is integer array) (1/2)

| コード名 | ｻﾌﾞﾙｰﾁﾝ名 | CALL文 | |
|---|---|---|---|
| dort | actvy | 修正前 | call edit ( act,  0,  i1,-nreg, 4hrgn=, ・・) |
| | | 修正後 | call edit ( act,izero,  i1,-nreg, 4hrgn=, ・・) |
| | anfout | 修正前 | call edit ( dirf,  0, mm, -im, 4h  i=, ・・) |
| | | 修正後 | call edit ( dirf,izero, mm, -im, 4h  i=, ・・) |
| | grdscl | 修正前 | call wot3( cban,  0, 6, -icmjcm, 'ij  ', ・・) |
| | | 修正後 | call wot3( cban, izero, 6, -icmjcm, 'ij  ', ・・) |
| | inpa | 修正前 | call ndxr (d(libis),ism,d(limbis),  0,-1,1) |
| | | 修正後 | call ndxr (d(libis),ism,d(limbis),izero,-1,1) |
| | | 修正前 | call ndxr (d(lmbms),msm,d(lmmbms),  0,-1,1) |
| | | 修正後 | call ndxr (d(lmbms),msm,d(lmmbms),izero,-1,1) |
| | | 修正前 | call ndxr (d(liljbg),igm,d(lnmbig),  0,lm,imsjm) |
| | | 修正後 | call ndxr (d(liljbg),igm,d(lnmbig),izero,lm,imsjm) |
| | input | 修正前 | call wot3(d(lijgsz),  0, -njszn, -niszn, ・・) |
| | | 修正後 | call wot3(d(lijgsz),izero, -njszn, -niszn, ・・) |
| | macmx | 修正前 | call edit (sig,  0, mtm, -ihp, 4hposn, ・・) |
| | | 修正後 | call edit (sig,izero, mtm, -ihp, 4hposn, ・・) |
| | nquadr | 修正前 | call edit (msoms,  0, -msm,-msm, 4hmset, ・・) |
| | | 修正後 | call edit (msoms,izero, -msm,-msm, 4hmset, ・・) |
| | quad | 修正前 | call edit( ilfrt,  0,-2*mnquad,-mma, ・・) |
| | | 修正後 | call edit( ilfrt,izero,-2*mnquad,-mma, ・・) |
| | | 修正前 | call edit (snpl(i13),  0, mms,1-lm, ・・) |
| | | 修正後 | call edit (snpl(i13),izero, mms,1-lm, ・・) |
| | sbsrin | 修正前 | call wot3 ( intsr,  0,-ism,-nintsr, ・・) |
| | | 修正後 | call wot3 ( intsr, izero,-ism,-nintsr, ・・) |
| | | 修正前 | call wot3 ( intfx,  0,-ism,-nintfx, ・・) |
| | | 修正後 | call wot3 ( intfx, izero,-ism,-nintfx, ・・) |
| | seqio | 修正前 | call dopc( -1, nt, 0, jerr,  0,0,0,0) |
| | | 修正後 | call dopc( -1, nt, 0, jerr,  izero,0,0,0) |

Table 2.10 Improvements of argument in dort (actual argument is integer constant or integer variable, and dummy argument is integer array) (2/2)

| コード名 | ｻﾌﾞﾙｰﾁﾝ名 | CALL文 | |
|---|---|---|---|
| | | 修正前 | call dopc( 3, nt, nt, jerr, 0,0,0,0) |
| | | 修正後 | call dopc( 3, nt, nt, jerr, izero,0,0,0) |
| | sorsum | 修正前 | call wot3 (flum(il1j+1), 0, lms, -ims, ・・) |
| | | 修正後 | call wot3 (flum(il1j+1),izero, lms, -ims, ・・) |
| | | 修正前 | call wot3 (sinm, 0, lms, -ims, 4h i=, ・・) |
| | | 修正後 | call wot3 (sinm, izero, lms, -ims, 4h i=, ・・) |
| | | 修正前 | call wot3 (sorm, 0, lms, -ims, 4h i=, ・・) |
| | | 修正後 | call wot3 (sorm, izero, lms, -ims, 4h i=, ・・) |
| | vario | 修正前 | call blkio( ii, lun, 0,0,0, ld, 0,0,0 ) |
| | | 修正後 | call blkio( ii, lun, izero,0,0, ld, 0,0,0 ) |
| | | 修正前 | call blkio( ii, lun, 0,0,0, ld, nrecmx, ・・) |
| | | 修正後 | call blkio( ii, lun, izero,0,0, ld, nrecmx, ・・) |
| | | 修正前 | call blkio (ii,lun, iary, ixx, irec, ・・) |
| | | 修正後 | call blkio (ii,lun, iww , ixx, irec, ・・) |
| | | 修正前 | call blkio( ii, lun, ibuf, nwds, irec, ・・) |
| | | 修正後 | call blkio( ii, lun, iww , nwds, irec, ・・) |
| | | 修正前 | call blkio( ii, lun,ibuf, nwds, irec, ・・) |
| | | 修正後 | call blkio( ii, lun, iww, nwds, irec, ・・) |

Table 2.11 Improvements of argument in tort (actual argument is integer constant or integer variable, and dummy argument is integer array) (1/2)

| コード名 | ｻﾌﾞﾙｰﾁﾝ名 | CALL文 | |
|---|---|---|---|
| tort | inpa | 修正前 | call ndxr(ld(1,libis),ism,ld(1,limbis),0, ・・) |
| | | 修正後 | call ndxr(ld(1,libis),ism,ld(1,limbis),izero, ・・ |
| | | 修正前 | call ndxr (ld(1,ljbjs),jsm,ld(1,ljmbjs),0 , ・・) |
| | | 修正後 | call ndxr (ld(1,ljbjs),jsm,ld(1,ljmbjs),izero, ・・ |
| | | 修正前 | call ndxr (ld(1,lmbms),msm,ld(1,lmmbms),0, ・・) |
| | | 修正後 | call ndxr (ld(1,lmbms),msm,ld(1,lmmbms),izero, ・・ |
| | | 修正前 | call ndxr( d(liljbg), igm, 0, 0, ・・) |
| | | 修正後 | call ndxr( d(liljbg), igm, izero, izero, ・・) |
| | macmx | 修正前 | call wot4( nou, sig, 0, 0, 0, ・・) |
| | | 修正後 | call wot4( nou, sig, izero, 0, 0, ・・) |
| | meshj | 修正前 | call wotv( npr, fjo, 0, 0,'Ofjo before・・) |
| | | 修正後 | call wotv( npr, fjo, izero,izero,'Ofjo before ・・ |
| | | 修正前 | call wotv( npr, fjo, 0, 0, 'Ofjo after・・) |
| | | 修正後 | call wotv( npr, fjo, izero,izero, 'Ofjo after ・・ |
| | meshk | 修正前 | call wotv( npr, fko, 0, 0, 'Ofko before ・・) |
| | | 修正後 | call wotv( npr, fko, izero,izero, 'Ofko before・・ |
| | quad | 修正前 | call edit (snpl(i9), 0, mms,1-lms,4hmom.,・・) |
| | | 修正後 | call edit (snpl(i9),izero, mms,1-lms,4hmom.,・・) |
| | refog | 修正前 | call wotv( npr, d(lfres), ibjk, jbk, 0, ・・) |
| | | 修正後 | call wotv( npr, d(lfres), ibjk, jbk, izero, ・・) |
| | respns | 修正前 | call wot4( nou, znres, 0,・・) |
| | | 修正後 | call wot4( nou, znres,izero,・・) |
| | | 修正前 | call wot4( nou, znres, 0,・・) |
| | | 修正後 | call wot4( nou, znres,izero,・・) |
| | | 修正前 | call wot4( nou, fkres, 0, ・・) |
| | | 修正後 | call wot4( nou, fkres, izero, ・・) |
| | sbsfix | 修正前 | call wot4( nou, fii, 0, 'Ofii',4, mma, ・) |
| | | 修正後 | call wot4( nou, fii,izero, 'Ofii',4, mma, ・) |

Table 2.11 Improvements of argument in tort (actual argument is integer constant or integer variable, and dummy argument is integer array) (2/2)

| | | | |
|---|---|---|---|
| | | 修正前 | call wot4( nou, fio, 0, 'Ofio',4, mma, ・・) |
| | | 修正後 | call wot4( nou, fio,izero, 'Ofio',4, mma, ・・) |
| | | 修正前 | call wot4( nou, fji, 0, 'Ofji',4, mma, ・・) |
| | | 修正後 | call wot4( nou, fji,izero, 'Ofji',4, mma, ・・) |
| | | 修正前 | call wot4( nou, fjo, 0, 'Ofjo',4, mma, ・・) |
| | | 修正後 | call wot4( nou, fjo,izero, 'Ofjo',4, mma, ・・) |
| | | 修正前 | call wot4( nou, fko, 0, 'Ofko',4,・・) |
| | | 修正後 | call wot4( nou, fko,izero, 'Ofko',4,・・) |
| | sdisin | 修正前 | call wot4( npr,shapm(i8), 0,'Odistri ・・) |
| | | 修正後 | call wot4( npr,shapm(i8),izero,'Odistri ・・) |
| | seqio | 修正前 | call dopc( -1, nt, 0, jerr, 0,0,0,0) |
| | | 修正後 | call dopc( -1, nt, 0, jerr, izero,0,0,0) |
| | | 修正前 | call dopc( 3, nt, nt, jerr, 0,0,0,0) |
| | | 修正後 | call dopc( 3, nt, nt, jerr, izero,0,0,0) |
| | source | 修正前 | call wotv( npr, flij, ibjk, jbk, 0, ・・) |
| | | 修正後 | call wotv( npr, flij, ibjk, jbk, izero, ・・) |
| | vario | 修正前 | call blkio (ii,lun, iary, ixx, irec, ld,・・) |
| | | 修正後 | call blkio (ii,lun, iww , ixx, irec, ld,・・) |
| | | 修正前 | call blkio( ii, lun, ibuf, nwds, irec,・・) |
| | | 修正後 | call blkio( ii, lun, iww , nwds, irec,・・) |
| | | 修正前 | call blkio( ii, lun,ibuf, nwds, irec, ・・) |
| | | 修正後 | call blkio( ii, lun, iww, nwds, irec, ・・) |
| | | 修正前 | call blkio( ii, lun, 0,0,0, ld, 0,0,0 ) |
| | | 修正後 | call blkio( ii, lun, izero,0,0, ld, 0,0,0 ) |
| | | 修正前 | call blkio( ii, lun, 0,0,0, ld, nrecmx, ・・) |
| | | 修正後 | call blkio( ii, lun, izero,0,0, ld, nrecmx, ・・) |
| | wrdir | 修正前 | call wot4( nou, dirf, 0, ' ', 0, nip, ・・) |
| | | 修正後 | call wot4( nou, dirf, izero, ' ', 0, nip, ・・) |

Table 2.12 Values of "irow" and "imode" on CRAY and Sun version

| imode | CRAY版 | Sun 版 |
|---|---|---|
| | irow | irow |
| ≦1 | 1 | 1 |
| 2 | 3 or 4 | 3 or 4 |
| 3 | 2 or 4 | 3 or 4 |
| 4 | 6 or 4 | 3 or 4 |
| 5 | 6 or 4 | 6 or 4 |
| 6 | 7 | 7 |
| 7 | 4 | 4 |
| 8 | 5 | 5 |
| 9 | 6 | 6 |

irowの値が
2の時、rowv
3の時、rowt
6の時、rowc
を呼び出す。

Table 2.13 Dynamic behaivor of original version of the dort code

| name | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ROWT | 2400000 | .2040E11 | 84.8 | .6519E11 | 85.5 | 40 | 73.4 | 3.0- 3.3 | 0 |
| WSOL | 7842 | .2168E10 | 9.0 | .2880E10 | 3.8 | 24 | 27.3 | 1.3- 1.3 | 0 |
| ROW | 30001 | .6266E09 | 2.6 | .2561E10 | 3.4 | 42 | 80.7 | 3.8- 4.3 | 0 |
| PLANE | 158 | .1835E09 | 0.8 | .2465E09 | 0.3 | 105 | 26.4 | 1.3- 1.3 | 0 |
| FLUX | 12 | .1520E09 | 0.6 | .2762E09 | 0.4 | 4 | 73.0 | 1.2- 2.2 | 0 |
| SOURCE | 1 | .1307E09 | 0.5 | .2379E09 | 0.3 | 5 | 70.8 | 1.2- 2.2 | 0 |
| CMSCLR | 157 | .1277E09 | 0.5 | .2229E09 | 0.3 | 84 | 44.0 | 1.7- 1.8 | 0 |
| REBAL | 157 | 69004493 | 0.3 | .3167E09 | 0.4 | 26 | 85.3 | 4.1- 4.9 | 0 |
| VARIO | 244932 | 42092621 | 0.2 | 40495935 | 0.1 | 1 | 3.0 | 0.9- 1.0 | 0 |
| CRED | 150506 | 41888580 | 0.2 | .1727E10 | 2.3 | 442 | 99.8 | 35.8- 46.5 | 0 |
| >CRIT | 93682 | --- | | --- | | --- | --- | --- | --- |
| WWESOL | 157 | 39833138 | 0.2 | .1816E10 | 2.4 | 2964 | 99.9 | 39.4- 53.0 | 0 |
| SORSUM | 1 | 35798297 | 0.1 | 35801885 | 0.0 | 43 | 0.0 | 1.0- 1.0 | 0 |
| TIMSUM | 1 | 17846364 | 0.1 | 17846725 | 0.0 | 2 | 0.0 | 1.0- 1.0 | 0 |
| >TIMON | 55598 | --- | | --- | | --- | --- | --- | --- |
| >TIMOFF | 55593 | --- | | --- | | --- | --- | --- | --- |
| CLEARX | 148471 | 14426091 | 0.1 | .5639E09 | 0.7 | 379 | 99.6 | 34.2- 43.8 | 0 |
| BTFLO | 30000 | 11172202 | 0.0 | .1152E09 | 0.2 | 48 | 95.2 | 9.2- 11.5 | 0 |
| REED | 22355 | 1742010 | 0.0 | 7603236 | 0.0 | 26 | 78.8 | 4.3- 4.4 | 0 |
| >RITE | 1445 | --- | | --- | | --- | --- | --- | --- |
| SEQIO | 23798 | 1473884 | 0.0 | 1473884 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| CMOVX | 16443 | 891809 | 0.0 | 6710117 | 0.0 | 22 | 92.6 | 6.5- 8.3 | 0 |
| CSETDM | 316 | 670042 | 0.0 | 11750460 | 0.0 | 77 | 99.4 | 14.3- 21.6 | 0 |
| SFLXIN | 1 | 623924 | 0.0 | 623924 | 0.0 | 95 | 0.0 | 1.0- 1.0 | 0 |

Table 2.14 Loop-list of original subroutine rowt

vectorize - loop list ------- ROWT ----------

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000001-00000352 | ----- | total | | 2400000 | .1615E11 | 79.2 | .1615E11 | 24.8 | 1 | 0.0 | 1.0- 1.0 |
| 00000061-00000062 | 1520 | do | V | 48000000 | .1551E10 | 7.6 | .2419E11 | 37.1 | 42 | 100.0 | 11.7- 19.5 |
| 00000197-00000198 | 4210 | do | V | 43200000 | .1396E10 | 6.8 | .2177E11 | 33.4 | 42 | 100.0 | 11.7- 19.5 |
| 00000058-00000063 | 1530 | do | S | 2400000 | .6240E09 | 3.1 | .6240E09 | 1.0 | 20 | 0.0 | 1.0- 1.0 |
| 00000194-00000199 | 4220 | do | S | 2160000 | .5616E09 | 2.8 | .5616E09 | 0.9 | 20 | 0.0 | 1.0- 1.0 |
| 00000185-00000189 | 4200 | do | V | 2160000 | 75600000 | 0.4 | .1179E10 | 1.8 | 42 | 100.0 | 11.7- 19.5 |
| 00000054-00000054 | 1510 | do | V | 2400000 | 45230769 | 0.2 | .7056E09 | 1.1 | 42 | 100.0 | 11.7- 19.5 |
| 00000068-00000070 | 1540 | do | V | 0 | 0 | 0.0 | 0 | 0.0 | 0 | 0 | 0 |
| 00000179-00000179 | 4080 | do | V | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 |
| 00000311-00000316 | 7060 | do | V | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 |
| 00000320-00000321 | 7070 | do | V | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 |
| 00000326-00000326 | 7110 | do | V | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 |

Table 2.15 Loop-list of original subroutine wsol

vectorize - loop list ------- WSOL ----------

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000122-00000122 | 270 | do | S | 1323560 | .6293E09 | 29.0 | .6293E09 | 21.9 | 23 | 0.0 | 1.0- 1.0 |
| 00000136-00000137 | 320 | do | S | 1277804 | .5665E09 | 26.1 | .5665E09 | 19.7 | 23 | 0.0 | 1.0- 1.0 |
| 00000167-00000167 | 420 | do | S | 1277804 | .5069E09 | 23.4 | .5069E09 | 17.6 | 23 | 0.0 | 1.0- 1.0 |
| 00000085-00000093 | 220 | do | S | 35953 | .1017E09 | 4.7 | .1017E09 | 3.5 | 51 | 0.0 | 1.0- 1.0 |
| 00000105-00000124 | 280 | do | S | 15684 | 90002080 | 4.2 | 90002080 | 3.1 | 84 | 0.0 | 1.0- 1.0 |
| 00000185-00000188 | 540 | do | V | 1323560 | 75215337 | 3.5 | .7822E09 | 27.2 | 24 | 100.0 | 7.8- 13.0 |
| 00000160-00000177 | 480 | do | S | 15684 | 52315040 | 2.4 | 52315040 | 1.8 | 83 | 0.0 | 1.0- 1.0 |
| 00000129-00000147 | 380 | do | S | 15684 | 51007164 | 2.4 | 51007164 | 1.8 | 83 | 0.0 | 1.0- 1.0 |
| 00000181-00000190 | 542 | do | S | 15684 | 43677480 | 2.0 | 43677480 | 1.5 | 84 | 0.0 | 1.0- 1.0 |
| 00000141-00000146 | 350 | do | S | 30072 | 25882738 | 1.2 | 25882738 | 0.9 | 35 | 0.0 | 1.0- 1.0 |
| 00000171-00000176 | 450 | do | S | 30072 | 22401376 | 1.0 | 22401376 | 0.8 | 35 | 0.0 | 1.0- 1.0 |
| 00000080-00000094 | 230 | do | M | 471 | 1053699 | 0.0 | 1258355 | 0.0 | 76 | 17.1 | 1.2- 1.2 |
| 00000102-00000193 | 800 | do | S | 7842 | 846936 | 0.0 | 846936 | 0.0 | 2 | 0.0 | 1.0- 1.0 |
| 00000154-00000156 | 410 | do | V | 15684 | 437776 | 0.0 | 4539714 | 0.2 | 26 | 100.0 | 7.8- 13.0 |
| 00000069-00000077 | 210 | do | V | 471 | 416954 | 0.0 | 1024307 | 0.0 | 60 | 63.4 | 2.4- 2.5 |
| 00000001-00000195 | ----- | total | | 7842 | 278874 | 0.0 | 278874 | 0.0 | 1 | 0.0 | 1.0- 1.0 |

Table 2.16 Vectorized subroutines

| コード名 | ベクトル化したサブルーチン |
|---|---|
| dort | csolswp, rowb, rowl, rown, rowv, rwoz, sorx |
| tort | csolswp, rown, rownv, rowy, rowsv |

Table 2.17 Dynamic behaivor of vector version of the dort code

| name | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ROWT | 2400000 | .1605E11 | 88.3 | .7672E11 | 87.4 | 40 | 84.5 | 4.4- 5.0 | 0 |
| WSOL | 7842 | .6290E09 | 3.5 | .2880E10 | 3.3 | 24 | 86.5 | 4.1- 5.0 | 0 |
| ROW | 30001 | .6266E09 | 3.4 | .2561E10 | 2.9 | 42 | 80.7 | 3.8- 4.3 | 0 |
| PLANE | 158 | .1835E09 | 1.0 | .2465E09 | 0.3 | 105 | 26.4 | 1.3- 1.3 | 0 |
| FLUX | 12 | .1520E09 | 0.8 | .2762E09 | 0.3 | 4 | 73.0 | 1.2- 2.2 | 0 |
| SOURCE | 1 | .1307E09 | 0.7 | .2379E09 | 0.3 | 5 | 70.8 | 1.2- 2.2 | 0 |
| CMSCLR | 157 | .1277E09 | 0.7 | .2229E09 | 0.3 | 84 | 44.0 | 1.7- 1.8 | 0 |
| REBAL | 157 | 69004493 | 0.4 | .3167E09 | 0.4 | 26 | 85.3 | 4.1- 4.9 | 0 |
| VARIO | 244932 | 42092621 | 0.2 | 40495935 | 0.0 | 1 | 3.0 | 0.9- 1.0 | 0 |
| CRED | 150506 | 41888580 | 0.2 | .1727E10 | 2.0 | 442 | 99.8 | 35.8- 46.5 | 0 |
| >CRIT | 93682 | --- | | --- | | --- | --- | --- | --- |
| WWESOL | 157 | 39833138 | 0.2 | .1816E10 | 2.1 | 2964 | 99.9 | 39.4- 53.0 | 0 |
| SORSUM | 1 | 35798297 | 0.2 | 35801885 | 0.0 | 43 | 0.0 | 1.0- 1.0 | 0 |
| TIMSUM | 1 | 17846364 | 0.1 | 17846725 | 0.0 | 2 | 0.0 | 1.0- 1.0 | 0 |
| >TIMON | 55598 | --- | | --- | | --- | --- | --- | --- |
| >TIMOFF | 55593 | --- | | --- | | --- | --- | --- | --- |
| CLEARX | 148471 | 14426091 | 0.1 | .5639E09 | 0.6 | 379 | 99.6 | 34.2- 43.8 | 0 |
| BTFLO | 30000 | 11172202 | 0.1 | .1152E09 | 0.1 | 48 | 95.2 | 9.2- 11.5 | 0 |
| REED | 22355 | 1742010 | 0.0 | 7603236 | 0.0 | 26 | 78.8 | 4.3- 4.4 | 0 |
| >RITE | 1445 | --- | | --- | | --- | --- | --- | --- |
| SEQIO | 23798 | 1473884 | 0.0 | 1473884 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| CMOVX | 16443 | 891809 | 0.0 | 6710117 | 0.0 | 22 | 92.6 | 6.5- 8.3 | 0 |
| CSETDM | 316 | 670042 | 0.0 | 11750460 | 0.0 | 77 | 99.4 | 14.3- 21.6 | 0 |
| SFLXIN | 1 | 623924 | 0.0 | 623924 | 0.0 | 95 | 0.0 | 1.0- 1.0 | 0 |

Table 2.18 Comparison between original version and vector version

| 入力データ | 時間モード | オリジナル版 | ベクトル化版 |
|---|---|---|---|
| j42.jffnj | CPU時間 | 2h 19m 38.32s | 1h 5m 49.41s |
| | VU 時間 | 11m 00.85s | 19m 27.44s |
| | 倍率 | 1.0 | 2.2 |
| j175.jffnj | CPU時間 | 4h 10m 39.68s | 2h 9m 41.04s |
| | VU 時間 | 22m 58.24s | 37m 50.69s |
| | 倍率 | 1.0 | 1.9 |

※コンパイルは共にVPP FORTRAN77/VP OPT(E) で行った。

```
setenv TARGET cray-c90     ⇦
mv d2p8p14.id dorus
head -180 dorus > ufirst
csh -x ufirst dort cray cflag
csh -x jcldor dos
csh -x jcldor dort
```

Fig. 2.1 Installation method of dort, gip, and grtuncl codes

```
csh -x jcldor visa
```

Fig. 2.2 Installation method of visa, and torsed codes

```
mv t2p8p14.id torus
csh -x ufirst tort cray cflag
csh -x jcltor tort
```

Fig. 2.3 Installation method of tort code

```
                .
                .
    setenv parmc '-V -A fast -e mxz -o aggress'  # -o off
                .
    cft77 $parmc -l bb.tor.lst -b bb.tor.o bb.tor.f
                .
                .
```

⇩

```
                .
                .
   #setenv parmc '-V -A fast -e mxz -o aggress'  # -o off
    setenv parmc '-V -A fast -e mx -o aggress'  # -o off
                .
    cft77 $parmc -l bb.tor.lst -b bb.tor.o bb.tor.f
                .
                .
```

Fig. 2.4 Modification of compiler option for t2p8p14.id

```
#!/bin/csh
#@$-eo
#@$-q cr004nss
#@$-lt 0:10:00
cd $QSUB_WORKDIR
../src/dort < ../input/dort/probpd.F >& ../output/dort/probpd.outlist
```

Fig. 2.5 Example of shell-script for execution on CRAY-T94

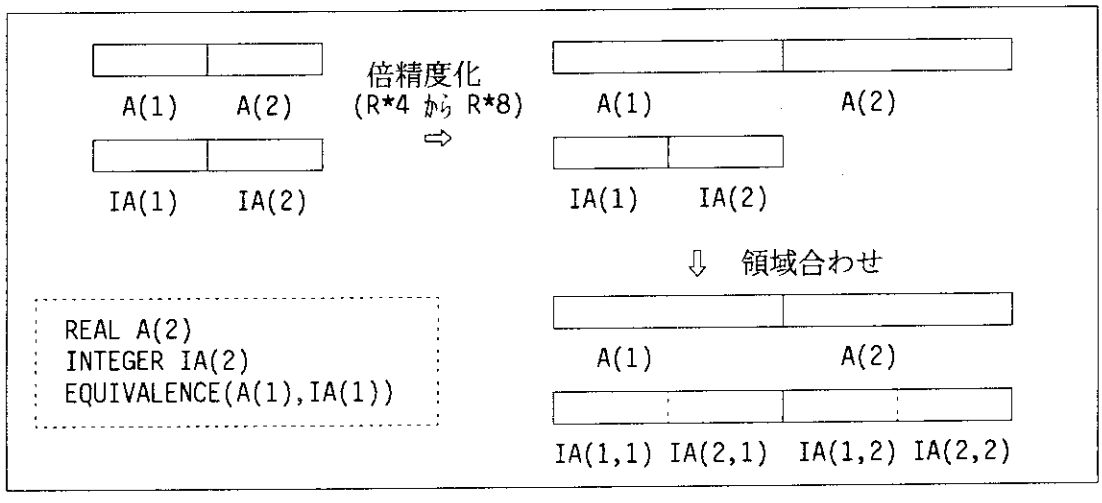


Fig. 2.6 Concept of change into the double precision

```
・ACTVY
  *STREAM77G ADJUST NEDUM,IMODD,LOCP,NBFNT,NBFII,LOCE,LOCI,NBFID,LOCV
  *STREAM77G ADJUST LOCF,LOCL,LOCDM1,LOCDM2,LOCDM3,LOCDM4,LOCDM5,LDUML
  *STREAM77G ADJUST NBFRI
  *STREAM77L ADJUST ICMAT,ICPOS,IJZN,IZMT,MIXT,NUCL,IZNRG
・BALCAL
  *STREAM77L ADJUST IZNRG,IZMT,IBJ,IBIS,ISET,IJZN
・BTFLO
  *STREAM77L ADJUST ISET,MSETI,MBMS,MUNP,IBJ,IJZN,IZNRG,IBIS,MMDN
・CMSCLR
  *STREAM77L ADJUST ICBI,JCBJ,ISET,IBIS,MBMS,ILJBG,IJZN,IZMT,IBJ,
・COEF
  *STREAM77L ADJUST IZN,IMT,IZNRG,NSIG
・FLUX
  *STREAM77L ADJUST IBJ,IBIS,IJZN,ILJBG,ISET,IZMT,NSIG,NMBIG
・FLXPRT
  *STREAM77L ADJUST NMBIG,IBJ,ILJBG
・FSCON
  *STREAM77L ADJUST ISET,IBJ,ILJBG,IJZN,IZMT,NMBIG,IBIS
・GEOM
  *STREAM77L ADJUST IBIS,IZMT,ICBI,JCBJ, JSZBJ, ISZBI
・GRDCOR
  *STREAM77L ADJUST ICBI,JCBJ,ISET,IBIS,MBMS,ILJBG,IJZN,IZMT,IBJ
・GRDDIF
  *STREAM77L ADJUST ICBI,JCBJ,ISET,IBIS,MBMS,ILJBG,IJZN,IZMT,IBJ
・GRDSCL
  *STREAM77L ADJUST ICBI,JCBJ,ISET,IBIS,MBMS,ILJBG,IJZN,IZMT,IBJ
・INCHK
  *STREAM77L ADJUST LD,LOCP
・INPA
  *STREAM77L ADJUST LD,IBIS
・ISODIF
  *STREAM77L ADJUST IJZN,IZMT,ITMBG,IZNRG,MBMS,IBJ,ILJBG,NSIG
・JEOM
  *STREAM77L ADJUST IBIS,IZMT,ICBI,JCBJ,JSZBJ,ISZBI
・MACMX
  *STREAM77L ADJUST MIXT,NUCL,MATL,NSIG
・MAPR
  *STREAM77L ADJUST IJZN,IZMT,IBJ
・MAPX
  *STREAM77L ADJUST M1,M2,IBJ,ISET,IBIS
・MAP3
  *STREAM77L ADJUST M1,M2,JL,JU,IGZN,ISET,IBIS
・MAXI
  *STREAM77L ADJUST K
・MAXIJ
  *STREAM77L ADJUST K,L,M
・MESH
  *STREAM77L ADJUST ILFRT, MUNP, MBMS, MSOMS
・MESHR
  *STREAM77L ADJUST MBMS, ISZNG, JSZBJ, ISZBI, IBIS, IJGSZ, MSOMS, MMDN, MSETI
・MINSA
  *STREAM77L ADJUST K
・MSUM
  *STREAM77L ADJUST K
・MSUMD
  *STREAM77L ADJUST K,L
```

Fig. 2.7 Conversion control lines in bb.dor.f (1/2)

```
・NDXR
  *STREAM77L ADJUST IPVT,IA1,IA2
・NNZRO
  *STREAM77L ADJUST K
・NQUADR
  *STREAM77L ADJUST MSOMS,IJGSZ
・PLANE
  *STREAM77L ADJUST IBJ,IBIS,ICBI,ILJBG,ISET,JCBJ,MBMS,MMDN,JNTFX
  *STREAM77L ADJUST MTETA,MTLVL,JNTSR,IJZN,IZMT,IZNRG
・QUAD
  *STREAM77L ADJUST MBMS,MMDN,MTEMU,MTETA,MTLVL
  *STREAM77L ADJUST NMBIG,ISCTG,ISNPN,MSOMS,ILFRT,MUNP
・REBAL
  *STREAM77L ADJUST ICBI,JCBJ,ISET, IBIS,MBMS,IBJ,ILJBG,KEYAJ,KEYAI
・ROW
  *STREAM77L ADJUST INTFX,MUMATE,INTSR,MSOMS,LFRT,MBMS,MSETI,MUNP
・SBABDO
  *STREAM77L ADJUST ISET,IBIS,MBMS
・SBSRIN
  *STREAM77L ADJUST ISET,IBJ,IBIS,MBMS,ICBI, JCBJ,IZNRG,INTSR,JNTSR
  *STREAM77L ADJUST INTFX,JNTFX,IJZN
・SDISIN
  *STREAM77L ADJUST IBJ,NMBIG,ILJBG
・SDISO
  *STREAM77L ADJUST ISET,IBIS,IBJ,ILJBG,IJZN,IZMT,NSIG,NMBIG
・SFLXIN
  *STREAM77L ADJUST IBJ,NMBIG,ILJBG
・SORSOR
  *STREAM77L ADJUST ISET,IBJ,ILJBG,NMBIG
・SORSUM
  *STREAM77L ADJUST ISET,IBJ,ILJBG,IJZN,IZMT,NMBIG,IBIS
・SORT
  *STREAM77L ADJUST IX
・SORX
  *STREAM77L ADJUST INDX,MBMS,MSETI,MMDN,IVM,IV1
・SOURCE
  *STREAM77L ADJUST IBJ,IJZN,IZMT,JBLK,NSIG,ILJBG,NMBIG,ITHYG,IBIS
  *STREAM77L ADJUST ISET,ITMBG,IZNRG
・SWEEP
  *STREAM77L ADJUST IZN,JMG,NZR
・SWEEQ
  *STREAM77L ADJUST IZN,JMG,NZR
・VLCAL
  *STREAM77L ADJUST IBJ,IBIS,ISET
・WRBNDI
  *STREAM77L ADJUST IC
・WRFOG
  *STREAM77L ADJUST IMBIS, ISET,NMBIG,LD
・WRSCL
  *STREAM77L ADJUST IZMT,IZNRG,IBIS,IMBIS,ISET,IC
・WSOL
  *STREAM77L ADJUST IFCNZ
・WWESOL
  *STREAM77L ADJUST IFCNZ
・ZLEAK
  *STREAM77L ADJUST IZN,NZR,
```

Fig. 2.7 Conversion control lines in bb.dor.f (2/2)

```
・AMINAF
  *STREAM77G ADJUST LOCP,LUNDUM,IDU262,IDU632,IDU641,LOCE,LOCI,IXXX
  *STREAM77G ADJUST LOCV,LOCF,NBFLX,NEDUM,IMODD,LOCL,LUNIT,LOCDM1
  *STREAM77G ADJUST LOCDM3,LOCDM4,LOCDM5,LOCDM6
  *STREAM77G ADJUST LD,IBIS,JBJS,IJBK,JMBJS,JSET,IBJK,JBK,NSCTG,LMSBG
  *STREAM77G ADJUST MBMS,MMDN,MTEMU,MTETA,MTLVL,NMBIG,NSCTG,ISNPN,ILFRT
  *STREAM77G ADJUST MUNP,MTXZI,MMDU,MSET,ICBI,JCBJ,JSZBJ,ISZBI
  *STREAM77G ADJUST KCBK,KSZBK,ISET,JSET,KEYAI,KEYAJ,KEYAK
  *STREAM77G ADJUST IJZN,IBJ,MSAV,MAP,N1,N2,IL,IU,JL,JU,KL,KU
  *STREAM77G ADJUST IZNRG,ILJBG,IZMT,MMBMS,NSIG,ITHYG,ITMBG
  *STREAM77G ADJUST MIXT,NUCL,MATL,KNTSR,KNTFX,INDX,IVM,IV1,LFRT
  *STREAM77G ADJUST INTFX,MUMATE,INTSR,IMBIS,IC,IACOP,JNTFX,JNTSR
・CINC
  *STREAM77L ADJUST I
・CMPRI
  *STREAM77L ADJUST I,J
```

Fig. 2.8 Conversion control lines in bb.tor.f

```
・GIP
  *STREAM77G ADJUST LOCL
・TAPE
  *STREAM77L ADJUST MB,MC,MID,MIDS,MCO,MCOS
```

Fig. 2.9 Conversion control lines in bb.gip.f

```
・ALLOT
  *STREAM77G ADJUST LIA,LIL,LOA,LWA,LCA,LOCP,NBFNT,NBFII,LOCE,LOCI,NBFID
  *STREAM77G ADJUST NBFID,LOCF,NBFLX,NEDUM,IMODD,LOCL,LOCDM1,LOCDM2
  *STREAM77G ADJUST LOCDM3,LOCDM4,LOCDM5,LDUML,LDTK,ISET,JSET
  *STREAM77G ADJUST M0,M2,MB,MC,MTT
```

Fig. 2.10 Conversion control lines in bb.unc.f

```
・CSETX
 *STREAM77L ADJUST K
・ DTFIN
 *STREAM77L ADJUST LDTK
・ VARIO
 *STREAM77L ADJUST LD
```

Fig. 2.11 Conversion control lines in bb.lib.f

```
 TWC(6 4 8 0 8) SRP
//*********************************************************************
//*               STREAM77  ( C-ENGINE )                              *
//*********************************************************************
//JOBPROC DD DSN=J9127.PROCLIB.CNTL,DISP=SHR
//CENG     EXEC CENG,REGION.STRM77=,
// A='CONV,READ,NOT,NOTAB',INC='INCLUDE'
//FT05F001 DD *
SUP(0400)
/*
//FT06F001 DD DSN=J9127.BB.UNCF.OUTLIST,DISP=(NEW,CATLG),
//         DCB=(LRECL=137,BLKSIZE=19043,RECFM=FBA,DSORG=PS),
//         SPACE=(TRK,(50,10),RLSE),UNIT=TSSWK
//FT12F001 DD DSN=J9127.BB.UNCF.FORT77,DISP=(NEW,CATLG),
//         DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB,DSORG=PO),
//         SPACE=(TRK,(30,10,30),RLSE),UNIT=D0340
//FT14F001 DD DSN=J9127.BB.UNCF.INCLUDE,DISP=(NEW,CATLG),
//         DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB,DSORG=PO),
//         SPACE=(TRK,(10,10,20),RLSE),UNIT=D0340
//SYSIN  DD DSN=J9127.BB.UNCW.FORT77,
//             DISP=SHR,LABEL=(,,,IN)
//SYSINC DD DSN=J9127.BB.UNC.INCLUDE,
//           DISP=SHR,LABEL=(,,,IN)
//SYSPRINT DD SYSOUT=*
//
```

Fig. 2.12 JCL for STREAM77

```
program gip
                    .
common/bulkbu/locl(1), liml, lint, lenint, lmco,   . . .
                    .
call fidos(3, nerr, nin, nou, locl, locl, locl)
                    .
end

subroutine fidos(ii2,j3,n5,n6,d,ldtk,loco)
                    .
dimension d(1), ldtk(1),loco(1)
                    .
```

Fig. 2.13 An example of original common block

```
    program gip
    implicit real*8 (a-h,o-z)
   .....................................................................
   : common/bulkbu/locl(2,36)                                          :
   : equivalence(locl(2,2),liml),(locl(2,3),lint),(locl(2,4),lenint),  :
   :1          (locl(2,5),lmco),   . . .                               :
   .....................................................................
                    .
    call fidos(3, nerr, nin, nou, locl, locl, locl)
                    .
    end

    subroutine fidos(ii2,j3,n5,n6,d,ldtk,loco)
    implicit real*8 (a-h,o-z)
    dimension d(1),   ldtk(2,1),   loco(2,1)
                    .
```

Fig. 2.14 An example of common block and integer Array modification

```
      subroutine tape
                .
      call wot3( crx(1,i04),0,i05,-ihp, 4hpos., 4hmat.,nou
     1        , 0.d0, 0, 12h0)
                .
      end
      subroutine wot3(x,ibj,jm,imi, ti, ti,nu,km,tk,tt,n)
                .
 1800 call edit( x(ijk+1),ibj,jm,im,ti, tj,nu)
                .
      end
      subroutine edit(x,ifj,nxj,nxi,id1,  id2, nou)
                .
      write(nou,9010)   id1, (id2, j,j=i1,i2)
                .
 9010 format('0',1x,a4,3x,a4,i3,7(6x,a4,i3) )
                .
      end
```

Fig. 2.15 Original version of subroutine tape, wot3, and edit

```
      subroutine tape
      implicit real*8 (a-h,o-z)
                .
      call wot3( crx(1,i04),0,i05,-ihp,4hpos.,4hmat.,nou
     1        , 0.d0, 0, 12h0)
                .
      end
      subroutine wot3(x, ibj,jm,imi,ti,tj,nu,km,tk,tt,n)
      implicit real*8 (a-h,o-z)
      character*4 ti,tj
                .
                .
 1800 call edit( x(ijk+1),ibj,jm,im,ti,tj,nu)
               .
      end
      subroutine edit(x,ifj,nxj,nxi,id1,id2,nou)
      implicit real*8 (a-h,o-z)
      character*4 id1,id2
                .
      write(nou,9010) id1,(id2,j,j=i1,i2)
                .
 9010 format('0',1x,a4,3x,a4,i3,7(6x,a4,i3) )
                .
      end
```

Fig. 2.16 Double precision version of subroutine tape, wot3, and edit

```
      subroutine tape
              •
      call seqio (2, nt4,  777, 1, 0)
              •
      end
      subroutine seqio(io, nt,a, m, irec)
      dimension a(1)
              •
      end
```

Fig. 2.17 Original version of subroutine tape and seqio

```
      subroutine tape
      implicit real*8 (a-h,o-z)
      dimension iww(2)
      equivalence (iww(1),ww)
              •
      iww(2) = 777
      call seqio (2, nt4, ww, 1, 0)
              •
      end
      subroutine seqio(io, nt, a, m, irec)
      implicit real*8 (a-h,o-z)
      dimension a(1)
              •
      end
```

Fig. 2.18 Double precision version of subroutine tape and seqio

```
      subroutine vario

              •
      call blkio(ii,lun,iary,ixx,irec,ld,nrecmx,nwblkm,ipax)
              •
      end
      subroutine blkio(ii,nt,l,m,irec,ld,nrecmx,nwblki,lenf)
      dimension ld(1),l(1)
              •
      end
```

Fig. 2.19 Original version of subroutine vario and blkio

```
      subroutine vario
      implicit real*8 (a-h,o-z)
      dimension iww(2),izero(2)
      data  iww/0,0/, izero/0,0/
                  •
      iww(2) = iary
      call blkio(ii,lun,iww ,ixx,irec,ld,nrecmx,nwblkm,ipax)
                  •
      call blkio(ii,lun,izero,0,0,ld,nrecmx,nwblkm,ipax)
                  •
      end
      subroutine blkio(ii,nt,l,m,irec,ld,nrecmx,nwblki,lenf)
      dimension ld(2,1),l(2,1)
                  •
      end
```

Fig. 2.20 Double precision version of subroutine vario and blkio

```
      cpu = second(x)
      chg = csecond(x)
      if(chg.eq.0) chg = cpu
      if(i.ge.1 .and. i.le.2 .and. xtime.eq.timh) go to 200
      wal = -1.
      wal = timef(x)*1.d-3
```

⇩

```
cvpp  cpu = second(x)
      call clock(cpu,0,2)
cvpp> chg = csecond(x)
cvpp  if(chg.eq.0) chg = cpu
      chg = cpu
cvpp<
      if(i.ge.1 .and. i.le.2 .and. xtime.eq.timh) go to 200
c##   wal = -1.
      wal = -1.d0
cvpp  wal = timef(x)*1.d-3
      call time(ix)
      wal = real(ix)
```

Fig. 2.21 Improvement of service routine

```
      subroutine mapr ( ijzn, izmt, ibj, jm, nout )
      dimension ijzn(1), izmt(1), ibj(1), k(120),label(2), ia(35)
     1   ia(35)
c
c - - - - - plots irregular 2-d map of zone/matl
c
      data ia(1), ia(2), ia(3), ia(4), ia(5), ia(6), ia(7), ia(8)
     1   /   1h1,   1h2,   1h3,   1h4,   1h5,   1h6,   1h7,   1h8      /
                .
                .
     1   ,ia(33),ia(34),ia(35), iblnk,      ixr, label(1), label(2)
     1   /   1hx,   1hy,   1hz,   1h ,     3hxrr,  4hzone,   4hmatl    /
      data line, ltable  / 120, 35 /
                .
                .
   25 do 11 l=m,ix
                .
      iia = mod(n-1,ltable)
   11 k(iik+1) = ia(iia+1)                          ⇦ 文字としての使用
c
      write (nout, 4) j,(k(l),l=1, lm)              ⇦ 文字としての使用
                .
                .
      do 32 lop = 1,3
      do 33 l = m,ix
      iik = l - m
   33 k(iik+1) = mod(l/110, 10)                     ⇦ 整数としての使用
      write (nout,31) ir, (k(i),i=1,lm)             ⇦ 整数としての使用
      ir = iblnk
                .
                .
   32 continue
                .
                .
    4 format ( 1x, i4, 4h... , 120a1)
   66 format ( 5h  yzt )
   31 format ( 5x, a4, 120i1)
                .
                .
      end
```

Fig. 2.22 Original subroutine mapr

```
      subroutine mapr ( ijzn, izmt, ibj, jm, nout )
          implicit  real*8 ( a-h,o-z )
          real*8          label
cvpp>>
      character*4 ia
      character*4 k
      integer ik(120)                          ⇦ 文字型配列と整数型配列の
      equivalence (k(1),ik(1))                 ⇦ 共有
      character*4 iblnk,ir,ixr
cvpp<<
c##   dimension ijzn(1), izmt(1), ibj(1), k(120),label(2), ia(35)
      dimension ijzn(2,1), izmt(2,1), ibj(2,1), k(120),label(2),
     1   ia(35)
c
c - - - - - plots irregular 2-d map of zone/matl
c
      data ia(1), ia(2), ia(3), ia(4), ia(5), ia(6), ia(7), ia(8)
     1   /   1h1,   1h2,   1h3,   1h4,   1h5,   1h6,   1h7,   1h8      /
                .
                .
     1   ,ia(33),ia(34),ia(35), iblnk,      ixr, label(1), label(2)
     1   /   1hx,   1hy,   1hz,   1h ,     3hxrr,  4hzone,   4hmatl    /
c
   25 do 11 l=m,ix
                .
      iia = mod(n-1,ltable)
   11 k(iik+1) = ia(iia+1)                          ⇦ 文字型配列の使用
c
      write (nout, 4) j,(k(l),l=1, lm)              ⇦ 文字型配列の使用
                .
                .
      do 32 lop = 1,3
      do 33 l = m,ix
      iik = l - m
cvpp3 k(iik+1) = mod(l/110, 10)
   33 ik(iik+1) = mod(l/110, 10)                    ⇦ 整数型配列の使用
cvpp  write (nout,31) ir, (k(i),i=1,lm)
      write (nout,31) ir, (ik(i),i=1,lm)            ⇦ 整数型配列の使用
      ir = iblnk
                .
                .
   32 continue
                .
    4 format ( 1x, i4, 4h... , 120a1)
   66 format ( 5h  yzt )
   31 format ( 5x, a4, 120i1)
                .
                .
      end
```

Fig. 2.23 Improvement of subroutine mapr

```
      subroutine loco
          implicit  real*8 ( a-h,o-z )
          real*8     locl1
          real*8     locp1
          real*8     locv1
          real*8     loce1
          real*8     locf1
          real*8     loci1
             .
             .
      character*18  compil
      data  compil /'comdate   comtime '/
      data locl1,locp1,locv1,loce1 / 4hlocl, 4hlocp, 4hlocv, 4hloce /
      data locf1,loci1 / 4hlocf, 4hloci /
             .
             .
c##   locl(1) = locl1
      locl(2,1)=locl1
c##   locp(1) = locp1
      locp(2,1)=locp1
c##   loce(1) = loce1
      loce(2,1)=loce1
c##   loci(1) = loci1
      loci(2,1)=loci1
c##   locv(1) = locv1
      locv(2,1)=locv1
c##   locf(1) = locf1
      locf(2,1)=locf1
             .
             .
      return
      end
```

Fig. 2.24 Original subroutine loco (STREAM77 version)

```
      subroutine loco
          implicit  real*8 ( a-h,o-z )
          character*4     locl1
          character*4     locp1
          character*4     locv1
          character*4     loce1
          character*4     locf1
          character*4     loci1
cvpp>>
      equivalence(locl1,ilocl1),(locp1,ilocp1),(locv1,ilocv1),
     1           (loce1,iloce1),(locf1,ilocf1),(loci1,ilocil)
cvpp<<
                .
                .
      character*18  compil
      data  compil /'comdate   comtime '/
      data locl1,locp1,locv1,loce1 / 4hlocl, 4hlocp, 4hlocv, 4hloce /
      data locf1,locil / 4hlocf, 4hloci /
                .
                .
c
c##   locl(1) = locl1
      locl(2,1)=ilocl1
c##   locp(1) = locp1
      locp(2,1)=ilocp1
c##   loce(1) = loce1
      loce(2,1)=iloce1
c##   loci(1) = loci1
      loci(2,1)=ilocil
c##   locv(1) = locv1
      locv(2,1)=ilocv1
c##   locf(1) = locf1
      locf(2,1)=ilocf1
                .
                .
      return
      end
```

Fig. 2.25 Improvement of subroutine loco

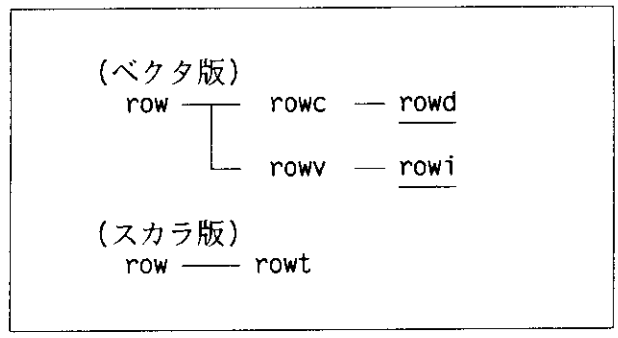


Fig. 2.26 Tree structuires of vector version and scalar version

```
      subroutine row (      dz,     imm,     lmx,     mmb,     imx,     cio,
        .
        .
      imode = mod(mode,10)
      irow = 1
        .
        .
c - - - - - alternatives
      go to (310, 310, 322, 325, 320, 355, 355, 355, 355), imode
c - - - - -  lin  scl  Owt thet  vec rown rowl rowb rowc
c  imode=      1    2    3    4    5    6    7    8    9
c
c - - - - - rowz rowv rowt rowl rowb rowc rown
c   irow=      1    2    3    4    5    6    7
c - - - - - 2= scalar weighted--rowt always
  310 irow = 3
      go to 330
c
c - - - - - 5= vector weighted--rowc always
  320 irow = 6
      go to 330
c
c - - - - - 3= zero weighted--rowv if vector; else rowt
  322 irow = 2
      pctwt = 0.d0
      go to 327
c
c - - - - - 4= theta weighted--rowc if vector; else rowt
  325 irow = 6
  327 continue
cnc
c     irow = 3
cnc
c - - - - - rowl - for all exceptional cases
  330 if(mode.ge.10 .or. ingeom.ge.10) irow = 4
      if(wmxx.ne.0.d0                ) irow = 4
      go to 360
c
c - - - - - 6=rown, 7=rowl, 8=rowb, 9=rowc
  355 irow = imode - 3
      if(imode.gt.6) go to 360
      irow = 7
      ifnodl = .true.
        .
        .
      end
```

Fig. 2.27 Original subroutine row

```
      subroutine row (      dz,     imm,     lmx,     mmb,     imx,     cio,
        .
        .
      imode = mod(mode,10)
      irow = 1
        .
        .
c - - - - - alternatives
      go to (310, 310, 322, 325, 320, 355, 355, 355, 355), imode
c - - - - -  lin  scl  Owt thet  vec rown rowl rowb rowc
c  imode=      1    2    3    4    5    6    7    8    9
c
c - - - - - rowz rowv rowt rowl rowb rowc rown
c   irow=      1    2    3    4    5    6    7
c - - - - - 2= scalar weighted--rowt always
  310 irow = 3
      go to 330
c
c - - - - - 5= vector weighted--rowc always
  320 irow = 6
      go to 330
c
c - - - - - 3= zero weighted--rowv if vector; else rowt
  322 irow = 2
      pctwt = 0.d0
      go to 327
c
c - - - - - 4= theta weighted--rowc if vector; else rowt
  325 irow = 6
  327 continue
cnc
c.tn>>
c     irow = 3
      irow = 3                ⇦  rowtの呼び出し設定
c.tn<<
cnc
c - - - - - rowl - for all exceptional cases
  330 if(mode.ge.10 .or. ingeom.ge.10) irow = 4
      if(wmxx.ne.0.d0                ) irow = 4
      go to 360
c
c - - - - - 6=rown, 7=rowl, 8=rowb, 9=rowc
  355 irow = imode - 3
      if(imode.gt.6) go to 360
      irow = 7
      ifnodl = .true.
        .
        .
      end
```

Fig. 2.28 Improvement of subroutine row

```
      subroutine reed (lun,irec,array,nwds,mode)
          implicit  real*8 ( a-h,o-z )
      dimension array(nwds)
        .
        .
c - - - - - read the record requested
   80 continue
      if(io.eq.1) read  (lun) array
      if(io.eq.2) write (lun) array
        .
        .
 9900 return
      end
```

Fig. 2.29 Original subroutine reed

```
      subroutine reed (lun,irec,array,nwds,mode)
          implicit  real*8 ( a-h,o-z )
      dimension array(nwds)
c.tn>>
      real*4 sarray(10000)
c.tn<<
        .
        .
c - - - - - read the record requested
   80 continue
c.tn>> read for real*4 data (unit=4)
c.tn  if(io.eq.1) read  (lun) array
      if(io.eq.1) then
        if(lun.eq.4) then
          read  (lun) (sarray(ii),ii=1,nwds)
          do 3000 ii = 1, nwds
             array(ii) = dble(sarray(ii))
 3000     continue
        else
          read  (lun) array
        end if
      end if
c.tn<<
      if(io.eq.2) write (lun) array
        .
        .
 9900 return
      end
```

Fig. 2.30 Improvement subroutine reed

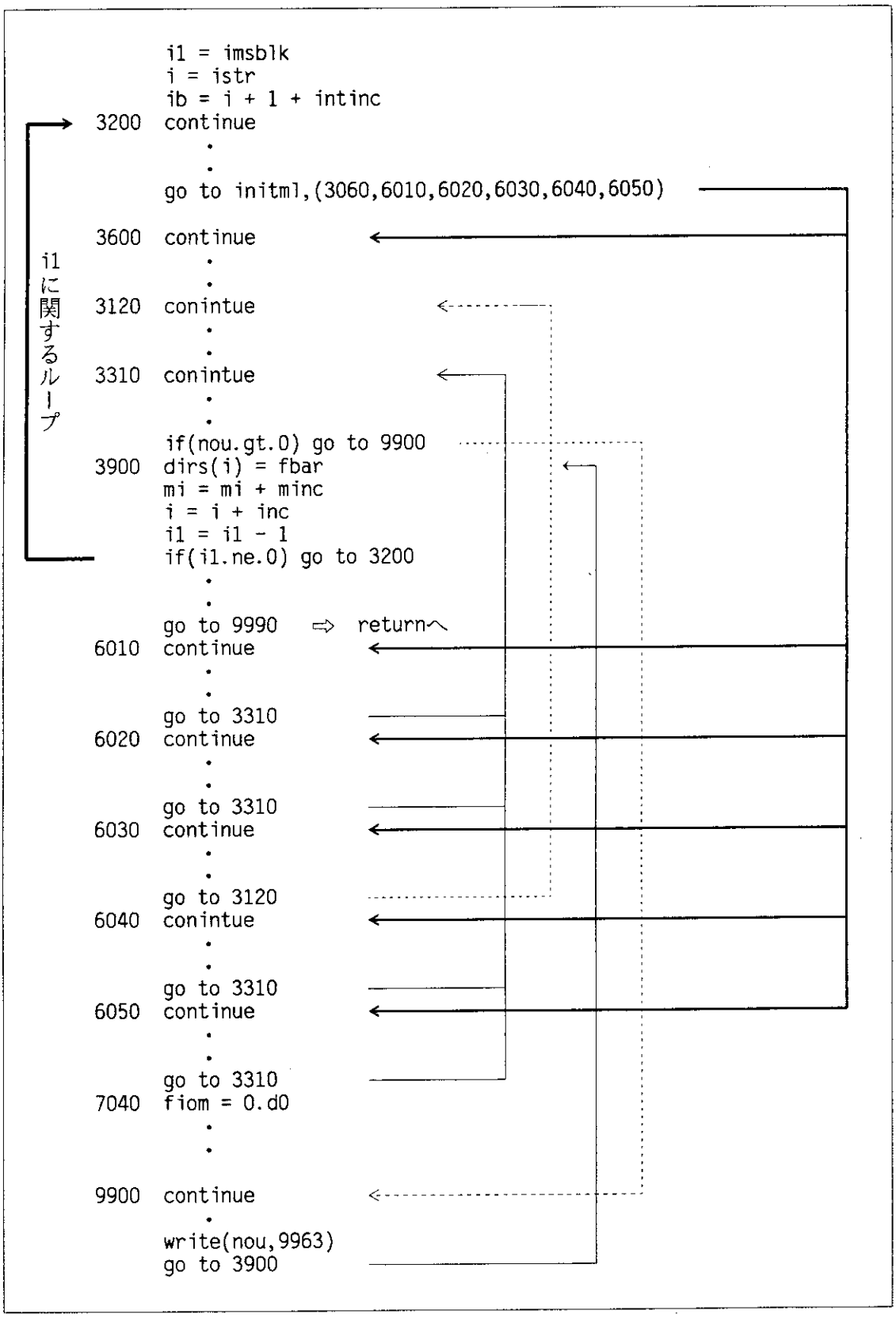


Fig. 2.31 Structure of subroutine rowt

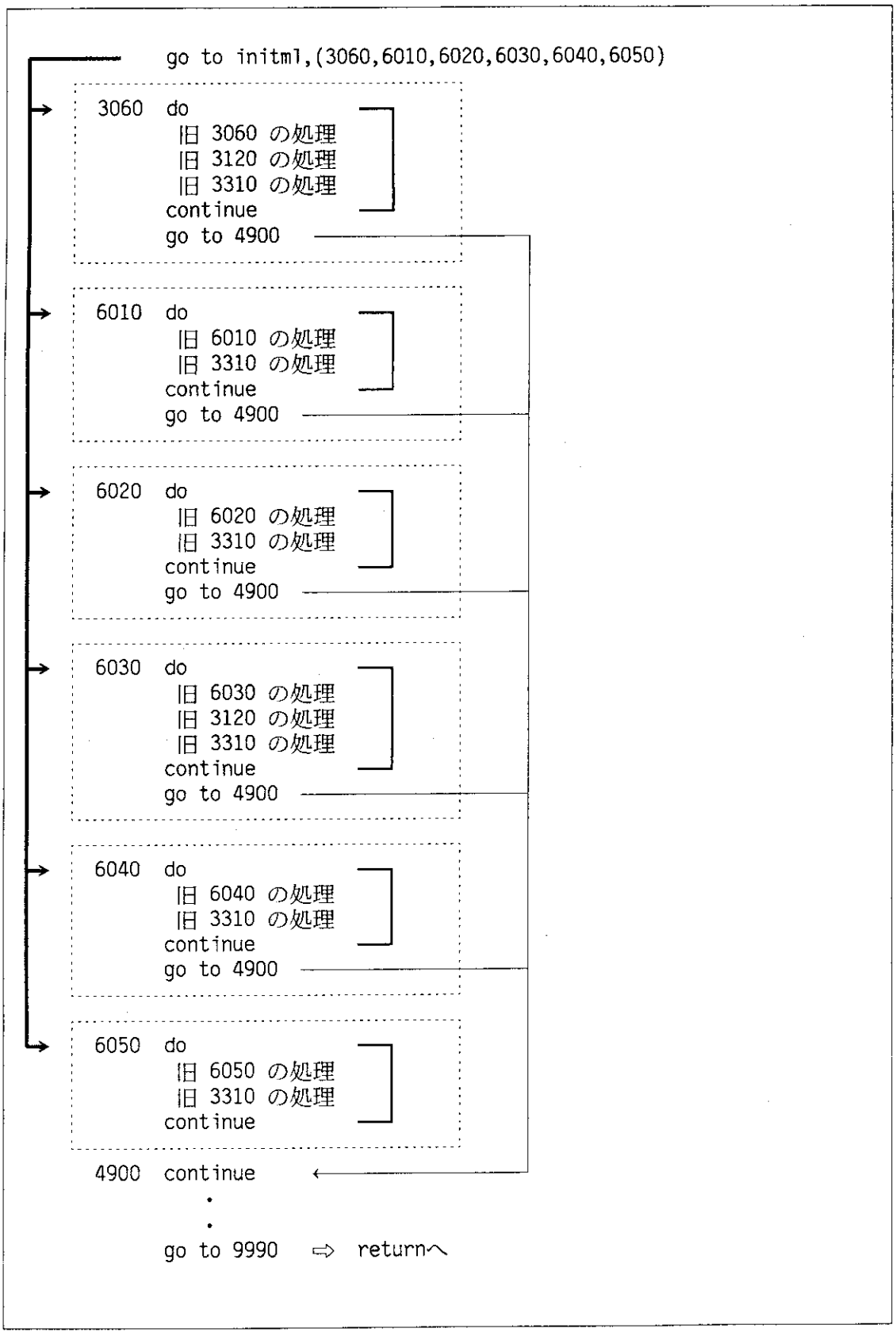


Fig. 2.32 Structure of subroutine rowt

```
c - - - - - begin i loop
      il = imsblk
      i = istr
      ib = i + 1 + intinc
      avima = dzmua*av(ib)
 3020 continue
      aheta = etama*ah(i)
      fmoi = fmo(i)
      wm = wmax
      wj = wmax
      wi = wmax
      fjomi = fjo(mi)
      go to initml,(3060,6010,6020,6030,6040,6050)
c - - - - - not plane, not zero weight, theta weighted
 3060 continue
      pclr(i) = pclr(i) - dzmuw*fiom*av(ib)
      e1 = av(i+1) - av(i)
      e2 = e1*dz2mu
      e1 = e1*bet2dz
      dav2 = e1 - e2
      dav3 = e2 + e2
      e1 = fmoi*(e1 + e2)
      e3=dirs(i)
      e1 = e1+e3
      fiom = fiom + fix(i)
      sorv = e3
      e4 = avima*fiom
      e3 = fjomi*aheta
      sorj = e4*pctwt  + e3 +e1
      sori = e3*pctwt + e4 +e1
      sorm = (e3 + e4)*pctwt + e1
      ib = ib + inc
      avima = dzmua*av(ib)
 3120 continue
c - - - - - calculate m - space weighting coefficient
      if(fmoi.le.0.d0) go to 3238
      e1 = fmoi*(sgtv2(i) + avima + aheta)
      e2 = e1 + e1 - sorm
      if(e2.le.e1) go to 3238
      e1 = (e1 + e1)/e2
      wm = 1.d0
      if(wm.gt.e1) go to 3238
      wm = e1
 3238 z = dav2*wm + dav3
c - - - - - calculate j - space weighting coefficient
 3310 continue
      dav2 = dav2 + sgtv2(i)
      if(fjomi.le.0.d0) go to 3328
      e1 = fjomi*(dav2 + avima)
      e2 = e1 + e1 - sorj
      if(e2.le.e1) go to 3328
      e1 = (e1 + e1)/e2
      wj = 1.d0
      if(wj.gt.e1) go to 3328
      wj = e1
 3328 y = aheta*wj
c - - - - - calculate i - space weighting coefficient
```

Fig. 2.33 Original il-loop in subroutine rowt (1/2)

```
      if(fiom.le.0.d0) go to 3428
      e1 = fiom*(dav2 + aheta)
      e2 = e1 + e1 - sori
      if(e2.le.e1) go to 3428
      e1 = (e1 + e1)/e2
      wi = 1.d0
      if(wi.gt.e1) go to 3428
      wi = e1
 3428 x = avima*wi - dav3
c - - - - - centered flux
      fbar = sorv + x*fiom + y*fjomi + z*fmoi
      fbar = fbar / (sgtv2(i) + sgtv2(i) + x + y + z)
c - - - - - extrapolate
      fix(i) = (fbar - fiom)*wi + fiom
      fjo(mi) = (fbar - fjomi)*wj + fjomi
      fmo(i) = (fbar - fmoi)*wm + fmoi
      if(nou.gt.0) go to 9900
 3900 dirs(i) = fbar
      fiom = fix(i)
      mi = mi + minc
      i = i + inc
      i1 = i1 - 1
      if(i1.ne.0) go to 3020
c - - - - - end i loop
```

Fig. 2.33 Original i1-loop in subroutine rowt (2/2)

```
c - - - - - not plane, not zero weight, weighted
 6030 continue
      pclr(i) = pclr(i) - dzmuw*fiom*av(ib)
      e1 = av(i+1) - av(i)
      e2 = e1*dz2mu
      e1 = e1*bet2dz
      dav2 = e1 - e2
      dav3 = e2 + e2
      e2 = fmoi*(e1 + e2)
      e1 = dirs(i)
      sorv = e1
      e1 = e1 + e2
      sorm = e1
      fiom = fiom + fix(i)
      sori = e1 + avima*fiom
      sorj = e1 + fjomi*aheta
      ib = ib + include
      avima = dzmua*av(ib)
      go to 3120
c
```

Fig. 2.34 Original '6030 continue' block in subroutine rowt

```
      iend = max(imsblk*inc, 1)
      do i = istr, iend, inc
                  .
                  .
      continue
```

Fig. 2.35 Do-loop of vector version (subroutine rowt)

```
c
      i1 = imsblk
      i = istr
      ib = i + 1 + intinc
cvp   avima = dzmua*av(ib)
      iend = max(imsblk*include,1)
      if(abs(iend-istr)/abs(include)+1.gt.nn) then
        write(6,*) ' ## VECTOR WORK AREA IS LUCK !!  in rowt '
        write(6,*) ' NN = ',abs(iend-istr)/abs(include)
      end if
      ibw = ib
      miw = mi
      n = 0
      do 8000 i = istr,iend,inc
         n = n + 1
         wfjo(n) = fjo(mi)            ⇦ ワーク配列の定義(fjo)
         mi = mi + minc
         wfmo(n) = fmo(i)             ⇦ ワーク配列の定義(fmo)
 8000 continue
      go to initml,(3060,6010,6020,6030,6040,6050)
```

Fig. 2.36 Definition of work arrays 'wfjo' and 'wfmo'

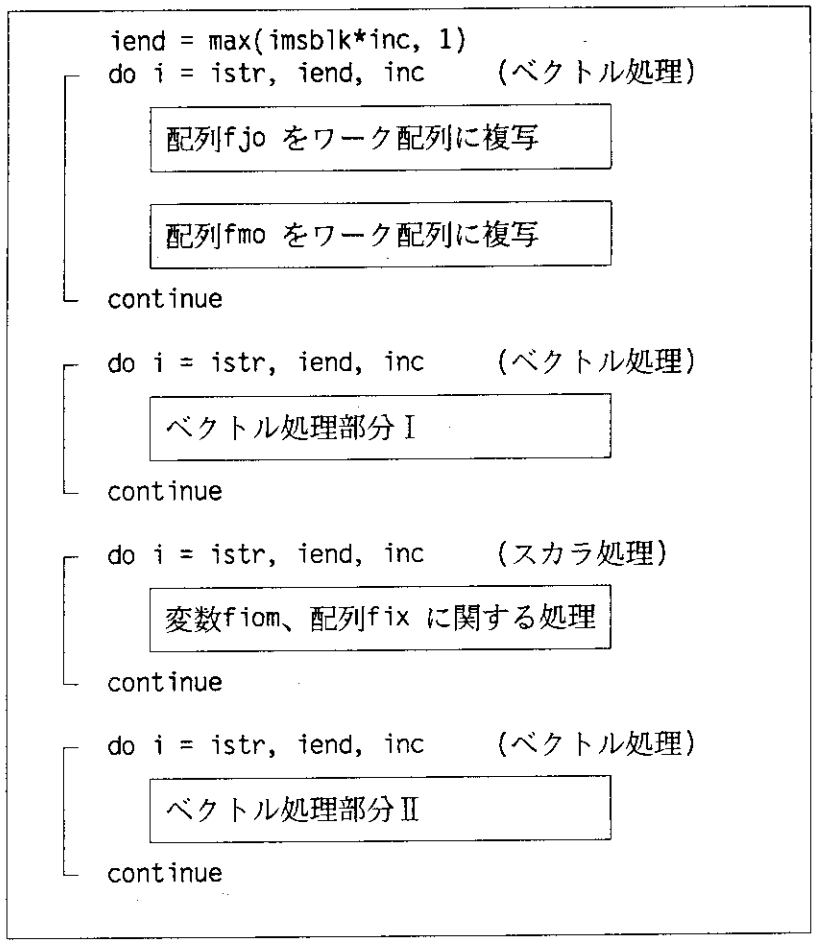


Fig. 2.37 Structure of vectorized do-loop

```
     6030 continue
          mi = miw
          n = 0
v         do 8320 i = istr, iend, include
v         n = n + 1
v         ahetaw(n) = etama*ah(i)
v         fmoi = fmo(i)
v         wmw(n) = wmax
v         wjw(n) = wmax
v         fjomi = fjo(mi)
     c - - - - - - not plane, not zero weight, weighted
     cvp  pclr(i) = pclr(i) - dzmuw*fiom*av(ib)
v         e1 = av(i+1) - av(i)
v         e2 = e1*dz2mu
v         e1 = e1*bet2dz
v         dav2w(n) = e1 - e2
v         dav3w(n) = e2 + e2
v         e2 = fmoi*(e1 + e2)
v         e1 = dirs(i)
v         sorv = e1
v         e1 = e1 + e2
v         e1w(n) = e1
v         sorm = e1
     cvp  fiom = fiom + fix(i)
v         avimaw(n) = dzmua*av(ib)
     cvp  sori = e1 + avima*fiom
v         sorj = e1 + fjomi*ahetaw(n)
     cvp  ib = ib + inc
v         avima2w(n) = dzmua*av(ib+include)
     c
     c - - - - - - calculate m - space weighting coefficient
     c##  if(fmoi.le.0.) go to 3238
v         if(fmoi.le.0.d0) go to 8318
v         e1 = fmoi*(sgtv2(i) + avima2w(n) + ahetaw(n))
v         e2 = e1 + e1 - sorm
v         if(e2.le.e1) go to 8318
v         e1 = (e1 + e1)/e2
v         wmw(n) = 1.d0
v         if(wmw(n).gt.e1) go to 8318
v         wmw(n) = e1
v     8318 zw(n) = dav2w(n)*wmw(n) + dav3w(n)
     c
     c - - - - - - calculate j - space weighting coefficient
     c3310 continue
v         dav2w(n) = dav2w(n) + sgtv2(i)
v         if(fjomi.le.0.d0) go to 8328
v         e1 = fjomi*(dav2w(n) + avima2w(n))
v         e2 = e1 + e1 - sorj
v         if(e2.le.e1) go to 8328
v         e1 = (e1 + e1)/e2
v         wjw(n) = 1.d0
v         if(wjw(n).gt.e1) go to 8328
v         wjw(n) = e1
v     8328 yw(n) = ahetaw(n)*wjw(n)
     c3900 dirs(i) = fbar
     cc    dirs(i) = fbar
     cc    fiom = fix(i)
```

Fig. 2.38 Vector version of original '6030 continue' block in subroutine rowt (1/2)

```
v          mi = mi + minc
     cvp   i = i + inc
v          il = il - 1
     cvp   if(il.ne.0) go to 3020
v          ib = ib + inc
v     8320 continue
           ib = ibw
           n = 0
s          do 8330 i = istr,iend,include
v          n = n + 1
v          wi = wmax
v          fmoi = wfmo(n)
v          fjomi = wfjo(n)
m          pclr(i) = pclr(i) - dzmuw*fiom*av(ib)
s          fiom = fiom + fix(i)
s          sori = elw(n) + avimaw(n)*fiom
     c
     c - - - - - calculate i - space weighting coefficient
s          if(fiom.le.0.d0) go to 8338
s          el = fiom*(dav2w(n) + ahetaw(n))
s          e2 = el + el - sori
s          if(e2.le.el) go to 8338
s          el = (el + el)/e2
s          wi = 1.d0
s          if(wi.gt.el) go to 8338
s          wi = el
m     8338 x = avima2w(n)*wi - dav3w(n)
     c - - - - - centered flux
m          fbarw(n) = dirs(i) + x*fiom + yw(n)*fjomi + zw(n)*fmoi
m          fbarw(n) = fbarw(n) / (sgtv2(i) + sgtv2(i) + x + yw(n) + zw(n))
     c
     c - - - - - extrapolate
s          fix(i) = (fbarw(n) - fiom)*wi + fiom
s          fiom = fix(i)
v          ib = ib + inc
v     8330 continue
           n = 0
           mi = miw
v          do 8340 i = istr,iend,include
v          n = n + 1
v          fmoi = wfmo(n)
v          fjomi = wfjo(n)
     c - - - - - extrapolate
v          fjo(mi) = (fbarw(n) - fjomi)*wjw(n) + fjomi
v          fmo(i) = (fbarw(n) - fmoi)*wmw(n) + fmoi
     c**
     ccc   if(nou.gt.0) go to 9900
     c**
     c3900 dirs(i) = fbar
v          dirs(i) = fbarw(n)
v          mi = mi + minc
     cvp   if(il.ne.0) go to 3020
v     8340 continue
           go to 4900
```

Fig. 2.38 Vector version of original '6030 continue' block in subroutine rowt (2/2)

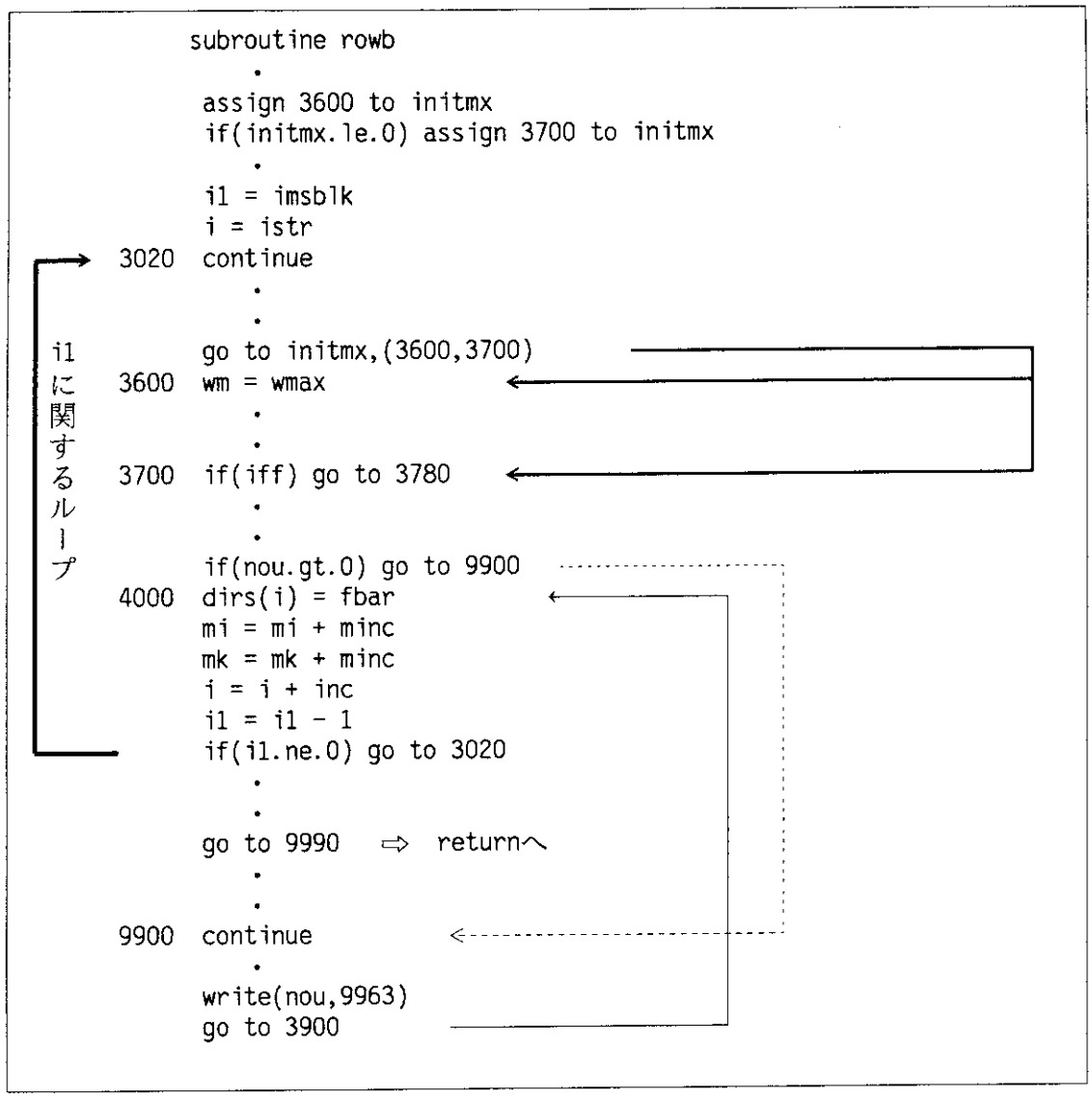


Fig. 2.39 Structure of subroutine rowb

```
c##    wm = 1.0
       wm = 1.0d0
c##    t  = 0.0
       t  = 0.0d0
c##    fjomi = 0.
       fjomi = 0.d0
c##    atxia = 0.
       atxia = 0.d0
c
c-----------------------------------------------------------------------
c - - - - - begin i loop
c
       i1 = imsblk
       i  = istr
 3020 continue
c
c - - - - - calculate m,i dependent terms
c           fmo and bet2dx must be 0 for initial dir, cf. bet2dz
       pclr(i) = pclr(i) - dzmuw*fiom*avib(i)
       fiom = fiom + fix(i)
       avima = dzmua*avia(i)
       avimu = dzmua*avib(i)
       avmufw = avimu*fiom*pctwt
       if(nvl.eq.0) go to 3030
       fjomi = fjo(mi)
       atxia = dzxia*at(i)
 3030 atxifw = atxia*fjomi*pctwt
       fkomi = fko(mk)
       aheta = etama*ah(i)
       ahetfw = aheta*fkomi*pctwt
c
       fmoi = fmo(i)
       davi = av(i+1) - av(i)
       davemu = davi*dz2mu
       davbet = davi*bet2dx
       dampm = davemu + davemu
       dabmm = davbet - davemu
       dabmms = dabmm + sgtv2(i)
       fmois = (davbet + davemu)*fmo(i) + dirs(i)
c
       wk = wmax
       wj = wmax
       wi = wmax
c
c - - - - - calculate m-space weighting coefficient
       go to initmx,( 3600, 3700)
 3600 wm = wmax
       if(fmoi.le.0.0d0 .or. iff) go to 3640
       e7 = (sgtv2(i) + avima + atxia + aheta)*fmoi
       e9 = e7 + e7
       e8 = e9 - fmois - avmufw - atxifw - ahetfw
       if(e8.gt.e7) wm = dmax1(1.d0,e9/e8)
 3640 t = dabmm*wm + dampm
 3700 if(iff) go to 3780
c
c - - - - - calculate k-space weighting coefficient
```

Fig. 2.40 Original il-loop in subroutine rowb (1/2)

```
      if(fkomi.le.0.0d0) go to 3720
      e5 = (dabmms + avima + atxia)*fkomi
      e9 = e5 + e5
      e6 = e9 - aheta*fkomi - fmois - avmufw - atxifw
      if(e6.gt.e5) wk = dmax1(1.d0,e9/e6)
c
c - - - - - calculate j-space weighting coefficient
 3720 if(fjomi.le.0.0d0) go to 3740
      e3 = (dabmms + avima + aheta)*fjomi
      e9 = e3 + e3
      e4 = e9 - atxia*fjomi - fmois - avmufw - ahetfw
      if(e4.gt.e3) wj = dmax1(1.d0,e9/e4)
c
c - - - - - calculate i-space weighting coefficient
 3740 if(fiom.le.0.0d0) go to 3780
      e1 = (dabmms + atxia + aheta)*fiom
      e9 = e1 + e1
      e2 = e9 - avimu*fiom - fmois - atxifw - ahetfw
      if(e2.gt.e1) wi = dmax1(1.d0,e9/e2)
c
c - - - - - centered flux
 3780 z = aheta*wk
      y = atxia*wj
      x = avima*wi - dampm
      fbar = dirs(i) + x*fiom + y*fjomi + z*fkomi + t*fmoi
      fbar = fbar / (sgtv2(i) + sgtv2(i) + x + y + z + t)
c
c - - - - - extrapolate
      fmo(i) = (fbar - fmoi)*wm + fmoi
      fko(mk) = (fbar - fkomi)*wk + fkomi
      if(nvl.gt.0) fjo(mi) = (fbar - fjomi)*wj + fjomi
      fix(i) = (fbar - fiom)*wi + fiom
c**
      if(nou.gt.0) go to 9900
c**
 4000 dirs(i) = fbar
      fiom = fix(i)
      mi = mi + minc
      mk = mk + mink
      i = i + inc
      il = il - 1
      if(il.ne.0) go to 3020
c - - - - - end i loop
```

Fig. 2.40 Original il-loop in subroutine rowb (2/2)

```
      end if
      miw = mi
      mkw = mk
      n = 0
      do 8610 i = istr, iend, inc
      n = n + 1
*vocl stmt,if(0)
      if(nvl.eq.0) then
        wfjo(n) = 0.d0
        atxiaw(n) = 0.d0
      else
        wfjo(n) =  fjo(mi)
        atxiaw(n) = dzxia*at(i)
      end if
      wfko(n) = fko(mk)
      mi = mi + minc
      mk = mk + mink
 8610 continue
```

Fig. 2.41 Definition of work arrays 'wfjo', 'atxiaw' and 'wfko'

```
        mi = miw
        mk = mkw
        n = 0
v       do 8620 i = istr, iend, inc
v       n = n + 1
  c
  c - - - - - calculate m,i dependent terms
  c           fmo and bet2dx must be 0 for initial dir, cf. bet2dz
  cvp   pclr(i) = pclr(i) - dzmuw*fiom*avib(i)
  cvp   fiom = fiom + fix(i)
v       avimaw(n) = dzmua*avia(i)
v       avimuw(n) = dzmua*avib(i)
  cvp   avmufw = avimu*fiom*pctwt
  cvp   if(nvl.eq.0) go to 3030
  cvp   fjomi = fjo(mi)
  cvp   atxia = dzxia*at(i)
  c3030 atxifw = atxia*fjomi*pctwt
v       fjomi = wfjo(n)
v  8601 atxifww(n) = atxiaw(n)*fjomi*pctwt
v       fkomi = fko(mk)
v       ahetaw(n) = etama*ah(i)
v       ahetfww(n) = ahetaw(n)*fkomi*pctwt
  c
v       fmoi = fmo(i)
v       davi = av(i+1) - av(i)
v       davemu = davi*dz2mu
v       davbet = davi*bet2dx
v       dampmw(n) = davemu + davemu
v       dabmmw(n) = davbet - davemu
v       dabmmsw(n) = dabmmw(n) + sgtv2(i)
v       fmoisw(n) = (davbet + davemu)*fmo(i) + dirs(i)
```

Fig 2.42 Vector version of original il-loop in subroutine rowb (1/3)

```
  c
v       wkw(n) = wmax
v       wjw(n) = wmax
v       wiw(n) = wmax
  c
  c - - - - - calculate m-space weighting coefficient
v       if(initmw.le.0) then
v         wmw(n) = 1.0d0
v       else
  c3600 wm = wmax
v       wmw(n) = wmax
v       end if
v  8609 continue
v       mi = mi + minc
v       mk = mk + mink
  cvp   i = i + inc
v       i1 = i1 - 1
  cvp   if(i1.ne.0) go to 3020
v  8620 continue
  cvp
        mi = miw
        mk = mkw
        n = 0
s       do 8630 i = istr, iend, inc
v       n = n + 1
m       pclr(i) = pclr(i) - dzmuw*fiom*avib(i)
s       fiom = fiom + fix(i)
m       avmufw = avimuw(n)*fiom*pctwt
v       fmoi = fmo(i)
v       fkomi = fko(mk)
v       fjomi = wfjo(n)
  cvp
v       if(initmw.le.0) then
v        t = 0.0d0
v       else
v       if(fmoi.le.0.0d0 .or. iff) go to 8602
v       e7 = (sgtv2(i) + avimaw(n) + atxiaw(n) + ahetaw(n))*fmoi
v       e9 = e7 + e7
m       e8 = e9 - fmoisw(n) - avmufw - atxifww(n) - ahetfww(n)
s       if(e8.gt.e7) wmw(n) = dmax1(1.d0,e9/e8)
  c3640 t = dabmm*wm + dampm
m  8602 t = dabmmw(n)*wmw(n) + dampmw(n)
v       end if
  cvp
  c3700 if(iff) go to 3780
v       if(iff) go to 8603
  c
  c - - - - - calculate k-space weighting coefficient
  c##   if(fkomi.le.0.0) go to 3720
v       if(fkomi.le.0.0d0) go to 8604
v       e5 = (dabmmsw(n) + avimaw(n) + atxiaw(n))*fkomi
s       e9 = e5 + e5
m       e6 = e9 - ahetaw(n)*fkomi - fmoisw(n) - avmufw - atxifww(n)
s       if(e6.gt.e5) wkw(n) = dmax1(1.d0,e9/e6)
  c
  c - - - - - calculate j-space weighting coefficient
```

Fig 2.42 Vector version of original i1-loop in subroutine rowb (2/3)

```
  cvp20 if(fjomi.le.0.0) go to 3740
v  8604 if(fjomi.le.0.0d0) go to 8605
v       e3 = (dabmmsw(n)+ avimaw(n) + ahetaw(n))*fjomi
s       e9 = e3 + e3
m       e4 = e9 - atxiaw(n)*fjomi - fmoisw(n) - avmufw - ahetfww(n)
s       if(e2.gt.e3) wjw(n) = dmax1(1.d0,e9/e4)
  c - - - - - calculate i-space weighting coefficient
  c3740 if(fiom.le.0.0) go to 3780
m  8605 if(fiom.le.0.0d0) go to 8603
s       e1 = (dabmmsw(n) + atxiaw(n) + ahetaw(n))*fiom
s       e9 = e1 + e1
s       e2 = e9 - avimuw(n)*fiom - fmoisw(n) - atxifww(n) - ahetfww(n)
s       if(e2.gt.e1) wiw(n) = dmax1(1.d0,e9/e2)
  c - - - - - centered flux
  c3780 z = aheta*wk
m  8603 z = ahetaw(n)*wkw(n)
s       y = atxiaw(n)*wjw(n)
s       x = avimaw(n)*wiw(n) - dampmw(n)
s       fbarw(n) = dirs(i) + x*fiom + y*fjomi + z*fkomi + t*fmoi
m       fbarw(n) = fbarw(n) / (sgtv2(i) + sgtv2(i) + x + y + z + t)

  cvp   fmo(i) = (fbar - fmoi)*wm + fmoi
  cvp   fko(mk) = (fbar - fkomi)*wk + fkomi
  cvp   if(nvl.gt.0) fjo(mi) = (fbar - fjomi)*wj + fjomi
s       fix(i) = (fbarw(n) - fiom)*wiw(n) + fiom
  c4000 dirs(i) = fbar
s       fiom = fix(i)
v       mi = mi + minc
v       mk = mk + mink
v  8630 continue
        mi = miw
        mk = mkw
        n = 0
v       do 8640 i = istr, iend, inc
v       n = n + 1
v       fmoi = fmo(i)
v       fkomi = wfko(n)
v       fjomi = wfjo(n)
v       fmo(i) = (fbarw(n) - fmoi)*wmw(n) + fmoi
v       fko(mk) = (fbarw(n) - fkomi)*wkw(n) + fkomi
v       if(nvl.gt.0) fjo(mi) = (fbarw(n) - fjomi)*wjw(n) + fjomi
v c4000 dirs(i) = fbar
v       dirs(i) = fbarw(n)
v       mi = mi + minc
v       mk = mk + mink
v  8640 continue
  c - - - - - end i loop
```

Fig 2.42 Vector version of original il-loop in subroutine rowb (3/3)

```
  cdir$ ivdep
  cvocl loop,novrec
s       do 270 ij=ijb,ije,jnc2
s   270 fx(ij) = qq(ij) - cbanjb(ij)*fx(ij-jnc) - cbanjt(ij)*fx(ij+jnc)
```

Fig. 2.43 Original version of 'do 270' in subroutine wsol

```
  cdir$ ivdep
  *vocl loop,novrec
v       do 270 ij=ijb,ije,jnc2
v   270 fx(ij) = qq(ij) - cbanjb(ij)*fx(ij-jnc) - cbanjt(ij)*fx(ij+jnc)
```

Fig. 2.44 Vector version of 'do 270' in subroutine wsol

# 3. FLOWGR コードのベクトル化

FLOWGR コードは富士通(株)製大型汎用計算機 M780 上で利用されているものであるが，当計算機上では 30 回のリスタートによる平均 30 時間もの実行時間を要するため，年間約 30 ケースしか実行できなかった (メモリ使用量 3MB 未満). そこで，富士通(株)製分散メモリ型ベクトル並列計算機 VPP500/42 (以下 VPP500) 上でのベクトル化を行った. 今回のベクトル化による速度性能の向上は，オリジナル版の VPP500 ベクトル実行に比較すると，ベクトル化版のベクトル実行では 4.3 倍であった. 本報告書では，主にベクトル化の作業内容について述べる.

## 3.1 コード概要

FLOWGR コードは，高温ガス炉の一次冷却系破断事故時における炉内への空気の侵入挙動及び炉内における化学反応を伴う多成分気体の流動を解析する [6].

原子炉系を破断孔，一次元流路，分岐から構成される一次元の流路網としてモデル化し，流路壁の空間温度分布及び時間変化や多成分気体を構成する気体の種類等を入力データで与える. 気体の種類としてはヘリウム，窒素，酸素，一酸化炭素，二酸化炭素，アルゴンを考える.

解析の対象とする期間は，配管破断が生じてから流路網外の重い気体と炉内冷却気体の混合気体の自然対流が発生するまでの間で，混合気体及び成分気体の質量保存則，混合気体の運動量保存則，混合気体のエネルギー保存則に基づく支配方程式を一次元座標メッシュにより離散化し数値解を求める. この間，指定した流路においては化学反応による一酸化炭素及び二酸化炭素の発生を取り扱う.

## 3.2 オリジナル版コードの修正

M780 で利用されているオリジナル版コードには，サブルーチン間で引数として使用されている配列サイズの相違や初期化されていない変数等が存在していた. VPP500 上でデバッグオプション "-Dasu" [5, 8] を付けたコンパイル & 実行を行うと，これらが原因でアベンドする. デバッグオプションを付けない通常のコンパイル & 実行には差し支えないが，実行をより確実にするため，ベクトル化を行う前にこれらについて修正を行った.

(1) 配列 QS について

サブルーチン SETO では COMMON ブロック VALNEW において COMMON 配列 QS(NSZ,7) が宣言されていた. この配列は，このルーチンから呼び出しているサブルーチン PCON へ渡されているが，PCON では DIMENSION により QS(NSZ,8) と宣言されていた. そこでサイズを揃えるため，サブルーチン SETO の QS(NSZ,7) を QS(NSZ,8) とした. これに伴い他のサブルーチンで宣言されている COMMON ブロック VALNEW においても，ブロックサイズを合わせるために QS(NSZ,7) を QS(NSZ,8) に変更した. 配列 QS についての変更内容を Fig. 3.1 に示す.

(2) サブルーチン OUT4 の変数 NMAX について

この変数は DO 4000 ループ内の IF 文が真にならない場合に未定義になるため，エラーが発生した．そこで，NMAX に値が定義されない場合の暫定処置として NMAX=1 を OUT4 に追加した．追加結果を Fig. 3.2 に示す．

(3) サブルーチン STEP，STEP2 の配列 QTMP，STMP について

この変数も初期化されていなかったために，エラーが発生した．そこで，これらについては，2 つのサブルーチンの先頭において DATA 文の追加により初期化することにした．追加結果を Fig. 3.3 に示す．

### 3.3 動的挙動解析

本コードのコスト分布を調査するため，入力データ TAKEDA と TAKEDAM を用いてオリジナル版コードについての動的挙動解析を行った．VPP500 上には動的挙動解析ツールとして SAMPLAR [7] が用意されているが，これはベクトル化を行う上で必要になる詳細な解析情報を得ることができない．そこで，解析には GSP 上に用意されている動的挙動解析ツール ANALYZER [4] を用いた．このツールを用いると GSP 上で実行した場合の，全体に対するサブルーチン毎の解析結果，各サブルーチンに対する DO ループ毎の解析結果と STATEMENT 毎の解析結果が得られる．全体に対するサブルーチン毎の解析結果について，TAKEDA を用いた場合を Table 3.1 に，TAKEDAM を用いた場合を Table 3.2 に示す．この解析結果より，PCON，SOLVE，SOR，CPMASS の 4 つのサブルーチンのコストが高い．よって，主にこれらのサブルーチンをベクトル化の対象とした．

### 3.4 ベクトル化

ここでは，PCON，SOLVE，SOR，CPMASS の 4 つのサブルーチンとこの他のサブルーチンに対して行ったベクトル化チューニングの内容について述べる．尚，TAKEDA，TAKEDAM の 2 つの入力データの違いによるコスト分布の差はほとんど無いので，以降の説明では，TAKEDA を用いた場合の各動的挙動解析結果を用いることにする．

#### 3.4.1 サブルーチン PCON

このルーチンに対する DO ループ毎の動的挙動解析結果を Table 3.3 に示す．この情報を基に各ループに対してチューニングを行った．以下ではそのチューニング内容を述べる．

(1) DO 500 ループについて

Table 3.3 によると，DO 521，DO 510，DO 520 ループのコストが高い．これらのループを含んでいるオリジナル版の DO 500 ループを Fig. 3.4 に示す．このような多重ループにおいてコンパイラによる自動ベクトル化の対象になるのは基本的に最内ループである．また，ベクトル実行においてはループの回転数のことをベクトル長と呼ぶが，ベクトル化の対象となるループのベクトル長はおよそ 10 以上であればベクトル実行することによって効率良く速度向上を得ることができる．逆に 10 以下であればベクトル実行のためのオーバーヘッドが目立ち，スカラ実行よりも遅くなってしまうことが大部分である．これらの DO 521，DO 510，DO 520 ループはベク

トル化の対象となっておりベクトル長は NGAS である．しかし，TAKEDA，TAKEDAM においては NGAS=5 であり，ベクトル実行においてはベクトル長が短過ぎる．従って，これらのループについては外側ループの DO 500 ループのベクトル長である NCELL(セルの数:TAKEDA，TAKEDAM において 85) を利用し，これを最内ループにするチューニングを行った．以降でそのチューニング過程を説明する．

・Fig. 3.4 から Fig. 3.5 への変更

Fig. 3.4 において 1 行目〜 11 行目の部分と 12 行目〜 26 行目の部分の 2 つの部分に DO 500 ループを分割し，前の部分を DO 500 ループ，後の部分を DO 501 ループとした．Fig. 3.4 の 1 行目〜 11 行目の部分で定義されている一次元配列 ETAA は，12 行目〜 26 行目の部分で参照されているが，分割後の 2 つのループ間でも正しい定義参照関係になるように，Fig. 3.5 では ETAAV という二次元配列を用意しそれを利用するようにした．このため，このルーチンの配列宣言部分で新たに ETAAV(NSZ,8) を宣言し，ETAA(8) の宣言はコメント化した．また，Fig. 3.4 の 2，3 行目で初期化を行っている配列 ETA，FLAM は，21 行目以降まで使用されていないので，Fig. 3.5 では DO 501 ループへ移すことにした．

・Fig. 3.5 から Fig. 3.6 への変更

Fig. 3.5 において 11 行目〜 28 行目の DO 501 ループを 11 行目〜 22 行目の部分と 23 行目〜 28 行目の部分の 2 つの部分に分割し，前の部分を DO 501 ループ，後の部分を DO 502 ループとした．Fig. 3.5 の 11 行目〜 22 行目で定義参照している一次元配列 FMA と DENOM は，23 行目〜 28 行目で参照定義されているが，分割後の 2 つのループ間でも正しい定義参照関係になるように，Fig. 3.6 では FMAV と DENOMV という二次元配列を用意しそれを利用するようにした．このため，このルーチンの配列宣言部分で新たに FMAV(NSZ,8) と DENOMV(NSZ,8) を宣言した．また，Fig. 3.5 の 12，13 行目で初期化を行っている配列 ETA，FLAM は，23 行目以降まで使用されないので，Fig. 3.6 では DO 502 ループへ移すことにした．尚，DO 502 ループの内側ループは DO 522 とした．

・Fig. 3.6 から Fig. 3.7 への変更

Fig. 3.6 において 12 行目〜 23 行目の DO 501 ループを 12 行目〜 15 行目の部分と 16 行目〜 23 行目の部分の 2 つの部分に分割し，前の部分を DO 501 ループ，後の部分を DO 503 ループとした．尚，三重ループ DO 503 ループの外側から 2 番目のループは DO 523 とした．

・Fig. 3.7 から Fig. 3.8 への変更

Fig. 3.7 において 28 行目〜 37 行目の DO 502 ループを 28 行目〜 30 行目の部分と 31 行目〜 37 行目の部分の 2 つの部分に分割し，前の部分を DO 502 ループ，後の部分を DO 504 ループとした．

・Fig. 3.8 から Fig. 3.9 への変更

Fig. 3.8 において 31 行目〜 38 行目の DO 504 ループを 31 行目〜 36 行目の部分と 37 行目〜 38 行目の部分の 2 つの部分に分割し，前の部分を DO 504 ループ，後の部分を DO 505 ループとした．

• Fig. 3.9 から Fig. 3.10 への変更

Fig. 3.9 における DO 500, DO 510 の二重ループ, DO 501, DO 520 の二重ループ, DO 503, DO 523, DO 521 の三重ループ, DO 504, DO 522 の二重ループの各最外ループを最内ループとした. これにより, ベクトル長 NCELL のループがコンパイラの自動ベクトル化の対象となり, 高速化されることとなった.

(2) DO 300 ループについて

オリジナル版のこのループを Fig. 3.11 に示す. 自動ベクトル化の対象となる最内ループの DO 310 ループのベクトル長は NGAS でありベクトル長が短く高速化には向かない. 従って, DO 300 ループのベクトル長である NCELL を利用し, これを最内ループにするチューニングを行った. チューニング方法は DO 500 ループと同様である. 次にチューニング過程を示す.

• Fig. 3.11 から Fig. 3.12 への変更

Fig. 3.11 において 1 行目〜 2 行目の部分と 3 行目〜 7 行目の部分の 2 つの部分に分割し, 前の部分を DO 300 ループ, 後の部分を DO 301 ループとした.

• Fig. 3.12 から Fig. 3.13 への変更

Fig. 3.12 において二重ループ DO 301 の最外ループを最内ループとした. これによりベクトル長 NCELL のループがコンパイラの自動ベクトル化の対象となり, 高速化されることとなった.

(3) DO 10, DO 20 ループについて

オリジナル版のこの 2 つのループを Fig. 3.14 に示す. これら 2 つのループ共二重ループであり, ベクトル化の対象になる内側ループのベクトル長が短い. 従って, 内側ループと外側ループの入れ替えを行い, 内側ループのベクトル長を NCELL にしてベクトル化させるようにした. 変更後のこれらのループを Fig. 3.15 に示す.

(4) DO 200 ループについて

オリジナル版のこのループを Fig. 3.16 に示す. このループについても (3) と同様に, 内側ループのベクトル長が短いため, 外側ループとの入れ換えを行い内側ループのベクトル長を NCELL にしてベクトル化させるようにした. 変更後のこのループを Fig. 3.17 に示す.

(5) ユーザ定義関数を呼び出している DO ループについて

このサブルーチンでは Fig. 3.13 の DO 310 ループ, Fig. 3.10 の DO 504 ループにおいてそれぞれユーザ定義関数 SHEAT, SCOND を呼び出している. このようにユーザ定義関数を呼び出しているループは VPP コンパイラの仕様によりベクトル化されない. これをベクトル化させるには, 呼び出されている関数の上位ループへのインライン展開を行う. インライン展開とは, 関数や副プログラムをそのままループ内へ展開することを言うが, このインライン展開は, 関数 SHEAT, SCOND をサブルーチン PCON を格納しているファイルに結合し, コンパイル時にコンパイルオプション "-Of" を指定することで可能になる. この "-Of" は M780 での "OPT(F)" に相当するものである.

こうすることにより, 関数 SHEAT, SCOND もベクトル化の対象となるが, この内, SHEAT は

コンパイラによってベクトル化されないコーディングがされていたため修正を加えた．オリジナル版の SHEAT を Fig. 3.18 に示す．SHEAT は IF-ELSEIF-ENDIF ブロックから成っているが，これらの条件に合致しない場合，SHEAT が未定義になるとコンパイラが判断するため，ベクトル化されなかった．そこで，このブロックに新たに ELSE ブロックを設けダミー処理として "SHEAT=0.0D0" を追加した．これにより，関数 SHEAT を呼び出している Fig. 3.13 の DO 130 ループがベクトル化されるようになった．関数 SHEAT の新たな IF-ELSEIF-ELSE-ENDIF ブロックを Fig. 3.19 に示す．

### 3.4.2 サブルーチン SOLVE

このルーチンに対する DO ループ毎の動的挙動解析結果を Table 3.4 に示す．また，オリジナル版のこのルーチンを Fig. 3.20 に示す．この情報を基に DO 130，DO 240 ループに対し施したベクトル化の内容について述べる．

(1) DO 130 ループについて

DO 130 ループのコストが高いのは，コンパイラが DO 130 ループはベクトル実行を行うと配列 A について回帰演算が発生すると判断したため，ベクトル化されていないことが原因である．しかし，ベクトル化の対象となるループの構造によっては，コンパイラは回帰演算が発生しない場合でも発生すると判断する場合があるため，ベクトル実行した場合に回帰演算が発生するかどうか調査した．

Fig. 3.20 の DO 130 ループにおいて，A(I,K) と A(I,JJ)，または A(I,K) と A(N,J) が同じアドレスをアクセスすると回帰演算が発生し，そうでなければ回帰演算は発生しない．尚，A(I,JJ) と A(N,J) は参照されているだけなので同じアドレスをアクセスしても回帰演算とは言わない．

・A(I,K) と A(N,J) について

この 2 つの配列の一次元目の添字 I と K に注目する．I，K 共に DO 130 ループの処理中では固定の値を持つ．Fig. 3.20 において N は 7 行目で，I は 18 行目で決められる．I は "I=N+L-1" という計算で決められ，L は N が固定の場合に 2 ～ MBAND まで変化する．これより，L は最低でも 2 であるため，"I=N+L-1" より I は常に N より大きい値を取ることが分かる．よって，I と N は DO 130 ループにおいて常に異なる値を持つことから，A(I,K) と A(N,J) は同じアドレスをアクセスすることは無いため，この 2 つの配列については回帰演算は発生しない．

・A(I,K) と A(I,JJ) について

この 2 つの配列については，二次元目の添字 K と JJ に注目する．K は DO 130 ループの DO 変数であり KI ～ KF まで変化し，JJ はループ処理中では固定の値を持つ．Fig. 3.20 において KI は 21 行目で KF は 22 行目で決められ，JJ は 17 行目で決められる．KI は "KI=MBAND-L+2"，JJ は "JJ=MBAND-L+1" という計算で決められる．これより，KI は JJ より大きい値を持ち，DO 130 ループにおいて DO 変数 K は JJ より常に大きい値を取ることが分かる．よって，A(I,K) と A(I,JJ) は同じアドレスをアクセスすることは無いため，この 2 つ

の配列については回帰演算は発生しない.

以上より, DO 130 ループにおいて配列 A について回帰演算は発生しない. よって, コンパイラに対して回帰演算が発生しないことを判断させる最適化制御行 "*vocl loop,novrec(A)" を DO 130 ループの直前に指定し, ベクトル化させるようにした. "(A)" は配列 A を意味する. 指定した DO 130 ループを Fig. 3.21 に示す.

(2) DO 240 ループについて

このルーチンのコストが高いのも, DO 130 ループと同様にコンパイラが DO 240 ループはベクトル化を行うと配列 B について回帰演算が発生すると判断したため, ベクトル化されていないことが原因である. そこで, このループに対してもベクトル実行した場合に回帰演算が発生するかどうかを調査した.

Fig. 3.20 の DO 240 ループにおいて, B(N) と B(K) が同じアドレスをアクセスすると回帰演算が発生し, そうでなければ回帰演算は発生しない.

配列 B(N) の添字 N は DO 240 ループ処理中では固定の値を持ち, 配列 B(K) の添字 K は DO 240 ループで変化する. K は 54 行目の "K=N+L-MBAND" という計算で決められているが, L は DO 240 ループにおいて LL ～ NCOLS まで変化する DO 変数である. この内, LL は DO 250 ループの外で "LL=MBAND+1" と定義されていることにより, DO 240 ループにおいて K の初期値は "K=N+MBAND+1-MBAND" より N+1 である. 従って, このループにおいては K の値は常に N より大きいことにより, B(N) と B(K) は同じアドレスをアクセスすることは無いため, この 2 つの配列については回帰演算は発生しない.

以上より, DO 130 ループと同様にコンパイラに対して回帰演算が発生しないことを判断させる最適化制御行 "*vocl loop,novrec(B)" を DO 240 ループの直前に指定して, ベクトル化させるようにした. 指定した DO 240 ループを Fig. 3.22 に示す.

### 3.4.3 サブルーチン SOR

Table 3.1 の V-effect の情報よりこのルーチンはスカラ実行よりも逆に遅くなっている場合がある. このルーチンに対する DO ループ毎の動的挙動解析結果を Table 3.5 に示す. これによると DO 2101, DO 2201, DO 2102, DO 2202 ループがスカラ実行よりも遅い原因になっていることがわかる. 更にオリジナル版のこのルーチンを Fig. 3.23 に示す. これらのループは三重ループである DO 2000 ループの最内ループであり, Table 3.5 からもわかるように平均ベクトル長が 1 である. このような短いベクトル長でベクトル実行を行うと, ベクトル実行のオーバーヘッドが大きくなりスカラ実行よりも遅くなってしまう.

これを回避するためのチューニングとしては, DO 2100, DO 2200 ループの回転数 NCELL を利用しこれを最内ループにするのが望ましい. また, このルーチンは全体に対してコストも高いため, チューニングを行うとベクトル化の効果が期待できる. しかし, チューニングを行うと配列 TMP について回帰演算が発生してしまい, 正当な計算が行われなくなってしまう. そこで, 速度向上を得られるようなチューニングは行わず, スカラ実行よりも遅くなることだけを回避することにした. これについては, 上の 4 つの DO ループの直前に, ベクトル実行を抑止してスカ

ラ実行を行わせるための最適化制御行 "*vocl loop,scalar" の指定を行った．指定を行ったこのルーチンを Fig. 3.24 に示す．

### 3.4.4 サブルーチン CPMASS

Table 3.1 の V-effect の情報よりこのルーチンはスカラ実行よりも逆に遅くなっている場合がある．このルーチンに対する DO ループ毎の動的挙動解析結果を Table 3.6 に示す．更にオリジナル版のこのルーチンを Fig. 3.25 に示す．Table 3.6 より DO 1000 ループが最もコストが高いが，Fig. 3.25 から分かるように，これは最外ループであるためベクトル化されていないことが原因である．次に DO 1400，DO 1300 ループのコストが高い．これは，最内ループにもかかわらず，コンパイラがそれぞれのループにおいて定義参照されている二次元配列 FMAT について回帰演算が発生すると判断したため，ベクトル化されていないことが原因である．しかし，この回帰演算を回避し仮にこの 2 つのループについてベクトル化を行ったとしても，Table 3.6 からも分かるように平均ベクトル長が 1 であるため，ベクトル化の効果が望めずスカラ実行よりも逆に遅くなることが予想される．従って，DO 1000 ループの回転数 NCELL を利用してベクトル化できるようにこのループの大幅なチューニングを行った．

(1) DO 1000 ループの分割

Fig. 3.25 の DO 1000 ループを 21 行目〜 28 行目の部分，29 行目〜 43 行目の部分と 44 行目〜 63 行目の部分の 3 つの部分に分割した．ループ ID は順に DO 1000，DO 1001，DO 1002 ループとした．これを Fig. 3.26 に示す．

DO 1000 ループと DO 1001 ループ以降を分割したのは，DO 1000 ループを NCELL でベクトル化させるために行ったものである．また，DO 1001 ループと DO 1002 ループ以降を分割した理由は次のとおりである．

Fig. 3.25 において 29 行目〜 43 行目の部分は非境界セルについての計算で，44 行目〜 63 行目の部分は境界セルについての計算である．ここで入力データ TAKEDA の場合の流路ネットワークモデル図を Fig. 3.27 に示す (TAKEDAM の場合も同様)．これから分かるように，大部分が非境界セルであり，更にこのルーチンについての STATEMENT 毎の動的挙動解析結果を Table 3.7 に示すと，true の情報 (IF 文の真率) からも分かるように，非境界セルについての計算が全体の 96.5% である．そこで，この非境界セルについての計算部分を NCELL でベクトル化するために分割を行った．

分割を行った結果，DO 1002 ループは全体の 3.5% しか実行されない．また，Fig. 3.27 からも境界セルは 3 つである．従って，DO 1002 ループを NCELL 回実行させる必要はないため，DO 1001 ループにおいて境界セルの場合をカウントし，そのカウント分を DO 1002 ループにおいて回転させるようにした．DO 1001 ループでの境界セルのカウント処理は，"IF(ICEL(I).EQ.0)" の ELSE として追加した．このカウント処理を行うため，境界セルの数 (DO 1002 ループの回転数) を格納する変数と境界セルの ID を格納する配列を新たに用意した．更に，境界セルが 3 つであることから DO 1002 ループのベクトル長は 3 になる．これをベクトル実行するとスカラ実行よりも遅くなるため，ベクトル実行を抑止する最適化制御行 "*vocl loop,scalar" を DO 1002 ループの直前に指定した．

(2) Fig. 3.26 の DO 1001 ループ

Fig. 3.26 の DO 1300 ループの DO 変数は IPR(I) ～ IPR(I+1)-1 まで回転する．FLOWGR コードにおいて，セル ID が I の時のプラス方向のセル ID またはジャンクション ID を得る場合は，それぞれの ID が格納されている配列に対しこの IPR(I) ～ IPR(I+1)-1 を添字にして得ている．すなわち，IPR(I) ～ IPR(I+1)-1 はプラス方向のセル ID またはジャンクション ID を得るためのポインタである．従って，セル ID が I の時，DO 1300 ループはプラス方向のセル ID またはジャンクション ID の数だけ回転する．言い替えれば，プラス方向のセル ID またはジャンクション ID へのポイント数だけ回転する．入力データ TAKEDA によって作成される Fig. 3.27 の流路ネットワークモデルにおいては，大部分のセルに対するプラス方向のセル ID またはジャンクション ID の数は 1 つであり，流路が分岐している部分についてのみ複数個存在する．従って，DO 1300 ループは大部分が回転が 1 であるため，平均ベクトル長が 1 になっている．これと同様のことが DO 1400 ループについても言え，DO 1400 ループの回転範囲である IPL(I) ～ IPL(I+1)-1 はマイナス方向のセル ID またはジャンクション ID を得るためのポインタである．従って，Fig. 3.27 においては DO 1400 ループも大部分が回転が 1 であるため，平均ベクトル長が 1 になっている．

以上の考察より，Fig. 3.26 における DO 1300，DO 1400 ループの大部分の回転が 1 であることを利用し，回転が 1 の時とそれ以外の時に処理を分けることにした．このようにするためには，DO 1300，DO 1400 ループの回転数を予め求めておく必要がある．プラス方向またはマイナス方向のセル ID またはジャンクション ID を得るためのポインタは，サブルーチン NETWRK において作成される流路ネットワークモデルに依存する．そこで，NETWRK の最後の部分に DO 1300，DO 1400 ループの回転数を求める処理を追加した．追加した処理を Fig. 3.28 に示す．ここでは，セル ID が I の時の，プラス方向のセル ID またはジャンクション ID へのポイント数 (DO 1300 ループの回転数) を ipr_loop(I) に，マイナス方向のセル ID またはジャンクション ID へのポイント数 (DO 1400 ループの回転数) を ipl_loop(I) に格納している．この 2 つの配列は，サブルーチン NETWRK とサブルーチン CPMASS に既存の COMMON ブロック POINTR に COMMON 変数 ipr_loop(NSZ)，ipl_loop(NSZ) として追加し，サブルーチン CPMASS で参照できるようにした．

この配列を利用して，Fig. 3.26 における DO 1300，DO 1400 ループを IF-ELSE-ENDIF ブロックにすることにより，回転数が 1 の時とそれ以外の時に処理を分けた．これを Fig. 3.29 に示す．尚，COMMON ブロック POINTR はサブルーチン ENERGY，MXMASS，PRESMD，READWT，SET0，SET1，SOR，STEP，STEP2 にも存在するため，それぞれに対しても COMMON 変数 ipr_loop(NSZ)，ipl_loop(NSZ) を追加した．

(3) Fig. 3.29 の DO 1001 ループの分割

次に，Fig. 3.29 の DO 1001 ループにおいて 5 行目～ 9 行目の部分，10 行目～ 17 行目の部分，19 行目～ 23 行目の部分と 24 行目～ 31 行目の部分の 4 つの部分に分割した．1 番目のループを DO 1001，2 番目のループを DO 1003，3 番目のループを DO 1004，4 番目のループを DO 1005 とした．これを Fig. 3.30 に示す．この内，DO 1003，DO 1005 ループはそれぞれポイント数が 1 以外の場合の計算であるため，極めて回転数が少ない．そこで，この 2 つの

ループを NCELL 回実行する必要はないため，DO 1001，DO 1004 ループにおいてポイント数が 1 以外の場合をカウントし，そのカウント分だけ DO 1003，DO 1005 ループにおいて回転させるようにした．カウント処理はそれぞれ DO 1001，DO 1004 ループの "if(ipr_loop(I).eq.1)" と "if(ipl_loop(I).eq.1)" の ELSE として追加した．このカウント処理を行うため，ポイント数 1 以外のセル ID の数 (DO 1003，DO 1005 ループの回転数) を格納する変数とポイント数 1 以外のセル ID を格納する配列を新たに用意した．また，Fig. 3.27 においてはポイント数が 1 以外の場合 (分岐している部分) はプラス方向マイナス方向共に 1 回である．この回数でベクトル実行するとベクトル長が短すぎてスカラ実行よりも遅くなってしまうため，ベクトル実行を抑止する最適化制御行 "*vocl loop,scalar"を DO 1003，DO 1005 ループの直前に指定した．以上，分割したことにより，DO 1001，DO 1003 ループは NCELL によってベクトル化されるようになった．

### 3.4.5 サブルーチン SET1 と STEP

この 2 つのルーチンは Table 3.1 によると全体に対してそれほどコストは高くない．このようなコストのあまり高くないルーチンに対してベクトル化を行うと，短時間ジョブでは効果は現れないが，ある程度の長時間ジョブになれば効果が現れて来る．そこで，FLOWGR コードのタイムステップを大きくして，この 2 つのルーチンのチューニング前とチューニング後の実行時間を比較したところ，約 6 時間強かかるジョブが約 5 時間半に短縮された．よって，これらのルーチンに対して行ったチューニング内容も有効なものとした．その内容を以下に示す．尚，STEP よりコストの高いサブルーチン DIFFSN，ZERO はコンパイラの自動ベクトル化機能により効率良いベクトル化がされており，更にチューニングできる部分は無かった．

(1) サブルーチン SET1

このルーチンに対する DO ループ毎の動的挙動解析結果を Table 3.8 に示す．このルーチンでコンパイラによってベクトル化されていない箇所は DO 400 ループだけある．この他はすべてコンパイラの自動ベクトル化機能により効率良いベクトル化がされてるため，DO 400 ループについてのみチューニングを行った．オリジナル版の DO 400 ループを Fig. 3.31 に示す．このループは Fig. 3.26 の DO 1001 ループと構造が似ていて，DO 410 ループは Fig. 3.26 の DO 1300 ループに，DO 420 ループは Fig. 3.26 の DO 1400 ループに相当する．従って，この DO 400 ループについても Fig. 3.26 の DO 1001 ループと同様のチューニングを行った．Fig. 3.31 において 1 行目〜 2 行目の部分，3 行目〜 4 行目の部分，5 行目〜 6 行目の部分と 7 行目〜 8 行目の部分の 4 つの部分に分割し，順に DO 400，DO 401，DO 402，DO 403 ループとした．更に，DO 401，DO 402 ループを回転数が 1 の場合とそれ以外の場合に分け，それ以外の場合のループをそれぞれ DO 404，DO 405 ループとした．この 2 つのループの直前には "*vocl loop,scalar" を指定した．チューニング結果を Fig. 3.32 に示す．

(2) サブルーチン STEP

このルーチンに対する DO ループ毎の動的挙動解析結果を Table 3.9 に示す．これによるとコンパイラによってベクトル化されていない箇所は DO 600，DO 800 ループである．この内，DO

800 ループはベクトル実行すると回帰演算が発生するためベクトル化不可能である．また，この他はすべてコンパイラの自動ベクトル化機能により，効率良いベクトル化がされている．そこで，DO 600 ループについてのみチューニングを行った．オリジナル版の DO 600 ループを Fig. 3.33 に示す．このループは 2 重ループであり内側ループがベクトル化されているが，内側ループのベクトル長が短い．そのため，外側ループとのループの入れ換えを行い，ベクトル長 NCELL でベクトル化させるように変更した．変更結果を Fig. 3.34 に示す．尚，このルーチンと構造が似ているサブルーチン STEP2 の DO 600 ループについても同様の変更を行った．

### 3.4.6 サブルーチン MXMASS と PRESMD

これら 2 つのルーチンは，Table 3.1 の V-effect の情報からわかるように，ベクトル実行したことによりスカラ実行よりも逆に遅くなっている場合がある．これらのルーチンの DO ループ毎の動的挙動解析結果を MXMASS について Table 3.10 に，PRESMD について Table 3.11 に示す．これらより，MXMASS については DO 1200，DO 1100 ループの，PRESMD については DO 1200，DO 1100 ループのベクトル長が 1 であることが，スカラ実行よりも遅い原因であることが分かる．しかし，この 2 つのルーチンは全体に対しコストは高くないので，大幅なチューニングを行ってもベクトル化の効果は望めない．そこで，これらのループについては，ベクトル実行を抑止してスカラ実行を行わせるための最適化制御行 "*vocl loop,scalar" を各ループの直前に指定するのみに留めた．指定を行った MXMASS の DO 1100，DO 1200 ループを Fig. 3.35 に，PRESMD の DO 1100，DO 1200 ループを Fig. 3.36 に示す．

## 3.5 計算結果の評価及びベクトル化の効果

ここでは，ベクトル化版 FLOWGR コードの計算結果とベクトル化の効果について述べる．

### 3.5.1 計算結果について

VPP500 において，入力データ TAKEDA と TAKEDAM を用いて，ベクトル化版コードのベクトル実行結果とオリジナル版コードのスカラ実行結果の比較を行った．この結果は完全には一致しない．そこで，VPP コンパイラでデフォルトの設定である演算評価方法の変更を抑止して，ベクトル化版コードのスカラ実行結果とオリジナル版コードのスカラ実行結果を比較したところ，全桁一致した．これより，ベクトル化版コードのベクトル実行結果とオリジナル版コードのスカラ実行結果が一致しないのは，VPP コンパイラによる演算評価方法の変更方法による違い，ベクトル実行するかスカラ実行するかで異なる最適化の違いと浮動小数点演算時の丸めの方法の違い (ベクトル計算では切捨て，スカラ計算では四捨五入) 等，コンパイラと使用するハードウェアの影響が考えられる．また，ユーザよりこの計算結果の違いは問題無いとの評価をいただいた．よって，ベクトル化版コードのベクトル実行結果は妥当であることとした．

### 3.5.2 ベクトル化の効果について

ベクトル化の効果を知るため，入力データ TAKEDA と TAKEDAM を用いたベクトル化版コードについての動的挙動解析を行った．コード全体に対するサブルーチン毎の解析結果を TAKEDA については Table 3.12 に，TAKEDAM については Table 3.13 に示す．これによると，チューニ

ングを行ったサブルーチン PCON はベクトル化率が 92.8% から 99.1% に，ベクトル化効果が 1.4-3.4 倍から 13.7-20.2 倍に向上している．サブルーチン SOLVE はベクトル化率が 29.9% から 66.4% に，ベクトル化効果が 1.4-1.4 倍から 2.4-2.6 倍に向上している．サブルーチン SOR はベクトル実行を抑止したので，ベクトル化率が低下した代わりに，スカラ実行より遅くなることは無くなった．サブルーチン CPMASS はベクトル化率が 22.4% から 96.3% に，ベクトル化効果が 0.9-1.1 倍から 10.2-13.2 倍に向上し，コード全体に対してのこのサブルーチンのコストの割合が小さくなった．

また，ベクトル化版コードの性能を測るため，VPP500 上でベクトル実行した場合の実行時間の測定を行い，M780 上での実行時間と比較した．実行時間は，プログラム終了直前の CPU 時間と開始直後の CPU 時間の差とし，終了直前と開始直後にそれぞれ CPU 時間を取得できるサービスルーチン "clock"を挿入して測定を行った．測定結果を Table 3.14 に示す．尚，測定は入力データ TAKEDA，TAKEDAM の両方について行った．この時の入力データによるタイムステップと CPU 制限時間は，TAKEDA ではそれぞれ 2500 回と 25 分，TAKEDAM ではそれぞれ 10000 回と 25 分で行った．

Table 3.14 より，VPP500 上でのベクトル化版の実行時間を M780 と比較すると，約 8 倍強の速度性能の向上が得られたことがわかる．更に，TAKEDA についてタイムステップ 1000000 回，CPU 制限時間 6000 分とし，長時間ジョブにした場合のベクトル化版の実行時間を測定した．この結果は約 5 時間 36 分であった．従って，VPP500 でベクトル化版コードを用いると，従来の M780 を利用する場合よりも充分に効率良い実行ができるようになったことが言える．

### 3.6 まとめ

今回のベクトル化では，係数行列を解くサブルーチン SOR はオリジナル版のアルゴリズムによりベクトル化不可能であった．また，SOLVE についても同様で，アルゴリズムにより充分な高速化が行えなかった．しかし，従来の M780 上での実行に比べると，約 8 倍強の性能向上が得られたため，今までよりも多ケースの問題についての計算が可能になるものと考える．最後に，本コードはサブルーチン SOR，SOLVE にベクトル化向きのアルゴリズムを採用すれば更に充分に高速化できるコードであると考える．

Table 3.1 Dynamic behavior of original FLOWGR(TAKEDA)

vectorize - total list

| name | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| PCON | 11626 | .3217E10 | 38.9 | .7926E10 | 34.5 | 8 | 92.8 | 1.4- 3.4 | 0 |
| SOLVE | 11605 | .1692E10 | 20.5 | .2319E10 | 10.1 | 27 | 29.9 | 1.4- 1.4 | 0 |
| SOR | 60721 | .1134E10 | 13.7 | .1132E10 | 4.9 | 2 | 46.7 | 0.9- 1.2 | 0 |
| CPMASS | 46420 | .6888E09 | 8.3 | .6658E09 | 2.9 | 2 | 22.4 | 0.9- 1.1 | 0 |
| SET1 | 11626 | .3256E09 | 3.9 | .7255E09 | 3.2 | 3 | 83.2 | 1.6- 2.8 | 0 |
| >SET1E | 11605 | --- | | --- | | --- | --- | --- | --- |
| DIFFSN | 186016 | .2862E09 | 3.5 | .5071E10 | 22.1 | 43 | 99.5 | 14.5- 21.9 | 0 |
| ZERO | 60721 | .1796E09 | 2.2 | .3166E10 | 13.8 | 83 | 99.4 | 14.4- 21.8 | 0 |
| STEP | 2510 | .1741E09 | 2.1 | .1024E10 | 4.5 | 36 | 96.7 | 3.5- 7.8 | 0 |
| MXMASS | 11605 | .1487E09 | 1.8 | .1399E09 | 0.6 | 2 | 40.8 | 0.8- 1.1 | 0 |
| SCOND | 4941050 | .1393E09 | 1.7 | .1393E09 | 0.6 | 1 | 0.0 | 1.0- 1.0 | 0 |
| SHEAT | 4941050 | .1117E09 | 1.3 | .1117E09 | 0.5 | 1 | 0.0 | 1.0- 1.0 | 0 |
| PRESMD | 11605 | 81049320 | 1.0 | 75386080 | 0.3 | 2 | 48.8 | 0.8- 1.1 | 0 |
| ENERGY | 2696 | 30526773 | 0.4 | 38072912 | 0.2 | 2 | 47.3 | 1.1- 1.4 | 0 |
| TWX | 428570 | 19714220 | 0.2 | 19714220 | 0.1 | 1 | 0.0 | 1.0- 1.0 | 0 |
| NEWOLD | 2500 | 12552949 | 0.2 | 37917500 | 0.2 | 7 | 97.1 | 1.6- 4.3 | 0 |
| DEVMAX | 72326 | 12551825 | 0.2 | .1282E09 | 0.6 | 43 | 95.1 | 9.1- 11.4 | 0 |
| VELOC | 11605 | 7220988 | 0.1 | .1367E09 | 0.6 | 45 | 99.8 | 15.2- 23.8 | 0 |
| REACT | 11626 | 7199886 | 0.1 | 99317298 | 0.4 | 43 | 97.8 | 11.8- 16.1 | 0 |
| TIMELP | 1 | 3792788 | 0.0 | 3792788 | 0.0 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DENSTY | 14301 | 1710620 | 0.0 | 29388555 | 0.1 | 43 | 99.3 | 14.1- 21.1 | 0 |
| SETO | 1 | 209070 | 0.0 | 547137 | 0.0 | 6 | 90.8 | 1.5- 3.5 | 0 |
| >SETOR | 20 | --- | | --- | | --- | --- | --- | --- |
| OUT3 | 11 | 81402 | 0.0 | 129734 | 0.0 | 6 | 60.5 | 1.2- 1.8 | 0 |
| READWT | 1 | 74678 | 0.0 | 102628 | 0.0 | 6 | 58.8 | 1.0- 1.6 | 0 |
| OUT4 | 11 | 65129 | 0.0 | 73244 | 0.0 | 52 | 11.7 | 1.1- 1.1 | 0 |
| NETWRK | 1 | 27440 | 0.0 | 128310 | 0.0 | 14 | 88.2 | 4.1- 5.3 | 0 |
| ROUT4 | 11 | 23496 | 0.0 | 23496 | 0.0 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DATAIN | 1 | 7681 | 0.0 | 7681 | 0.0 | 32 | 0.0 | 1.0- 1.0 | 0 |
| INPUT | 1 | 2285 | 0.0 | 2795 | 0.0 | 6 | 30.6 | 1.1- 1.3 | 0 |
| SEE | 10 | 1210 | 0.0 | 1210 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| QUAD | 2 | 863 | 0.0 | 5432 | 0.0 | 10 | 99.4 | 3.8- 8.7 | 0 |
| SUMARY | 1 | 142 | 0.0 | 142 | 0.0 | 10 | 0.0 | 1.0- 1.0 | 0 |
| MAIN | 1 | 37 | 0.0 | 37 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| RESTRT | 1 | 27 | 0.0 | 27 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| BLOCDT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RIN4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| STEP2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (total) | --- | .8274E10 | 100.0 | .2299E11 | 100.0 | --- | 82.9 | 2.0- 3.3 | --- |

Table 3.2 Dynamic behavior of original FLOWGR(TAKEDAM)

vectorize - total list

| name | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| PCON | 21458 | .5930E10 | 39.0 | .1461E11 | 33.6 | 8 | 92.8 | 1.4- 3.4 | 0 |
| SOLVE | 21440 | .3126E10 | 20.5 | .4285E10 | 9.8 | 27 | 29.9 | 1.4- 1.4 | 0 |
| SOR | 107200 | .2000E10 | 13.1 | .1997E10 | 4.6 | 2 | 46.8 | 0.9- 1.2 | 0 |
| CPMASS | 85760 | .1273E10 | 8.4 | .1230E10 | 2.8 | 2 | 22.4 | 0.9- 1.1 | 0 |
| SET1 | 21458 | .5743E09 | 3.8 | .1166E10 | 2.7 | 3 | 82.4 | 1.4- 2.6 | 0 |
| >SET1E | 21440 | --- | | --- | | --- | --- | --- | --- |
| DIFFSN | 343328 | .5282E09 | 3.5 | .9360E10 | 21.5 | 43 | 99.5 | 14.5- 21.9 | 0 |
| ZERO | 107200 | .3170E09 | 2.1 | .5590E10 | 12.8 | 83 | 99.4 | 14.4- 21.8 | 0 |
| STEP2 | 10008 | .2751E09 | 1.8 | .1694E10 | 3.9 | 36 | 97.3 | 3.6- 8.3 | 0 |
| MXMASS | 21440 | .2748E09 | 1.8 | .2585E09 | 0.6 | 2 | 40.8 | 0.8- 1.1 | 0 |
| SCOND | 9119650 | .2572E09 | 1.7 | .2572E09 | 0.6 | 1 | 0.0 | 1.0- 1.0 | 0 |
| SHEAT | 9119650 | .2061E09 | 1.4 | .2061E09 | 0.5 | 1 | 0.0 | 1.0- 1.0 | 0 |
| PRESMD | 21440 | .1497E09 | 1.0 | .1393E09 | 0.3 | 2 | 48.8 | 0.8- 1.1 | 0 |
| DENSTY | 872120 | .1043E09 | 0.7 | .1792E10 | 4.1 | 43 | 99.3 | 14.1- 21.1 | 0 |
| TWX | 1702975 | 78336850 | 0.5 | 78336850 | 0.2 | 1 | 0.0 | 1.0- 1.0 | 0 |
| NEWOLD | 10000 | 50211795 | 0.3 | .1517E09 | 0.3 | 7 | 97.1 | 1.6- 4.3 | 0 |
| TIMELP | 1 | 24539858 | 0.2 | 24539858 | 0.1 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DEVMAX | 128640 | 21927847 | 0.1 | .2346E09 | 0.5 | 43 | 95.6 | 9.5- 12.0 | 0 |
| VELOC | 21440 | 13340628 | 0.1 | .2526E09 | 0.6 | 45 | 99.8 | 15.2- 23.8 | 0 |
| REACT | 21458 | 13315303 | 0.1 | .1838E09 | 0.4 | 43 | 97.8 | 11.8- 16.2 | 0 |
| SET0 | 1 | 186012 | 0.0 | 483117 | 0.0 | 6 | 90.6 | 1.5- 3.5 | 0 |
| >SETOR | 17 | --- | | --- | | --- | --- | --- | --- |
| OUT3 | 11 | 79963 | 0.0 | 125994 | 0.0 | 6 | 59.4 | 1.2- 1.8 | 0 |
| READWT | 1 | 71126 | 0.0 | 94812 | 0.0 | 6 | 56.4 | 1.0- 1.5 | 0 |
| OUT4 | 11 | 65148 | 0.0 | 73064 | 0.0 | 52 | 11.4 | 1.1- 1.1 | 0 |
| NETWRK | 1 | 27440 | 0.0 | 128310 | 0.0 | 14 | 88.2 | 4.1- 5.3 | 0 |
| ROUT4 | 11 | 25366 | 0.0 | 25366 | 0.0 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DATAIN | 1 | 7681 | 0.0 | 7681 | 0.0 | 32 | 0.0 | 1.0- 1.0 | 0 |
| INPUT | 1 | 2286 | 0.0 | 2796 | 0.0 | 6 | 30.5 | 1.1- 1.3 | 0 |
| SEE | 10 | 1210 | 0.0 | 1210 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| QUAD | 2 | 863 | 0.0 | 5432 | 0.0 | 10 | 99.4 | 3.8- 8.7 | 0 |
| SUMARY | 1 | 142 | 0.0 | 142 | 0.0 | 10 | 0.0 | 1.0- 1.0 | 0 |
| MAIN | 1 | 37 | 0.0 | 37 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| RESTRT | 1 | 27 | 0.0 | 27 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| BLOCDT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ENERGY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RIN4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| STEP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (total) | --- | .1522E11 | 100.0 | .4351E11 | 100.0 | --- | 83.4 | 2.0- 3.4 | --- |

Table 3.3 Dynamic behavior of DO loops in subroutine PCON

vectorize - loop list ------- PCON

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000089-00000093 | 521 | do | V | 4941050 | .2014E10 | 62.6 | .5238E10 | 66.1 | 5 | 100.0 | 1.3- 3.9 |
| 00000078-00000084 | 510 | do | V | 988210 | .5169E09 | 16.1 | .1344E10 | 17.0 | 5 | 100.0 | 1.3- 3.9 |
| 00000086-00000097 | 520 | do | S | 988210 | .3113E09 | 9.7 | .3113E09 | 3.9 | 5 | 0.0 | 1.0- 1.0 |
| 00000067-00000069 | 310 | do | S | 988210 | .1235E09 | 3.8 | .1235E09 | 1.6 | 5 | 0.0 | 1.0- 1.0 |
| 00000053-00000055 | 130 | do | V | 46504 | 72975508 | 2.3 | .1067E09 | 1.3 | 85 | 33.3 | 1.5- 1.5 |
| 00000031-00000031 | 10 | do | V | 988210 | 62713327 | 1.9 | .1631E09 | 2.1 | 5 | 100.0 | 1.3- 3.9 |
| 00000075-00000099 | 500 | do | S | 11626 | 37551980 | 1.2 | 37551980 | 0.5 | 85 | 0.0 | 1.0- 1.0 |
| 00000060-00000060 | 200 | do | V | 988210 | 25845492 | 0.8 | 67198280 | 0.8 | 4 | 100.0 | 1.3- 3.9 |
| 00000050-00000050 | 125 | do | V | 186016 | 22703491 | 0.7 | .4427E09 | 5.6 | 85 | 100.0 | 15.6- 24.7 |
| 00000065-00000070 | 300 | do | S | 11626 | 9882100 | 0.3 | 9882100 | 0.1 | 85 | 0.0 | 1.0- 1.0 |
| 00000059-00000060 | 200 | do | S | 11626 | 5929260 | 0.2 | 5929260 | 0.1 | 85 | 0.0 | 1.0- 1.0 |
| 00000030-00000031 | 10 | do | S | 11626 | 4941050 | 0.2 | 4941050 | 0.1 | 85 | 0.0 | 1.0- 1.0 |
| 00000047-00000051 | 120 | do | S | 46504 | 2929752 | 0.1 | 2929752 | 0.0 | 5 | 0.0 | 1.0- 1.0 |
| 00000106-00000114 | 610 | do | V | 11626 | 2027097 | 0.1 | 39528400 | 0.5 | 85 | 100.0 | 15.6- 24.7 |
| 00000045-00000045 | 110 | do | V | 46504 | 1418968 | 0.0 | 27669880 | 0.3 | 85 | 100.0 | 15.6- 24.7 |
| 00000044-00000056 | 100 | do | S | 11626 | 837072 | 0.0 | 837072 | 0.0 | 4 | 0.0 | 1.0- 1.0 |
| 00000001-00000118 | ----- | total | | 11626 | 825446 | 0.0 | 825446 | 0.0 | 1 | 0.0 | 1.0- 1.0 |
| 00000034-00000035 | 20 | do | S | 0 | 0 | 0.0 | 0 | 0.0 | 0 | 0 | 0 |
| 00000035-00000035 | 20 | do | V | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 |

Table 3.4 Dynamic behavior of DO loops in subroutine SOLVE

vectorize - loop list ------- SOLVE

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000025-00000028 | 130 | do | S | 1613095 | .6363E09 | 37.6 | .6363E09 | 27.4 | 27 | 0.0 | 1.0- 1.0 |
| 00000017-00000029 | 140 | do | S | 986425 | .4632E09 | 27.4 | .4632E09 | 20.0 | 27 | 0.0 | 1.0- 1.0 |
| 00000053-00000056 | 240 | do | S | 986425 | .3406E09 | 20.1 | .3406E09 | 14.7 | 27 | 0.0 | 1.0- 1.0 |
| 00000039-00000044 | 220 | do | V | 986425 | 90079126 | 5.3 | .4664E09 | 20.1 | 27 | 89.3 | 4.5- 5.7 |
| 00000012-00000014 | 110 | do | V | 963215 | 75500121 | 4.5 | .3266E09 | 14.1 | 27 | 85.1 | 3.9- 4.7 |
| 00000008-00000030 | 150 | do | | 11605 | 41975285 | 2.5 | 41975285 | 1.8 | 85 | 0.0 | 1.0- 1.0 |
| 00000035-00000045 | 230 | do | S | 11605 | 33143880 | 2.0 | 33143880 | 1.4 | 85 | 0.0 | 1.0- 1.0 |
| 00000051-00000057 | 250 | do | S | 11605 | 10850675 | 0.6 | 10850675 | 0.5 | 85 | 0.0 | 1.0- 1.0 |
| 00000001-00000065 | ----- | total | | 11605 | 371360 | 0.0 | 371360 | 0.0 | 1 | 0.0 | 1.0- 1.0 |

Table 3.5 Dynamic behavior of DO loops in subroutine SOR

vectorize - loop list ------- SOR ----------

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000030-00000045 | 2200 | do | S | 60721 | .3364E09 | 29.7 | .3364E09 | 29.7 | 85 | 0.0 | 1.0- 1.0 |
| 00000015-00000028 | 2100 | do | S | 60721 | .2632E09 | 23.2 | .2632E09 | 23.2 | 85 | 0.0 | 1.0- 1.0 |
| 00000017-00000020 | 2101 | do | V | 5161285 | .1322E09 | 11.7 | .1146E09 | 10.1 | 1 | 99.8 | 0.7- 1.3 |
| 00000032-00000035 | 2201 | do | V | 5161285 | .1322E09 | 11.7 | .1146E09 | 10.1 | 1 | 99.8 | 0.7- 1.3 |
| 00000022-00000025 | 2102 | do | V | 5161285 | .1316E09 | 11.6 | .1141E09 | 10.1 | 1 | 99.9 | 0.7- 1.3 |
| 00000037-00000040 | 2202 | do | V | 5161285 | .1316E09 | 11.6 | .1141E09 | 10.1 | 1 | 99.9 | 0.7- 1.3 |
| 00000009-00000009 | 1000 | do | V | 60721 | 1852769 | 0.2 | 36128995 | 3.2 | 85 | 100.0 | 15.6- 24.7 |
| 00000051-00000051 | 3000 | do | V | 60721 | 1852769 | 0.2 | 36128995 | 3.2 | 85 | 100.0 | 15.6- 24.7 |
| 00000001-00000054 | ----- | total | | 60721 | 1639467 | 0.1 | 1639467 | 0.1 | 1 | 0.0 | 1.0- 1.0 |
| 00000013-00000047 | 2000 | do | | 60721 | 1396583 | 0.1 | 1396583 | 0.1 | 1 | 0.0 | 1.0- 1.0 |

Table 3.6 Dynamic behavior of DO loops in subroutine CPMASS

vectorize - loop list ------- CPMASS ----------

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000022-00000063 | 1000 | do | S | 46420 | .2953E09 | 42.9 | .2953E09 | 44.4 | 85 | 0.0 | 1.0- 1.0 |
| 00000039-00000043 | 1400 | do | M | 3806440 | .1968E09 | 28.6 | .1849E09 | 27.8 | 1 | 41.7 | 0.8- 1.1 |
| 00000032-00000036 | 1300 | do | M | 3806440 | .1944E09 | 28.2 | .1833E09 | 27.5 | 1 | 39.4 | 0.8- 1.1 |
| 00000001-00000066 | ----- | total | | 46420 | 2274580 | 0.3 | 2274580 | 0.3 | 1 | 0.0 | 1.0- 1.0 |

Table 3.7 Dynamic behavior of statements in subroutine CPMASS (1/2)

```
vectorize - statement list -- CPMASS ------------------------------------------------------------------------------------------
isn       ex-count   true    v-cost    s-cost v o c--------1---------2---------3---------4---------5---------6---------7--
00000001     46420           1810380   1810380          SUBROUTINE CPMASS(KNUM,FMAT,ANS,NCELL,NWIDTH,ICH,
00000002                                               >                 WOLD,RHOLD,RHNEW,RHU,RDN,QNEW,QS)
00000003                                          C*
00000004                                                IMPLICIT REAL*8 (A-H,O-Z)
00000005                                                PARAMETER (NSZ=200)
00000006                                                COMMON /CRPCON/ NO2,NCO,NCO2
00000007                                                COMMON /PARINT/ TI,TF,DELT,CHI
00000008                                                COMMON /POINTR/ IPL(NSZ),IPR(NSZ),ICP(NSZ),ICM(NSZ),IJP(NSZ),
00000009                                               >                IJM(NSZ)
00000010                                                COMMON /CELJUN/ NRVOL(NSZ),NLVOL(NSZ),CLN(NSZ),CD(NSZ),CTHE(NSZ),
00000011                                               >       NJL(NSZ),NJR(NSZ),NSI(NSZ),DSI(NSZ),IJUN(NSZ),ICEL(NSZ),
00000012                                               >       DST(NSZ),XJ(NSZ),XC(NSZ),DJ(NSZ),VCEL(NSZ),LB(9),RAT(NSZ),
00000013                                               >       GV(NSZ),AL(NSZ),COSR(NSZ),COSL(NSZ)
00000014                                                 DIMENSION      WOLD(1),RHOLD(1),RHNEW(1),
00000015                                               >                RHU(1),RDN(1),QNEW(1),QS(1),ICH(1)
00000016                                                REAL*8 ANS(NCELL),FMAT(NCELL,2*NWIDTH+1)
00000017                                          C*
00000018     46420            92840     92840          CALL ZERO(FMAT,ANS,NWIDTH,NCELL)
00000019                                          C*
00000020     46420            46420     46420          NWP = NWIDTH + 1
00000021     46420           232100    232100 S        DO 1000 I = 1, NCELL
00000022   3945700         67076900  67076900 V          ANS(I) = VCEL(I)*WOLD(I)*RHOLD(I)+VCEL(I)*DELT*QNEW(I)
00000023   3945700         39457000  39457000 M          FMAT(I,NWP) = VCEL(I)*RHNEW(I)
00000024   3945700   100.0 23674200  23674200 V          IF(ICH(NSI(I)).NE.0)THEN
00000025   3945700    50.0 15782800  15782800 V            IF(KNUM.EQ.NO2.OR.KNUM.EQ.NCO)THEN
00000026   1972850         33538450  33538450 M              FMAT(I,NWP) = FMAT(I,NWP) - VCEL(I)*DELT*QS(I)
00000027   3945700                0         0 V            ENDIF
00000028   3945700                0         0 V          ENDIF
00000029   3945700    96.5 15782800  15782800 V          IF(ICEL(I).EQ.0) THEN
00000030                                          C*PLUS
00000031   3806440         41870840  41870840 M            DO 1300 L = IPR(I),IPR(I+1)-1
00000032   3899280         17996677  15597120 V              NWPP = NWP+ICP(L)-I
00000033   3899280          8998338   7798560 V              JP   = IJP(L)
00000034   3899280         79785268  74086320 M              FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0)
00000035   3899280         74086320  74086320 S              FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP)
00000036   3899280         13497508  11697840 V    1300      CONTINUE
```

Table 3.7 Dynamic behavior of statements in subroutine CPMASS (2/2)

| | | | | | | |
|---|---|---|---|---|---|---|
| 00000037 | | | | | | C*MINUS |
| 00000038 | 3806440 | | 41870840 | 41870840 | M | DO 1400 L = IPL(I),IPL(I+1)-1 |
| 00000039 | 3852860 | | 17782431 | 15411440 | V | NWPM = NWP+ICM(L)-I |
| 00000040 | 3852860 | | 8891215 | 7705720 | V | JM = IJM(L) |
| 00000041 | 3852860 | | 74686209 | 69351480 | M | FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0) |
| 00000042 | 3852860 | | 73204340 | 73204340 | S | FMAT(I,NWP) = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM) |
| 00000043 | 3852860 | | 22228038 | 19264300 | V | 1400 CONTINUE |
| 00000044 | 139260 | 66.7 | 1114080 | 1114080 | V | ELSEIF(ICEL(I).EQ.-1.OR.ICEL(I).EQ.1) THEN |
| 00000045 | | | | | | C*BREAK POINT |
| 00000046 | 92840 | | 464200 | 464200 | M | FMAT(I,NWP) = 1.0 |
| 00000047 | 92840 | | 557040 | 557040 | V | ANS(I) = WOLD(I) |
| 00000048 | 46420 | 0.0 | 185680 | 185680 | V | ELSEIF(ICEL(I).EQ.-2) THEN |
| 00000049 | | | | | | C*CLOSED END (PLUS) |
| 00000050 | 0 | | 0 | 0 | V | IPRI = IPR(I) |
| 00000051 | 0 | | 0 | 0 | V | JP = IJP(IPRI) |
| 00000052 | 0 | | 0 | 0 | S | NWPP = NWP + 1 |
| 00000053 | 0 | | 0 | 0 | M | FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0) |
| 00000054 | 0 | | 0 | 0 | M | FMAT(I,NWP) = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP) |
| 00000055 | 46420 | 100.0 | 185680 | 185680 | V | ELSEIF(ICEL(I).EQ. 2) THEN |
| 00000056 | | | | | | C*CLOSED END (MINUS) |
| 00000057 | 46420 | | 92840 | 92840 | V | IPLI = IPL(I) |
| 00000058 | 46420 | | 92840 | 92840 | V | JM = IJM(IPLI) |
| 00000059 | 46420 | | 46420 | 46420 | S | NWPM = NWP - 1 |
| 00000060 | 46420 | | 835560 | 835560 | M | FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0) |
| 00000061 | 46420 | | 881980 | 881980 | M | FMAT(I,NWP) = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM) |
| 00000062 | 3945700 | | 0 | 0 | V | ENDIF |
| 00000063 | 3945700 | | 11837100 | 11837100 | V | 1000 CONTINUE |
| 00000064 | | | | | | C* |
| 00000065 | 46420 | | 0 | 0 | | RETURN |
| 00000066 | 46420 | | 92840 | 92840 | | END |

Table 3.8 Dynamic behavior of DO loops in subroutine SET1

vectorize - loop list ------- SET1 ----------

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000025-00000034 | 210 | do | V | 1023088 | .1127E09 | 34.6 | .2930E09 | 40.4 | 4 | 100.0 | 1.3- 3.9 |
| 00000086-00000092 | 400 | do | S | 23231 | .1047E09 | 32.1 | .1047E09 | 14.4 | 85 | 0.0 | 1.0- 1.0 |
| 00000088-00000088 | 410 | do | V | 1974635 | 39644595 | 12.2 | 34358649 | 4.7 | 1 | 100.0 | 0.7- 1.3 |
| 00000090-00000090 | 420 | do | V | 1974635 | 39188910 | 12.0 | 33963722 | 4.7 | 1 | 100.0 | 0.7- 1.3 |
| 00000022-00000051 | 200 | do | V | 11626 | 12337869 | 3.8 | .1270E09 | 17.5 | 88 | 95.2 | 9.1- 11.5 |
| 00000080-00000083 | 300 | do | V | 23231 | 11899037 | 3.7 | 42930888 | 5.9 | 88 | 76.2 | 3.5- 3.7 |
| 00000095-00000113 | 500 | do | V | 23231 | 4562977 | 1.4 | 88978046 | 12.3 | 88 | 100.0 | 15.6- 24.7 |
| 00000001-00000116 | ----- | total | | 23231 | 615737 | 0.2 | 615737 | 0.1 | 1 | 0.0 | 1.0- 1.0 |
| 00000054-00000074 | 250 | do | V | 0 | 0 | 0.0 | 0 | 0.0 | 0 | 0 | 0 |
| 00000057-00000066 | 260 | do | V | 0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 |

Table 3.9 Dynamic behavior of DO loops in subroutine STEP

vectorize - loop list ------- STEP ----------

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000131-00000133 | 365 | do | V | 986425 | 59185500 | 34.0 | .1539E09 | 15.0 | 4 | 100.0 | 1.3- 3.9 |
| 00000169-00000171 | 600 | do | V | 986425 | 57667923 | 33.1 | .1499E09 | 14.6 | 4 | 100.0 | 1.3- 3.9 |
| 00000120-00000122 | 310 | do | V | 46420 | 13611094 | 7.8 | 69761440 | 6.8 | 85 | 84.8 | 4.9- 5.4 |
| 00000128-00000137 | 360 | do | V | 11605 | 8641857 | 5.0 | 57305490 | 5.6 | 85 | 89.5 | 6.2- 7.1 |
| 00000058-00000058 | 110 | do | V | 11605 | 8114821 | 4.7 | .3798E09 | 37.1 | 4675 | 100.0 | 40.3- 54.6 |
| 00000114-00000116 | 308 | do | V | 46420 | 7513749 | 4.3 | 48690670 | 4.8 | 85 | 89.1 | 6.0- 6.9 |
| 00000168-00000171 | 600 | do | S | 11605 | 5918550 | 3.4 | 5918550 | 0.6 | 85 | 0.0 | 1.0- 1.0 |
| 00000091-00000097 | 260 | do | V | 11605 | 2836626 | 1.6 | 52975109 | 5.2 | 88 | 99.8 | 15.1- 23.4 |
| 00000214-00000219 | 800 | do | S | 2510 | 2444740 | 1.4 | 2444740 | 0.2 | 88 | 0.0 | 1.0- 1.0 |
| 00000106-00000123 | 300 | do | | 11605 | 1456212 | 0.8 | 1456212 | 0.1 | 4 | 0.0 | 1.0- 1.0 |
| 00000047-00000180 | 8888 | do | | 2510 | 1415913 | 0.8 | 1415913 | 0.1 | 5 | 0.0 | 1.0- 1.0 |
| 00000069-00000074 | 200 | do | V | 11605 | 1331302 | 0.8 | 25960385 | 2.5 | 88 | 100.0 | 15.6- 24.7 |
| 00000054-00000054 | 100 | do | V | 11605 | 1163476 | 0.7 | 22687775 | 2.2 | 85 | 100.0 | 15.6- 24.7 |
| 00000082-00000082 | 250 | do | V | 11605 | 1163476 | 0.7 | 22687775 | 2.2 | 85 | 100.0 | 15.6- 24.7 |
| 00000076-00000076 | 220 | do | V | 11605 | 628455 | 0.4 | 12254880 | 1.2 | 88 | 100.0 | 15.6- 24.7 |
| 00000064-00000064 | 120 | do | V | 11605 | 354101 | 0.2 | 6904975 | 0.7 | 85 | 100.0 | 15.6- 24.7 |
| 00000163-00000163 | 510 | do | V | 11605 | 354101 | 0.2 | 6904975 | 0.7 | 85 | 100.0 | 15.6- 24.7 |
| 00000001-00000222 | ----- | total | | 2510 | 173544 | 0.1 | 173544 | 0.0 | 1 | 0.0 | 1.0- 1.0 |
| 00000153-00000153 | 410 | do | V | 2696 | 164525 | 0.1 | 3208240 | 0.3 | 85 | 100.0 | 15.6- 24.7 |

Table 3.10 Dynamic behavior of DO loops in subroutine MXMASS

vectorize - loop list ------- MXMASS

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000028-00000033 | 1200 | do | M | 951610 | 55570096 | 37.4 | 51050395 | 36.5 | 1 | 57.5 | 0.8- 1.2 |
| 00000021-00000026 | 1100 | do | M | 951610 | 53990031 | 36.3 | 49715820 | 35.5 | 1 | 55.9 | 0.8- 1.1 |
| 00000018-00000051 | 1000 | do | S | 11605 | 38888355 | 26.1 | 38888355 | 27.8 | 85 | 0.0 | 1.0- 1.0 |
| 00000001-00000054 | ----- | total | | 11605 | 278520 | 0.2 | 278520 | 0.2 | 1 | 0.0 | 1.0- 1.0 |

Table 3.11 Dynamic behavior of DO loops in subroutine PRESMD

vectorize - loop list ------- PRESMD

| isn | do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000016-00000038 | 1000 | do | S | 11605 | 38366130 | 47.3 | 38366130 | 50.9 | 85 | 0.0 | 1.0- 1.0 |
| 00000023-00000025 | 1200 | do | V | 951610 | 22228038 | 27.4 | 19264300 | 25.6 | 1 | 100.0 | 0.7- 1.3 |
| 00000019-00000021 | 1100 | do | V | 951610 | 20246262 | 25.0 | 17546760 | 23.3 | 1 | 100.0 | 0.7- 1.3 |
| 00000001-00000041 | ----- | total | | 11605 | 208890 | 0.3 | 208890 | 0.3 | 1 | 0.0 | 1.0- 1.0 |

Table 3.12 Dynamic behavior of tuned FLOWGR(TAKEDA)

vectorize - total list

| name | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| SOR | 104795 | .1876E10 | 29.7 | .1994E10 | 4.9 | 2 | 6.3 | 1.1- 1.1 | 0 |
| SOLVE | 20412 | .1630E10 | 25.8 | .4080E10 | 9.9 | 27 | 66.4 | 2.4- 2.6 | 0 |
| PCON | 20433 | .9189E09 | 14.6 | .1527E11 | 37.2 | 71 | 99.1 | 13.7- 20.2 | 0 |
| DIFFSN | 326928 | .5030E09 | 8.0 | .8913E10 | 21.7 | 43 | 99.5 | 14.5- 21.9 | 0 |
| ZERO | 104795 | .3099E09 | 4.9 | .5465E10 | 13.3 | 83 | 99.4 | 14.4- 21.8 | 0 |
| SET1 | 20433 | .2722E09 | 4.3 | .1282E10 | 3.1 | 13 | 97.1 | 2.7- 6.5 | 0 |
| >SET1E | 20412 | --- | | --- | | --- | --- | --- | --- |
| MXMASS | 20412 | .2461E09 | 3.9 | .2461E09 | 0.6 | 2 | 0.0 | 1.0- 1.0 | 0 |
| STEP | 2510 | .2041E09 | 3.2 | .1785E10 | 4.3 | 62 | 97.5 | 5.6- 11.2 | 0 |
| PRESMD | 20412 | .1326E09 | 2.1 | .1326E09 | 0.3 | 2 | 0.0 | 1.0- 1.0 | 0 |
| CPMASS | 81648 | .1033E09 | 1.6 | .1201E10 | 2.9 | 30 | 96.3 | 10.2- 13.2 | 0 |
| ENERGY | 2735 | 30968370 | 0.5 | 38623670 | 0.1 | 2 | 47.3 | 1.1- 1.4 | 0 |
| DEVMAX | 125207 | 21435589 | 0.3 | .2268E09 | 0.6 | 43 | 95.4 | 9.4- 11.9 | 0 |
| TWX | 428570 | 19714220 | 0.3 | 19714220 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| VELOC | 20412 | 12700974 | 0.2 | .2405E09 | 0.6 | 45 | 99.8 | 15.2- 23.8 | 0 |
| NEWOLD | 2500 | 12552949 | 0.2 | 37917500 | 0.1 | 7 | 97.1 | 1.6- 4.3 | 0 |
| REACT | 20433 | 9739064 | 0.2 | .1177E09 | 0.3 | 43 | 96.7 | 10.5- 13.8 | 0 |
| TIMELP | 1 | 3792788 | 0.1 | 3792788 | 0.0 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DENSTY | 23147 | 2768737 | 0.0 | 47567085 | 0.1 | 43 | 99.3 | 14.1- 21.1 | 0 |
| SETO | 1 | 209070 | 0.0 | 547137 | 0.0 | 6 | 90.8 | 1.5- 3.5 | 0 |
| >SETOR | 20 | --- | | --- | | --- | --- | --- | --- |
| OUT3 | 11 | 81402 | 0.0 | 129734 | 0.0 | 6 | 60.5 | 1.2- 1.8 | 0 |
| READWT | 1 | 74677 | 0.0 | 102627 | 0.0 | 6 | 58.8 | 1.0- 1.6 | 0 |
| OUT4 | 11 | 65129 | 0.0 | 73244 | 0.0 | 52 | 11.7 | 1.1- 1.1 | 0 |
| NETWRK | 1 | 31032 | 0.0 | 131902 | 0.0 | 11 | 85.8 | 3.8- 4.7 | 0 |
| ROUT4 | 11 | 23496 | 0.0 | 23496 | 0.0 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DATAIN | 1 | 7681 | 0.0 | 7681 | 0.0 | 32 | 0.0 | 1.0- 1.0 | 0 |
| INPUT | 1 | 2285 | 0.0 | 2795 | 0.0 | 6 | 30.6 | 1.1- 1.3 | 0 |
| SEE | 10 | 1210 | 0.0 | 1210 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| QUAD | 2 | 863 | 0.0 | 5432 | 0.0 | 10 | 99.4 | 3.8- 8.7 | 0 |
| SUMARY | 1 | 142 | 0.0 | 142 | 0.0 | 10 | 0.0 | 1.0- 1.0 | 0 |
| MAIN | 1 | 44 | 0.0 | 44 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| RESTRT | 1 | 27 | 0.0 | 27 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| BLOCDT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SHEAT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SCOND | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RIN4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| STEP2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (total) | --- | .6310E10 | 100.0 | .4111E11 | 100.0 | --- | 90.2 | 5.8- 7.1 | --- |

Table 3.13 Dynamic behavior of tuned FLOWGR(TAKEDAM)

vectorize - total list

| name | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| SOR | 119500 | .2140E10 | 28.9 | .2275E10 | 4.6 | 2 | 6.3 | 1.1- 1.1 | 0 |
| SOLVE | 23900 | .1908E10 | 25.7 | .4777E10 | 9.7 | 27 | 66.4 | 2.4- 2.6 | 0 |
| PCON | 23918 | .1074E10 | 14.5 | .1785E11 | 36.2 | 71 | 99.1 | 13.7- 20.2 | 0 |
| DIFFSN | 382688 | .5888E09 | 7.9 | .1043E11 | 21.2 | 43 | 99.5 | 14.5- 21.9 | 0 |
| ZERO | 119500 | .3534E09 | 4.8 | .6232E10 | 12.6 | 83 | 99.4 | 14.4- 21.8 | 0 |
| SET1 | 23918 | .2906E09 | 3.9 | .1310E10 | 2.7 | 13 | 98.2 | 2.5- 6.4 | 0 |
| >SET1E | 23900 | --- | | --- | | --- | --- | --- | --- |
| MXMASS | 23900 | .2882E09 | 3.9 | .2882E09 | 0.6 | 2 | 0.0 | 1.0- 1.0 | 0 |
| STEP2 | 10008 | .1901E09 | 2.6 | .1875E10 | 3.8 | 62 | 98.0 | 6.3- 12.7 | 0 |
| PRESMD | 23900 | .1553E09 | 2.1 | .1553E09 | 0.3 | 2 | 0.0 | 1.0- 1.0 | 0 |
| CPMASS | 95600 | .1210E09 | 1.6 | .1406E10 | 2.9 | 30 | 96.3 | 10.2- 13.2 | 0 |
| DENSTY | 874580 | .1046E09 | 1.4 | .1797E10 | 3.6 | 43 | 99.3 | 14.1- 21.1 | 0 |
| TWX | 1702975 | 78336850 | 1.1 | 78336850 | 0.2 | 1 | 0.0 | 1.0- 1.0 | 0 |
| NEWOLD | 10000 | 50211795 | 0.7 | .1517E09 | 0.3 | 7 | 97.1 | 1.6- 4.3 | 0 |
| TIMELP | 1 | 24539858 | 0.3 | 24539858 | 0.0 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DEVMAX | 143400 | 24054112 | 0.3 | .2680E09 | 0.5 | 43 | 95.9 | 9.8- 12.6 | 0 |
| VELOC | 23900 | 14871315 | 0.2 | .2816E09 | 0.6 | 45 | 99.8 | 15.2- 23.8 | 0 |
| REACT | 23918 | 10117535 | 0.1 | .1128E09 | 0.2 | 43 | 95.9 | 9.8- 12.6 | 0 |
| SETO | 1 | 186012 | 0.0 | 483117 | 0.0 | 6 | 90.6 | 1.5- 3.5 | 0 |
| >SETOR | 17 | --- | | --- | | --- | --- | --- | --- |
| OUT3 | 11 | 79963 | 0.0 | 125994 | 0.0 | 6 | 59.4 | 1.2- 1.8 | 0 |
| READWT | 1 | 71125 | 0.0 | 94811 | 0.0 | 6 | 56.4 | 1.0- 1.5 | 0 |
| OUT4 | 11 | 65148 | 0.0 | 73064 | 0.0 | 52 | 11.4 | 1.1- 1.1 | 0 |
| NETWRK | 1 | 31032 | 0.0 | 131902 | 0.0 | 11 | 85.8 | 3.8- 4.7 | 0 |
| ROUT4 | 11 | 25366 | 0.0 | 25366 | 0.0 | 44 | 0.0 | 1.0- 1.0 | 0 |
| DATAIN | 1 | 7681 | 0.0 | 7681 | 0.0 | 32 | 0.0 | 1.0- 1.0 | 0 |
| INPUT | 1 | 2286 | 0.0 | 2796 | 0.0 | 6 | 30.5 | 1.1- 1.3 | 0 |
| SEE | 10 | 1210 | 0.0 | 1210 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| QUAD | 2 | 863 | 0.0 | 5432 | 0.0 | 10 | 99.4 | 3.8- 8.7 | 0 |
| SUMARY | 1 | 142 | 0.0 | 142 | 0.0 | 10 | 0.0 | 1.0- 1.0 | 0 |
| MAIN | 1 | 37 | 0.0 | 37 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| RESTRT | 1 | 27 | 0.0 | 27 | 0.0 | 1 | 0.0 | 1.0- 1.0 | 0 |
| BLOCDT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ENERGY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| OUT5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SHEAT | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SCOND | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RIN4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| STEP | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (total) | --- | .7418E10 | 100.0 | .4932E11 | 100.0 | --- | 90.5 | 5.9- 7.3 | --- |

Table 3.14 Measured results of processing time

| 入力データ | M780 | VPP500 （オリジナル版） | VP500 （ベクトル化版） |
|---|---|---|---|
| TAKEDA | 900 sec | 477 sec | 111 sec |
| TAKEDAM | 1091 sec | 562 sec | 132 sec |

ENERGY, NEWOLD, OUT5, SET0, SET1, STEP, STEP2, TIMELP の各ルーチンにおいて

```
      COMMON /VALNEW/ TMNEW,TNEW(NSZ),VNEW(NSZ), . . . .
                      . . . . . .

                      . . . . . .
                     QNEW(NSZ,8),FCOFN(NSZ),CNEW(NSZ),QS(NSZ,7)
                                                      ⇧
                                                    変更前
                        ↓
      COMMON /VALNEW/ TMNEW,TNEW(NSZ),VNEW(NSZ), . . . .
                      . . . . . .

                      . . . . . .
                     QNEW(NSZ,8),FCOFN(NSZ),CNEW(NSZ),QS(NSZ,8)
                                                      ⇧
                                                    変更後
```

Fig. 3.1 Modification of common block VALNEW

```
                .
                .
                .
C.VPP-modify
      NMAX=1                    ← 追加した初期化処理
C.VPP-modify
      VMAX = 0.0D0
      DO 4000 N = 1,NJUNC
        IF(ABS(V(N)).GT.VMAX) THEN
          VMAX = ABS(V(N))
          NMAX = N
        ENDIFF
 4000 CONTINUE
                .
                .
                .
```

Fig. 3.2 Modification of variable "NMAX" in subroutine OUT4

STEP, STEP2 の各ルーチンの先頭において

```
      PARAMETER(NSZ=200,RLXT=0.1)
C.VPP-modify
      PARAMETER(NSZZ=NSZ*8)
C.VPP-modify
                .
                .
                .
                .
C.VPP-modify
      data qtmp /NSZZ*0.0d0/
      data stmp /NSZZ*0.0d0/
C.VPP-modify
                .
                .
                .
```

Fig. 3.3 Modification of two variables "qtmp" and "stmp" in subroutine STEP and STEP2

```
 1  m      DO 500 I = 1,NCELL
 2  m        ETA(I) = 0.0
 3  s        FLAM(I)= 0.0
 4  v        DO 510 K = 1,NGAS
 5  v          FMA    = FMS(KINDG(K))
 6  v          EKA    = EK(KINDG(K))
 7  v          SGA    = SIGMA(KINDG(K))
 8  v          TASA   = T(I)/EKA
 9  v          OMVA   = AE/TASA**BE+CE*EXP(-DE*TASA)+EE*EXP(-FE*TASA)
10  v          ETAA(K)= 26.69E-7*SQRT(FMA*T(I))/SGA**2/OMVA
11  v  510   CONTINUE
12  s        DO 520 K = 1,NGAS
13  s          FMA    = FMS(KINDG(K))
14  s          DENOM  = 0.0D0
15  v          DO 521 KK = 1,NGAS
16  v            FMB=FMS(KINDG(KK))
17  v            PAB = (1.0+(ETAA(K)/ETAA(KK))**0.5*(FMB/FMA)**0.25)**2
18        >          / SQRT(8.0*(1.0+(FMA/FMB)))
19  v            DENOM = DENOM + X(I,KK)*PAB
20  v  521     CONTINUE
21  m          ETA(I) = ETA(I)+X(I,K)*ETAA(K)/DENOM
22  s          FLAMA  = SCOND(FMA,CPX(I,K),R,ETAA(K),KINDG(K))
23  m          FLAM(I)= FLAM(I) + X(I,K)*FLAMA/DENOM
24  v  520   CONTINUE
25  m        ALPHA(I)= FNU(NSI(I))*FLAM(I)/CD(I)
26  v  500 CONTINUE
```

Fig. 3.4 DO 500 loop in subroutine PCON

```
      C.ORG DIMENSION CPX(NSZ,8),ETAA(8),DAB(NSZ),FCOF(NSZ)
      C.VPP
            DIMENSION CPX(NSZ,8),ETAAV(NSZ,8),DAB(NSZ),FCOF(NSZ) ← 新たな
      C.VPP                                                        配列宣言
 1          DO 500 I = 1,NCELL
 2            DO 510 K = 1,NGAS
 3              FMA    = FMS(KINDG(K))
 4              EKA    = EK(KINDG(K))
 5              SGA    = SIGMA(KINDG(K))
 6              TASA   = T(I)/EKA
 7              OMVA   = AE/TASA**BE+CE*EXP(-DE*TASA)+EE*EXP(-FE*TASA)
 8              ETAAV(I,K)= 26.69E-7*SQRT(FMA*T(I))/SGA**2/OMVA
 9      510   CONTINUE
10      500 CONTINUE
11          DO 501 I=1,NCELL
12            ETA(I) = 0.0
13            FLAM(I)= 0.0
14            DO 520 K = 1,NGAS
15              FMA    = FMS(KINDG(K))
16              DENOM  = 0.0D0
17              DO 521 KK = 1,NGAS
18                FMB=FMS(KINDG(KK))
19                PAB = (1.0+(ETAAV(I,K)/ETAAV(I,KK))**0.5*(FMB/FMA)**0.25)**2
20         >            / SQRT(8.0*(1.0+(FMA/FMB)))
21                DENOM = DENOM + X(I,KK)*PAB
22      521       CONTINUE
23              ETA(I) = ETA(I)+X(I,K)*ETAAV(I,K)/DENOM
24              FLAMA  = SCOND(FMA,CPX(I,K),R,ETAAV(I,K),KINDG(K))
25              FLAM(I)= FLAM(I) + X(I,K)*FLAMA/DENOM
26      520   CONTINUE
27            ALPHA(I)= FNU(NSI(I))*FLAM(I)/CD(I)
28      501 CONTINUE
```

Fig. 3.5 Division of DO 500 loop in Fig. 3.4

```
   C.VPP
        REAL*8 FMAV(NSZ,8),DENOMV(NSZ,8)     ← 新たな配列宣言
   C.VPP
 1      DO 500 I = 1,NCELL
 2        DO 510 K = 1,NGAS
 3          FMA    = FMS(KINDG(K))
 4          EKA    = EK(KINDG(K))
 5          SGA    = SIGMA(KINDG(K))
 6          TASA   = T(I)/EKA
 7          OMVA   = AE/TASA**BE+CE*EXP(-DE*TASA)+EE*EXP(-FE*TASA)
 8          ETAAV(I,K)= 26.69E-7*SQRT(FMA*T(I))/SGA**2/OMVA
 9   510    CONTINUE
10   500 CONTINUE
12      DO 501 I=1,NCELL
13        DO 520 K = 1,NGAS
14          FMAV(I,K)    = FMS(KINDG(K))
15          DENOMV(I,K)  = 0.0D0
16          DO 521 KK = 1,NGAS
17            FMB=FMS(KINDG(KK))
18            PAB = (1.0+(ETAAV(I,K)/ETAAV(I,KK))**0.5*
19       >(FMB/FMAV(I,K))**0.25)**2 / SQRT(8.0*(1.0+(FMAV(I,K)/FMB)))
20            DENOMV(I,K) = DENOMV(I,K) + X(I,KK)*PAB
21   521      CONTINUE
22   520    CONTINUE
23   501 CONTINUE
24      DO 502 I=1,NCELL
25         ETA(I) = 0.0
26         FLAM(I)= 0.0
27         DO 522 K=1,NGAS
28           ETA(I) = ETA(I)+X(I,K)*ETAAV(I,K)/DENOMV(I,K)
29           FLAMA  = SCOND(FMAV(I,K),CPX(I,K),R,ETAAV(I,K),KINDG(K))
30           FLAM(I)= FLAM(I) + X(I,K)*FLAMA/DENOMV(I,K)
31   522    CONTINUE
32          ALPHA(I)= FNU(NSI(I))*FLAM(I)/CD(I)
33   502 CONTINUE
```

Fig. 3.6 Division of DO 501 loop in Fig. 3.5

```
1       DO 500 I = 1,NCELL
2         DO 510 K = 1,NGAS
3           FMA    = FMS(KINDG(K))
4           EKA    = EK(KINDG(K))
5           SGA    = SIGMA(KINDG(K))
6           TASA   = T(I)/EKA
7           OMVA   = AE/TASA**BE+CE*EXP(-DE*TASA)+EE*EXP(-FE*TASA)
8           ETAAV(I,K)= 26.69E-7*SQRT(FMA*T(I))/SGA**2/OMVA
9   510   CONTINUE
10  500 CONTINUE
12      DO 501 I=1,NCELL
13        DO 520 K = 1,NGAS
14          FMAV(I,K)    = FMS(KINDG(K))
15          DENOMV(I,K)  = 0.0D0
16  520   CONTINUE
17  501 CONTINUE
18      DO 503 I=1,NCELL
19        DO 523 K=1,NGAS
20          DO 521 KK = 1,NGAS
21            FMB=FMS(KINDG(KK))
22            PAB = (1.0+(ETAAV(I,K)/ETAAV(I,KK))**0.5*
23     >(FMB/FMAV(I,K))**0.25)**2 / SQRT(8.0*(1.0+(FMAV(I,K)/FMB)))
24            DENOMV(I,K) = DENOMV(I,K) + X(I,KK)*PAB
25  521       CONTINUE
26  523     CONTINUE
27  503 CONTINUE
28      DO 502 I=1,NCELL
29         ETA(I) = 0.0
30         FLAM(I)= 0.0
31         DO 522 K=1,NGAS
32           ETA(I) = ETA(I)+X(I,K)*ETAAV(I,K)/DENOMV(I,K)
33           FLAMA  = SCOND(FMAV(I,K),CPX(I,K),R,ETAAV(I,K),KINDG(K))
34           FLAM(I)= FLAM(I) + X(I,K)*FLAMA/DENOMV(I,K)
35  522    CONTINUE
36         ALPHA(I)= FNU(NSI(I))*FLAM(I)/CD(I)
37  502 CONTINUE
```

Fig. 3.7 Division of DO 501 loop in Fig. 3.6

```
1       DO 500 I = 1,NCELL
2         DO 510 K = 1,NGAS
3           FMA    = FMS(KINDG(K))
4           EKA    = EK(KINDG(K))
5           SGA    = SIGMA(KINDG(K))
6           TASA   = T(I)/EKA
7           OMVA   = AE/TASA**BE+CE*EXP(-DE*TASA)+EE*EXP(-FE*TASA)
8           ETAAV(I,K)= 26.69E-7*SQRT(FMA*T(I))/SGA**2/OMVA
9     510   CONTINUE
10    500 CONTINUE
11        DO 501 I=1,NCELL
12          DO 520 K = 1,NGAS
13            FMAV(I,K)    = FMS(KINDG(K))
14            DENOMV(I,K)  = 0.0D0
15    520     CONTINUE
16    501 CONTINUE
17        DO 503 I=1,NCELL
18          DO 523 K=1,NGAS
19            DO 521 KK = 1,NGAS
20              FMB=FMS(KINDG(KK))
21              PAB = (1.0+(ETAAV(I,K)/ETAAV(I,KK))**0.5*
22       >(FMB/FMAV(I,K))**0.25)**2 / SQRT(8.0*(1.0+(FMAV(I,K)/FMB)))
23              DENOMV(I,K) = DENOMV(I,K) + X(I,KK)*PAB
24    521       CONTINUE
25    523     CONTINUE
26    503 CONTINUE
27        DO 502 I=1,NCELL
28          ETA(I) = 0.0
29          FLAM(I)= 0.0
30    502 CONTINUE
31        DO 504 I=1,NCELL
32          DO 522 K=1,NGAS
33             ETA(I) = ETA(I)+X(I,K)*ETAAV(I,K)/DENOMV(I,K)
34             FLAMA  = SCOND(FMAV(I,K),CPX(I,K),R,ETAAV(I,K),KINDG(K))
35             FLAM(I)= FLAM(I) + X(I,K)*FLAMA/DENOMV(I,K)
36    522     CONTINUE
37          ALPHA(I)= FNU(NSI(I))*FLAM(I)/CD(I)
38    504 CONTINUE
```

Fig. 3.8 Division of DO 502 loop in Fig. 3.7

```
      DO 500 I = 1,NCELL
        DO 510 K = 1,NGAS
          FMA    = FMS(KINDG(K))
          EKA    = EK(KINDG(K))
          SGA    = SIGMA(KINDG(K))
          TASA   = T(I)/EKA
          OMVA   = AE/TASA**BE+CE*EXP(-DE*TASA)+EE*EXP(-FE*TASA)
          ETAAV(I,K)= 26.69E-7*SQRT(FMA*T(I))/SGA**2/OMVA
  510   CONTINUE
  500 CONTINUE
      DO 501 I=1,NCELL
        DO 520 K = 1,NGAS
          FMAV(I,K)    = FMS(KINDG(K))
          DENOMV(I,K)  = 0.0D0
  520   CONTINUE
  501 CONTINUE
      DO 503 I=1,NCELL
        DO 523 K=1,NGAS
          DO 521 KK = 1,NGAS
            FMB=FMS(KINDG(KK))
            PAB = (1.0+(ETAAV(I,K)/ETAAV(I,KK))**0.5*
     >(FMB/FMAV(I,K))**0.25)**2 / SQRT(8.0*(1.0+(FMAV(I,K)/FMB)))
            DENOMV(I,K) = DENOMV(I,K) + X(I,KK)*PAB
  521     CONTINUE
  523   CONTINUE
  503 CONTINUE
      DO 502 I=1,NCELL
        ETA(I) = 0.0
        FLAM(I)= 0.0
  502 CONTINUE
      DO 504 I=1,NCELL
        DO 522 K=1,NGAS
          ETA(I) = ETA(I)+X(I,K)*ETAAV(I,K)/DENOMV(I,K)
          FLAMA  = SCOND(FMAV(I,K),CPX(I,K),R,ETAAV(I,K),KINDG(K))
          FLAM(I)= FLAM(I) + X(I,K)*FLAMA/DENOMV(I,K)
  522   CONTINUE
  504 CONTINUE
      DO 505 I=1,NCELL
        ALPHA(I)= FNU(NSI(I))*FLAM(I)/CD(I)
  505 CONTINUE
```

Fig. 3.9 Division of DO 504 loop in Fig. 3.8

```
 1 S       DO 510 K = 1,NGAS
 2 V       DO 500 I = 1,NCELL
 3 V           FMA    = FMS(KINDG(K))
 4 V           EKA    = EK(KINDG(K))
 5 V           SGA    = SIGMA(KINDG(K))
 6 V           TASA   = T(I)/EKA
 7 V           OMVA   = AE/TASA**BE+CE*EXP(-DE*TASA)+EE*EXP(-FE*TASA)
 8 V           ETAAV(I,K)= 26.69E-7*SQRT(FMA*T(I))/SGA**2/OMVA
 9 V   500 CONTINUE
10 S   510 CONTINUE
11 S       DO 520 K = 1,NGAS
12 V       DO 501 I=1,NCELL
13 V           FMAV(I,K)    = FMS(KINDG(K))
14 V           DENOMV(I,K)  = 0.0D0
15 V   501 CONTINUE
16 S   520 CONTINUE
17 S       DO 523 K=1,NGAS
18 S       DO 521 KK = 1,NGAS
19 V       DO 503 I=1,NCELL
20 V             FMB=FMS(KINDG(KK))
21 V             PAB = (1.0+(ETAAV(I,K)/ETAAV(I,KK))**0.5*
22        >(FMB/FMAV(I,K))**0.25)**2 / SQRT(8.0*(1.0+(FMAV(I,K)/FMB)))
23 V             DENOMV(I,K) = DENOMV(I,K) + X(I,KK)*PAB
24 V   503 CONTINUE
25 S   521 CONTINUE
26 S   523 CONTINUE
27 V       DO 502 I=1,NCELL
28 V         ETA(I) = 0.0
29 V         FLAM(I)= 0.0
30 V   502 CONTINUE
31 S       DO 522 K=1,NGAS
32 V       DO 504 I=1,NCELL
33 V          ETA(I) = ETA(I)+X(I,K)*ETAAV(I,K)/DENOMV(I,K)
34 V          FLAMA  = SCOND(FMAV(I,K),CPX(I,K),R,ETAAV(I,K),KINDG(K))
35 V          FLAM(I)= FLAM(I) + X(I,K)*FLAMA/DENOMV(I,K)
36 V   504 CONTINUE
37 S   522 CONTINUE
38 V       DO 505 I=1,NCELL
39 V          ALPHA(I)= FNU(NSI(I))*FLAM(I)/CD(I)
40 V   505 CONTINUE
```

Fig. 3.10 Tuned DO 500 loop in Fig. 3.4

```
1  S      DO 300 I = 1,NCELL
2  S        CP(I)  = 0.0D0
3  S        DO 310 K = 1,NGAS
4  S          CPX(I,K) = SHEAT(T(I),KINDG(K))
5  V          CP(I)    = CP(I) + X(I,K)*CPX(I,K)
6  V  310   CONTINUE
7  S  300 CONTINUE
```

Fig. 3.11 DO 300 loop in subroutine PCON

```
1        DO 300 I = 1,NCELL
2          CP(I)  = 0.0D0
3    300 CONTINUE
4        DO 301 I=1,NCELL
5          DO 310 K = 1,NGAS
6            CPX(I,K) = SHEAT(T(I),KINDG(K))
7            CP(I)    = CP(I) + X(I,K)*CPX(I,K)
8    310   CONTINUE
9    301 CONTINUE
```

Fig. 3.12 Division of DO 300 loop in Fig. 3.11

```
1  V      DO 300 I = 1,NCELL
2  V        CP(I)  = 0.0D0
3  V  300 CONTINUE
4  S        DO 310 K = 1,NGAS
5  V      DO 301 I=1,NCELL
6  V          CPX(I,K) = SHEAT(T(I),KINDG(K))
7  V          CP(I)    = CP(I) + X(I,K)*CPX(I,K)
8  V  301 CONTINUE
9  S  310   CONTINUE
```

Fig. 3.13 Tuned DO 300 loop in Fig. 3.11

```
1          IF(IMOL.EQ.0) THEN
2  S         DO 10 I = 1,NCELL
3  V         DO 10 K = 1,NGAS
4  V   10      X(I,K) = W(I,K)*FM(I)/FMS(KINDG(K))
5          ELSE
6  S         DO 20 I = 1,NCELL
7  V         DO 20 K = 1,NGAS
8  V   20      W(I,K) = X(I,K)*FMS(KINDG(K))/FM(I)
9          ENDIF
```

Fig. 3.14 DO 10 and DO 20 loops in subroutine PCON

```
1          IF(IMOL.EQ.0) THEN
2  S         DO 10 K = 1,NGAS
3  V         DO 10 I = 1,NCELL
4  V   10      X(I,K) = W(I,K)*FM(I)/FMS(KINDG(K))
5          ELSE
6  S         DO 20 K = 1,NGAS
7  V         DO 20 I = 1,NCELL
8  V   20      W(I,K) = X(I,K)*FMS(KINDG(K))/FM(I)
9          ENDIF
```

Fig. 3.15 Tuned DO 10 and DO 20 loops in subroutine PCON

```
1  S       DO 200 I =1,NCELL
2  V         DO 200 K = 1,NGAS-1
3  V  200     DIF(I,K) = FCOF(I)*DIF(I,K)
```

Fig. 3.16 DO 200 loop in subroutine PCON

```
1  S         DO 200 K = 1,NGAS-1
2  V       DO 200 I =1,NCELL
3  V  200     DIF(I,K) = FCOF(I)*DIF(I,K)
```

Fig. 3.17 Tuned DO 200 loop in subroutine PCON

```
      FUNCTION SHEAT(TT,II)
C*
C*SPECIFIC HEATS FOR CONSTANT PRESSURE  IN UNIT OF JOULE/KG/KELVIN
C*    KIND=1:AR, =2:HE, =3:AIR, =4,CO, =5:CO2, =6:N2, =7:O2
C*
      IMPLICIT REAL*8 (A-H,O-Z)
      DATA AN2,BN2,CN2,DN2
     >    /0.938314D+3,2.95732D-1,-7.31507D-5,5.81796D-9/
      DATA AO2,BO2,CO2,DO2
     >    /0.817026D+3,3.87124D-1,-1.41476D-4,1.99678D-8/
      DATA AAIR,BAIR,CAIR,DAIR
     >    /0.905673D+3,3.10391D-1,-8.59483D-5,8.56212D-9/
      DATA ACO,BCO,CCO,DCO
     >    /0.929207D+3,3.43656D-1,-1.00711D-4,1.01493D-8/
      DATA ACO2,BCO2,CCO2,DCO2
     >    /0.619542D+3,9.43157D-1,-3.92290D-4,5.44078D-8/
C*
      IF(II.EQ.1) THEN
        SHEAT = 0.5218E+3
      ELSEIF(II.EQ.2) THEN
        SHEAT = 5.193E+3
      ELSEIF(II.EQ.3) THEN
        SHEAT = AAIR+BAIR*TT+CAIR*TT**2+DAIR*TT**3
      ELSEIF(II.EQ.4) THEN
        SHEAT = ACO+BCO*TT+CCO*TT**2+DCO*TT**3
      ELSEIF(II.EQ.5) THEN
        SHEAT = ACO2+BCO2*TT+CCO2*TT**2+DCO2*TT**3
      ELSEIF(II.EQ.6) THEN
        SHEAT = AN2+BN2*TT+CN2*TT**2+DN2*TT**3
      ELSEIF(II.EQ.7) THEN
        SHEAT = AO2+BO2*TT+CO2*TT**2+DO2*TT**3
      ENDIF
C*
      RETURN
      END
```

Fig. 3.18 Function SHEAT

```
      IF(II.EQ.1) THEN
        SHEAT = 0.5218E+3
      ELSEIF(II.EQ.2) THEN
        SHEAT = 5.193E+3
      ELSEIF(II.EQ.3) THEN
        SHEAT = AAIR+BAIR*TT+CAIR*TT**2+DAIR*TT**3
      ELSEIF(II.EQ.4) THEN
        SHEAT = ACO+BCO*TT+CCO*TT**2+DCO*TT**3
      ELSEIF(II.EQ.5) THEN
        SHEAT = ACO2+BCO2*TT+CCO2*TT**2+DCO2*TT**3
      ELSEIF(II.EQ.6) THEN
        SHEAT = AN2+BN2*TT+CN2*TT**2+DN2*TT**3
      ELSEIF(II.EQ.7) THEN
        SHEAT = AO2+BO2*TT+CO2*TT**2+DO2*TT**3
C.VPP-modify
      ELSE
        SHEAT = 0.0D0
C.VPP-modify
      ENDIF
```

Fig. 3.19 Modification of IF-BLOCK in function SHEAT

```
1           SUBROUTINE SOLVE(NEQ,NCOLS,MBAND,A,B,IPUNT)
2           IMPLICIT REAL*8 (A-H,O-Z)
3           REAL*8 A(NEQ,NCOLS),B(NEQ)
4     C*
5           IPUNT = 0
6           KMIN  = MBAND + 1
7           DO 150 N = 1,NEQ
8             IF(A(N,MBAND).EQ.0.0D0) GO TO 999
9             IF(A(N,MBAND).EQ.1.0D0) GO TO 120
10            C  = 1.0D0 / A(N,MBAND)
11    V       DO 110 K = KMIN, NCOLS
12    V         IF(A(N,K).EQ.0.0D0) GO TO 110
13    V         A(N,K) = C*A(N,K)
14    V  110  CONTINUE
15       120  CONTINUE
16    S       DO 140 L = 2,MBAND
17    V         JJ = MBAND - L + 1
18    V         I  = N + L - 1
19    V         IF(I.GT.NEQ) GO TO 140
20    V         IF( A(I,JJ).EQ.0.0D0) GO TO 140
21    V         KI = MBAND + 2 - L
22    V         KF = NCOLS + 1 - L
23    V         J  = MBAND
24    S         DO 130 K = KI, KF
25    V           J = J + 1
26    V           IF( A(N,J).EQ.0.0D0) GO TO 130
27    M             A(I,K) = A(I,K) - A(I,JJ)*A(N,J)
28    V  130    CONTINUE
29    V  140  CONTINUE
30       150 CONTINUE
31    C*
32    C*REDUCTION OF A LOAD VECTOR B
33    C*
34    S       DO 230 N = 1, NEQ
35    V         IF( A(N,MBAND).EQ.1.0D0) GO TO 210
36    M         B(N) = B(N) / A(N,MBAND)
37       210  CONTINUE
38    V         DO 220 L = 2, MBAND
39    V           JJ = MBAND - L + 1
40    V           I  = N + L - 1
41    V           IF(I.GT.NEQ) GO TO 220
42    V           IF( A(I,JJ).EQ.0.0D0) GO TO 220
43    V           B(I) = B(I) - A(I,JJ)*B(N)
44    V  220  CONTINUE
45    V  230 CONTINUE
46    C*
47    C*BACK-SUBSTITUTION
48    C*
49          LL = MBAND + 1
50    S     DO 250 M = 1, NEQ
51            N = NEQ + 1 - M
52    S       DO 240 L = LL,NCOLS
53    V         IF( A(N,L).EQ.0.0D0) GO TO 240
54    V         K = N + L - MBAND
55    S         B(N) = B(N) - A(N,L)*B(K)
56    V  240  CONTINUE
57    S  250 CONTINUE
58          RETURN
59    C*
60       999 CONTINUE
61          IPUNT = 1
62          WRITE(6,*) '+++++ SET OF EQUATIONS ARE SINGULAR +++++'
63    C*
64          STOP
65          END
```

Fig. 3.20 Subroutine SOLVE

```
                    .
                    .
                    .
  *vocl loop,novrec(A)
V           DO 130 K = KI, KF
V             J = J + 1
V             IF( A(N,J).EQ.0.0D0) GO TO 130
V               A(I,K) = A(I,K) - A(I,JJ)*A(N,J)
V   130     CONTINUE
                    .
                    .
                    .
```

Fig. 3.21 Tuned DO 130 loop in subroutine SOLVE

```
                    .
                    .
                    .
  *vocl loop,novrec(B)
V         DO 240 L = LL,NCOLS
V           IF( A(N,L).EQ.0.0D0) GO TO 240
V           K = N + L - MBAND
V           B(N) = B(N) - A(N,L)*B(K)
V   240   CONTINUE
                    .
                    .
                    .
```

Fig. 3.22 Tuned DO 240 loop in subroutine SOLVE

```
      SUBROUTINE SOR(FMAT,ANS,PRE,NCELL,NWIDTH)
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER(NSZ=200,ITER=100,OMEGA=1.0D0,EPS=1.0D-3)
      COMMON /POINTR/ IPL(NSZ),IPR(NSZ),ICP(NSZ),ICM(NSZ),IJP(NSZ),
     >                IJM(NSZ)
      REAL*8 ANS(NCELL),PRE(NCELL),TMP(NSZ),FMAT(NCELL,2*NWIDTH+1)
C*
V     DO 1000 N =1,NCELL
V1000   TMP(N) = PRE(N)
C*
      NWP = NWIDTH + 1
      DO 2000 LOOP = 1,ITER,1
        DEVMAX = 0.0
S       DO 2100 I = 1, NCELL
S       WORK = ANS(I)
V       DO 2101 L = IPR(I),IPR(I+1)-1
V         IF(ICP(L).EQ.0)GO TO 2101
V         IP = ICP(L)+NWP-I
V         WORK = WORK-FMAT(I,IP)*TMP(ICP(L))
V2101   CONTINUE
V       DO 2102 L = IPL(I),IPL(I+1)-1
V         IF(ICM(L).EQ.0)GO TO 2102
V         IM = ICM(L)+NWP-I
V         WORK = WORK-FMAT(I,IM)*TMP(ICM(L))
V2102   CONTINUE
S       WORK = WORK/FMAT(I,NWP)
S       TMP(I) = OMEGA*(WORK-TMP(I))+TMP(I)
S2100   CONTINUE
S       DO 2200 I = NCELL,1,-1
S       WORK = ANS(I)
V       DO 2201 L = IPR(I),IPR(I+1)-1
V         IF(ICP(L).EQ.0)GO TO 2201
V         IP = ICP(L)+NWP-I
V         WORK = WORK-FMAT(I,IP)*TMP(ICP(L))
V2201   CONTINUE
V       DO 2202 L = IPL(I),IPL(I+1)-1
V         IF(ICM(L).EQ.0)GO TO 2202
V         IM = ICM(L)+NWP-I
V         WORK = WORK-FMAT(I,IM)*TMP(ICM(L))
V2202   CONTINUE
S       WORK   = WORK/FMAT(I,NWP)
S       TMP(I) = OMEGA*(WORK-TMP(I))+TMP(I)
S       IF(ABS(WORK).LE.1.0D-50)GO TO 2200
M       DEVMAX = MAX(ABS(TMP(I)/WORK-1.D0),DEVMAX)
V2200   CONTINUE
        IF(DEVMAX.LE.EPS) GO TO 2001
 2000 CONTINUE
 2001 CONTINUE
C*
V     DO 3000 N = 1,NCELL
V3000   ANS(N) = TMP(N)
C*
      RETURN
      END
```

Fig. 3.23 Subroutine SOR

```
      SUBROUTINE SOR(FMAT,ANS,PRE,NCELL,NWIDTH)
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER(NSZ=200,ITER=100,OMEGA=1.0D0,EPS=1.0D-3)
      COMMON /POINTR/ IPL(NSZ),IPR(NSZ),ICP(NSZ),ICM(NSZ),IJP(NSZ),
     >                IJM(NSZ)
      REAL*8 ANS(NCELL),PRE(NCELL),TMP(NSZ),FMAT(NCELL,2*NWIDTH+1)
C*
V     DO 1000 N =1,NCELL
V1000   TMP(N) = PRE(N)
C*
      NWP = NWIDTH + 1
      DO 2000 LOOP = 1,ITER,1
        DEVMAX = 0.0
S       DO 2100 I = 1, NCELL
S       WORK = ANS(I)
*vocl loop,scalar                                    ← 指定
        DO 2101 L = IPR(I),IPR(I+1)-1
          IF(ICP(L).EQ.0)GO TO 2101
          IP = ICP(L)+NWP-I
          WORK = WORK-FMAT(I,IP)*TMP(ICP(L))
2101    CONTINUE
*vocl loop,scalar                                    ← 指定
        DO 2102 L = IPL(I),IPL(I+1)-1
          IF(ICM(L).EQ.0)GO TO 2102
          IM = ICM(L)+NWP-I
          WORK = WORK-FMAT(I,IM)*TMP(ICM(L))
2102    CONTINUE
S       WORK = WORK/FMAT(I,NWP)
S       TMP(I) = OMEGA*(WORK-TMP(I))+TMP(I)
S2100   CONTINUE
S       DO 2200 I = NCELL,1,-1
S       WORK = ANS(I)
*vocl loop,scalar                                    ← 指定
        DO 2201 L = IPR(I),IPR(I+1)-1
          IF(ICP(L).EQ.0)GO TO 2201
          IP = ICP(L)+NWP-I
          WORK = WORK-FMAT(I,IP)*TMP(ICP(L))
2201    CONTINUE
*vocl loop,scalar                                    ← 指定
        DO 2202 L = IPL(I),IPL(I+1)-1
          IF(ICM(L).EQ.0)GO TO 2202
          IM = ICM(L)+NWP-I
          WORK = WORK-FMAT(I,IM)*TMP(ICM(L))
2202    CONTINUE
S       WORK   = WORK/FMAT(I,NWP)
S       TMP(I) = OMEGA*(WORK-TMP(I))+TMP(I)
S       IF(ABS(WORK).LE.1.0D-50)GO TO 2200
M       DEVMAX = MAX(ABS(TMP(I)/WORK-1.D0),DEVMAX)
V2200   CONTINUE
        IF(DEVMAX.LE.EPS) GO TO 2001
2000  CONTINUE
2001  CONTINUE
C*
V     DO 3000 N = 1,NCELL
V3000   ANS(N) = TMP(N)
C*
      RETURN
      END
```

Fig. 3.24 Tuned subroutine SOR

```
1          SUBROUTINE CPMASS(KNUM,FMAT,ANS,NCELL,NWIDTH,ICH,
2         >                  WOLD,RHOLD,RHNEW,RHU,RDN,QNEW,QS)
3    C*
4          IMPLICIT REAL*8 (A-H,O-Z)
5          PARAMETER (NSZ=200)
6          COMMON /CRPCON/ NO2,NCO,NCO2
7          COMMON /PARINT/ TI,TF,DELT,CHI
8          COMMON /POINTR/ IPL(NSZ),IPR(NSZ),ICP(NSZ),ICM(NSZ),IJP(NSZ),
9         >                IJM(NSZ)
10         COMMON /CELJUN/ NRVOL(NSZ),NLVOL(NSZ),CLN(NSZ),CD(NSZ),CTHE(NSZ),
11        >        NJL(NSZ),NJR(NSZ),NSI(NSZ),DSI(NSZ),IJUN(NSZ),ICEL(NSZ),
12        >        DST(NSZ),XJ(NSZ),XC(NSZ),DJ(NSZ),VCEL(NSZ),LB(9),RAT(NSZ),
13        >        GV(NSZ),AL(NSZ),COSR(NSZ),COSL(NSZ)
14         DIMENSION      WOLD(1),RHOLD(1),RHNEW(1),
15        >               RHU(1),RDN(1),QNEW(1),QS(1),ICH(1)
16         REAL*8 ANS(NCELL),FMAT(NCELL,2*NWIDTH+1)
17   C*
18         CALL ZERO(FMAT,ANS,NWIDTH,NCELL)
19   C*
20         NWP = NWIDTH + 1
21 S       DO 1000 I = 1, NCELL
22 V         ANS(I) = VCEL(I)*WOLD(I)*RHOLD(I)+VCEL(I)*DELT*QNEW(I)
23 M         FMAT(I,NWP) = VCEL(I)*RHNEW(I)
24 V         IF(ICH(NSI(I)).NE.0)THEN
25 V           IF(KNUM.EQ.NO2.OR.KNUM.EQ.NCO)THEN
26 M             FMAT(I,NWP) = FMAT(I,NWP) - VCEL(I)*DELT*QS(I)
27 V           ENDIF
28 V         ENDIF
29 V         IF(ICEL(I).EQ.0) THEN
30   C*PLUS
31 M           DO 1300 L = IPR(I),IPR(I+1)-1
32 V             NWPP = NWP+ICP(L)-I
33 V             JP   = IJP(L)
34 M             FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0)
35 S             FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP)
36 V  1300     CONTINUE
37   C*MINUS
38 M           DO 1400 L = IPL(I),IPL(I+1)-1
39 V             NWPM = NWP+ICM(L)-I
40 V             JM   = IJM(L)
41 M             FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0)
42 S             FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM)
43 V  1400     CONTINUE
44 V         ELSEIF(ICEL(I).EQ.-1.OR.ICEL(I).EQ.1) THEN
45   C*BREAK POINT
46 M           FMAT(I,NWP) = 1.0
47 V           ANS(I) = WOLD(I)
48 V         ELSEIF(ICEL(I).EQ.-2) THEN
49   C*CLOSED END (PLUS)
50 V           IPRI = IPR(I)
51 V           JP   = IJP(IPRI)
52 S           NWPP = NWP + 1
53 M           FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0)
54 M           FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP)
55 V         ELSEIF(ICEL(I).EQ. 2) THEN
56   C*CLOSED END (MINUS)
57 V           IPLI = IPL(I)
58 V           JM   = IJM(IPLI)
59 S           NWPM = NWP - 1
60 M           FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0)
61 M           FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM)
62 V         ENDIF
63 V  1000 CONTINUE
64   C*
65         RETURN
66         END
```

Fig. 3.25 Subroutine CPMASS

```
      C.DEBUG-start
            integer border__list(NSZ)        ← 境界セルIDを格納する配列
      C.DEBUG-end

 1 V        DO 1000 I = 1, NCELL                          ← DO 1000 ループ
 2 V          ANS(I) = VCEL(I)*WOLD(I)*RHOLD(I)+VCEL(I)*DELT*QNEW(I)
 3 V          FMAT(I,NWP) = VCEL(I)*RHNEW(I)
 4 V          IF(ICH(NSI(I)).NE.0)THEN
 5 V            IF(KNUM.EQ.NO2.OR.KNUM.EQ.NCO)THEN
 6 V              FMAT(I,NWP) = FMAT(I,NWP) - VCEL(I)*DELT*QS(I)
 7 V            ENDIF
 8 V          ENDIF
 9 V   1000 CONTINUE
10
11           iborder=0
12 S         do 1001 I=1,NCELL                           ← DO 1001 ループ
13 V           IF(ICEL(I).EQ.0) THEN
14    C*PLUS
15 M             DO 1300 L = IPR(I),IPR(I+1)-1
16 V               NWPP = NWP+ICP(L)-I
17 V               JP   = IJP(L)
18 M               FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0)
19 S               FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP)
20 V    1300       CONTINUE
21    C*MINUS
22 M             DO 1400 L = IPL(I),IPL(I+1)-1
23 V               NWPM = NWP+ICM(L)-I
24 V               JM   = IJM(L)
25 M               FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0)
26 S               FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM)
27 V    1400       CONTINUE
28 V           ELSE                                      ← 境界セルカウント処理
29 M             iborder=iborder+1
30 M             border__list(iborder)=I
31 V           ENDIF
32 V    1001 CONTINUE
33
34    *vocl loop,scalar
35           DO 1002 k=1,iborder                         ← DO 1002 ループ
36             IF(ICEL(border__list(k)).EQ.-1.OR.ICEL(border__list(k)).EQ.1) THEN
37    C*BREAK POINT
38               FMAT(border__list(k),NWP) = 1.0
39               ANS(border__list(k)) = WOLD(border__list(k))
40             ELSEIF(ICEL(border__list(k)).EQ.-2) THEN
41    C*CLOSED END (PLUS)
42               IPRI = IPR(border__list(k))
43               JP   = IJP(IPRI)
44               NWPP = NWP + 1
45               FMAT(border__list(k),NWPP) = -MAX(-RHU(JP),RDN(JP)-
46          &0.5*RHU(JP),0.0)
47               FMAT(border__list(k),NWP)  = FMAT(border__list(k),NWP)-
48          &FMAT(border__list(k),NWPP)+RHU(JP)
49             ELSEIF(ICEL(border__list(k)).EQ. 2) THEN
50    C*CLOSED END (MINUS)
51               IPLI = IPL(border__list(k))
52               JM   = IJM(IPLI)
53               NWPM = NWP - 1
54               FMAT(border__list(k),NWPM) = -MAX(RHU(JM),RDN(JM)+
55          &0.5*RHU(JM),0.0)
56               FMAT(border__list(k),NWP)  = FMAT(border__list(k),NWP)-
57          &FMAT(border__list(k),NWPM)-RHU(JM)
58             ENDIF
59     1002 CONTINUE
```

Fig. 3.26 Division of DO 1000 loop in Fig. 3.25

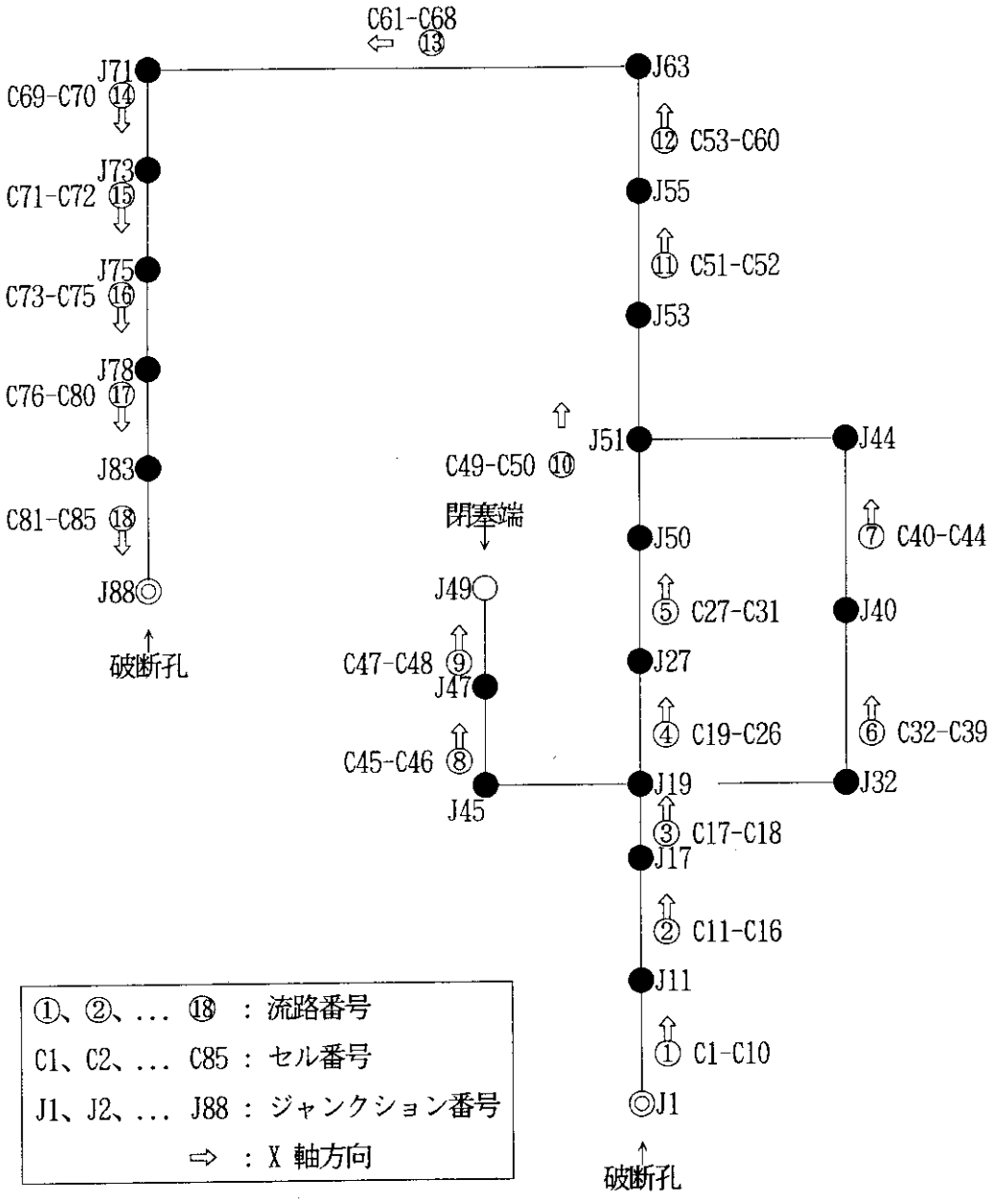


Fig. 3.27 Stream network model of FLOWGR(TAKEDA)

```
      COMMON /POINTR/ IPL(NSZ),IPR(NSZ),ICP(NSZ),ICM(NSZ),IJP(NSZ),
     >                IJM(NSZ),ipr_loop(NSZ),ipl_loop(NSZ)
                                  ↑追加したCOMMON変数↑

                     C.VPP
                           do 10 i=1,NCELL
                           icount=0
                     *vocl loop,scalar
                           do 11 l=IPR(I),IPR(I+1)-1
                             icount=icount+1
                        11 continue
                           ipr_loop(I)=icount
                        10 continue

                           do 20 i=1,NCELL
                           icount=0
                     *vocl loop,scalar
                           do 21 l=IPL(I),IPL(I+1)-1
                             icount=icount+1
                        21 continue
                           ipl_loop(I)=icount
                        20 continue
                     C.VPP
```

Fig. 3.28 Transaction of counting for plus or minus pointer about cells

```
1          iborder=0
2          do 1001 I=1,NCELL                          ← DO 1001 ループ
3            IF(ICEL(I).EQ.0) THEN
4    C*PLUS
5             if(ipr_loop(I).eq.1) then               ← ポイント数が1の場合
6               NWPP = NWP+ICP(IPR(I))-I
7               JP   = IJP(IPR(I))
8               FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0)
9               FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP)
10            else                                    ← 1以外の場合
11             DO 1300 L = IPR(I),IPR(I+1)-1
12              NWPP = NWP+ICP(L)-I
13              JP   = IJP(L)
14              FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0)
15              FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP)
16     1300     CONTINUE
17            endif
18   C*MINUS
19            if(ipl_loop(I).eq.1) then               ← ポイント数が1の場合
20              NWPM = NWP+ICM(IPL(I))-I
21              JM   = IJM(IPL(I))
22              FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0)
23              FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM)
24            else                                    ← 1以外の場合
25             DO 1400 L = IPL(I),IPL(I+1)-1
26              NWPM = NWP+ICM(L)-I
27              JM   = IJM(L)
28              FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0)
29              FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM)
30     1400     CONTINUE
31            endif
32           ELSE                                     ← 境界セルカウント処理
33             iborder=iborder+1
34             border_list(iborder)=I
35           ENDIF
36     1001 CONTINUE
```

Fig. 3.29 Modification of DO 1300 and DO 1400 loops

```
C.VPP
      integer p1_else(NSZ)          ← ポイント数1以外のセルIDを
C.VPP                                   格納する配列

      ip1_else=0
V     do 1001 I=1,NCELL             ← DO 1001 ループ
V       IF(ICEL(I).EQ.0) THEN
V        if(ipr_loop(I).eq.1) then
C*PLUS
V           NWPP = NWP+ICP(IPR(I))-I
V           JP   = IJP(IPR(I))
V           FMAT(I,NWPP) = -MAX(-RHU(JP),RDN(JP)-0.5*RHU(JP),0.0)
V           FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPP)+RHU(JP)
V         else
V          ip1_else=ip1_else+1
V          p1_else(ip1_else)=I
V         endif
V       ENDIF
V 1001 continue

*vocl loop,scalar                   ← DO 1003 ループ
      do 1003 k=1,ip1_else
          DO 1300 L = IPR(p1_else(k)),IPR(p1_else(k)+1)-1
            NWPP = NWP+ICP(L)-p1_else(k)
            JP   = IJP(L)
            FMAT(p1_else(k),NWPP) = -MAX(-RHU(JP),RDN(JP)-
     &0.5*RHU(JP),0.0)
            FMAT(p1_else(k),NWP)  = FMAT(p1_else(k),NWP)-
     &FMAT(p1_else(k),NWPP)+RHU(JP)
 1300     CONTINUE
 1003 continue

      ip1_else=0
      iborder=0
V     do 1004 I=1,NCELL             ← DO 1004 ループ
V       IF(ICEL(I).eq.0) THEN
V        if(ip1_loop(I).eq.1) then
C*MINUS
V           NWPM = NWP+ICM(IPL(I))-I
V           JM   = IJM(IPL(I))
V           FMAT(I,NWPM) = -MAX(RHU(JM),RDN(JM)+0.5*RHU(JM),0.0)
V           FMAT(I,NWP)  = FMAT(I,NWP)-FMAT(I,NWPM)-RHU(JM)
V         else
V          ip1_else=ip1_else+1
V          p1_else(ip1_else)=I
V         endif
V       ELSE
V          iborder=iborder+1
V          border_list(iborder)=I
V       ENDIF
V 1004 continue

*vocl loop,scalar
      do 1005 k=1,ip1_else          ← DO 1005 ループ
          DO 1400 L = IPL(p1_else(k)),IPL(p1_else(k)+1)-1
            NWPM = NWP+ICM(L)-p1_else(k)
            JM   = IJM(L)
            FMAT(p1_else(k),NWPM) = -MAX(RHU(JM),RDN(JM)+
     &0.5*RHU(JM),0.0)
            FMAT(p1_else(k),NWP)  = FMAT(p1_else(k),NWP)-
     &FMAT(p1_else(k),NWPM)-RHU(JM)
 1400     CONTINUE
 1005 continue
```

Fig. 3.30 Tuned DO 1001 loop in Fig. 3.26

```
 1 S      DO 400 N = 1,NCELL
 2 S        UIN(N) = 0.0
 3 V        DO 410 L = IPR(N),IPR(N+1)-1
 4 V  410       UIN(N) = UIN(N)+XJ(IJP(L))*VNEW(IJP(L))
 5 V        DO 420 L = IPL(N),IPL(N+1)-1
 6 V  420       UIN(N) = UIN(N)+XJ(IJM(L))*VNEW(IJM(L))
 7 S        UIN(N) = UIN(N)/2.0/XC(N)
 8 S  400 CONTINUE
```

Fig. 3.31 DO 400 loop in subroutine SET1

```
  C.VPP
        integer p1__else(NSZ)
  C.VPP

        ip1__else=0

V       DO 400 N = 1,NCELL
V         UIN(N) = 0.0
V   400 continue

V       do 401 N=1,NCELL
V       if(ipr_loop(N).eq.1) then
V         UIN(N) = UIN(N)+XJ(IJP(IPR(N)))*VNEW(IJP(IPR(N)))
V       else
V         ip1__else=ip1__else+1
V         p1__else(ip1__else)=N
V       endif
V   401 continue

  *vocl loop,scalar
        do 404 k=1,ip1__else
          DO 410 L = IPR(p1__else(k)),IPR(p1__else(k)+1)-1
    410       UIN(p1__else(k)) = UIN(p1__else(k))+XJ(IJP(L))*VNEW(IJP(L))
    404 continue

V       do 402 N=1,NCELL
V       if(ip1__loop(N).eq.1) then
V         UIN(N) = UIN(N)+XJ(IJM(IPL(N)))*VNEW(IJM(IPL(N)))
V       else
V         ip1__else=ip1__else+1
V         p1__else(ip1__else)=N
V       endif
V   402 continue

  *vocl loop,scalar
        do 405 k=1,ip1__else
          DO 420 L = IPL(p1__else(k)),IPL(p1__else(k)+1)-1
    420       UIN(p1__else(k)) = UIN(p1__else(k))+XJ(IJM(L))*VNEW(IJM(L))
    405 continue

V       do 403 N=1,NCELL
V         UIN(N) = UIN(N)/2.0/XC(N)
V   403 CONTINUE
```

Fig. 3.32 Tuned DO 400 loop in subroutine SET1

```
S       DO 600 J = 1,NCELL
V       DO 600 K = 1,NGAS-1
V         QNEW(J,K) = (1.0-RLXQ)*QNEW(J,K)+ RLXQ*QTMP(J,K)
V         QS  (J,K) = (1.0-RLXQ)*QS(J,K)  + RLXQ*STMP(J,K)
V  600 CONTINUE
```

Fig. 3.33 DO 600 loop in subroutine STEP

```
S       DO 600 K = 1,NGAS-1
V       DO 600 J = 1,NCELL
V         QNEW(J,K) = (1.0-RLXQ)*QNEW(J,K)+ RLXQ*QTMP(J,K)
V         QS  (J,K) = (1.0-RLXQ)*QS(J,K)  + RLXQ*STMP(J,K)
V  600 CONTINUE
```

Fig. 3.34 Tuned DO 600 loop in subroutine STEP

```
                          .
                          .
*vocl loop,scalar
           DO 1100 L = IPR(I),IPR(I+1)-1
             JP = IJP(L)
             IP = ICP(L)+NWP-I
             FMAT(I,IP)= BETA(JP)*RHM(JP)*XJ(JP)
             ANS(I) = ANS(I)-GAMM(JP)*RHM(JP)*XJ(JP)
             FMAT(I,NWP) = FMAT(I,NWP)-FMAT(I,IP)
 1100      CONTINUE
*vocl loop,scalar
           DO 1200 L = IPL(I),IPL(I+1)-1
             JM = IJM(L)
             IM = ICM(L)+NWP-I
             FMAT(I,IM)= BETA(JM)*RHM(JM)*XJ(JM)
             ANS(I) = ANS(I) + GAMM(JM)*RHM(JM)*XJ(JM)
             FMAT(I,NWP) = FMAT(I,NWP)-FMAT(I,IM)
 1200      CONTINUE
                          .
                          .
```

Fig. 3.35 Tuned DO 1100 and DO 1200 loops in subroutine MXMASS

```
                          .
                          .
*vocl loop,scalar
       DO 1000 I = 1, NCELL
         ANS(I) = ANS(I) + VCEL(I)*QNEW(I)
         IF(ICEL(I).EQ.0) THEN
           DO 1100 L = IPR(I),IPR(I+1)-1
             JP = IJP(L)
             ANS(I) = ANS(I)-GAMM(JP)*RHM(JP)*XJ(JP)
 1100      CONTINUE
*vocl loop,scalar
           DO 1200 L = IPL(I),IPL(I+1)-1
             JM = IJM(L)
             ANS(I) = ANS(I) + GAMM(JM)*RHM(JM)*XJ(JM)
 1200      CONTINUE
                          .
                          .
```

Fig. 3.36 Tuned DO 1100 and DO 1200 loops in subroutine PRESMD

# 4. RBUU コードのベクトル化

富士通(株)製分散メモリ型ベクトル並列計算機VPP500/42(以下VPP500)において，相対論的ボルツマン・ウェーリング・ウーレンベック法によるシミュレーション・コードRBUUのベクトル化，及びその並列化の調査を行った．本コードは，一度IBMマシン用にベクトル化，及び並列化の高速化チューニングが施されていたが，VPP500にインストールしたところベクトル化率が落ち速度低下を招いていた．また，並列化処置された箇所はVPP500ではコメント扱いされていた．従って，第1段階ではVPP500上でベクトル化チューニングを行い，次いで並列化の可能性について調査した．以下にこれらの作業内容について報告する．

## 4.1 コード概要

RBUUコードは，2つの原子核の衝突反応におけるRBUU方程式[9, 10, 11]を，テスト粒子を用いたシミュレーションにより解くコードである．各時間ステップ毎に粒子生成の確率を計算し観測量を与えると共に，テスト粒子を原子核の分布に従うように配置し，その配置から密度分布を計算して場を与える．その場から粒子に働く力を計算し，運動方程式を解く．これを繰り返しながらテスト粒子の時間発展を追うものである．

## 4.2 ベクトル化

ベクトル化を進めるにあたりユーザより4つの入力データが提供された．それぞれをテスト・ケース1〜4として，各ケースごとに動的挙動解析を行い，高速化のためのチューニングを行った．各テスト・ケースの一覧をTable 4.1に，対応する入力データをFig. 4.1に示す．

### 4.2.1 動的挙動解析

計算コストの分布状況，及び自動ベクトル化の状況を調査するため，動的挙動解析ツールであるSAMPLAR，及びANALYZERを利用した．

SAMPLARは，VPP500上で動作するツールであり，実行可能プログラムを対象にしている．これによりルーチン単位の計算コスト分布を得ることができる．一方，ANALYZERの方はGSP上で動作するツールであり，ソースプログラムを対象にしている．ルーチン単位，ループ単位，文単位の計算コスト，実行回数，及びループの回転数等の情報が得られる．ルーチン単位の計算コスト分布に関しては，RBUUが実際に動作するのがVPP500上であるため，VPP-500上の解析結果であるSAMPLARのものの方がより実情に近い値を示している．

これらツールを利用するに際し，ANALYZERの制約によりDOループの変更を行った．RBUUコードの中で使用されているDOループは，殆どがDO 〜 END DOの構文のものであり，ANALYZERではエラー扱いされてしまった．ANALYZERがDO 〜 END DOの構文を許していないFORTRAN77 EX/VPを使用しているためである．そこで，これらのDOループをDO 〜 CONTINUEの構文に変更した．動的挙動解析で使用した実行用シェル・スクリプト(SAMPLAR

実行，SAMPLARの対象とした実行可能プログラム生成，ANALYZER実行)をFig. 4.2～4.4に示す.

各テスト・ケースにおける，ベクトル化対応前のSAMPLARの出力結果をFig. 4.5に示す. 各ケースの上位6ルーチンは，それぞれ3%以上のコスト比率を占めており，全体で84～91%を占めている．特にDENSルーチンにコストが集中(40～49%)している．そこでベクトル化チューニングの対象ルーチンとして，少なくともいずれかのケースで上位6ルーチンの1つになっているDENS, SMEAR, ITERAT, XSORT, ABSOR, PERMUT, RCDELTの7ルーチンを選択した. 各テスト・ケースにおける，上記7ルーチンのベクトル化対応前のANALYZERの出力結果(抜粋：routine list, loop list)をFig. 4.6～4.12に示す．各ルーチンの計算コストの分布状況，ベクトル化状況は以下のとおりであった.

(1)DENSルーチン(Fig. 4.6)

DO 1331・1341の二重ループにルーチン全体の76～80%の計算コストが集中している．内側ループは回転数が小さく，つまりベクトル長が短く，外側のループはベクトル化されていない．また，ルーチン全体としてのベクトル化率は62～76%とあまり高くない.

(2)SMEARルーチン(Fig. 4.7)

ルーチン全体としてのベクトル化率は99.6%と高く，ベクトル化効果も34～44倍と高い性能が出ている．ただ15.8%のコスト比率を占めるDO 2511がベクトル化されていない.

(3)ITERATルーチン(Fig. 4.8)

全体の約86～88%の計算コストを占めているDO 1831は自動ベクトル化されており，ベクトル長も102541と長い割りには，ベクトル化率35.5%，ベクトル化効果1.5と性能が出ていない.

(4)XSORTルーチン(Fig. 4.9)

全体の約75～86%の計算コストを占めているDO 2861, DO 2821, DO 2851, DO 2811, 及びDO 2791の各DOループ内では，ベクトル化，非ベクトル化の文が入り混ざっており，自動ベクトル化が不十分である.

(5)ABSORルーチン(Fig. 4.10)

全体の約96～100%の計算コストを占めているDO 1141, DO 1171内の自動ベクトル化が不十分である.

(6)PERMUTルーチン(Fig. 4.11)

ベクトル化率，ベクトル化効果共に，約20%，1.0～1.2と低い．また，ベクトル長がテスト・ケース3，4において短くなっている.

(7)RCDELTルーチン(Fig. 4.12)

DO 2111・2121の二重ループは，テスト・ケース1～3において，約50～70%のコスト比率を占めており，内側ループの回転数が外側のものに比べて極端に小さい．また外側ループはベクトル化されていない.

### 4.2.2 チューニング

動的挙動解析により得たデータを参考にして，高速化のためのチューニングを実施した．以下にその作業内容を記述する．

#### 4.2.2.1 DENS ルーチン

DENS ルーチンは計算コストに関して言えば，2つの二重ループ DO 1331・1341，DO 1371・1381 により約 98 ～ 99% が占められている．それぞれは 76 ～ 80%，18 ～ 23% である．本作業では，この 2 つの二重ループのチューニングを行った．

(1) 二重ループ (DO 1331・1341) の変更

RBUU コードの計算コスト中では DENS ルーチンの占める比率が高く，中でも DO 1331・1341 の比率が高い．従って，本二重ループのコスト・ダウンが RBUU コードの高速化の鍵を握っている．

本二重ループにおいては，外側ループの DO 1331 にコストが偏っており，その中の IF 文はベクトル化されておらずその真率は低い (テスト・ケース 1 ～ 4 で 1.0，1.6，11.9，17.0%) ことが判った．また，DO 1341 の平均ベクトル長が短い (テスト・ケース 1 ～ 4 で 15，6，3，4) ことも判った．そこで，変更の第 1 ステップとして，DO 1331 のコスト・ダウンを図るため，NGP(I) .GT. 0 を満たす I を抽出する処理を独立させた．そして，この条件を満たす I について，DO 1341 を実施するよう変更した．この抽出処理をベクトル化 (ベクトル長：102541) することによりコスト・ダウンを図った．第 1 ステップにおける変更前，及び変更後のソース・コードを対比して Fig. 4.13 に示す．

変更の第 2 ステップとして，DO 1341 のコスト・ダウンを図るため FAC*XMEFF(J)/P(0,J) の項の計算を集中化してベクトル処理できるようにした．この時の J の範囲は，Fig. 4.13 における NG1(I) の最小値～NG2(I) の最大値である．これは IGP(I) が増加関数であること，及び NGP(I) が非負であるということを利用している．FAC*XMEFF(J)/P(0,J) の計算では対象外の J についても計算するが，ベクトル処理 (平均ベクトル長：NX*NX*NZ × IF 文の真率) することによりコスト・ダウンを図った．第 2 ステップの変更後のソース・コードを Fig. 4.14 に示す．

第 3 ステップとして，NGP(I) .GT. 0 を満たす I の抽出処理と FAC*XMEFF(J)/P(0,J) を集中計算する時の NG1(I) の最小値，NG2(I) の最大値の計算処理を XSORT ルーチンの中で実施するように変更した．IGP(I)，NGP(I) の値は XSORT ルーチンで定義され，他のルーチンでは変更されないこと，及び XSORT ルーチンと DENS ルーチンの実行回数の比が約 1：6 ～ 7 であることから，コスト・ダウンが期待できる．第 3 ステップの変更後のソース・コードを Fig. 4.15 に示す．Fig. 4.15 では，更に NGP(I) .eq. 1 の抽出処理も行っている．

(2) 二重ループ (DO 1371・1381) の変更

DO 1371・1381 の二重ループは，構造的に (1) の DO 1331・1341 と同じであり，変更方法も同じである．変更前，及び変更後のソース・コードを Fig. 4.16 に対比して示す．

#### 4.2.2.2 SMEAR ルーチン

本ルーチンにおいては，DO 2511 のベクトル化，及び処理の簡略化を行った．

(1)DO 2511 のベクトル化

DO 2511 がベクトル化されていない理由は，「配列 VTAR におけるデータの定義引用が回帰的参照の可能性があるためベクトル化できない」というものであった．実際は，配列 VTAR の第1次元の添字式 1 + (MAXX+IY) * NX と NX + (MAXX+IY) * NX とはそのとり得る値が重ならずベクトル化可能であった．そのため，*VOCL LOOP,NOVREC(VTAR) 文を挿入してベクトル化した．

(2) 処理の簡略化

次の 2 つの処理に関して，簡略化した．

- 配列 VSRC の値を配列 VTAR に代入する際，配列 RANDX を中継している処理 (DO 2491)
- IBM 版の並列化対処のための変数 ITASK による DO ループ (DO 2411, DO 2461)

配列 RANDX については中継せず，直接代入しても結果は同じため，直接代入するように変更した．結果として DO 2491 が不要となった．ITASK の方は，意味がないため対応する DO ループを削除した．(1)，(2) の変更内容を Fig. 4.17 に示す．

#### 4.2.2.3 ITERAT ルーチン

本ルーチンにおいては，ANALYZER の結果の中でベクトル化効果の出ていない 2 つの DO ループ DO 1771, DO 1831 内の IF 文の構造を変更した．いずれの場合も，条件が互いに排反となる 2 つの IF 文が続けて記述されており，真率が一方に偏っているにもかかわらず，どちらもマスク操作による方法でベクトル化されていた．このためベクトル化率，ベクトル化効果，共に小さくなっていた．排反条件を利用して，2 つの IF 文を IF ～ THEN ～ ELSE ～ END IF の構造に変更した．変更内容を Fig. 4.18 に示す．

#### 4.2.2.4 XSORT ルーチン

XSORT ルーチンはグリッド上の空間位置情報を元にテスト粒子の並べ替えを行うルーチンであり，グリッドのある範囲の外に飛び出したものは粒子列の後方に移動し，シミュレーションの対象から外している．ここでは，シミュレーション対象の粒子に対する処理と対象外の粒子に対する処理を分けることにより，全体の処理を簡略化すると共に部分ベクトル化されている DO ループ (DO 2791, DO 2851) の分割を行った．これにより，ベクトル化率，及びベクトル化効果の向上を狙った．変更内容を Fig. 4.19 に示す．

4.2.2.5 ABSOR ルーチン

ABSOR ルーチンは，概略 DO 1141， DO 1151， DO 1171 の三重ループで構成されている．ANALYZER の結果 (Fig. 4.10) よりベクトル長は DO 1141 が 102541 と長く，DO 1151 は 0 に近い値である．DO 1171 のベクトル長はテスト・ケース 1～4 で 0， 7， 17， 12 である．また，計算コストについて言えば，DO 1141 がテスト・ケース 1， 2 において約 80% 以上を占めており，DO 1171 がテスト・ケース 3， 4 において約 70% を占めている．そこで，ここでは，2 つの対応を行った．1 つは DO 1151 以下が実際に実行される時の I，つまり NGPA(I) .GT. 0 を満たす I を抽出する処理を独立させた．これにより，ベクトル化率の向上を図り，DO 1141 のコスト・ダウンを図った．2 つめは，DO 1171 をベクトル化可能部分とベクトル化不可能部分とに分割し，ベクトル化率，及びベクトル化効果の向上を図った．処理の順序性を考慮して DO 1171 を DO 1171， DO 1172， DO 1173， DO 1174， DO 1175 の 5 つに分割し，各ループ間のデータ引き継ぎのために作業用の配列を導入した．例えば以下のような変更を行った．変更内容を Fig. 4.20 に示す．

```
PXCM=BETAX*TRAFO+PP(1)    ⇒    XCMa(IB)=BETAXa(IB)*TRAFOa(IB)+PP(1,IB)
```

4.2.2.6 PERMUT ルーチン

ANALYZER の結果 (Fig. 4.11) より，本ルーチンのベクトル長が短いことが判ったため，ルーチン全体をスカラ化した．スカラ化するため，*VOCL TOTAL,SCALAR 文を本ルーチンの先頭に挿入し，これにより，ベクトル実行のオーバヘッドに対応した．

4.2.2.7 RCDELT ルーチン

DO 2111・2121 の二重ループは，DENS ルーチンの DO 1331・1341 と同じように，内部の IF 文の真率が低く (テスト・ケース 1～4 で 0.7， 0.7， 5.1， 9.7%)，内側ループのベクトル長も短い (テスト・ケース 1～4 で 11， 6， 3， 3)．このため，DENS ルーチンの場合と同じように NGP(I).GT.1 を満たす I を抽出することによりコスト・ダウンを図った．DO 2111・2121 の変更内容を Fig. 4.21 に示す．

4.2.2.8 その他

TPCORRA ルーチン内で実行されている SIGMA，及び OMEGA のスメア処理 (CALL SMEAR) は冗長な処理であることが判ったため，実行しないよう対処した．TPCORRA ルーチンは MAIN ルーチンから呼ばれるが，その直前で呼ばれる TPPREDA ルーチンの中で SIGMA，及び OMEGA のスメア処理が行われている．その後，TPCORRA ルーチンが呼ばれるまでスメア結果は更新されないため，TPCORRA ルーチン内でのスメア処理が冗長になっていた．

### 4.2.3 計算結果の評価及び効果

#### 4.2.3.1 計算結果の評価

VPP500 上で FORTRAN プログラムを実行する場合，同じソース・コードを用いたとしても，スカラ演算とベクトル演算との丸め方法の違い，コンパイラによる最適化方法の違いにより計算の出力結果に違いが生じる．従って，ベクトル化の変更等，何らかの変更により計算の出力結果に違いが生じるのは一般的である．今回の変更作業においても，変更前のオリジナル版コードと変更後のベクトル化版コードとを，同じコンパイル・オプションによりコンパイルしても，その出力結果に違いが生じていた．この出力結果の違いが今回の変更によるものなのか，丸め方法・最適化方法等の計算環境によるものなのかの確認を行った．

確認方法は，基本的には，計算環境を同じくするため，オリジナル版，ベクトル化版共に最適化抑止したスカラー・コンパイルを行いその計算結果を比較するのであるが，ここでは計算時間短縮のため，以下のような簡便な方法で行った．

- 丸め方法を合わせるためオリジナル版，ベクトル化版共にスカラー演算の丸め方法を切り捨てモードにするため MAIN ルーチンの先頭に `CALL CHROUND(3)` を挿入した．
- オリジナル版，ベクトル化版共に最適化を抑止するが，オリジナル版の方はスカラー・コンパイルを行い，ベクトル化版の方はベクトル・コンパイルを行った．

  オリジナル版のコンパイル・オプション： `-Oe,-PE -Wv,-sc`
  ベクトル化版のコンパイル・オプション： `-Oe,-PE`

本確認作業で対象とした出力結果は，テスト・ケース 1 ～ 4 の summry.dat，及び output.dat である．diff コマンドにより出力結果を比較したが，いずれの場合も summry.dat のタイム・スタンプの値を除き，オリジナル版とベクトル化版のものが全桁一致した．以上のことから，今回の変更作業が計算の誤りを引き起こしていないことが確認できた．

#### 4.2.3.2 効果

オリジナル版，及びベクトル化版の実行時間を測定した．測定対象としたのは，テスト・ケース 1 ～ 4 であり，測定値を Table 4.2 に示す．また各ケースのベクトル化版の SAMPLAR の結果を Fig. 4.22 に示す．この Fig. 4.22 と Fig. 4.5 とから，ルーチン単位のコスト・ダウンがおおよそどの程度のものか判る．テスト・ケース 1 ～ 4 における各変更ルーチン毎のコスト・ダウン率を Table 4.3 に示す．

今回の変更により，ベクトル化版ベクトル実行は，オリジナル版ベクトル実行に比べて 1.85 ～ 2.08 倍，オリジナル版スカラ実行に比べて 8.4 ～ 14 倍の高速化を実現することができた．ルーチン単位にみると DENS ルーチンのコスト・ダウンが大きく貢献しているのが判る．反面，ITERAT ルーチンの変更はあまり効果を上げていない．ANALYZER の結果では，変更後はベクトル化率 100%，ベクトル化効果約 40 ～ 55 倍と向上しているが，VPP500 と GSP という走行環境の差が数値に現れたものと思われる．実際 ITERAT ルーチンは変更前でも，高速化されていたものと思われる．

DENS ルーチンの次にコスト・ダウンに貢献しているのが ABSOR ルーチンである．それが顕著に現れるのは，Table 4.4 に示すテスト・ケース 5 の場合であった．これは，実際に使うテスト・ケースの中で最大級のものである．

今回の変更の中で ABSOR ルーチンだけ変更していない ABSOR 対応未のコードと ABSOR ルーチンも対応した ABSOR 対応済コードの 2 つを用いてテスト・ケース 5 の実行，及び SAMPLAR の実行を行った．Table 4.5 にその測定値を示す．ABSOR の対応により，ABSOR ルーチン単独では 79% のコスト・ダウン，実行時間で 19% の削減効果があった．

## 4.3 並列化調査

本コードの高速化を更に進めるため，並列化の可能性について検討を行った．

### 4.3.1 コード調査

RBUU コードの機能的な構成は，以下のようになっている．

(1) 初期化処理

テスト・データ (input.cfg) 等の入力を行い，各種変数等の初期化を行っている．

(2) 時間ステップ処理

1 〜 NTMAX の時間ステップ・ループを形成し，各ステップ毎にテスト粒子法により運動方程式を解き，時間経過のシミュレートを行っている．ここで NTMAX は入力データでる．

(3) 終了処理

RBUU コードの構造図を Fig. 4.23 に示す．

### 4.3.2 並列性の調査

並列化を進める場合，その分割対象の候補として時間・粒子・領域が挙げられる．これらについて並列性の調査を行った．

#### 4.3.2.1 時間分割による並列化

時間ステップ NT = n+1 (n > 0) の計算をする場合，前の時間ステップ n のデータ，例えば，R(1 〜 3,I)，P(0 〜 3,I)，XMEFF(I) 等を必要としており，各時間ステップの間で計算の独立性が保たれていない．時間に関する並列性を見出すことはできない．

#### 4.3.2.2 粒子分割による並列化

計算対象の粒子には，陽子・中性子であれば 1 〜 NTOTAL まで，反陽子であれば 1 〜 NANTOT までの番号が割り振られて，それぞれの位置，運動量，質量等が対応付けられている．各時間ステップにおいて，粒子の運動に伴いその空間の位置が変わることにより，この番号は変わる．粒子順番の並べ替え処理，グリッド外に飛び出した粒子を計算対象外にする処理等が行われる．

番号変更の処理は，陽子・中性子の場合，RCDELT，TPSORT，XSORTの各ルーチンで行っており，反陽子の場合，RCDELT，XSORTAの各ルーチンで行っている．粒子分割する場合，こうした番号変更の処理が行われるまでの区間を並列化の対象にすることができる．この区間が並列化の1単位となり，この単位ごとに各PE間のデータ整合をとらなければならない．

RBUUコードのメインルートに関して，DOループを最小単位とした処理関連図をFig. 4.24に示す．これは，Fig. 4.23をさらに詳細化したものであり，DOループの繰り返し属性，及びテスト・ケース5における1処理単位の時間を載せている．ここでの時間は，実測値ではなく，SAMPLAR，及びANALYZERの結果を元に計算により算出した概略平均値である．

各PE間のデータ整合のオーバヘッドを考慮すると，Fig. 4.24の中では，TPPRED，TPCORR，TPSORT，XSORTの各ルーチンが粒子分割の候補として考えられる．TPPRED，TPCORRは処理構造が似ており，並列化の対象はそれぞれDO2661，DO 2621である．1処理時間はそれぞれのDOループで206.2 ms，186.6 msであり，いずれの場合も，各PE間でデータの整合が必要となるのは配列P(0:3,MAXPAR)である．ここで，MAXPARには式(4.1)の関係がありテスト・ケース5では，416000である．

$$(\text{標的粒子の質量}+\text{入射粒子の質量})\times\text{粒子数}\leq\text{MAXPAR} \tag{4.1}$$

TPSORTルーチンの場合，並列化対象となるのは，計算コストが集中しているDO 2711とDO 2751である．DO 2711の1処理時間は39.5 msであり，各PE間でデータの整合が必要となるのは配列IGP(NX*NX*NZ)である．ここで，NX*NX*NZ = 102541(=41*41*61)である．DO 2751は，粒子番号Iに関して計算の順序性があり並列化できない．XSORTルーチンはTPSORTルーチンと処理構造が似ており，同じ理由により，並列化候補はDO 2811となる．1処理時間は39.8 ms，データ整合が必要なのは配列IGP(NX*NX*NZ)である．

#### 4.3.2.3 領域分割による並列化

オリジナル・コードのSMEARルーチンの中にIBM版の並列化対応の箇所がある．Z方向を分割した並列化である．SMEARに関しては，例えば，ITERATルーチンの中でSMEARルーチンを呼び出した直後に以下の(4.2)のような処理が行われている．IIZ(I)により他PEのデータを引用している．VPP500の並列化方式は分散メモリ方式であり，こうした引用がある場合には，各PE間でデータ整合等の対応が必要となる．SMEARルーチンの処理時間は数ミリ秒であり，SIGMS等を各PE間でデータ転送するオーバヘッドを考慮すると，VPP500ではIBM版のような対応はとれない．

$$\text{XEMEFF(I)} = \text{XMREST(I)} - \text{GS} * \text{SIGMS(IIX(I), IIY(I), IIZ(I))} \tag{4.2}$$

このような引用は，この他に以下の箇所で行われている．並列化のオーバヘッドを考慮すると，Fig. 4.24の中では，ABSOR，DENSのルーチンとRCDELTルーチンのDO 2111・2121ループが領域分割の対象として考えられる．

ITERAT:DO 1841,1891　　RCDELT:DO 2131,2161
TPCORR:DO 2621　　TPCOOA:DO 2641
TPPRED:DO 2661　　TPPREDA:DO 2681 等

ABSORルーチンの場合，計算コストはDO 1141・1151・1171の三重ループに集中している．DO 1141は，DO 1151の処理がグリッドのセル位置番号LGに関して計算の順序性を保つ必要があるため，分割できない．従って，並列化する場合DO 1151を分割することになるが，平均ベクトル長は2.67と短い．DO 1151 以下の1処理時間は，197.5msであり，データ整合が必要となるのは，配列PROB(MAXANT) とNDENS(0:200)である．ここで，MAXANT = 30000である．DENSルーチンの場合，並列化候補はDO 1342・1343，及びDO 1382・1383の二重ループである．1処理時間はそれぞれ24.0ms，69.8msであり，データ整合が必要となるのは，それぞれ配列RHSVEI(NX*NX*NZ) とRHVVEI(NX*NX*NZ,0:3)である．RCDELTルーチンの場合，二重ループDO 2111・2121の1処理時間は134.8msであり，データ整合が必要となる配列はIPART1(MAXPAR/2)，IPART2(MAXPAR/2)，LASTCOL(MAXPAR)である．

### 4.3.3　並列化効果の予測

最大級のテスト・ケース5(Fig. 4.24)をモデルにして，その並列化効果について記述する．

#### 4.3.3.1　オーバヘッドの測定

粒子番号の変更，他PEのデータに対する引用により各PE間でデータの整合をとる必要性が生じる．このような場合，グローバル関数SUMにより整合をとることができる．並列化の可能性を検討するため，「4.3.2 並列性の調査」で整合の対象となった配列またはその大きさが同じ配列に関して，この関数のオーバヘッドを実測した．測定方法は，DENS，TPPRED，XSORTAの各ルーチンの中で対象とする配列を更新しているDOループを選択し，このDOループの処理時間を2つの方法で測定した．1つは対応箇所をSPRED DO ～ END SPREADで括り，その箇所の走行時間を測定する．もう1つはSPRED DO ～ END SPREAD SUM( )に変えてその箇所の走行時間を測定する．この2つの差をRBUU環境下でのオーバヘッドと想定した．実際，テスト・ケース5において，時間ステップ1～10までをPE数= 2，4，8，16で測定した．実測値をTable 4.6に示す．

#### 4.3.3.2　並列化効果

グローバル関数SUMによるオーバヘッドの程度は，データ量，PE数，その他諸々の計算環境により異なる．ここでは，「4.3.3.1 オーバヘッドの測定」で得た実測値をRBUU走行環境下におけるオーバヘッドと想定してテスト・ケース5における並列化の効果を求めた．その結果をTable 4.7に示す．

Table 4.7におけるコスト比率は，各並列化対象箇所の計算コストが全体の中で占める割合を百分率で示している．並列化効果欄の数値は，全体の中での並列化による想定コスト・ダウン率

$(CD_T)$であり，式(4.3)～(4.5)により求めた．

$$CD_T = (C_W/100)\times CD_R\times 100 \tag{4.3}$$

$$CD_R = (T_O - T_P)/T_O \tag{4.4}$$

$$T_P = O_V + T_O/npe \tag{4.5}$$

ここで，

$C_W(\%)$ ：コスト比率
$CD_R$ ：並列化対象箇所内のコスト・ダウン率
$T_O(ms)$ ：並列化対象箇所の走行時間（1処理単位）
$T_P(ms)$ ：並列化対象箇所の並列化後の想定走行時間（1処理単位）
$O_V(ms)$ ：オーバヘッド時間
$npe$ ： PE 数

以上のように，RBUUは全体を通した並列化はできないし，一連の固まった処理群さえも並列化の対象にできない．計算コストの大きいところを小刻みに並列化する方法しか見出せない．

Table 4.7から判るように，最大級のテスト・ケースであるテスト・ケース5の場合でさえ，4 PEで高々 10%のコスト・ダウン，倍率にして 1.1倍の効率しか見込まれない(実測値は，CPU TIME:49:56.73，倍率:1.1倍であった)．そして，PE数を上げてもオーバヘッドによりこれ以上の効率向上は期待できないことが判る．

## 4.4 まとめ

今回変更対象としたRBUUコードは，これまでにIBMマシン用にベクトル化・並列化の対応がとられたものであった．変更前のオリジナル・コードを最適化オプション-Oeでベクトル・コンパイルしたもので，ベクトル化率が89.8～92.3%，スカラ実行比4.53～7.16倍であった．従って，ベクトル化による高速化は，当初あまり望めないのではないかと思われた．しかし，実際には今回の変更により，ベクトル化率は93.9～98.0%，スカラ実行比8.4～13.4倍，実行時間で変更前のものより1.85～2.08倍の処理速度の向上を図ることができた．

これは，主にDENSルーチンのチューニング効果による．処理ロジックの見通しが悪くなったが，全体で33～38%のコスト・ダウンに寄与している．倍率にして1.5～1.6倍に相当する．またテスト・ケース1～4ではコスト比率が高くなかったABSORルーチンが，最大級のテスト・ケース5では20.6%を占めていた．ABSORルーチンのベクトル化チューニングによりテスト・ケース5で19%のコスト・ダウンを図ることができた．これにより，実際に利用する範囲内におけるベクトル化チューニングは，概ね終了したものと考える．

一方，並列化に関しては，有効な並列化方法を見出すことができなかった．計算コストの大きい箇所を並列化する小刻みな並列化方式では，コストの集中度がテスト・ケースや時間ステッ

ブにより異なるため，汎用的な並列化を実現できない．例えば，ABSORルーチンに関して言えば，テスト・ケース5において1処理時間が200ms以上であったのは，900ステップのうち，時間ステップ149～574であった．時間ステップ132までは10ms以下であった．最大級のテスト・ケースにおいてさえ，ベクトル化版は約1時間で実行可能であり，この点で並列化の必要性は高いものとは言えなくなった．今後，粒子数，グリッドの大きさを大きくとることにより，長時間のテスト・ケースが現れることも考えられる．その場合でも，並列化対象部の計算コスト比率が高くないため，本処理方式を踏襲する限り，1倍強程度の並列化効果しか期待できない．

Table 4.1 Test cases

| | ﾃｽﾄ・ｹｰｽ1 | ﾃｽﾄ・ｹｰｽ2 | ﾃｽﾄ・ｹｰｽ3 | ﾃｽﾄ・ｹｰｽ4 |
|---|---|---|---|---|
| 標的原子 | ——— | C | Si | Eu |
| 標的原子の質量 | ——— | 12 | 28 | 197 |
| 標的原子の陽子数 | ——— | 6 | 14 | 63 |
| 入射原子 | Ca | H | Si | Eu |
| 入射原子の質量 | 40 | 1 | 28 | 197 |
| 入射原子の陽子数 | 20 | 1 | 14 | 63 |
| テスト粒子数 | 400 | 800 | 800 | 200 |
| 時間ステップ数 | 900 | 500 | 500 | 500 |

Table 4.2 Execution time of original version and vectorized version

| | | ﾃｽﾄ・ｹｰｽ1 | ﾃｽﾄ・ｹｰｽ2 | ﾃｽﾄ・ｹｰｽ3 | ﾃｽﾄ・ｹｰｽ4 |
|---|---|---|---|---|---|
| (1)オリジナル版(ｽｶﾗ) *1 | | 1:20:10.67 | 42:49.63 | 1:03:17.84 | 1:28:23.43 |
| (2)オリジナル版(ﾍﾞｸﾄﾙ) *2 | | 11:12.14<br>( 5:00.29) | 6:21.40<br>( 2:43.97) | 13:30.99<br>( 7:02.62) | 19:30.80<br>(10:29.73) |
| (3)ベクトル化版(ﾍﾞｸﾄﾙ) *3 | | 5:58.00<br>( 4:24.17) | 3:03.08<br>( 2:14.02) | 6:30.18<br>( 3:36.97) | 10:33.22<br>( 5:18.57) |
| 倍率 | (1) → (2) | 7.16 | 6.74 | 4.68 | 4.53 |
| | (2) → (3) | 1.88 | 2.08 | 2.08 | 1.85 |
| | (1) → (3) | 13.4 | 14.0 | 9.7 | 8.4 |

(備考) 測定値はCPU 時間、VU時間であり、時：分：秒である。( ) 内がVU時間。
倍率は、CPU 時間の比である。
＊1～3のコンパイラ・オプションは以下のとおりである。
＊1： -Oe -Wv,-sc
＊2： -Oe
＊3： -Oe
＊3： -Oe

Table 4.3 Cost down ratios in updated routines

| | ﾃｽﾄ・ｹｰｽ1 | ﾃｽﾄ・ｹｰｽ2 | ﾃｽﾄ・ｹｰｽ3 | ﾃｽﾄ・ｹｰｽ4 |
|---|---|---|---|---|
| DBNS ルーチン | 85 % | 89 % | 77 % | 70 % |
| SMBAR ルーチン | 11 % | 13 % | 14 % | 13 % |
| ITBRAT ルーチン | 14 % | 5 % | 2 % | 2 % |
| XSORT ルーチン | 13 % | 25 % | 25 % | 30 % |
| ABSOR ルーチン | 95 % | 89 % | 80 % | 71 % |
| PERMUT ルーチン | 68 % | 77 % | 83 % | 82 % |
| RCDBLT ルーチン | 78 % | 86 % | 50 % | 30 % |

(備考) コスト・ダウン率 (%) ＝ ($C_{oe}-C_{VEC}$) ÷ $C_{oe}\times 100$
$C_{oe}$ ：Table 4.1 におけるオリジナル版(ベクトル)のサンプラのカウント値
$C_{VEC}$ ：Table 4.1 におけるベクトル化版(ベクトル)のサンプラのカウント値

Table 4.4 Test case 5

| | 標 的 側 | | | 入 射 側 | | | ﾃｽﾄ粒子数 | 時間ｽﾃｯﾌﾟ数 |
|---|---|---|---|---|---|---|---|---|
| | 原子 | 質量 | 陽子数 | 原子 | 質量 | 陽子数 | | |
| ﾃｽﾄ・ｹｰｽ 5 | Pb | 208 | 82 | Pb | 208 | 82 | 1000 | 900 |

※他のデータはテスト・ケース 4 と同じ

Table 4.5 Execution time and result of SAMPLER in test case 5

| | ABSOR 対応未コード | ABSOR 対応済コード |
|---|---|---|
| 実行時間 | CPU Time 1:07:38.48<br>VU Time 23:38.24 | CPU Time 54:55.98<br>VU Time 25:26.62 |
| サンプラの結果 | Count Percent<br>ABSOR : 85802 20.6% 1 位<br>( DBNS 17.9% 2 位) | Count Percent<br>ABSOR : 17830 5.3% 6 位<br>( DBNS 23.0% 1 位) |

Table 4.6 Execution times of global function SUM

| 測定配列 ＼ PE 数 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|
| RHSVBI( NX*NX*NZ) 要素数： 102541 | 8.1 | 12.6 | 20.1 | 36.7 |
| RHVVBI( NX*NX*NZ, 0:3 ) 要素数： 410164 | 20.6 | 44.6 | 77.3 | 145.4 |
| Rw( MAXPAR ) 要素数： 416000 | 39.1 | 53.8 | 83.7 | 150.4 |
| Pw( 0:3, MAXPAR ) 要素数：1664000 | 153.7 | 214.0 | 334.5 | 597.6 |
| IVA( MAXANT ) 要素数： 30000 | 3.2 | 4.9 | 5.8 | 10.4 |
| NDBNSw( 0:200 ) 要素数： 201 | 0 | 0 | 0.1 | 0.2 |

(単位：ミリ秒)

Table 4.7 Effect of parallelization

| 並列化対象箇所 | | コスト比率 (%) | 走行時間 (1処理単位) (ms) | 並列化効果 (%) PE = 2 | PE = 4 | PE = 8 | PE = 16 |
|---|---|---|---|---|---|---|---|
| TPPRED DO 2661 | | 5.2 | 206.2 | × | × | × | × |
| TPCORR DO 2621 | | 4.7 | 186.6 | × | × | × | × |
| TPSORT DO 2711 | | 1.1 | 39.5 | △ 0.3 | △ 0.5 | △ 0.4 | — 0.0 |
| XSORT DO 2811 | | 4.3 | 39.8 | △ 1.3 | △ 1.9 | △ 1.6 | △ 0.1 |
| ABSOR DO 1511 | | 5.3 | 197.5 | △ 2.0 | △ 3.2 | — | — |
| DENS | DO 1342･1343 | 14.8 | 24.0 | △ 2.3 | △ 3.3 | △ 0.6 | × |
| | DO 1382･1383 | 7.5 | 69.8 | △ 1.5 | △ 0.8 | × | × |
| RCDELT DO 2111･2121 | | 3.6 | 134.8 | × | × | × | × |

△：数%程度のコスト・ダウンしか期待できない
×：オーバヘッドの方が大きく並列化の対象にできない

```
7.97,6.808,2.79,3.97,51.674,-14.659 :K=200, M*=0.83, RHOO=0.17 ML2
0.17,0.83   RHOO, XMSTAR          : RHO , BFF, MASS                    (REAL)
40,40       MASSTA,MASSPR         : TARGET/PROJECTILE MASS             (INT)
20,20       MSTAPR,MSPRPR         : NUMBER OF PCUT80TONS               (INT)
  00.0 ELAB                       : KIN. ENERGY OF PROJECTILE          (REAL)
0., 0., 0.00        ZOTA,ZOPR,B   : DISPLACEM. OF PROJ., IMPACT P.     (REAL)
0.1,1.0,1.0         DT,DX,DZ      : SPACE AND TIME STEP SIZE           (REAL)
0.25        ALPHA                 : PING PARAM.                        (REAL)
1           ICOLIN                : -1 CASC., 0 VLAS., 1 ALL           (INT)
0           JPART                 : PARTICLE PRODUCTION
1           NNUCL                 : # OF NUCLEI (=1 ONLY PROJ.)        (INT)
2           INSYS                 : =0 LAB SYSTEM, ELSE CM             (INT)
2           INITD                 : 1: BOX   2: WOODS-SAXON            (INT)
25          ISEED                 : ODD !! RANDOM NUMBER               (INT)
9990        LIMIT                 : ALLOWED CPU TIME                   (INT)
900         NTMAX                 : NUMBER OF TIMESTEPS                (INT)
400         NUM                   : NUMBER OF CUT80PARTICLES           (INT)
0,50        IVSAM,ITSTEP          : =0 NO VSAM; # OF VSAM TIMESTEPS    (INT)
100         IPSKIP                : FRACTION OF TIMESTP. FOR VSAM      (INT)
100         NFRSUM                : SUMMARY AFTER NFRSUM TIMESTEPS     (INT)
100         NFRPRI                : PRINTOUT AFTER NFRPRI TIMESTEP     (INT)
1.0         CSFAC                 : SCALING FACTOR FOR X-SECTION       (REAL)
0.0         FACGS                 : SCALING FACTOR FOR GSBAR           (REAL)
0.0         FACGV                 : SCALING FACTOR FOR GVBAR           (REAL)
```

Fig. 4.1 Input data for test case 1 (1/4)

```
7.97,6.808,2.79,3.97,51.674,-14.659 :K=200, M*=0.83, RHOO=0.17 ML2
0.17,0.83   RHOO, XMSTAR          : RHO , BFF, MASS                    (REAL)
12,1        MASSTA,MASSPR         : TARGET/PROJECTILE MASS             (INT)
 6,1        MSTAPR,MSPRPR         : NUMBER OF PCUT80TONS               (INT)
4000.0 ELAB                       : KIN. ENERGY OF PROJECTILE          (REAL)
5.,-10., 0.00       ZOTA,ZOPR,B   : DISPLACEM. OF PROJ., IMPACT P.     (REAL)
0.1,0.8,0.8         DT,DX,DZ      : SPACE AND TIME STEP SIZE           (REAL)
0.25        ALPHA                 : PING PARAM.                        (REAL)
1           ICOLIN                : -1 CASC., 0 VLAS., 1 ALL           (INT)
0           JPART                 : PARTICLE PRODUCTION
2           NNUCL                 : # OF NUCLEI (=1 ONLY PROJ.)        (INT)
2           INSYS                 : =0 LAB SYSTEM, ELSE CM             (INT)
2           INITD                 : 1: BOX   2: WOODS-SAXON            (INT)
25          ISEED                 : ODD !! RANDOM NUMBER               (INT)
9990        LIMIT                 : ALLOWED CPU TIME                   (INT)
500         NTMAX                 : NUMBER OF TIMESTEPS                (INT)
800         NUM                   : NUMBER OF CUT80PARTICLES           (INT)
0,50        IVSAM,ITSTEP          : =0 NO VSAM; # OF VSAM TIMESTEPS    (INT)
100         IPSKIP                : FRACTION OF TIMESTP. FOR VSAM      (INT)
100         NFRSUM                : SUMMARY AFTER NFRSUM TIMESTEPS     (INT)
100         NFRPRI                : PRINTOUT AFTER NFRPRI TIMESTEP     (INT)
1.0         CSFAC                 : SCALING FACTOR FOR X-SECTION       (REAL)
0.0         FACGS                 : SCALING FACTOR FOR GSBAR           (REAL)
0.0         FACGV                 : SCALING FACTOR FOR GVBAR           (REAL)
```

Fig. 4.1 Input data for test case 2 (2/4)

```
7.97,6.808,2.79,3.97,51.674,-14.659 :K=200, M*=0.83, RHO0=0.17 ML2
0.17,0.83   RHO0, XMSTAR          : RHO , BFF, MASS                    (REAL)
28,28       MASSTA,MASSPR         : TARGET/PROJECTILE MASS             (INT)
14,14       MSTAPR,MSPRPR         : NUMBER OF PCUT80TONS               (INT)
2100.0 ELAB                       : KIN. ENERGY OF PROJECTILE          (REAL)
8.,-8.,0.00         ZOTA,ZOPR,B   : DISPLACEM. OF PROJ.,IMPACT P.      (REAL)
0.1,0.8,0.8         DT,DX,DZ      : SPACE AND TIME STEP SIZE           (REAL)
0.25        ALPHA                 : PING PARAM.                        (REAL)
1           ICOLIN                : -1 CASC.,0 VLAS.,1 ALL             (INT)
0           JPART                 : PARTICLE PRODUCTION
2           NNUCL                 : # OF NUCLEI (=1 ONLY PROJ.)        (INT)
1           INSYS                 : =0 LAB SYSTEM, ELSE CM             (INT)
2           INITD                 : 1: BOX   2: WOODS-SAXON            (INT)
25          ISEED                 : ODD !! RANDOM NUMBER               (INT)
9990        LIMIT                 : ALLOWED CPU TIME                   (INT)
500         NTMAX                 : NUMBER OF TIMESTEPS                (INT)
800         NUM                   : NUMBER OF TEST-PARTICLES           (INT)
0,50        IVSAM,ITSTEP          : =0 NO VSAM; # OF VSAM TIMESTEPS    (INT)
100         IPSKIP                : FRACTION OF TIMESTP. FOR VSAM      (INT)
100         NFRSUM                : SUMMARY AFTER NFRSUM TIMESTEPS     (INT)
100         NFRPRI                : PRINTOUT AFTER NFRPRI TIMESTEP     (INT)
1.0         CSFAC                 : SCALING FACTOR FOR X-SECTION       (REAL)
0.0         FACGS                 : SCALING FACTOR FOR GSBAR           (REAL)
0.0         FACGV                 : SCALING FACTOR FOR GVBAR           (REAL)
```

Fig. 4.1 Input data for test case 3 (3/4)

```
7.97,6.808,2.79,3.97,51.674,-14.659 :K=200, M*=0.83, RHO0=0.17 ML2
0.17,0.83   RHO0, XMSTAR          : RHO , BFF, MASS                    (REAL)
197,197     MASSTA,MASSPR         : TARGET/PROJECTILE MASS             (INT)
 63, 63     MSTAPR,MSPRPR         : NUMBER OF PCUT80TONS               (INT)
2000.0 ELAB                       : KIN. ENERGY OF PROJECTILE          (REAL)
13.,-13.,0.00       ZOTA,ZOPR,B   : DISPLACEM. OF PROJ.,IMPACT P.      (REAL)
0.1,0.8,0.8         DT,DX,DZ      : SPACE AND TIME STEP SIZE           (REAL)
0.25        ALPHA                 : PING PARAM.                        (REAL)
1           ICOLIN                : -1 CASC.,0 VLAS.,1 ALL             (INT)
0           JPART                 : PARTICLE PRODUCTION
2           NNUCL                 : # OF NUCLEI (=1 ONLY PROJ.)        (INT)
2           INSYS                 : =0 LAB SYSTEM, ELSE CM             (INT)
2           INITD                 : 1: BOX   2: WOODS-SAXON            (INT)
13          ISEED                 : ODD !! RANDOM NUMBER               (INT)
99990       LIMIT                 : ALLOWED CPU TIME                   (INT)
500         NTMAX                 : NUMBER OF TIMESTEPS                (INT)
200         NUM                   : NUMBER OF CUT80PARTICLES           (INT)
0,50        IVSAM,ITSTEP          : =0 NO VSAM; # OF VSAM TIMESTEPS    (INT)
100         IPSKIP                : FRACTION OF TIMESTP. FOR VSAM      (INT)
100         NFRSUM                : SUMMARY AFTER NFRSUM TIMESTEPS     (INT)
100         NFRPRI                : PRINTOUT AFTER NFRPRI TIMESTEP     (INT)
1.0         CSFAC                 : SCALING FACTOR FOR X-SECTION       (REAL)
0.0         FACGS                 : SCALING FACTOR FOR GSBAR           (REAL)
0.0         FACGV                 : SCALING FACTOR FOR GVBAR           (REAL)
```

Fig. 4.1 Input data for test case 4 (4/4)

```
#!/bin/csh -f
#@$-eo
#@$-q vppm
#@$-C exsmpler
cd $QSUB_WORKDIR
setenv F7PARASAMP file:samp.data,interval:1,type:vtime
echo 'running RBUU/vpp-vector -Oe Test'
timex -H ../xrbuu.vec.Oe < input.cfg
echo 'RBUU/vpp-vector finished'
parasamp ../xrbuu.vec.Oe
```

Fig. 4.2 Shell script for execution of SAMPLER

```
#!/bin/csh -f                          ※最適化オプション：－Oe
#@$-q gspc                                        (省略時解釈値)
#@$-eo
#@$-C DCNJOY
cd $QSUB_WORKDIR
frtpx -o xrbuu.vec.Oe -Z CMP.vecoe -Bpe -Si *.f
```

Fig. 4.3 Shell script for compiling and linking

```
#!/bin/csh -f                          ※srbuu.org.f ：
#@$-eo                                     対象ソースプログラムを
#@$-q gsps                                 1本化したファイル
#@$-lM 200mb
#@$-lT 1:00:00
#@$-C exanalze
cd $QSUB_WORKDIR
echo 'running analyzer for RBUU.Oe'
afrt -px240 -e -f analyze.out srbuu.org.f < input.cfg
echo 'analyzer for RBUU finished'
```

Fig. 4.4 Shell script for execution of ANALYZER

<テスト・ケース１>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 27682 | 40.3 | 531 | dens__ |
| 17063 | 24.8 | 1564 | smear__ |
| 6905 | 10.1 | 1559 | iterat__ |
| 5845 | 8.5 | 2008 | xsort__ |
| 2330 | 3.4 | - | absor__ |
| 2212 | 3.2 | 706 | rcdelt__ |
| 1266 | 1.8 | 22 | permut__ |
| 1203 | 1.8 | 588 | tpcorr__ |
| 1151 | 1.7 | 566 | tppred__ |
| 636 | 0.9 | - | suraf__ |
| 552 | 0.8 | - | resel__ |
| 464 | 0.7 | 2011 | xsorta__ |
| 451 | 0.7 | 2040 | tpsort__ |
| 176 | 0.3 | - | matri2__ |
| 137 | 0.2 | 41 | init__ |
| 106 | 0.2 | - | cndns__ |
| 92 | 0.1 | 460 | tpblck__ |
| 92 | 0.1 | - | cross__ |
| 80 | 0.1 | 2037 | MAIN___ |
| 60 | 0.1 | - | ranf__ |
| 33 | 0.0 | - | ran__ |
| 33 | 0.0 | - | rnset__ |
| 32 | 0.0 | - | pboo__ |
| 26 | 0.0 | - | __start |
| 18 | 0.0 | - | cptime__ |
| 15 | 0.0 | 41 | etot__ |
| 10 | 0.0 | 512 | pdens__ |
| 7 | 0.0 | - | smrini__ |
| 5 | 0.0 | 41 | summry__ |
| 2 | 0.0 | - | anoutm__ |
| 1 | 0.0 | 2048 | asigin__ |
| 1 | 0.0 | - | absout__ |
| 68686 | | 1381 | TOTAL |

<テスト・ケース２>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 16883 | 43.1 | 427 | dens__ |
| 9546 | 24.3 | 1588 | smear__ |
| 3218 | 8.2 | 1673 | iterat__ |
| 3177 | 8.1 | 1974 | xsort__ |
| 1774 | 4.5 | 4 | absor__ |
| 1160 | 3.0 | 606 | rcdelt__ |
| 688 | 1.8 | 14 | permut__ |
| 558 | 1.4 | - | resel__ |
| 380 | 1.0 | 598 | tpcorr__ |
| 362 | 0.9 | 557 | tppred__ |
| 270 | 0.7 | 2048 | tpsort__ |
| 268 | 0.7 | 1921 | xsorta__ |
| 228 | 0.6 | - | suraf__ |
| 119 | 0.3 | - | siga__ |
| 106 | 0.3 | 80 | r4__ |
| 105 | 0.3 | - | init__ |
| 104 | 0.3 | - | matri2__ |
| 41 | 0.1 | - | ranf__ |
| 35 | 0.1 | 1998 | MAIN___ |
| 30 | 0.1 | - | rnset__ |
| 28 | 0.1 | - | cross__ |
| 26 | 0.1 | - | cndns__ |
| 22 | 0.1 | - | ran__ |
| 15 | 0.0 | 264 | tpblck__ |
| 11 | 0.0 | - | __start |
| 9 | 0.0 | 41 | etot__ |
| 9 | 0.0 | 202 | tppreda__ |
| 7 | 0.0 | - | cptime__ |
| 6 | 0.0 | - | pdens__ |
| 6 | 0.0 | - | pboo__ |
| 6 | 0.0 | 285 | tpcorra__ |
| 5 | 0.0 | - | anoutm__ |
| 3 | 0.0 | - | absout__ |
| 1 | 0.0 | - | r3__ |
| 1 | 0.0 | - | mogri__ |
| 1 | 0.0 | - | smrini__ |
| 1 | 0.0 | 41 | summry__ |
| 1 | 0.0 | - | antiou__ |
| 39210 | | 1395 | TOTAL |

<テスト・ケース３>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 41074 | 49.3 | 72 | dens__ |
| 10000 | 12.0 | 1586 | smear__ |
| 7047 | 8.5 | 2037 | xsort__ |
| 5726 | 6.9 | 1372 | iterat__ |
| 4792 | 5.7 | 4 | absor__ |
| 4337 | 5.2 | 6 | permut__ |
| 1631 | 2.0 | 346 | rcdelt__ |
| 1398 | 1.7 | 612 | tpcorr__ |
| 1352 | 1.6 | 581 | tppred__ |
| 985 | 1.2 | 2048 | tpsort__ |
| 979 | 1.2 | - | siga__ |
| 806 | 1.0 | - | suraf__ |
| 743 | 0.9 | 86 | r4__ |
| 564 | 0.7 | - | resel__ |
| 322 | 0.4 | 710 | init__ |
| 307 | 0.4 | 1743 | xsorta__ |
| 218 | 0.3 | - | cross__ |
| 200 | 0.2 | - | ranf__ |
| 156 | 0.2 | 733 | tpblck__ |
| 133 | 0.2 | 1960 | MAIN___ |
| 124 | 0.1 | - | rnset__ |
| 124 | 0.1 | 140 | cndns__ |
| 108 | 0.1 | - | matri2__ |
| 68 | 0.1 | - | ran__ |
| 54 | 0.1 | - | pboo__ |
| 34 | 0.0 | - | __start |
| 30 | 0.0 | 527 | tpcorra__ |
| 18 | 0.0 | 489 | tppreda__ |
| 15 | 0.0 | 512 | pdens__ |
| 9 | 0.0 | 41 | etot__ |
| 8 | 0.0 | - | cptime__ |
| 4 | 0.0 | 41 | summry__ |
| 4 | 0.0 | - | antiou__ |
| 3 | 0.0 | - | anoutm__ |
| 2 | 0.0 | - | smrini__ |
| 2 | 0.0 | - | sigtot__ |
| 2 | 0.0 | - | absout__ |
| 1 | 0.0 | 2048 | asigin__ |
| 1 | 0.0 | - | kinpar__ |
| 1 | 0.0 | 11 | apboo__ |
| 1 | 0.0 | 2026 | r3__ |
| 1 | 0.0 | - | tpblin__ |
| 83384 | | 907 | TOTAL |

<テスト・ケース４>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 56253 | 46.9 | 55 | dens__ |
| 11808 | 9.8 | 2039 | xsort__ |
| 10535 | 8.8 | 1587 | smear__ |
| 9194 | 7.7 | 1180 | iterat__ |
| 8233 | 6.9 | 6 | permut__ |
| 4807 | 4.0 | 4 | absor__ |
| 3000 | 2.5 | 389 | rcdelt__ |
| 2744 | 2.3 | 615 | tpcorr__ |
| 2636 | 2.2 | 574 | tppred__ |
| 1691 | 1.4 | 2048 | tpsort__ |
| 1622 | 1.4 | 73 | r4__ |
| 1549 | 1.3 | - | suraf__ |
| 1208 | 1.0 | - | cross__ |
| 979 | 0.8 | - | siga__ |
| 560 | 0.5 | - | resel__ |
| 477 | 0.4 | 1379 | init__ |
| 430 | 0.4 | 261 | tpblck__ |
| 421 | 0.4 | - | ranf__ |
| 316 | 0.3 | - | cndns__ |
| 294 | 0.2 | 1846 | xsorta__ |
| 268 | 0.2 | - | rnset__ |
| 259 | 0.2 | - | pboo__ |
| 243 | 0.2 | 1998 | MAIN___ |
| 135 | 0.1 | - | ran__ |
| 107 | 0.1 | - | matri2__ |
| 97 | 0.1 | - | __start |
| 26 | 0.0 | 442 | tppreda__ |
| 25 | 0.0 | 512 | pdens__ |
| 21 | 0.0 | 540 | tpcorra__ |
| 12 | 0.0 | - | anoutm__ |
| 8 | 0.0 | 41 | etot__ |
| 8 | 0.0 | - | cptime__ |
| 6 | 0.0 | 1245 | summry__ |
| 6 | 0.0 | - | antiou__ |
| 3 | 0.0 | - | smrini__ |
| 1 | 0.0 | - | inecho__ |
| 1 | 0.0 | - | mogri__ |
| 1 | 0.0 | - | r3__ |
| 1 | 0.0 | - | rcout__ |
| 1 | 0.0 | - | zwist__ |
| 1 | 0.0 | - | tpblin__ |
| 1 | 0.0 | - | absout__ |
| 119988 | | 835 | TOTAL |

Fig. 4.5 Computational cost distributions in RBUU original version

```
* TEST-CACE 1 * -Oe
vectorize - routine list
routine    ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect  overhd
DENS          21708  .1796E11  18.9  .4617E11   3.3        254    65.5    2.4-  2.6       0
vectorize - loop list ------ DENS
 do-id  kind  v   ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect
 1331   do    S      18088  .1325E11  73.8  .1325E11  28.7     102541     0.0    1.0-  1.0
 1371   do    S       3620  .2652E10  14.8  .2652E10   5.7     102541     0.0    1.0-  1.0
 1341   do    V   19309914  .1024E10   5.7  .6656E10  14.4         15   100.0    3.9-  9.1
 1381   do    V    3864018  .6059E09   3.4  .3938E10   8.5         15   100.0    3.9-  9.1
 1311   do    V      14480  .2221E09   1.2  .1039E11  22.5     102541   100.0   40.3- 54.6
 1321   do    V      18088  .1982E09   1.1  .9274E10  20.1     102541   100.0   40.3- 54.6

* TEST-CASE 2 * -Oe
vectorize - routine list
routine    ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect  overhd
DENS          12161  .1068E11  20.2  .2377E11   3.1        160    62.1    2.0-  2.3       0
vectorize - loop list ------ DENS
 do-id  kind  v   ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect
 1331   do    S      10131  .7508E10  70.3  .7508E10  31.6     102541     0.0    1.0-  1.0
 1371   do    S       2030  .1504E10  14.1  .1504E10   6.3     102541     0.0    1.0-  1.0
 1341   do    V   16875181  .9013E09   8.4  .2343E10   9.9          6   100.0    1.3-  3.9
 1381   do    V    3377137  .5336E09   5.0  .1387E10   5.8          6   100.0    1.3-  3.9
 1311   do    V       8120  .1245E09   1.2  .5828E10  24.5     102541   100.0   40.3- 54.6
 1321   do    V      10131  .1110E09   1.0  .5194E10  21.9     102541   100.0   40.3- 54.6

* TEST-CASE 3 * -Oe
vectorize - routine list
routine    ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect  overhd
DENS          13010  .1793E11  24.3  .3889E11   4.3         24    70.2    1.6-  2.5       0
vectorize - loop list ------ DENS
 do-id  kind  v   ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect
 1331   do    S      10980  .9763E10  54.5  .9763E10  25.1     102541     0.0    1.0-  1.0
 1341   do    V  .1344E09  .3953E10  22.1  .1028E11  26.4          3   100.0    1.3-  3.9
 1381   do    V   26121859  .2142E10  11.9  .5568E10  14.3          3   100.0    1.3-  3.9
 1371   do    S       2030  .1823E10  10.2  .1823E10   4.7     102541     0.0    1.0-  1.0
 1311   do    V       8120  .1245E09   0.7  .5828E10  15.0     102541   100.0   40.3- 54.6
 1321   do    V      10980  .1203E09   0.7  .5630E10  14.5     102541   100.0   40.3- 54.6

* TEST-CASE 4 * -Oe
vectorize - routine list
routine    ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect  overhd
DENS          14180  .2572E11  27.0  .5645E11   5.3         19    75.9    1.5-  2.6       0
vectorize - loop list ------ DENS
 do-id  kind  v   ex-count     v-cost     %     s-cost     %     v-leng  v-rate     v-effect
 1331   do    S      12150  .1168E11  45.4  .1168E11  20.7     102541     0.0    1.0-  1.0
 1341   do    V  .2114E09  .7921E10  30.8  .2059E11  36.5          4   100.0    1.3-  3.9
 1381   do    V   35248963  .3910E10  15.2  .1017E11  18.0          4   100.0    1.3-  3.9
 1371   do    S       2030  .1951E10   7.6  .1951E10   3.5     102541     0.0    1.0-  1.0
 1321   do    V      12150  .1331E09   0.5  .6229E10  11.0     102541   100.0   40.3- 54.6
 1311   do    V       8120  .1245E09   0.5  .5828E10  10.3     102541   100.0   40.3- 54.6
```

Fig. 4.6 Dynamic behavior of DENS routine in RBUU original version

```
* TEST-CACE 1 * -Oe
vectorize - routine list
routine    ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect  overhd
SMEAR         47000  .2696E11  28.4  .1043E13  75.4     764    99.6    34.0- 44.0       0
vectorize - loop list ------ SMEAR
 do-id  kind  v   ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect
 2501   do    V    4052002  .1037E11  38.5  .4853E12  46.5     1597   100.0   40.3- 54.6
 2441   do    V    4189358  .8577E10  31.8  .4014E12  38.5     1597   100.0   40.3- 54.6
 2511   do    S    4052002  .4267E10  15.8  .4267E10   0.4       39     0.0    1.0- 1.0
 2451   do    V    4189358  .3145E10  11.7  .1472E12  14.1     1597   100.0   40.3- 54.6
 2491   do    V    4052002  .4103E09   1.5  .4267E10   0.4       39   100.0    7.8- 13.0
 2431   do    S    2867000  79597802   0.3  79597802   0.0        1     0.0    1.0- 1.0

* TEST-CASE 2 * -Oe
vectorize - routine list
routine    ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect  overhd
SMEAR         26298  .1508E11  28.5  .5832E12  77.0     764    99.6    34.0- 44.0       0
vectorize - loop list ------ SMEAR
 do-id  kind  v   ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect
 2501   do    V    2266485  .5801E10  38.5  .2715E12  46.5     1597   100.0   40.3- 54.6
 2441   do    V    2343315  .4798E10  31.8  .2245E12  38.5     1597   100.0   40.3- 54.6
 2511   do    S    2266485  .2387E10  15.8  .2387E10   0.4       39     0.0    1.0- 1.0
 2451   do    V    2343315  .1759E10  11.7  .8233E11  14.1     1597   100.0   40.3- 54.6
 2491   do    V    2266485  .2295E09   1.5  .2387E10   0.4       39   100.0    7.8- 13.0
 2431   do    S    1604178  44522985   0.3  44522985   0.0        1     0.0    1.0- 1.0

* TEST-CASE 3 * -Oe
vectorize - routine list
routine    ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect  overhd
SMEAR         27996  .1575E11  21.4  .6090E12  67.6     761    99.6    34.0- 44.0       0
vectorize - loop list ------ SMEAR
 do-id  kind  v   ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect
 2501   do    V    2366667  .6057E10  38.5  .2835E12  46.5     1597   100.0   40.3- 54.6
 2441   do    V    2446893  .5010E10  31.8  .2345E12  38.5     1597   100.0   40.3- 54.6
 2511   do    S    2366667  .2492E10  15.8  .2492E10   0.4       39     0.0    1.0- 1.0
 2451   do    V    2446893  .1837E10  11.7  .8597E11  14.1     1597   100.0   40.3- 54.6
 2491   do    V    2366667  .2396E09   1.5  .2492E10   0.4       39   100.0    7.8- 13.0
 2431   do    S    1707756  46490967   0.3  46490967   0.0        1     0.0    1.0- 1.0

* TEST-CASE 4 * -Oe
vectorize - routine list
routine    ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect  overhd
SMEAR         30336  .1666E11  17.5  .6445E12  60.0     757    99.6    34.0- 44.0       0
vectorize - loop list ------ SMEAR
 do-id  kind  v   ex-count    v-cost     %     s-cost     %     v-leng  v-rate    v-effect
 2501   do    V    2504727  .6410E10  38.5  .3000E12  46.5     1597   100.0   40.3- 54.6
 2441   do    V    2589633  .5302E10  31.8  .2481E12  38.5     1597   100.0   40.3- 54.6
 2511   do    S    2504727  .2637E10  15.8  .2637E10   0.4       39     0.0    1.0- 1.0
 2451   do    V    2589633  .1944E10  11.7  .9098E11  14.1     1597   100.0   40.3- 54.6
 2491   do    V    2504727  .2536E09   1.5  .2637E10   0.4       39   100.0    7.8- 13.0
 2431   do    S    1850496  49203027   0.3  49203027   0.0        1     0.0    1.0- 1.0
```

Fig. 4.7 Dynamic behavior of SMEAR routine in RBUU original vergion

＊ TEST-CACE 1 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ITERAT | 3614 | .4230E11 | 44.6 | .2304E12 | 16.7 | 2999 | 83.4 | 5.4- 5.5 | 0 |

vectorize - loop list ——— ITERAT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1831 | do | V | 10846 | .3712E11 | 87.8 | .5672E11 | 24.6 | 102541 | 35.3 | 1.5- 1.5 |
| 1771 | do | V | 220454 | .1750E10 | 4.1 | .1372E11 | 6.0 | 1681 | 89.1 | 7.7- 8.0 |
| 1861 | do | V | 661606 | .1283E10 | 3.0 | .6006E11 | 26.1 | 1681 | 100.0 | 40.3- 54.6 |
| 1811 | do | V | 440908 | .8552E09 | 2.0 | .4002E11 | 17.4 | 1681 | 100.0 | 40.3- 54.6 |
| 1911 | do | V | 14456 | .4751E09 | 1.1 | .2224E11 | 9.6 | 102541 | 100.0 | 40.3- 54.6 |
| 1841 | do | V | 10846 | .3263E09 | 0.8 | .1527E11 | 6.6 | 16000 | 100.0 | 40.3- 54.6 |

＊ TEST-CASE 2 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ITERAT | 2021 | .2355E11 | 44.4 | .1236E12 | 16.3 | 2943 | 82.7 | 5.2- 5.3 | 0 |

vectorize - loop list ——— ITERAT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1831 | do | V | 6068 | .2077E11 | 88.2 | .3173E11 | 25.7 | 102541 | 35.3 | 1.5- 1.5 |
| 1771 | do | V | 123281 | .9763E09 | 4.1 | .7669E10 | 6.2 | 1681 | 89.2 | 7.7- 8.0 |
| 1861 | do | V | 370148 | .7179E09 | 3.0 | .3360E11 | 27.2 | 1681 | 100.0 | 40.3- 54.6 |
| 1811 | do | V | 246562 | .4782E09 | 2.0 | .2238E11 | 18.1 | 1681 | 100.0 | 40.3- 54.6 |
| 1911 | do | V | 8084 | .2657E09 | 1.1 | .1243E11 | 10.1 | 102541 | 100.0 | 40.3- 54.6 |
| 1881 | do | V | 370148 | .1462E09 | 0.6 | .6844E10 | 5.5 | 1681 | 100.0 | 40.3- 54.6 |

＊ TEST-CASE 3 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ITERAT | 2021 | .2723E11 | 36.9 | .1642E12 | 18.2 | 3173 | 85.2 | 5.9- 6.1 | 0 |

vectorize - loop list ——— ITERAT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1831 | do | V | 6917 | .2367E11 | 86.9 | .3617E11 | 22.0 | 102541 | 35.3 | 1.5- 1.5 |
| 1771 | do | V | 123281 | .9774E09 | 3.6 | .7670E10 | 4.7 | 1681 | 89.2 | 7.7- 8.0 |
| 1861 | do | V | 421937 | .8184E09 | 3.0 | .3830E11 | 23.3 | 1681 | 100.0 | 40.3- 54.6 |
| 1841 | do | V | 6917 | .5322E09 | 2.0 | .2491E11 | 15.2 | 40916 | 100.0 | 40.3- 54.6 |
| 1811 | do | V | 246562 | .4782E09 | 1.8 | .2238E11 | 13.6 | 1681 | 100.0 | 40.3- 54.6 |
| 1791 | do | V | 4042 | .3069E09 | 1.1 | .1436E11 | 8.7 | 40384 | 100.0 | 40.3- 54.6 |

＊ TEST-CASE 4 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ITERAT | 2021 | .3227E11 | 33.9 | .2176E12 | 20.2 | 3414 | 87.0 | 6.6- 6.9 | 0 |

vectorize - loop list ——— ITERAT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1831 | do | V | 8087 | .2768E11 | 85.8 | .4228E11 | 19.4 | 102541 | 35.3 | 1.5- 1.5 |
| 1841 | do | V | 8087 | .1121E10 | 3.5 | .5247E11 | 24.1 | 73726 | 100.0 | 40.3- 54.6 |
| 1771 | do | V | 123281 | .1006E10 | 3.1 | .7698E10 | 3.5 | 1681 | 88.8 | 7.5- 7.8 |
| 1861 | do | V | 493307 | .9568E09 | 3.0 | .4478E11 | 20.6 | 1681 | 100.0 | 40.3- 54.6 |
| 1791 | do | V | 4042 | .5604E09 | 1.7 | .2623E11 | 12.1 | 73734 | 100.0 | 40.3- 54.6 |
| 1811 | do | V | 246562 | .4782E09 | 1.5 | .2238E11 | 10.3 | 1681 | 100.0 | 40.3- 54.6 |

Fig. 4.8 Dynamic behavio of ITERAT routine in RBUU original version

```
* TEST-CACE 1 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect  overhd
XSORT          3624  .2308E10   2.4  .2818E11   2.0     35977    93.8  11.7- 12.7       0
vectorize - loop list ------- XSORT -------------------------------------------
 do-id  kind  v   ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect
 2861   do    S       3620  .1043E10  45.2  .1043E10   3.7     16000     0.0   1.0- 1.0
 2821   do    S       3620  .6950E09  30.1  .6950E09   2.5     16000     0.0   1.0- 1.0
 2791   do    M       3624  83009471   3.6  .3885E10  13.8     16000   100.0  40.3- 54.6
 2841   do    V       3624  79402769   3.4  .3716E10  13.2    102540   100.0  40.3- 54.6
 2831   do    V       3624  55582481   2.4  .2601E10   9.2    102541   100.0  40.3- 54.6
 2801   do    V       3624  39701772   1.7  .1858E10   6.6    102541   100.0  40.3- 54.6

* TEST-CASE 2 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect  overhd
XSORT          2036  .9431E09   1.8  .1174E11   1.6     29318    94.0  12.0- 12.9       0
vectorize - loop list ------- XSORT -------------------------------------------
 do-id  kind  v   ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect
 2861   do    S       1855  .3368E09  35.7  .3368E09   2.9     10086     0.0   1.0- 1.0
 2821   do    S       1855  .2245E09  23.8  .2245E09   1.9     10086     0.0   1.0- 1.0
 2851   do    S        181  79828190   8.5  79828190   0.7      9590     0.0   1.0- 1.0
 2811   do    S        181  65948428   7.0  65948428   0.6      9590     0.0   1.0- 1.0
 2841   do    V       2036  44609282   4.7  .2088E10  17.8    102540   100.0  40.3- 54.6
 2831   do    V       2036  31226802   3.3  .1461E10  12.4    102541   100.0  40.3- 54.6

* TEST-CASE 3 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect  overhd
XSORT          2036  .3593E10   4.9  .3339E11   3.7     63159    91.2   9.0- 9.5       0
vectorize - loop list ------- XSORT -------------------------------------------
 do-id  kind  v   ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect
 2861   do    S       1790  .1318E10  36.7  .1318E10   3.9     40895     0.0   1.0- 1.0
 2821   do    S       1790  .8784E09  24.4  .8784E09   2.6     40895     0.0   1.0- 1.0
 2851   do    S        246  .4085E09  11.4  .4085E09   1.2     36134     0.0   1.0- 1.0
 2811   do    S        246  .3375E09   9.4  .3375E09   1.0     36134     0.0   1.0- 1.0
 2791   do    M       2036  .1176E09   3.3  .5500E10  16.5     40320   100.0  40.3- 54.6
 2901   do    V       2036  49114701   1.4  .2299E10   6.9    161280   100.0  40.3- 54.6

* TEST-CASE 4 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect  overhd
XSORT          2036  .6442E10   6.8  .5713E11   5.3    100347    90.7   8.6- 9.1       0
vectorize - loop list ------- XSORT -------------------------------------------
 do-id  kind  v   ex-count     v-cost     %     s-cost     %    v-leng  v-rate    v-effect
 2861   do    S       1808  .2417E10  37.5  .2417E10   4.2     74268     0.0   1.0- 1.0
 2821   do    S       1808  .1611E10  25.0  .1611E10   2.8     74268     0.0   1.0- 1.0
 2851   do    S        228  .7153E09  11.1  .7153E09   1.3     68247     0.0   1.0- 1.0
 2811   do    S        228  .5910E09   9.2  .5910E09   1.0     68247     0.0   1.0- 1.0
 2791   do    M       2036  .2145E09   3.3  .1004E11  17.6     73594   100.0  40.3- 54.6
 2901   do    V       2036  89646008   1.4  .4195E10   7.3    294375   100.0  40.3- 54.6
```

Fig. 4.9 Dynamic behavior of XSORT routine in RBUU original version

＊ TEST-CACE 1 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ABSOR | 900 | .1569E10 | 1.7 | .1569E10 | 0.1 | 1 | 0.0 | 1.0- 1.0 | 0 |

vectorize - loop list ——— ABSOR

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1141 | do | S | 900 | .1569E10 | 100.0 | .1569E10 | 100.0 | 102541 | 0.0 | 1.0- 1.0 |
| ——— | total | | 903 | 18938 | 0.0 | 18938 | 0.0 | 1 | 0.0 | 1.0- 1.0 |
| 1221 | do | S | 1 | 1005 | 0.0 | 1005 | 0.0 | 201 | 0.0 | 1.0- 1.0 |
| 1211 | do | S | 1 | 755 | 0.0 | 755 | 0.0 | 151 | 0.0 | 1.0- 1.0 |
| 1191 | do | V | 2 | 55 | 0.0 | 1510 | 0.0 | 151 | 100.0 | 19.5- 35.1 |
| 1201 | do | V | 2 | 55 | 0.0 | 2010 | 0.0 | 201 | 100.0 | 28.6- 42.9 |

＊ TEST-CASE 2 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ABSOR | 500 | .1122E10 | 2.1 | .1137E10 | 0.2 | 1 | 2.2 | 1.0- 1.0 | 0 |

vectorize - loop list ——— ABSOR

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1141 | do | S | 500 | .8716E09 | 77.7 | .8716E09 | 76.7 | 102541 | 0.0 | 1.0- 1.0 |
| 1171 | do | S | 82808 | .2360E09 | 21.0 | .2360E09 | 20.8 | 7 | 0.0 | 1.0- 1.0 |
| 1181 | do | V | 611385 | 8465331 | 0.8 | 22009860 | 1.9 | 4 | 100.0 | 1.3- 3.9 |
| 1151 | do | | 51270500 | 4328297 | 0.4 | 4328297 | 0.4 | 0 | 0.0 | 1.0- 1.0 |
| 1161 | do | V | 82808 | 1146572 | 0.1 | 2981088 | 0.3 | 4 | 100.0 | 1.3- 3.9 |
| ——— | total | | 503 | 10538 | 0.0 | 10538 | 0.0 | 1 | 0.0 | 1.0- 1.0 |

＊ TEST-CASE 3 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ABSOR | 500 | .3644E10 | 4.9 | .3804E10 | 0.4 | 1 | 6.8 | 1.0- 1.1 | 0 |

vectorize - loop list ——— ABSOR

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1171 | do | S | 390653 | .2637E10 | 72.4 | .2637E10 | 69.3 | 17 | 0.0 | 1.0- 1.0 |
| 1141 | do | S | 500 | .8716E09 | 23.9 | .8716E09 | 22.9 | 102541 | 0.0 | 1.0- 1.0 |
| 1181 | do | V | 6832497 | 94603805 | 2.6 | .2460E09 | 6.5 | 4 | 100.0 | 1.3- 3.9 |
| 1151 | do | | 51270500 | 34663926 | 1.0 | 34663926 | 0.9 | 0 | 0.0 | 1.0- 1.0 |
| 1161 | do | V | 390653 | 5409042 | 0.1 | 14063508 | 0.4 | 4 | 100.0 | 1.3- 3.9 |
| ——— | total | | 503 | 10538 | 0.0 | 10538 | 0.0 | 1 | 0.0 | 1.0- 1.0 |

＊ TEST-CASE 4 ＊ -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| ABSOR | 500 | .3305E10 | 3.5 | .3448E10 | 0.3 | 1 | 6.7 | 1.0- 1.1 | 0 |

vectorize - loop list ——— ABSOR

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 1171 | do | S | 513165 | .2290E10 | 69.3 | .2290E10 | 66.4 | 12 | 0.0 | 1.0- 1.0 |
| 1141 | do | S | 500 | .8716E09 | 26.4 | .8716E09 | 25.3 | 102541 | 0.0 | 1.0- 1.0 |
| 1181 | do | V | 5932975 | 82148885 | 2.5 | .2136E09 | 6.2 | 4 | 100.0 | 1.3- 3.9 |
| 1151 | do | | 51270500 | 53734626 | 1.6 | 53734626 | 1.6 | 0 | 0.0 | 1.0- 1.0 |
| 1161 | do | V | 513165 | 7105362 | 0.2 | 18473940 | 0.5 | 4 | 100.0 | 1.3- 3.9 |
| ——— | total | | 503 | 10538 | 0.0 | 10538 | 0.0 | 1 | 0.0 | 1.0- 1.0 |

Fig. 4.10 Dynamic behavior of ABSOR routine in RBUU original version

```
* TEST-CACE 1 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect overhd
PERMUT       650822  .2658E09  0.3  .3295E09  0.0        15   21.4    1.2- 1.2        0
vectorize - loop list -------- PERMUT -----------------------------------------
 do-id  kind  v   ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect
 1951   do    S     650822  .2395E09  90.1  .2395E09  72.7       22   0.0    1.0- 1.0
 ------ total       650822  19524660   7.3  19524660   5.9        1   0.0    1.0- 1.0
 1941   do    V     650822   6773646   2.5  70445915  21.4       22 100.0    7.8- 13.0

* TEST-CASE 2 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect overhd
PERMUT       369783  92483835  0.2  .1119E09  0.0         9   20.5    1.2- 1.2        0
vectorize - loop list -------- PERMUT -----------------------------------------
 do-id  kind  v   ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect
 1951   do    S     369783  77866953  84.2  77866953  69.6       12   0.0    1.0- 1.0
 ------ total       369783  11093490  12.0  11093490   9.9        1   0.0    1.0- 1.0
 1941   do    V     369783   3523392   3.8  22902045  20.5       12 100.0    3.9- 9.1

* TEST-CASE 2 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect overhd
PERMUT      2634184  .3892E09  0.5  .4397E09  0.0         4   18.6    1.0- 1.2        0
vectorize - loop list -------- PERMUT -----------------------------------------
 do-id  kind  v   ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect
 1951   do    S    2634184  .2787E09  71.6  .2787E09  63.4        6   0.0    1.0- 1.0
 ------ total      2634184  79025520  20.3  79025520  18.0        1   0.0    1.0- 1.0
 1941   do    V    2634184  31524598   8.1  81963955  18.6        6 100.0    1.3- 3.9

* TEST-CASE 4 * -Oe
vectorize - routine list ------------------------------------------------------
routine    ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect overhd
PERMUT      4990136  .7786E09  0.8  .8809E09  0.1         5   18.9    1.0- 1.2        0
vectorize - loop list -------- PERMUT -----------------------------------------
 do-id  kind  v   ex-count     v-cost    %     s-cost    %    v-leng v-rate    v-effect
 1951   do    S    4990136  .5650E09  72.6  .5650E09  64.1        7   0.0    1.0- 1.0
 ------ total      4990136  .1497E09  19.2  .1497E09  17.0        1   0.0    1.0- 1.0
 1941   do    V    4990136  63915375   8.2  .1662E09  18.9        7 100.0    1.3- 3.9
```

Fig. 4.11 Dynamic behavior of PERMUT routine in RBUU original version

* TEST-CACE 1 * -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| RCDELT | 900 | .1156E10 | 1.2 | .2509E10 | 0.2 | 160 | 56.1 | 2.1- 2.2 | 0 |

vectorize - loop list ------- RCDELT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 2111 | do | S | 900 | .6590E09 | 57.0 | .6590E09 | 26.3 | 102541 | 0.0 | 1.0- 1.0 |
| 2131 | do | V | 900 | .4045E09 | 35.0 | .1572E10 | 62.6 | 7661 | 75.9 | 3.8- 3.9 |
| 2161 | do | | 900 | 60452325 | 5.2 | 60452325 | 2.4 | 87 | 0.0 | 1.0- 1.0 |
| 2121 | do | V | 650822 | 25458096 | 2.2 | .1655E09 | 6.6 | 11 | 100.0 | 3.9- 9.1 |
| 2141 | do | V | 900 | 3491618 | 0.3 | 45398789 | 1.8 | 134 | 95.8 | 11.0- 14.5 |
| 2231 | do | S | 900 | 999306 | 0.1 | 999306 | 0.0 | 87 | 0.0 | 1.0- 1.0 |

* TEST-CASE 2 * -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| RCDELT | 500 | .5280E09 | 1.0 | .9444E09 | 0.1 | 147 | 47.2 | 1.7- 1.8 | 0 |

vectorize - loop list ------- RCDELT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 2111 | do | S | 500 | .3663E09 | 69.4 | .3663E09 | 38.8 | 102541 | 0.0 | 1.0- 1.0 |
| 2131 | do | V | 500 | .1280E09 | 24.2 | .5024E09 | 53.2 | 4424 | 76.2 | 3.9- 4.0 |
| 2121 | do | V | 369783 | 20416772 | 3.9 | 53083608 | 5.6 | 6 | 100.0 | 1.3- 3.9 |
| 2161 | do | | 500 | 10829159 | 2.1 | 10829159 | 1.1 | 28 | 0.0 | 1.0- 1.0 |
| 2141 | do | V | 500 | 851225 | 0.2 | 8241970 | 0.9 | 44 | 95.8 | 8.1- 11.0 |
| 2171 | do | | 141 | 519108 | 0.1 | 519108 | 0.1 | 11 | 0.0 | 1.0- 1.0 |

* TEST-CASE 3 * -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| RCDELT | 500 | .1132E10 | 1.5 | .2580E10 | 0.3 | 26 | 62.9 | 2.2- 2.3 | 0 |

vectorize - loop list ------- RCDELT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 2131 | do | V | 500 | .4553E09 | 40.2 | .1770E10 | 68.6 | 15525 | 75.9 | 3.8- 3.9 |
| 2111 | do | S | 500 | .4116E09 | 36.4 | .4116E09 | 16.0 | 102541 | 0.0 | 1.0- 1.0 |
| 2121 | do | V | 2634184 | .1433E09 | 12.7 | .1863E09 | 7.2 | 3 | 100.0 | 1.0- 1.6 |
| 2161 | do | | 500 | .1069E09 | 9.4 | .1069E09 | 4.1 | 276 | 0.0 | 1.0- 1.0 |
| 2341 | do | S | 500 | 4100205 | 0.4 | 4100205 | 0.2 | 139 | 0.0 | 1.0- 1.0 |
| 2141 | do | V | 500 | 3730964 | 0.3 | 55897338 | 2.2 | 298 | 96.0 | 13.5- 15.9 |

* TEST-CASE 4 * -Oe

vectorize - routine list

| routine | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect | overhd |
|---|---|---|---|---|---|---|---|---|---|
| RCDELT | 500 | .2108E10 | 2.2 | .5430E10 | 0.5 | 18 | 65.1 | 2.4- 2.6 | 0 |

vectorize - loop list ------- RCDELT

| do-id | kind | v | ex-count | v-cost | % | s-cost | % | v-leng | v-rate | v-effect |
|---|---|---|---|---|---|---|---|---|---|---|
| 2131 | do | V | 500 | .9194E09 | 43.6 | .3576E10 | 65.8 | 31378 | 75.9 | 3.8- 3.9 |
| 2161 | do | | 500 | .5296E09 | 25.1 | .5296E09 | 9.8 | 1366 | 0.0 | 1.0- 1.0 |
| 2111 | do | S | 500 | .4587E09 | 21.8 | .4587E09 | 8.4 | 102541 | 0.0 | 1.0- 1.0 |
| 2121 | do | V | 4990136 | .1448E09 | 6.9 | .3765E09 | 6.9 | 3 | 100.0 | 1.3- 3.9 |
| 2341 | do | S | 500 | 18496854 | 0.9 | 18496854 | 0.3 | 627 | 0.0 | 1.0- 1.0 |
| 2141 | do | V | 500 | 16565628 | 0.8 | .2724E09 | 5.0 | 1452 | 96.0 | 15.6- 17.3 |

Fig. 4.12 Dynamic behavior of RCDELT routine in RBUU original version

（変更前）

```
s          DO 1331 I = 1, NX*NX*NZ
s            IF( NGP(I) .GT. 0) THEN
v              DO 1341 J = IGP(I)+1,IGP(I)+NGP(I)
v                RHSVEI(I) = RHSVEI(I) + FAC * XMEFF(J) / P(0,J)
v     1341       CONTINUE
s            END IF
s     1331 CONTINUE
```

⇩

（変更後）

```
             JX = 0
v            DO 1330 I = 1, NX*NX*NZ
v              IF( NGP(I) .GT. 0) THEN
v              JX = JX + 1
v              NV(JX) = I
v              NG1(JX) = IGP(I) + 1
v              NG2(JX) = IGP(I) + NGP(I)
v              END IF
v     1330 CONTINUE
s            DO 1340 I = 1, JX
v              DO 1341 J = NG1(I), NG2(I)
v              RHSVEI(NV(I)) = RHSVEI(NV(I)) + FAC * XMEFF(J) / P(0,J)
v     1341 CONTINUE
s     1340 CONTINUE
```

Fig. 4.13　Modification of DENS routine　(1-st step)

```
             JX = 0
v            DO 1330 I = 1, NX*NX*NZ
v              IF( NGP(I) .GT. 0) THEN
v              JX = JX + 1
v              NV(JX) = I
v              NG1(JX) = IGP(I) + 1
v              NG2(JX) = IGP(I) + NGP(I)
v              END IF
v     1330 CONTINUE
             ngmin = NG1(1)
             ngmax = NG2(1)
v            do 10 I = 2,JX
v              ngmin = MIN(ngmin,NG1(I))
v              ngmax = MAX(ngmax,NG2(I))
v       10 continue
v              do 20 I = ngmin,ngmax
v                pw0(I) = FAC * XMEFF(I) / P(0,I)
v       20 continue
s            DO 1340 I = 1, JX
     *VOCL LOOP,SCALAR
 2             DO 1341 J = NG1(I), NG2(I)
 2             RHSVEI(NV(I)) = RHSVEI(NV(I)) + pw0(J)
 2    1341 CONTINUE
s     1340 CONTINUE
```

Fig. 4.14　Modification of DENS routine　(2-nd step)

(DENSルーチン)

```
v           do 20 I = ngmin,ngmax
v             pw1(I) = FAC * XMEFF(I) / P(0,I)
v      20 continue
v           do 1340 I = 1, NX0
v             RHSVEI(NV0(I)) = pw1(NG0(I))
v    1340   continue
s           DO 1341 I = 1, NX1
    *VOCL LOOP,SCALAR
  2           DO 1342 J = NG1(I), NG2(I)
  2           RHSVEI(NV1(I)) = RHSVEI(NV1(I)) + pw1(J)
  2  1342 CONTINUE
s    1341 CONTINUE
```

(XSORT ルーチンへの追加処理)

```
          . . .

          NX0 = 0
          NX1 = 0
v         do 2965 I = 1, NX*NX*NZ
v           if(NGP(I) .gt. 0) then
v             if(NGP(I) .eq. 1) then
v               NX0 = NX0 + 1
v               NV0(NX0) = I
v               NG0(NX0) = IGP(I) + 1
v             else
v               NX1 = NX1 + 1
v               NV1(NX1) = I
v               NG1(NX1) = IGP(I) + 1
v               NG2(NX1) = IGP(I) + NGP(I)
v             end if
v           end if
v    2965 continue
          ngmin = MIN(NG0(1),NG1(1))
          ngmax = NG0(NX0)
v         do 2966 I = 1, NX1
v           ngmax = MAX(ngmax,NG2(I))
v    2966 continue
```

Fig. 4.15 Modification of DENS routine (3-rd step)

(変更前)

```
s           DO 1371 I = 1, NX*NX*NZ
s             IF( NGP(I) .GT. 0) THEN
v               DO 1381 J=IGP(I)+1, IGP(I)+NGP(I)
v                  RHVVEI(I,0) = RHVVEI(I,0) + FAC
v                  RHVVEI(I,1) = RHVVEI(I,1) + FAC * P(1,J) / P(0,J)
v                  RHVVEI(I,2) = RHVVEI(I,2) + FAC * P(2,J) / P(0,J)
v                  RHVVEI(I,3) = RHVVEI(I,3) + FAC * P(3,J) / P(0,J)
v    1381 CONTINUE
s             END IF
s    1371 CONTINUE
```

⇩

(変更後)

```
v           do 1380 I = ngmin,ngmax
v             pw1(I) = FAC * P(1,I) / P(0,I)
v             pw2(I) = FAC * P(2,I) / P(0,I)
v             pw3(I) = FAC * P(3,I) / P(0,I)
v    1380 continue
v           do 1381 I = 1, NX0
v              RHVVEI(NV0(I),0) =  FAC
v              RHVVEI(NV0(I),1) =  pw1(NG0(I))
v              RHVVEI(NV0(I),2) =  pw2(NG0(I))
v              RHVVEI(NV0(I),3) =  pw3(NG0(I))
v    1381 CONTINUE
s           DO 1382 I = 1, NX1
    *VOCL LOOP,SCALAR
  2           DO 1383 J = NG1(I), NG2(I)
  2             RHVVEI(NV1(I),0) = RHVVEI(NV1(I),0) + FAC
  2             RHVVEI(NV1(I),1) = RHVVEI(NV1(I),1) + pw1(J)
  2             RHVVEI(NV1(I),2) = RHVVEI(NV1(I),2) + pw2(J)
  2             RHVVEI(NV1(I),3) = RHVVEI(NV1(I),3) + pw3(J)
  2  1383 CONTINUE
s    1382 CONTINUE
```

Fig. 4.16 Modification of DO 1371･1381 in DENS routine

(変更前)

```
s          DO 2411 IT = 1, ITASK
s            DO 2421 IZ = MINIZ1(IT), MAXIZ1(IT)
s              DO 2431 I = 0, IDIMS
v
:                    . . .
v
s     2431 CONTINUE
s     2421 CONTINUE
s     2411 CONTINUE
s          DO 2461 IT=1,ITASK
s            DO 2471 IZ = MINIZ(IT),MAXIZ(IT)
s              DO 2481 I = 0, IDIMS
v                 DO 2491 IY = -MAXX+1, MAXX-1
v                   RANDX( IY, 0, IT) = VSRC( 1  + (MAXX+IY) * NX, IZ,I)
v                   RANDX( IY, 1, IT) = VSRC( NX + (MAXX+IY) * NX, IZ,I)
v      2491 CONTINUE

                           . . .

s 3              DO 2511 IY = -MAXX+1, MAXX-1
s 3                 VTAR( 1 + (MAXX+IY) * NX, IZ,I) = RANDX( IY, 0, IT )
s 3                 VTAR( NX + (MAXX+IY) * NX, IZ,I) = RANDX( IY, 1, IT )
s 3   2511 CONTINUE
s     2481 CONTINUE
s     2471 CONTINUE
s     2461 CONTINUE
```

⇩

(変更後)

```
      C     DO 2411 IT = 1, ITASK
s             DO 2421 IZ = MINIZ1(IT), MAXIZ1(IT)
s               DO 2431 I = 0, IDIMS
v
:                    . . .
v
s      2431 CONTINUE
s      2421 CONTINUE
      C2411 CONTINUE
      C     DO 2461 IT=1,ITASK
s             DO 2471 IZ = MINIZ(IT),MAXIZ(IT)
s               DO 2481 I = 0, IDIMS
      C            DO 2491 IY = -MAXX+1, MAXX-1
      C              RANDX( IY, 0, IT) = VSRC( 1  + (MAXX+IY) * NX, IZ,I)
      C              RANDX( IY, 1, IT) = VSRC( NX + (MAXX+IY) * NX, IZ,I)
      C 2491 CONTINUE

                           . . .

      *VOCL LOOP,NOVREC(VTAR)
v                DO 2511 IY = -MAXX+1, MAXX-1
v                   VTAR( 1 + (MAXX+IY) * NX, IZ,I)
          +                           = VSRC( 1  + (MAXX+IY) * NX, IZ,I)
v                   VTAR( NX + (MAXX+IY) * NX, IZ,I)
          +                           = VSRC( NX + (MAXX+IY) * NX, IZ,I)
v      2511 CONTINUE
s      2481 CONTINUE
s      2471 CONTINUE
      C2461 CONTINUE
```

Fig. 4.17 Modification of SMEAR routine

(変更前)

```
S         DO 1761 IZ = -MAXZ, MAXZ
v           DO 1771 I = 1, NX * NX
v              IF( SIGMAT(I,IZ,T1(0)) .LE. BOUND ) SITMAT(I,IZ,1) = 0.0
v              IF( SIGMAT(I,IZ,T1(0)) .GT. BOUND )
v         >    SITMAT(I,IZ,1) =  SIGMAT(I,IZ,T1(0)) - BOUND
v              SITMAT(I,IZ,2) =  SIGMAT(I,IZ,T1(0)) + BOUND
v     1771 CONTINUE
s     1761 CONTINUE


v           DO 1831 I = 1, NX * NX * NZ
v             TDIFF = FT1VEC(I,IREL2) - FT1VEC(I,IREL1)
v             ADIFF = ABS( TDIFF )
v              IF( ADIFF .GT. EPS )
          +      SITVEC(I,IREL3) = SITVEC(I,IREL2) -
          +         ( SITVEC(I,IREL2) - SITVEC(I,IREL1)   ) / TDIFF
          +        * FT1VEC(I,IREL2)
v              IF( ADIFF .LE. EPS ) SITVEC(I,IREL3) = SITVEC(I,IREL2)
v     1831 CONTINUE
```

⇩

(変更後)

```
s         DO 1761 IZ = -MAXZ, MAXZ
v           DO 1771 I = 1, NX * NX
v             IF( SIGMAT(I,IZ,T1(0)) .LE. BOUND ) THEN
v                SITMAT(I,IZ,1) = 0.0
v             ELSE
v                SITMAT(I,IZ,1) =  SIGMAT(I,IZ,T1(0)) - BOUND
v             END IF
v             SITMAT(I,IZ,2) =  SIGMAT(I,IZ,T1(0)) + BOUND
v     1771 CONTINUE
s     1761 CONTINUE

v           DO 1831 I = 1, NX * NX * NZ
v             TDIFF = FT1VEC(I,IREL2) - FT1VEC(I,IREL1)
v             ADIFF = ABS( TDIFF )
v             IF( ADIFF .GT. EPS ) THEN
v               SITVEC(I,IREL3) = SITVEC(I,IREL2) -
        +          ( SITVEC(I,IREL2) - SITVEC(I,IREL1)   ) / TDIFF
        +         * FT1VEC(I,IREL2)
v             ELSE
v               SITVEC(I,IREL3) = SITVEC(I,IREL2)
v             END IF
v     1831 CONTINUE
```

Fig. 4.18 Modification of ITERAT routine

(変 更 前)

```
                          .  .  .
M         DO 2791 I = 1, NTOTAL
V           IIX(I) = INT( R(1,I) * XDX + 50.5 ) - 50
V           IIY(I) = INT( R(2,I) * XDX + 50.5 ) - 50
V           IIZ(I) = INT( R(3,I) * XDZ + 50.5 ) - 50
M           IF( IIX(I) .GT. MAXX2 .OR. IIX(I) .LT. -MAXX2) WARNING = .TRUE.
M           IF( IIY(I) .GT. MAXX2 .OR. IIY(I) .LT. -MAXX2) WARNING = .TRUE.
M           IF( IIZ(I) .GT. MAXZ2 .OR. IIZ(I) .LT. -MAXZ2) WARNING = .TRUE.
V           IV(I) = 1 + ( MAXX+IIX(I) ) + NX * ( MAXX+IIY(I) )
          +           + NX*NX * ( MAXZ+IIZ(I) )
V     2791 CONTINUE
                          .  .  .
          IF( WARNING) THEN
M           DO 2811 I = 1, NTOTAL
M           IF( IIX(I) .LE. MAXX2 .AND. IIX(I) .GE. -MAXX2 .AND.
          +       IIY(I) .LE. MAXX2 .AND. IIY(I) .GE. -MAXX2 .AND.
          +       IIZ(I) .LE. MAXZ2 .AND. IIZ(I) .GE. -MAXZ2)
          +     IGP(IV(I)) = IGP(IV(I)) + 1
V     2811 CONTINUE
          ELSE
S8          DO 2821 I = 1, NTOTAL
M8            IGP(IV(I)) = IGP(IV(I)) + 1
V8    2821 CONTINUE
          END IF
                          .  .  .
          IF( WARNING ) THEN
M           DO 2851 I = 1, NTOTAL
V             IF( IIX(I) .LE. MAXX2 .AND. IIX(I) .GE. -MAXX2 .AND.
          +         IIY(I) .LE. MAXX2 .AND. IIY(I) .GE. -MAXX2 .AND.
          +         IIZ(I) .LE. MAXZ2 .AND. IIZ(I) .GE. -MAXZ2) THEN
M               ICOPY(I) = IGP(IV(I))
M               IGP(IV(I)) = IGP(IV(I)) - 1
V             ELSE
V               ICOPY(I) = NLAST
V               NLAST = NLAST - 1
V             END IF
V     2851 CONTINUE
          ELSE
S8          DO 2861 I = 1, NTOTAL
M8            ICOPY(I) = IGP(IV(I))
S8            IGP(IV(I)) = IGP(IV(I)) - 1
V8    2861 CONTINUE
          END IF
                          .  .  .
```

(変 更 後)

```
                          .  .  .
          JX = 0
V         DO 2791 I = 1, NTOTAL
V           IIX(I) = INT( R(1,I) * XDX + 50.5 ) - 50
V           IIY(I) = INT( R(2,I) * XDX + 50.5 ) - 50
V           IIZ(I) = INT( R(3,I) * XDZ + 50.5 ) - 50
V           IF( IIX(I) .GT. MAXX2 .OR. IIX(I) .LT. -MAXX2  .OR.
          |       IIY(I) .GT. MAXX2 .OR. IIY(I) .LT. -MAXX2  .OR.
          +       IIZ(I) .GT. MAXZ2 .OR. IIZ(I) .LT. -MAXZ2) then
V             JX = JX + 1
V             IX(JX) = I
V             IV(I) = 0
V           else
V             IV(I) = 1 + ( MAXX+IIX(I) ) + NX * ( MAXX+IIY(I) )
          +           + NX*NX * ( MAXZ+IIZ(I) )
V           end if
V     2791 CONTINUE
                          .  .  .
S 2         DO 2811 I = 1, NTOTA
M 2           IGP(IV(I)) = IGP(IV(I)) + 1
V 2   2811 CONTINUE
                          .  .  .
S 2         DO 2851 I = 1, NTOTA
M 2           ICOPY(I) = IGP(IV(I))
S 2           IGP(IV(I)) = IGP(IV(I)) - 1
V 2   2851 CONTINUE
V           DO 2861 J = 1, JX
V             ICOPY(IX(J)) = NLAST
V             NLAST = NLAST - 1
V     2861 CONTINUE
                          .  .  .
```

Fig. 4.19 Modification of XSORT routine

（変 更 前）

```
s         DO 1141 LG = 1, NX * NX * NZ
            DO 1151 LA = IGPA(LG) + 1, IGPA(LG) + NGPA(LG)
v             DO 1161 I = 0,3
v               PAP(I) = PA(I,LA)
v   1161      CONTINUE
              EMAP = XMEFFA(LA)
              SUM = 0.0
              VECDEN = 0.0
s             DO 1171 LB =  IGP(LG) + 1, IGP(LG) + NGP(LG)
v               DO 1181 I = 0,3
v                 PP(I) = P(I,LB)
v   1181        CONTINUE
s               EMPP = XMEFF(LB)
s               TOTE = PP(0) + PAP(0)
s               BETAX = (PP(1) + PAP(1)) / TOTE
s               BETAY = (PP(2) + PAP(2)) / TOTE
s               BETAZ = (PP(3) + PAP(3)) / TOTE
s               GAMMA = 1.0 / SQRT( 1.0 - BETAX**2 - BETAY**2 - BETAZ**2)
s               TRAFO = GAMMA * ( GAMMA *
     +                  (PP(1)*BETAX + PP(2)*BETAY + PP(3)*BETAZ)
     +                / ( GAMMA + 1.0 ) - PP(0)  )
s               PXCM = BETAX * TRAFO + PP(1)
s               PYCM = BETAY * TRAFO + PP(2)
s               PZCM = BETAZ * TRAFO + PP(3)
s               PRCM = SQRT ( PXCM**2 + PYCM**2 + PZCM**2 )
s               EP   = SQRT ( EMPP**2 + PRCM**2)
s               EA   = SQRT ( EMAP**2 + PRCM**2)
s               SRT = EA + EP
s               S   = SRT * SRT
s               PNEW = SIGA(S,EMPP, EMAP)
v               INDEX = NINT(PNEW)
v               NUSIG(INDEX) = NUSIG(INDEX) + 1
v               FLUS =  PRCM/ EP+ PRCM/EA
v               PNEW = -PNEW * FLUS /10.0
v               SUM = SUM + PNEW
v               VECDEN = VECDEN + FAC
v   1171      CONTINUE
                  . . .
```

（変 更 後）

```
              JX = 0
v             do 10 LG = 1, NX * NX * NZ
v               if ( NGPA(LG) .GT. 0 ) then
v                 JX = JX + 1
v                 IX(JX) = LG
v                 IG1(JX) = IGPA(LG) + 1
v                 IG2(JX) = IGPA(LG) + NGPA(LG)
v               end if
v          10 continue
              do 1141 LG = 1,JX
                do 1151 LA = IG1(LG), IG2(LG)
                  PAP0 = PA(0,LA)
                  PAP1 = PA(1,LA)
                  PAP2 = PA(2,LA)
                  PAP3 = PA(3,LA)
                  EMAP = XMEFFA(LA)
                  SUM = 0.0
                  VECDEN = 0.0
                  IB = 0
v                 do 1171 LB = IGP(IX(LG)) + 1, IGP(IX(LG)) + NGP(IX(LG))
v                   IB = IB + 1
v                   PP(0,IB) = P(0,LB)
v                   PP(1,IB) = P(1,LB)
v                   PP(2,IB) = P(2,LB)
v                   PP(3,IB) = P(3,LB)
v                   EMPPa(IB) = XMEFF(LB)
v                   TOTEa(IB) = PP(0,IB) + PAP0
v                   BETAXa(IB) = (PP(1,IB) + PAP1) / TOTEa(IB)
v                   BETAYa(IB) = (PP(2,IB) + PAP2) / TOTEa(IB)
v                   BETAZa(IB) = (PP(3,IB) + PAP3) / TOTEa(IB)
v                   GAMMAa(IB) = 1.0 / SQRT( 1.0 - BETAXa(IB)**2
     +                                         -BETAYa(IB)**2 - BETAZa(IB)**2)
v                   TRAFOa(IB) = GAMMAa(IB) * ( GAMMAa(IB) *
     +                (PP(1,IB)*BETAXa(IB) + PP(2,IB)*BETAYa(IB)
     +                  + PP(3,IB)*BETAZa(IB)) / (GAMMAa(IB) + 1.0 ) - PP(0,IB))
v                   PXCMa(IB) = BETAXa(IB) * TRAFOa(IB) + PP(1,IB)
v                   PYCMa(IB) = BETAYa(IB) * TRAFOa(IB) + PP(2,IB)
v                   PZCMa(IB) = BETAZa(IB) * TRAFOa(IB) + PP(3,IB)
v                   PRCMa(IB) = SQRT(PXCMa(IB)**2+PYCMa(IB)**2+PZCMa(IB)**2)
v                   EPa(IB)   = SQRT ( EMPPa(IB)**2 + PRCMa(IB)**2)
v                   EAa(IB)   = SQRT ( EMAP**2 + PRCMa(IB)**2)
v                   SRTa(IB) = EAa(IB) + EPa(IB)
v                   Sa(IB)   = SRTa(IB) * SRTa(IB)
v                   FLUSa(IB) =  PRCMa(IB)/ EPa(IB)+ PRCMa(IB)/EAa(IB)
v     1171        CONTINUE
s                 do 1172 LB = 1, IB
s                   PNEWa(LB) = SIGA(Sa(LB),EMPPa(LB), EMAP)
s     1172        CONTINUE
v                 do 1173 LB = 1, IB
v                   INDEXa(LB) = NINT(PNEWa(LB))
v     1173        CONTINUE
     *VOCL LOOP,SCALAR
  2               do 1174 LB = 1, IB
  2                 NUSIG(INDEXa(LB)) = NUSIG(INDEXa(LB)) + 1
  2   1174        CONTINUE
v                 do 1175 LB = 1, IB
v                   PNEWa(LB) = -PNEWa(LB) * FLUSa(LB) /10.0
v                   SUM = SUM + PNEWa(LB)
v                   VECDEN = VECDEN + FAC
v     1175        CONTINUE
                      . . .
```

Fig. 4.20 Modification of ABSOR routine

（変更前）

```
s          DO 2111 I = 1, NX*NX*NZ
s            IF( NGP(I) .GT. 1 ) THEN
s              CALL PERMUT( NGP(I), ICF, SBBD)
v              DO 2121 J = 1, NGP(I) / 2
v                ICNT = ICNT + 1
v                IPART1(ICNT)  = IGP(I) + ICF( 2*J - 1)
v                IPART2(ICNT)  = IGP(I) + ICF( 2*J )
v                LASTCOL(ICNT) = NGP(I)
v     2121       CONTINUE
s            END IF
s     2111 CONTINUE
```

⇩

（変更後）

```
v          jx = 0
v          do 2110 i = 1,NX*NX*NZ
v            if(NGP(i) .gt. 1) then
v              jx = jx + 1
v              ix(jx) = i
v            end if
v     2110 continue

s          do 2111 i = 1, jx
s            call PERMUT( NGP(ix(i)), ICF, SBBD)
     *vocl loop,scalar
  2          do 2121 J = 1, NGP(ix(i)) / 2
  2            ICNT = ICNT + 1
  2            IPART1(ICNT)  = IGP(ix(i)) + ICF( 2*J - 1)
  2            IPART2(ICNT)  = IGP(ix(i)) + ICF( 2*J )
  2            LASTCOL(ICNT) = NGP(ix(i))
  2   2121 CONTINUE
s     2111 CONTINUE
```

Fig. 4.21 Modification of DO 2111･2121 in RCDBLT routine

<テスト・ケース１>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 15116 | 41.2 | 1594 | smear |
| 5929 | 16.2 | 1579 | iterat |
| 5081 | 13.9 | 1695 | xsort |
| 4108 | 11.2 | 2016 | dens |
| 1227 | 3.3 | 556 | tpcorr |
| 1174 | 3.2 | 550 | tppred |
| 560 | 1.5 | － | resel |
| 536 | 1.5 | － | suraf |
| 497 | 1.4 | 301 | rcdelt |
| 471 | 1.3 | 2036 | tpsort |
| 461 | 1.3 | 2011 | xsorta |
| 410 | 1.1 | － | permut |
| 180 | 0.5 | － | matri2 |
| 159 | 0.4 | － | cndns |
| 125 | 0.3 | 2048 | init |
| 105 | 0.3 | 1682 | absor |
| 105 | 0.3 | 2032 | MAIN |
| 89 | 0.2 | － | cross |
| 86 | 0.2 | 464 | tpblck |
| 63 | 0.2 | － | rnset |
| 60 | 0.2 | － | ranf |
| 25 | 0.1 | － | pboo |
| 21 | 0.1 | － | ran |
| 19 | 0.1 | － | cptime |
| 18 | 0.0 | － | pdens |
| 13 | 0.0 | － | __start |
| 13 | 0.0 | 41 | etot |
| 6 | 0.0 | 376 | summry |
| 3 | 0.0 | － | absout |
| 2 | 0.0 | － | anoutm |
| 1 | 0.0 | － | tpcorra |
| 36663 | | 1492 | TOTAL |

<テスト・ケース２>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 8269 | 44.1 | 1594 | smear |
| 3050 | 16.3 | 1665 | iterat |
| 2391 | 12.7 | 1603 | xsort |
| 1829 | 9.7 | 2020 | dens |
| 560 | 3.0 | － | resel |
| 386 | 2.1 | 557 | tpcorr |
| 363 | 1.9 | 555 | tppred |
| 277 | 1.5 | 2048 | tpsort |
| 240 | 1.3 | 1916 | xsorta |
| 195 | 1.0 | － | suraf |
| 189 | 1.0 | 1085 | absor |
| 162 | 0.9 | 251 | rcdelt |
| 160 | 0.9 | － | permut |
| 114 | 0.6 | 45 | r4 |
| 109 | 0.6 | － | matri2 |
| 102 | 0.5 | 41 | init |
| 86 | 0.5 | － | siga |
| 57 | 0.3 | － | ranf |
| 41 | 0.2 | 2017 | MAIN |
| 37 | 0.2 | － | cndns |
| 34 | 0.2 | － | rnset |
| 16 | 0.1 | 524 | tpblck |
| 13 | 0.1 | － | ran |
| 12 | 0.1 | － | cross |
| 11 | 0.1 | － | __start |
| 11 | 0.1 | － | cptime |
| 11 | 0.1 | 294 | tppreda |
| 8 | 0.0 | 41 | etot |
| 7 | 0.0 | 284 | tpcorra |
| 6 | 0.0 | － | pboo |
| 3 | 0.0 | － | pdens |
| 3 | 0.0 | － | r3 |
| 3 | 0.0 | － | absout |
| 1 | 0.0 | 41 | summry |
| 1 | 0.0 | － | rcout |
| 1 | 0.0 | － | mogri |
| 1 | 0.0 | － | anoutm |
| 18759 | | 1537 | TOTAL |

<テスト・ケース３>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 9306 | 23.3 | 2031 | dens |
| 8641 | 21.6 | 1594 | smear |
| 5624 | 14.1 | 1346 | iterat |
| 5273 | 13.2 | 1875 | xsort |
| 1385 | 3.5 | 612 | tpcorr |
| 1333 | 3.3 | 573 | tppred |
| 993 | 2.5 | 2044 | tpsort |
| 950 | 2.4 | 289 | absor |
| 822 | 2.1 | － | suraf |
| 813 | 2.0 | － | siga |
| 809 | 2.0 | 486 | rcdelt |
| 746 | 1.9 | 61 | r4 |
| 741 | 1.9 | － | permut |
| 560 | 1.4 | － | resel |
| 319 | 0.8 | 468 | init |
| 282 | 0.7 | 1906 | xsorta |
| 240 | 0.6 | － | cross |
| 201 | 0.5 | － | ranf |
| 159 | 0.4 | 788 | tpblck |
| 138 | 0.3 | － | rnset |
| 137 | 0.3 | 2012 | MAIN |
| 122 | 0.3 | － | cndns |
| 109 | 0.3 | － | matri2 |
| 77 | 0.2 | － | ran |
| 65 | 0.2 | － | pboo |
| 30 | 0.1 | 589 | tpcorra |
| 20 | 0.1 | 521 | tppreda |
| 16 | 0.0 | 512 | pdens |
| 12 | 0.0 | － | cptime |
| 9 | 0.0 | － | __start |
| 9 | 0.0 | 41 | etot |
| 7 | 0.0 | 2048 | anoutm |
| 3 | 0.0 | 41 | summry |
| 3 | 0.0 | － | absout |
| 3 | 0.0 | － | antiou |
| 2 | 0.0 | 11 | zwist |
| 1 | 0.0 | － | sigtot |
| 1 | 0.0 | － | mogri |
| 1 | 0.0 | － | main |
| 39962 | | 1380 | TOTAL |

<テスト・ケース４>
Synthesis Information

| Count | Percent | VL | Name |
|---|---|---|---|
| 16780 | 25.9 | 2042 | dens |
| 9203 | 14.2 | 1595 | smear |
| 9040 | 13.9 | 1151 | iterat |
| 8321 | 12.8 | 1919 | xsort |
| 2679 | 4.1 | 603 | tpcorr |
| 2590 | 4.0 | 569 | tppred |
| 2113 | 3.3 | 458 | rcdelt |
| 1697 | 2.6 | 2039 | tpsort |
| 1624 | 2.5 | 88 | r4 |
| 1622 | 2.5 | － | suraf |
| 1461 | 2.3 | － | permut |
| 1433 | 2.2 | － | cross |
| 1393 | 2.1 | 294 | absor |
| 906 | 1.4 | － | siga |
| 560 | 0.9 | － | resel |
| 501 | 0.8 | － | ranf |
| 444 | 0.7 | 781 | init |
| 436 | 0.7 | 254 | tpblck |
| 393 | 0.6 | 2048 | cndns |
| 312 | 0.5 | － | rnset |
| 309 | 0.5 | － | pboo |
| 282 | 0.4 | 1860 | xsorta |
| 233 | 0.4 | － | ran |
| 232 | 0.4 | 2040 | MAIN |
| 110 | 0.2 | － | matri2 |
| 36 | 0.1 | 501 | tppreda |
| 28 | 0.0 | 512 | pdens |
| 25 | 0.0 | 508 | tpcorra |
| 10 | 0.0 | 41 | etot |
| 8 | 0.0 | － | cptime |
| 7 | 0.0 | － | __start |
| 7 | 0.0 | 1769 | anoutm |
| 6 | 0.0 | 2026 | r3 |
| 4 | 0.0 | － | antiou |
| 3 | 0.0 | 2048 | summry |
| 3 | 0.0 | － | absout |
| 3 | 0.0 | － | main |
| 2 | 0.0 | － | sigtot |
| 1 | 0.0 | － | rcout |
| 1 | 0.0 | － | apboo |
| 1 | 0.0 | － | zwist |
| 64819 | | 1274 | TOTAL |

Fig. 4.2.2 Computational cost distributions in RBUU vectorized version

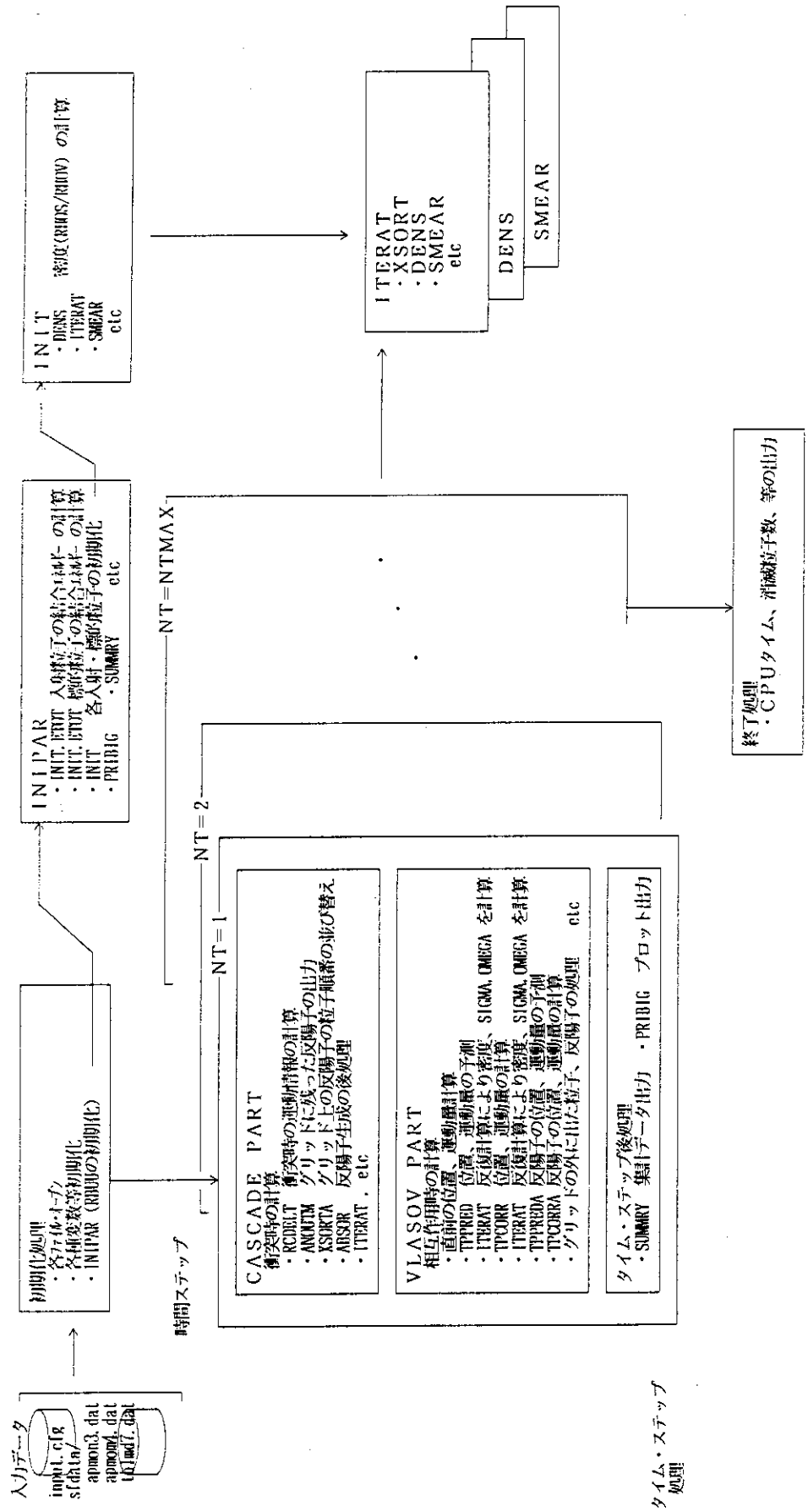


Fig. 4.2.3 Configuration of RDUU code

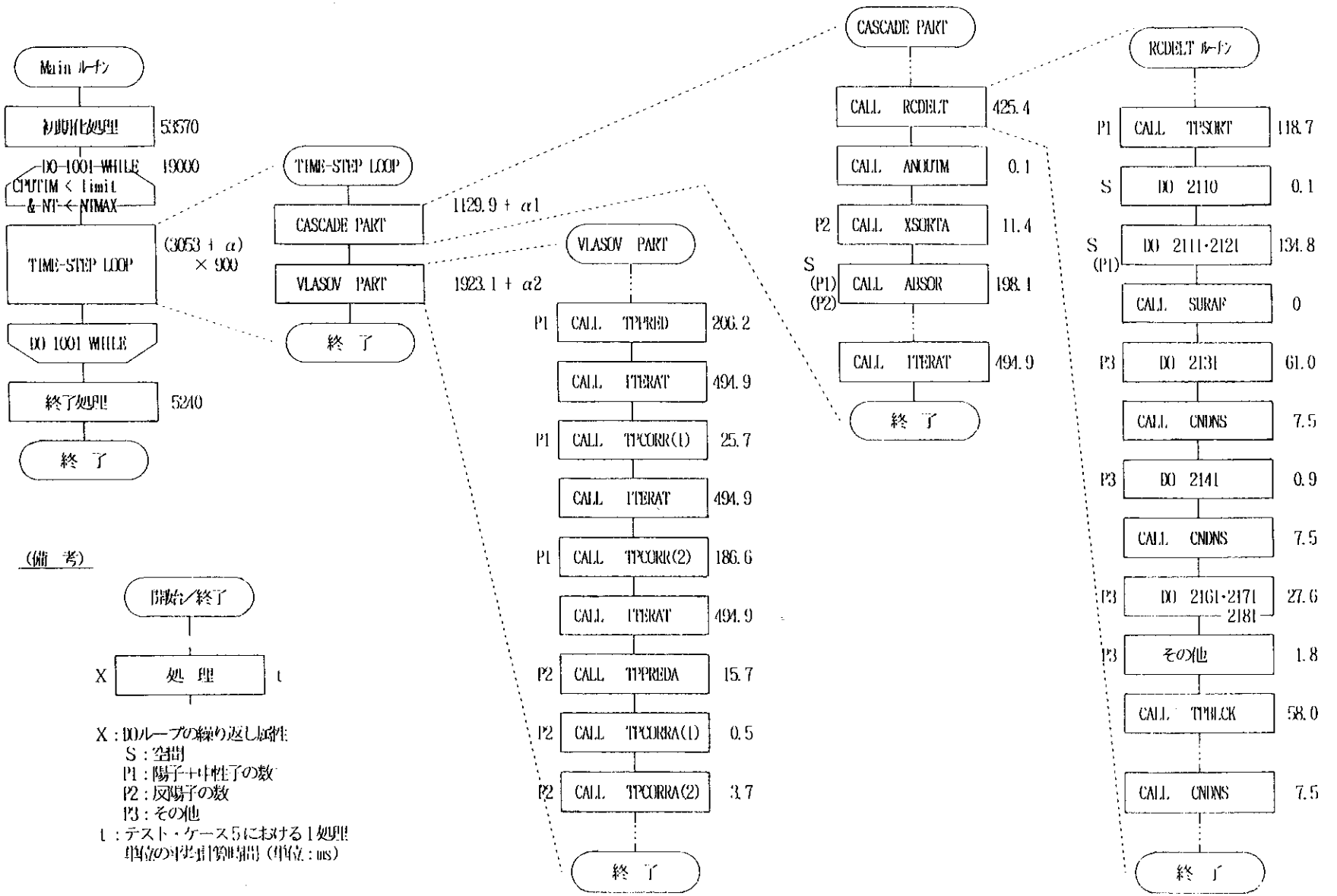


Fig. 4.2.4 Process flow in RIUU code (1／2)

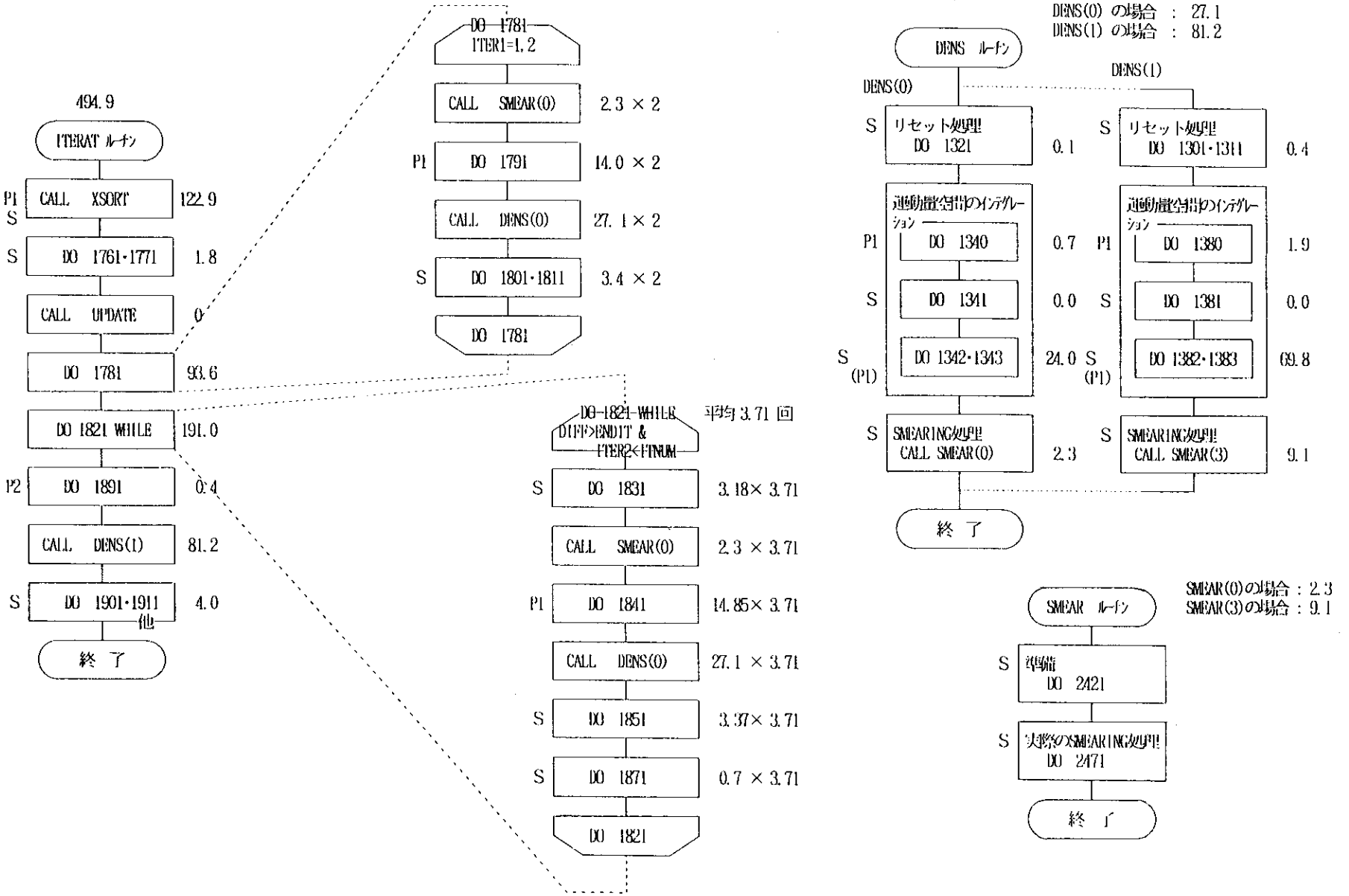


Fig. 4.2.4 Process flow in RIBUU code (2/2)

# 5. おわりに

平成６年度末に，VPP500 が導入されて，丸２年が経過した．この間，利用者数も順調に増加し，実行される原子力コードも多数に及んでいる．計算科学技術推進センター情報システム管理課では，ユーザからの依頼に応じて，これらの原子力コードを VPP500 向けに最適なベクトル化，並列化を施すチューニングを行ってきている．この作業は，大規模シミュレーション・ジョブの計算時間を短縮するだけでなく，単一プロセッサ上ではメモリ不足から実行できないジョブを並列化により可能にするなど，計算機のスループットの向上，ユーザの仕事時間の短縮のみならず，計算可能なジョブの範囲を拡大するなど，非常に意義のある仕事と考えている．

本報告書で記述した原子力コードのベクトル化については，その効果が顕著に表れているものが多いが，一方では VPP500 のベクトル並列計算機としての能力を十分に引き出せない原子力コードも数多く存在することが分かっている．このようなコードについては，平成９年度から導入された，１プロセッサの処理能力の高いスカラ並列型の AP3000 計算機システムへの移行も現在進めている．原子力コードの性格に応じて，計算機を使い分けていくことが，計算機資源の効率的利用，ユーザの仕事の効率化の観点から重要であると考えている．本報告書が，これらの仕事に関係する人々に多少とも参考になれば幸いである．

# 謝　辞

本作業を行う上で，作業を依頼された原子炉工学部高エネルギー中性子研究室の今野力氏 (2 章)，元高温工学部熱利用システム研究室の菱田誠氏 (3 章)，先端基礎研究センターハドロン輸送研究グループの丸山智幸氏 (4 章) には，コード内容の把握について御協力頂きました．また，VPP500 システムのハードウェア，ソフトウェアに関する相談については (株) 富士通 R&D システム部石附茂氏に，本作業を円滑に遂行するための各種事務処理については山田圭子氏に御協力を頂きました．ここにこれらの方々に感謝の意を表します．

最後に，本報告書を執筆する機会を与えて下さいました計算科学技術推進センター長浅井清氏，前次長秋元正幸氏 (現原子炉安全工学部部長)，(株) 富士通 R&D システム部前部長橋本道夫氏及び現部長平沢健一氏に感謝致します．

## 5. おわりに

平成６年度末に，VPP500 が導入されて，丸２年が経過した．この間，利用者数も順調に増加し，実行される原子力コードも多数に及んでいる．計算科学技術推進センター情報システム管理課では，ユーザからの依頼に応じて，これらの原子力コードを VPP500 向けに最適なベクトル化，並列化を施すチューニングを行ってきている．この作業は，大規模シミュレーション・ジョブの計算時間を短縮するだけでなく，単一プロセッサ上ではメモリ不足から実行できないジョブを並列化により可能にするなど，計算機のスループットの向上，ユーザの仕事時間の短縮のみならず，計算可能なジョブの範囲を拡大するなど，非常に意義のある仕事と考えている．

本報告書で記述した原子力コードのベクトル化については，その効果が顕著に表れているものが多いが，一方では VPP500 のベクトル並列計算機としての能力を十分に引き出せない原子力コードも数多く存在することが分かっている．このようなコードについては，平成９年度から導入された，１プロセッサの処理能力の高いスカラ並列型の AP3000 計算機システムへの移行も現在進めている．原子力コードの性格に応じて，計算機を使い分けていくことが，計算機資源の効率的利用，ユーザの仕事の効率化の観点から重要であると考えている．本報告書が，これらの仕事に関係する人々に多少とも参考になれば幸いである．

## 謝　辞

本作業を行う上で，作業を依頼された原子炉工学部高エネルギー中性子研究室の今野力氏 (2 章)，元高温工学部熱利用システム研究室の菱田誠氏 (3 章)，先端基礎研究センターハドロン輸送研究グループの丸山智幸氏 (4 章) には，コード内容の把握について御協力頂きました．また，VPP500 システムのハードウェア，ソフトウェアに関する相談については (株) 富士通 R&D システム部石附茂氏に，本作業を円滑に遂行するための各種事務処理については山田圭子氏に御協力を頂きました．ここにこれらの方々に感謝の意を表します．

最後に，本報告書を執筆する機会を与えて下さいました計算科学技術推進センター長浅井清氏，前次長秋元正幸氏 (現原子炉安全工学部部長)，(株) 富士通 R&D システム部前部長橋本道夫氏及び現部長平沢健一氏に感謝致します．

# 参考文献

[1] RSIC COMPUTER CODE COLLECTION TORT-DORT Two- and Three-Dimensional Discrete Ordinates Transport Version 2.8.14 Oak Ridge National Laboratory , March 1994

[2] CRAY T94/4128 newblock 利用者講習会資料日本クレイ株式会社, 1996年3月

[3] 「FACOM OS IV/F4 MSP STREAM77 説明書」, 富士通(株), 1992年5月.

[4] 「UXP/M アナライザ 使用手引書(FORTRAN,VP 用) V10L20 用」, 富士通(株), 1992年2月.

[5] 「UXP/M VPP FORTRAN77 EX/VP 使用手引書 V12用」, 富士通(株), 1994年9月.

[6] 一次冷却系配管破断事故時の多成分気体流動解析コードの整備, (財)原子力データセンター, 1994年3月.

[7] 「UXP/M VPP アナライザ 使用手引書 V10用」, 富士通(株), 1994年1月.

[8] 「原研 VPP500/42 システム利用手引 第2版」, 日本原子力研究所 計算科学技術推進センター 情報システム管理室, 1995年7月.

[9] S.Teis,W.Cassing,T.Maruyama and U.Mosel, Phys. Lett. B319 (1993) 47

[10] A.Lang,W.Cassing,U.Mosel and K.Weber, Nucl. Phys. A541 (1992) 507

[11] B.Blättel,V.Koch,W.Cassing and U.Mosel Phys. rev. C38(1988) 1767