JAERI-M
87-019

HTGR炉心2次元垂直断面モデルによる地震解析コードSONATINA-2Vのベクトル化

1987年2月

鶴岡　卓哉*・牧野　光弘**・幾島　毅・石黒美佐子

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

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

JAERI-M reports are issued irregularly.
Inquiries about availability of the reports should be addressed to Information Division Department of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 319-11, Japan.

編集兼発行　日本原子力研究所
印　　刷　㈱高野高速印刷

ＨＴＧＲ炉心２次元垂直断面モデルによる地震解析コード SONATINA－2Vのベクトル化

日本原子力研究所東海研究所計算センター
鶴岡卓哉[*]・牧野光弘[**]・幾島　毅[+]・石黒美佐子

（１９８７年1月２８日受理）

ブロック状燃料高温ガス炉（HTGR）の炉心2次元垂直断面モデルによる地震解析コードSONATINA－２Ｖをベクトル化した。このコードは，連立常微分方程式で記述される燃料ブロックの挙動をルンゲ・クッタ法により数値的に解いている。この数値解法は，ある時刻において燃料ブロックに働く力を並列に計算することができる。この並列性を利用できるようにプログラム構造を変更し，さらにベクトル計算機の機能を有効に利用できるようにプログラムを書き直した。その結果，FACOM VP－100でのベクトル化コードのベクトル・モード処理時間はオリジナル・コードのスカラ・モード処理時間の約1/6に短縮された。本報告書では，本コードの並列性を利用するためのプログラム構造の変更及びベクトル計算機向き最適化手法について述べる。さらに，ベクトル計算機で高い処理性能を得るためにはプログラム構造の考察が重要であることを指摘する。

＋　燃料安全工学部
＊　外来研究員（財団法人　原子力データ・センター）
＊＊　外来研究員（富士通株式会社）

Vectorization of the SONATINA-2V Code for Seismic Analysis of the Two-dimensional Vertical Slice HTGR Core

Takuya TSURUOKA,* Mitsuhiro MAKINO,** Takeshi IKUSHIMA+ and Misako ISHIGURO

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

(Received January 28, 1987)

The SONATINA-2V code for analyzing the dynamic behavior of fuel blocks in the vertical slice of the HTGR (High Temperature Gas Reactor) core under the seismic perturbation has been vectorized. In this code, the behavior of fuel blocks which is described by a set of ordinary differential equations is numerically solved by the Runge-Kutta Method. In this numerical method, the equations can be calculated in parallel at each time step. In order to utilize this parallelism, the program structure has been changed, and the program has been optimized to effectively utilize the vector instructions of the vector processor. The processing time of the vectorized program on the FACOM VP-100 in the vector mode is reduced to about 1/6 in comparison with that of the original program in the scalar mode. The restructure of the program for representing the parallelism and the technique for the vector optimization are described. It is pointed out that the consideration for the program structure is important to obtain the high performance on the vector computer.

Keywords: SONATINA-2V, HTGR, Vectorization, FACOM VP-100, Seismic, Computer Codes

---

+ Facility Safety Analysis Laboratory

* on leave from the Nuclear Energy Data Center (NEDAC)

** on leave from Fujitsu Limited

目　　次

CONTENTS

1. はじめに

SONATINA－2V[1]コードは，ブロック状燃料高温ガス炉の炉心2次元垂直断面モデルによる地震解析プログラムである。SONATINA－2Vで採用した解析モデルは，図1(a)に示すような円筒形状を持つ高温ガス炉の2次元垂直断面を図1(b)のようにモデル化して取り扱う。この垂直断面は，詳細には図1(c)に示すような要素で近似して計算している，炉心黒鉛ブロックは剛体として近似し，その運動は次のような形の運動方程式

$$\frac{dx_{k,\ell,i}}{dt} = x_{k,\ell,i+1} \qquad (i=1,3,5) \tag{1.a}$$

$$\frac{dx_{k,\ell,i}}{dt} = f_{k,\ell,i}(x_{1,\ell,1},\ x_{1,\ell,2},\ \cdots x_{k,\ell,i},\ \cdots x_{m,n,6}) \tag{1.b}$$

$$(i=2,4,6)$$
$$(k=1,\cdots,m,\ \ell=1,\cdots,n)$$

によって記述される。ここで添字のk，ℓは黒鉛ブロックの水平方向・垂直方向の番号を示す。xの添字i＝1, 3, 5はx_1, x_2, x_3がそれぞれ黒鉛ブロックの傾いた角度及び水平・垂直方向の変位であることを示している。また，xの添字i＝2, 4, 6はx_2, x_4, x_6がそれぞれ黒鉛ブロックの角速度及び水平・垂直方向の速度であることをそれぞれ示している。一方，（1.b）式の右辺は各黒鉛ブロックに働く力の成分を表す。ここで，力としては，図2に示すようなスプリングーダッシュポット近似によるブロック間の衝撃力，ブロックに付いているダウェル・ピンとソケット間の摩擦力，ブロックの重さによる重力などを考慮している。SONATINA－2Vでは，(1)式の連立常微分方程式を，ルンゲ・クッタ法またはオイラー法を用いて数値的に解いている。長時間の現象を模擬するときには，このコードの処理時間は非常に長くなる。そのために，ベクトル計算機FACOM VPの利用による処理時間の短縮が考えられる。

黒鉛ブロックの運動を記述する(1)式は，黒鉛ブロック間に衝撃及び摩擦力を介して相互作用があるために，各黒鉛ブロックの運動を独立に計算することはできない。しかし，(1)式の数値解法としてルンゲ・クッタ法またはオイラー法という単段階法を採用したために，並列処理が可能になった。すなわち，単段階法では，ある時刻tですでに求められている位置及び速度を用いて右辺の力を計算するために，右辺の計算はすべて独立に計算することができる。また，各力を計算した後に，次の時刻t＋Δtの位置及び速度を計算する場合にも，時刻tにおける速度と力がすべて既知であるために独立に計算することができる。ベクトル化は，この独立性を利用して行えば良い。

ところで，連立常微分方程式系(1)式をルンゲ・クッタ法で解くプログラムを作成する時に，各力の成分を一つのサブルーチンとして計算するという条件のもとで，考えられるプログラム構造としては図3に示すような2つの構造がある。プログラム構造Ⅰは，力を計算するサブルーチンFORCEの中にブロックに関するDOループ構造を持ち，一つのブロックを決め，それに対して働く力の成分をFORCE1，FORCE2．．．FORCEnと順に計算していく。一方，プログラム構造Ⅱは，力の成分を計算するサブルーチンFORCE1，FORCE2．．．FORCEnの中にブロックに関するDOループ

構造を持ち，一つの力の成分を決め，それを全ブロックについて計算を行う。プログラムの論理構造としては，ブロック又は力の成分のいずれかを優先して計算を行うという違いしかない。また，プログラムの処理効率から考えると，従来の汎用計算機（スカラ計算機）では，プログラム構造Ⅰの方が力を計算する関数呼び出しの回数がプログラム構造Ⅱより多いためオーバーヘッドが大きくなるが，ブロック数が極端に多くなければ処理時間にあまり大きな違いはない。しかし，DOループをベクトル命令で処理するベクトル計算機では，DOループをどのように構成するかによって処理性能に大きな違いがでてくることが考えられる。

SONATINA－2Vコードは，ベクトル計算機の利用を考慮せずに開発されているため，プログラム構造として上述のプログラム構造Ⅰを採用している。そのために，プログラムのほとんどの部分がベクトル化されていない。そこで，プログラムの構造をプログラム構造Ⅱに変更し，FACOM VP－100によって処理効率を改善した。燃料ブロックが水平方向に12個及び垂直方向に13個ある問題において，再構成されたコードのFACOM VP－100 ベクトル・モードでの処理時間はオリジナル・コードのスカラ・モードでの処理時間に比べ約1/6に短縮された。本報告では，プログラム構造の変更方法とその効果について報告し，ベクトル計算機を利用するプログラムにおいてプログラム構造の考察が重要であることを述べる。

本報告書の構成は，第2章でオリジナル・コードの構造及びFORTRANプログラムの動的解析ツールFORTUNE を使用した解析結果について述べる。次に，第3章では，ベクトル化のためのプログラムの変更方法及びベクトル化の手法について記述する。第4章ではベクトル化したSONATINA－2Vコードの実行結果について述べ，ベクトル・モードとスカラ・モードで実行した場合の演算結果の違いについて考察する。最後に，本報告の結論を第5章に要約する。

2. プログラムの解析

ベクトル化を行う範囲を明確にするために，SONATINA-2Vコードの処理時間の分布及びプログラム構造を解析した。ここで，ベクトル化のために使用したテスト・データはブロック状燃料高温ガス炉の垂直2次元炉心の1/2振動実験模型に対するものである。燃料ブロックの個数は，全体で156個（水平方向に12個及び垂直方向に13個）である。ベクトル化のテストとしては，地震発生から0.1秒までの現象を計算している。

図4にSONATINA-2Vを動的解析ツールFORTUNE[2)]によって解析したルーチン単位の解析結果を示す。SONATINA-2Vには，全部で21ルーチンがあるが，その中の8個のルーチン（FUN, BLOCK, DOWEL, VSPRNG, JULY31, MOMNT, MXCL, FRIC）で全演算コスト（処理時間に比例する量）の97.2%を占めている。

上記の8ルーチンの呼び出し関係は，JULY31とMXCLが最上位ルーチンであり，MXCLは他の7個のルーチンとは独立である。図5にJULY31以下の7ルーチンのTREE構造を示す。JULY31は，連立常微分方程式系のルンゲ・クッタ法またはオイラー法の制御ルーチンである。JULY31の下位に関数副プログラムFUNがある。FUNにはDERIVとBDERIVという副入口が2箇所ある。FUN及びBDERIVは(1)式の右辺に相当する各ブロックに働く力を計算するためのルーチンである。FUNで使用されている残りのルーチンVSPRNG, DOWEL, MOMNT, BLOCK, 及びFRICは，各ブロックに働く力の成分を計算している。VSPRNGでは，ブロック間に働く垂直バネ力とモーメントを計算している。DOWELでは，ブロックを結合しているダウエル・ピンに働く垂直バネ力とモーメントを計算している。MOMNTは，各ブロックの重さとガス圧の差による力とそのモーメントを計算している。FRICは，各ブロック間に働く摩擦力を計算している。BLOCKでは，各ブロックの衝突の仕方を分類している。

このプログラム構造は，前述の図3のプログラム構造Iであり，FUNの中にブロックに関するDOループ構造を持ち，FUNで決められた一つのブロックについて下位ルーチンで力の計算を行っている。そのため，最下位のルーチンにはほとんどDOループ構造がない。そこで，オリジナル・コードをFACOM FORTRAN77/VPコンパイラによって，自動ベクトル化を行うと，ほとんどのDOループはベクトル化されるが，ブロック数が水平方向に12個及び垂直方向に13個の問題ではベクトル化率は9.4%である。また，実際の装置でもブロック数は水平方向に最大30個及び垂直方向に最大20個であるために，そのベクトル長は最大でもベクトル計算の効率が出にくい20から30程度の小さな数である。

3. ベクトル化

ベクトル化作業は，次に示す二段階で行った。第一段階は，並列処理可能な部分をＤＯループとして表現できるようにプログラムの構造を変更した。次に第二段階では，サブルーチン単位にベクトル演算が効率的に行えるように書き替えた。この書き替えたベクトル化コードとオリジナル・コードはスカラ処理においては，計算結果は完全に一致するようにしている。

3.1 プログラム構造の変更

プログラム構造の変更方法としては，上位ルーチンＤＯループの下位ルーチンへの移動及び下位ルーチンの上位ルーチンへの展開（インライン展開）の二つを考えた。図６にこのプログラム構造の変更の簡単な例を示す。図6(a)に示す上位ルーチンＤＯループの下位ルーチンへの移動は，下位ルーチンが比較的ステップ数の多い場合に行った。この方法では，上位ルーチンＤＯループ中でサブルーチンや関数副プログラムを呼び出している文の前後でＤＯループを分割する。この時，分割したＤＯループの間でデータの受け渡しが必要となるスカラ変数は作業用配列（この例では，Ｘ¥，Ｙ¥，Ｚ¥）を用意して，ＤＯループ間及びルーチン間で論理構造が変化しないようにする必要がある。一方，図6(b)に示すインライン展開は，下位ルーチンのステップ数が少ない場合に行った。

以下では，ベクトル化のために行ったプログラムの変更をサブルーチン別に示す。オリジナル・コードでは境界条件の取り扱いのために，最下層にあるブロックとそれ以外のブロックを区別して処理している。しかし，ベクトル化の説明ではこの区別は本質的ではないので特別に断らない限り両者を区別しないで，一般のブロックについて記述する。

3.1.1 JULY31

ＪＵＬＹ３１は，ブロックの運動を記述する連立常微分方程式(1)をルンゲ・クッタ法で解くための制御ルーチンである。

このルーチンで行ったベクトル化のための変更としては，ブロック毎に計算されている力の成分を合成して加速度を計算するための関数DERIVをインライン展開した。図7(A)はオリジナル・コードのＪＵＬＹ３１の一部である。この文番号３１０の三重ＤＯループで呼ばれているDERIVの内容を展開したのが図7(B)である。計算結果ＡＣ，Ｔ１は異なるＤＯループ変数ＫＩＤＸ，ＬＩＤＸ，ＩＩＤＸに対してそれぞれ独立であるので，ＤＯループの順序を入れ換えても計算結果が変わらない。そこでＤＯループ長の短い変数ＩＩＤＸを最も外側に移動し，ブロックの垂直方向インデックスＬＩＤＸと水平方向インデックスＫＩＤＸをその内側のＤＯループになるように構造を変更した。次に，最外ＤＯループ変数ＩＩＤＸは，SONATINA-2Ｖで扱っている変位及び速度という物理量を示し，その物理量の数は６と決まっているので，このＩＩＤＸに関する演算を図7(c)のように展開した。この手法は外側ＤＯループのアンローリングと呼ばれている。更に，この図には示していないが，このＬＩＤＸとＫＩＤＸに関する二重ＤＯループに対して，次節で述べるＤＯループの一重化を行っている。

3.1.2 FUN

関数副プログラムFUNでは各ブロックに働く力を計算する。ベクトル化のための変更として上位ルーチンDOループを下位ルーチンに移動した。図8(a)に示すように，FUNは水平方向と垂直方向の各ブロックに対応するインデックスKとLに関する二重DOループがある。このDOループの中で，ブロックに働く力の成分を計算するサブルーチンVSPRNG，DOWEL，MOMNT，BLOCK 及びFRICを呼び出している。オリジナル・コードでは，各ブロック毎に各力の成分を全て計算している。しかし，常微分方程式の数値解法としてルンゲ・クッタ法を採用するという近似のもとでは，各力とモーメントの成分はブロック毎に独立に計算できるので，図8(b)のように，FUNにある二重DOループを各力の成分を計算するサブルーチンに移動して，力の成分毎に全ブロックをまとめて計算するように変更した。

3.1.3 FRIC

関数副プログラムFRICは，各ブロックの速度の関数である動摩擦力と速度に依存しない静摩擦力を計算する。

オリジナル・プログラムはFRICを関数副プログラムとして計算していたが，ベクトル化版ではFUNの構造変更に伴ってFRICをサブルーチン化した。そして，ベクトル化版ではFRICを呼ぶ前にFUNで全ブロックの速度が計算されているので，その速度を使用して全ブロックに働く摩擦力をまとめて計算するようにした。

3.1.4 VSPRNG・DOWEL

VSPRNGはブロックに働く垂直バネ力とそのモーメントを計算する。一方，DOWELは各ブロックの上面及び下面の中心から左右に一定距離だけ離れた2点にあるダウェル・ピンに働く垂直ばね力とそのモーメントを計算する。この2ルーチンは，計算の対象がブロックとダウェル・ピンという違いがあるが，ベクトル化という観点からはプログラムの構造が同じなので，ここで一緒に取り扱う。

図9(a)及び(b)にVSPRNG及びDOWELのオリジナル・コードのプログラム構造を示す。SONATINA-2Vでは，各ブロックに働く垂直力は各ブロックの上面及び下面の中心から左右に一定距離だけ離れた2点に図2のスプリング・ダッシュポットがそれぞれあると仮定している。また，ダウェル・ピンに働く垂直力もスプリング・ダッシュポットで近似している。そのために，ループ長2のブロックの右側と左側のスプリング・ダッシュポットに関するDOループがある。FUNの構造変更に伴い，全ブロックに関する二重DOループをVSPRNGとDOWELに持ち込み，全ブロックの上下面の左右にあるスプリング・ダッシュポットによる垂直バネ力とモーメントをまとめて計算するように変更した。図9(c)及び(d)にVSPRNG及びDOWELのベクトル化コードのプログラム構造を示す。この各ブロックの上下面の左右にあるスプリング・ダッシュポットに働く垂直バネ力とモーメントの合成の計算では上下に重なったブロックの境界面に注目して計算しているため，ブロックの垂直方向に関しては再帰演算になるような累積計算があり，単純にブロックに関するDOループと左右にあるスプリング・ダッシュポットに関するDOループの順序をかえることはできない。そのために，DOループを力の成分を計算するDOループと累積計算を行うDOループに分割し，ベクトル化版とスカラ版では，スカラ演算を行ったときに計算順序が変更されないようにしている。各ブロックに関する二重DOループについては，次節で述べるDOループの一重化を行った。

3.1.5 MOMNT

MOMNTでは，ブロックの重さとガス圧の差による力とそのモーメントを計算している。ＦＵＮの構造変更に伴う二重ＤＯループの移動によって，ブロックの重さとガス圧の差によるモーメントを全ブロックに関して，まとめて行うように変更した。図１０(A)はベクトル化のためにプログラム構造を変更した後のサブルーチンMOMNT中のＤＯループである。文番号１８３０の二重ＤＯループについては，内側のループ変数がＬであり，配列の第二添字についてベクトル化をしているために記憶領域上で一定間隔毎にあるデータをアクセスをしている。また，二重ＤＯループの内側にある文番号1820のＤＯループでは，外側ループ変数の値が変わる毎にベクトル長が減少していく演算になっている。そのためベクトル長が小さい場合には，ベクトル演算の効果が得られにくい。そこで，このＤＯループを図１０(B)のように改めた。まず文番号１８３０のＤＯループを，文番号１８２０のＤＯループの前後で分割した。そして，文番号１８２０のＤＯループについては，変数ＬＤＭＹに関するＤＯループと変数Ｋに関するＤＯループを入れ換えて，ＤＯループ変数Ｋによる連続アクセスで，しかもベクトル長がＬ及びＬＤＭＹの値によって変化しないベクトル演算を行えるようにした。文番号８２０と文番号１８３５に見られるＤＯループの一重化については3.3.2で述べる。また，ベクトル化版の文番号８２０と文番号８３０のＤＯループでＬの値が２ではなく１から始まっているのは，オリジナル版では別に処理している最下層のブロックをまとめて処理しているためである。

3.1.6 BLOCK

BLOCKは異なるブロック列間の水平方向の衝突力とそのモーメントを計算している。このBLOCKでは，ブロックの衝突する場合によって，８種類の衝突の仕方があり，それを分類し，それぞれの衝突に対してスプリングの歪度とその時間変化量を計算している。ＦＵＮの構造変更によるブロックに関する二重ＤＯループを持ち込むだけで全体が自動ベクトル化される。BLOCKでは衝突が起こったかどうかを判定して衝突後の反発係数などを計算している。そして，この衝突は全ブロックに対して起こる訳ではなく，また衝突が８種類に分類されているために，衝突後の処理しなければならないブロックの数は非常に少なくなる。そのため，衝突後の処理部分までベクトル化すると，その部分はベクトル長の小さい計算であり，処理効率が悪いことがわかった。そこで，図１１に示すように衝突後の処理部分をスカラ処理するようにＤＯループを分割した。図１１(A)はベクトル化前の８種類の衝突の一つを扱っている部分を例として示している。ここでＡＬＰＨとＸＩという変数は，隣合ったブロックの間隔を表すパラメータであり，ＡＬＰＨが正または零でかつＸＩが正の値の時に隣合ったブロックが衝突したとみなす。図１１(B)に示すベクトル化版では文番号３０００のＤＯループで衝突前の処理，文番号８０００のＤＯループで衝突後の処理を行っている。文番号３０００のＤＯループで衝突が起こりうる場合にはＩＦＬＧ¥２の値を１に設定しＸＩの計算値と衝突後の計算で必要になる量を作業用配列に保存しておく。このＤＯループは，ベクトル処理を行う。次に文番号８０００のループでＩＦＬＧ¥２が１になっているブロックに対して衝突後の計算をスカラ処理する。

3.2 ベクトル化の作業用配列

ベクトル化のためのプログラム構造の変更で作業用配列が必要となる。この作業用配列は，メインルーチンで記憶領域を確保し，各サブルーチンに引数として受け渡ししている。ベクトル化に伴い，

作業用配列の数が多くなると，各ルーチンの呼び出しに要するオーバーヘッドが大きくなる。そのために，このオーバーヘッドを減らす方法を考える必要がある。その一方法として図１２に示すようなENTRY文を使用する方法を採用した。図１２(A)に構造変更によって引数の数が多くなった段階でのJULY３１のサブルーチン文を示す。ここで最後に¥記号の付いている１２個の引数が，ベクトル化のために追加された作業用配列である。この引数を減らすために，図１２(B)に示すようにENTRY文を使用する。JULY３１の上位ルーチンでJULY３１が呼ばれる前にENTRY名SETJLYを呼び出すと，作業用配列のアドレス設定と言う準備作業を行う。このようにすると，SETJLYを一回呼ぶだけで，以後は，作業用配列のアドレスが変わらなけば，JULY３１で呼び出す場合には作業用配列を仮引数に加える必要がなくなる。この手順を最下位のルーチンまで繰り返す。ベクトル化を行った他の各ルーチンFUN，VSPRNG，DOWEL，MOMNT，BLOCKについて，上述のJULY３１と同様の方法で作業用配列を使用するための準備をENTRY文を利用して行っている。

3.3　各サブルーチン共通の変更

ここでは，ベクトル化のために構造を変えた個々のサブルーチンや関数副プログラムに対して共通に用いたベクトル化手法について説明を行う。また，用いた手法を各ルーチン毎に表１にまとめておく。

3.3.1　共通部分の省略

この作業は本来のベクトル化とは関係ないスカラー演算の最適化であるが，処理時間の短縮に効果があるのでここに記述する。オリジナル版では，図１３(A)に示すようにブロックが傾いた場合のブロックの高さ及び幅を計算するのに各サブルーチン毎に，ＳＩＮとＣＯＳの値を計算している。このＳＩＮとＣＯＳは計算時間を要する組み込み関数であり，また，プログラムで使用しているＳＩＮとＣＯＳの値は，前の時間ステップで求められているブロックの傾き角度を使用して求める値であるので，各時間ステップ毎に最初に一度計算すれば良いことがわかる。そこで，図１３(B)に示すように，ＳＩＮとＣＯＳの計算のみを行うサブルーチンＳＩＮＣＯＳをつくり，関数副プログラムＦＵＮのはじめにＳＩＮとＣＯＳの値をサブルーチンＳＩＮＣＯＳで計算し，作業用配列ＳＮＴ¥，ＣＳＴ¥に格納しておく。以後，ＳＩＮとＣＯＳを使用している場所ではＳＮＴ¥とＣＳＴ¥の値を使用するように変更した。

3.3.2　ＤＯループの一重化

SONATINA-2Vでは，解析対象の装置のブロック数に上限があり，水平方向に最大３０個また垂直方向に最大２０個である。そしてこの２つの値のどちらかが，ＤＯループの長さとなる。プログラムのなかでは，この水平方向と垂直方向に関する部分は二重ＤＯループとしてあらわれる。図１４(A)の例に示すように自動ベクトル化の範囲では配列の宣言の値と二重ＤＯループの繰り返し回数が異なるために多重ＤＯループのベクトル化の対象外[3]であり，この二重ＤＯループは最内ＤＯループ変数でしかベクトル化されない。そのままでは，ベクトル長が小さいために大きなベクトル化効果を期待できない。そこで図１４(B)に示すようなＤＯループの一重化を行い，ベクトル長を大きくした。この方法では，一重ＤＯループ変数から二重ＤＯループの時の変数を計算しているためのオーバーヘッド

があるが，ベクトル化しないサブルーチンとの整合性及びベクトル化後のプログラムの読みやすさなどの点から，全ての配列を無理に一次元化していない。ここで配列ＳＮＴ¥及びＣＳＴ¥は，大きさが（Ｍ，Ｎ）で，ＤＯループの繰り返し回数と一致しているため一次元配列にしている。配列Ｘは配列宣言の大きさとＤＯループの長さが一致していないので二次元配列のまま使用している。

4. ベクトル化の結果

4.1 ベクトル化の結果

表2にSONATINA-2Vのオリジナル版及びベクトル化版のFORTUNEコスト及びベクトル化率を各サブルーチン毎にまとめた。ベクトル化版では，FUNとBLOCKを除きベクトル化率が97.0%以上になっている。FUNとBLOCKのベクトル化率が低いのは，ベクトル化の効果が期待出来ないベクトル長の短いDOループを強制的にスカラ処理しているためである。全体のベクトル化率としては，オリジナル版の9.4%からベクトル化版の92.3%に向上した。表3には，オリジナル版をスカラ・モードで実行した場合及びベクトル化版をスカラ・モードとベクトル・モードで実行した場合の実行時間t(秒)，オリジナル版のスカラ・モード実行時間を基準とした各実行時間の比r1及びベクトル化版のスカラ・モード実行時間を基準とした各実行時間の比r2を示している。

オリジナル版とベクトル化版をそれぞれスカラ・モードで実行した場合に，ベクトル化版の方がオリジナル版に比べて処理時間が1／1.34に短縮されている。これは，ベクトル化版はプログラムの構造変更及び共通計算を減少したために関数呼び出しの回数が非常に少なくなっているためである。表4にサブルーチンの呼び出し回数をまとめる。まず，プログラムの構造変更のためにFUN以下のサブルーチン（BLOCK，DOWEL，MOMNTとVSPRNG）の呼び出し回数が1／156（＝1／(12×13)）に減少している，FRICは関数副プログラムからサブルーチンに変更され呼び出し回数が1／182（＝1／(14×13)）に減っている，そしてDERIVはインライン展開されたために呼び出し回数が1/19に減少している。このために，サブルーチンの呼び出しに伴うオーバーヘッドが減少した。次に，ベクトル化版では，組み込み関数SINとCOSの計算を各時間ステップで一回のみ行い，以後のその値を使うように変更したために，組み込み関数SINとCOSの使用回数の合計がオリジナル版と比較して約1/5に減少している。

ベクトル化版のスカラ・モードとベクトル・モードでは，ベクトル・モードの方がスカラ・モードと比べて処理時間が1／4.62（オリジナル版のスカラ・モードの実行時間に比べて1／6.21）に短縮されている。これは，プログラム全体のベクトル化率が約92.3%と高いことと二重DOループの一重化によってベクトル長が156と比較的大きいためである。

ここで，プログラム構造と処理性能について考えてみる。ベクトル版では組み込み関数SINとCOSの使用が最適化されているため，プログラム構造の違いによるベクトル化版とオリジナル版の処理性能を比較するときには，この違いを較正しなければならない。ベクトル化版の総処理時間とFORTUNE のコスト分布から推定されるサブルーチンSINCOSの処理時間79.04×0.074＝5.84秒とオリジナル版とベクトル化版で組み込み関数SINとCOSの呼び出し回数が1/5に減少していることから，オリジナル版でベクトル化版と同様に組み込み関数SINとCOSの呼び出し回数が最適化されているとしたときの総処理時間は106.12−5.84×4＝82.76秒である。即ち，スカラ処理の範囲ではオリジナル版とベクトル化版の処理時間には差がほとんどないことがわかる。しかし，両プログラムをベクトル処理をすると4.62倍処理性能に差があらわれる。このように，プログラム設計を行う場合に，ベクトル計算機を使用することを考えたプログラム構造を選択することが重要である。

4.2 計算結果の違いについて

スカラ・モードとベクトル・モードによる計算結果は，各演算については全く違いがない。しかし，組み込み関数の展開式やベクトル計算を効率よく行うための最適化に伴う計算順序の違いによって，その計算結果に差を生じることがあるが，その差は工学的に要求される有効数字の範囲では無視できるほど小さい。SONATINA-2Vのベクトル化版でもスカラ・モード（オリジナルのスカラ・モードと同一）とベクトル・モードで計算結果に違いがあらわれている。この計算結果の違いについて，数値的及び物理的に考察してみる。

数値的には，ベクトル化版で組み込み関数ＳＩＮとＣＯＳをスカラ・モードで計算し，かつプログラムをFORTRAN77／VPコンパイラのデバック・モードであるNOEVLオプション[3]（スカラ・モードでコンパイルした場合と演算順序を同一にする）を指定してベクトル・モードで実行するとその計算結果は完全にスカラ・モードでの計算結果と一致する。このことは，ベクトル化版のベクトル・モードの結果が，スカラ・モードの結果と異なる理由が，ベクトル計算用とスカラ計算用の組み込み関数ＳＩＮとＣＯＳの精度が1bit 程度異なることとベクトル計算とスカラ計算の演算順序が異なっていることから来ることを示している。

物理的にベクトル・モードとスカラ・モードの計算結果の違いを以下で考察する。ここで，使用した計算体系は図1(c)に示したブロック状燃料高温ガス炉の垂直２次元炉心の1/2振動実験模型を使用した。二つのモードによる計算結果の比較を図１５と図１６に示す。図１５(a)及び(c)は炉心周辺カラムの変位時刻歴応答曲線であり，図中の実線はスカラ・モード，点線はベクトル・モードによるものである。

二つのモードによる計算結果において，差はわずかである。図１５(c)及び(d)は炉心周辺カラムの衝突力歴応答曲線であり，変位よりも大きな差がみられる。この衝突力の最大値のカラム軸方向の分布を図１６に示す。二つのモードによる計算結果の差は最大で約４０％であるが，カラムの最大衝突力差はわずかに３％である。以下，この差について考えてみる。

ブロック状燃料から構成された高温ガス炉の炉心地震応答解析コードSONATINA-2Vにおいて，衝突現象は図2(a)に示したばねとダッシュポットを用いた粘弾性モデルで計算する。衝突の瞬間，ブロック間のギャップが零または負になる（すなわちオーバラップする）と衝突力が発生する。隣り合った二つのブロックｉとｊの間のキャップを δ_{ij} とすると，ばねとダッシュポットの力は変位差 $X_i - X_j$ と速度差 $V_i - V_j$ の関数として表される。

ばね力 F_s

$$F_s = K_{ij}(X_i - X_j - \delta_{ij}) \quad ; \quad X_j < |X_i - \delta_{ij}| \qquad (2.a)$$

$$F_s = 0 \quad ; \quad X_j \geqq |X_i - \delta_{ij}| \qquad (2.b)$$

ダッシュポット力 F_d

$$F_d = C_{ij}(V_i - V_j) \quad ; \quad X_j < |X_i - \delta_{ij}| \qquad (3.a)$$

$$F_d = 0 \quad ; \quad X_j \geqq |X_i - \delta_{ij}| \qquad (3.b)$$

衝突力 F

$$F = F_s + F_d \qquad (4)$$

ここで，X，V，K，Cはそれぞれ，変位，速度，ばね定数，粘性係数である。

上の式で明らかなように，衝突力は２物体間のギャップのオーバラップ量によって変化する。さらに，ばね定数は一般に大きい値であるので，わずかのオーバラップ量によっても大きな衝突力を発生する。よってスカラ・モードとベクトル・モードで用いている組み込み関数のわずかな差によってブロック変位のわずかな差が生じ，これによってブロックの衝突力に差が生ずる。また，解析コードの内部では，計算結果の発散を避けるために，積分時間刻みを変化させている。その結果，始めの計算結果の差が，少しづつ伝播して，最終計算結果では数１０％の差を生じることがある。しかし幸いなことに，計算が発散しない限り最大応答値の差は図１６から明らかなように，スカラ・モードとベクトル・モードにおいて大きくない。これはエネルギーバランスから考えても妥当なことと思われる。

5. おわりに

ブロック状燃料高温ガス炉の炉心２次元垂直断面モデルによる地震解析コードSONATINA－２Vをベクトル化した。このコードでは，ブロックの運動を記述する連立常微分方程式をルンゲ・クッタ法によって解いている。この解法は，陽解法であるために，ある時間ステップですべてのブロックについての計算を並列にすることができ，またこの問題にはその他の並列性が存在しない。オリジナルのSONATINA－２Vコードは，このブロックに関する並列性をベクトル計算という形で表現できないプログラム構造であった。そこで，そのプログラム構造をベクトル計算向きの構造に変更した。さらに，燃料ブロックに関する二重ＤＯループを一重ＤＯループ化してベクトル長を大きくすることによりベクトル演算の処理効率を向上させた。このベクトル化版をブロック状燃料高温ガス炉の垂直２次元炉心の1／2振動実験模型のデータでテスト計算した結果，サブルーチン及び組み込み関数の呼び出し回数が減少したために，オリジナル版とベクトル化版をそれぞれスカラ・モードで実行した場合に，ベクトル化版の方がオリジナル版に比べて処理時間が1／1.34に短縮された。同一のデータに対してベクトル化版のスカラ・モードとベクトル・モードでは，ベクトル・モードの方がスカラ・モードと比べて処理時間が１／4.62（オリジナル版のスカラ・モードの実行時間に比べて１／6.21）に短縮された。

SONATINA－２Vのベクトル化にみられたように，プログラム構造の選択によってスカラ処理の範囲では処理時間にあまり違いがないが，ベクトル処理では処理時間に大きな違いがあり得るので，プログラム設計時にはベクトル計算機に適合するプログラム構造かどうかを検討することが重要である。

謝　　辞

計算センター室長浅井清氏には，本報告書を書く機会を与えて頂くと共に，日常的に激励していただきました。計算センター藤井実氏には，本報告書を検討していただきました．深く感謝致します。

また，著者の一人鶴岡卓哉は，(財)原子力データ・センター専務理事三井田純一博士及び開発第二課長桜井文雄氏に外来研究員として本研究を行う機会を与えていただきましたことを感謝いたします。

参　考　文　献

1) IKUSHIMA T.: SONATINA－2A A computer program for seismic analysis of the two dimensional vertical slice HTGR core JAERI 1279 (1982).

2) 富士通 FACOM OS Ⅳ FORTUNE使用手引書 70SP－5730－1 (1984).

3) 富士通 FACOM OS Ⅳ／F4 MSP FORTRAN77／VP使用手引 78SP－5680－2 (1984).

5. おわりに

ブロック状燃料高温ガス炉の炉心２次元垂直断面モデルによる地震解析コードSONATINA-2Vをベクトル化した。このコードでは，ブロックの運動を記述する連立常微分方程式をルンゲ・クッタ法によって解いている。この解法は，陽解法であるために，ある時間ステップですべてのブロックについての計算を並列にすることができ，またこの問題にはその他の並列性が存在しない。オリジナルのSONATINA-2Vコードは，このブロックに関する並列性をベクトル計算という形で表現できないプログラム構造であった。そこで，そのプログラム構造をベクトル計算向きの構造に変更した。さらに，燃料ブロックに関する二重ＤＯループを一重ＤＯループ化してベクトル長を大きくすることによりベクトル演算の処理効率を向上させた。このベクトル化版をブロック状燃料高温ガス炉の垂直２次元炉心の1／2振動実験模型のデータでテスト計算した結果，サブルーチン及び組み込み関数の呼び出し回数が減少したために，オリジナル版とベクトル化版をそれぞれスカラ・モードで実行した場合に，ベクトル化版の方がオリジナル版に比べて処理時間が1／1.34に短縮された。同一のデータに対してベクトル化版のスカラ・モードとベクトル・モードでは，ベクトル・モードの方がスカラ・モードと比べて処理時間が１／4.62（オリジナル版のスカラ・モードの実行時間に比べて１／6.21）に短縮された。

SONATINA-2Vのベクトル化にみられたように，プログラム構造の選択によってスカラ処理の範囲では処理時間にあまり違いがないが，ベクトル処理では処理時間に大きな違いがあり得るので，プログラム設計時にはベクトル計算機に適合するプログラム構造かどうかを検討することが重要である。

謝　　辞

計算センター室長浅井清氏には，本報告書を書く機会を与えて頂くと共に，日常的に激励していただきました。計算センター藤井実氏には，本報告書を検討していただきました，深く感謝致します。

また，著者の一人鶴岡卓哉は，㈶原子力データ・センター専務理事三井田純一博士及び開発第二課長桜井文雄氏に外来研究員として本研究を行う機会を与えていただきましたことを感謝いたします。

参　考　文　献

1) IKUSHIMA T.: SONATINA-2A A computer program for seismic analysis of the two dimensional vertical slice HTGR core JAERI 1279 (1982).

2) 富士通 FACOM OS Ⅳ FORTUNE使用手引書 70SP-5730-1 (1984).

3) 富士通 FACOM OS Ⅳ/F4 MSP FORTRAN77/VP使用手引 78SP-5680-2 (1984).

5. おわりに

ブロック状燃料高温ガス炉の炉心２次元垂直断面モデルによる地震解析コードSONATINA-2Vをベクトル化した。このコードでは，ブロックの運動を記述する連立常微分方程式をルンゲ・クッタ法によって解いている。この解法は，陽解法であるために，ある時間ステップですべてのブロックについての計算を並列にすることができ，またこの問題にはその他の並列性が存在しない。オリジナルのSONATINA-2Vコードは，このブロックに関する並列性をベクトル計算という形で表現できないプログラム構造であった。そこで，そのプログラム構造をベクトル計算向きの構造に変更した。さらに，燃料ブロックに関する二重ＤＯループを一重ＤＯループ化してベクトル長を大きくすることによりベクトル演算の処理効率を向上させた。このベクトル化版をブロック状燃料高温ガス炉の垂直２次元炉心の1／2振動実験模型のデータでテスト計算した結果，サブルーチン及び組み込み関数の呼び出し回数が減少したために，オリジナル版とベクトル化版をそれぞれスカラ・モードで実行した場合に，ベクトル化版の方がオリジナル版に比べて処理時間が1／1.34に短縮された。同一のデータに対してベクトル化版のスカラ・モードとベクトル・モードでは，ベクトル・モードの方がスカラ・モードと比べて処理時間が１／4.62（オリジナル版のスカラ・モードの実行時間に比べて１／6.21）に短縮された。

SONATINA-2Vのベクトル化にみられたように，プログラム構造の選択によってスカラ処理の範囲では処理時間にあまり違いがないが，ベクトル処理では処理時間に大きな違いがあり得るので，プログラム設計時にはベクトル計算機に適合するプログラム構造かどうかを検討することが重要である。

謝　　辞

計算センター室長浅井清氏には，本報告書を書く機会を与えて頂くと共に，日常的に激励していただきました。計算センター藤井実氏には，本報告書を検討していただきました，深く感謝致します。

また，著者の一人鶴岡卓哉は，㈶原子力データ・センター専務理事三井田純一博士及び開発第二課長桜井文雄氏に外来研究員として本研究を行う機会を与えていただきましたことを感謝いたします。

参　考　文　献

1) IKUSHIMA T.: SONATINA-2A A computer program for seismic analysis of the two dimensional vertical slice HTGR core JAERI 1279 (1982).

2) 富士通 FACOM OS Ⅳ FORTUNE使用手引書 70SP-5730-1 (1984).

3) 富士通 FACOM OS Ⅳ/F4 MSP FORTRAN77/VP使用手引 78SP-5680-2 (1984).

Table 1 The summary of the methods for vector optimization.
L-MERGE stands for LOOP MERGE, L-UNROLL for LOOP UNROLLING, L-INDEX means optimization of vectorized do loop index, and F-REDUCT means reduction of built-in functions.
V-RATIO is the vectorization ratio in each subroutine with vector length indicated in the TABLE.

SUBROUTINE	METHOD FOR VECTOR OPTIMIZATION	VECTOR LENGTH (V-RATIO)
JULY31	L-MERGE, L-UNROLL, L-INDEX	156 (95)
FUN	L-MERGE, F-REDUCT	144 (52) 12-14 (21)
SINCOS	L-MERGE	156 (99)
ZERO	L-MERGE	156 (99)
VSPRNG	L-MERGE, L-DIST	144 (95) 12 (4)
DOWEL	L-MERGE, L-DIST	144 (96) 12 (4)
MOMNT	L-INDEX, L-MERGE	156-144 (40) 12 (60)
BLOCK	F-REDUCT, L-DIST	143 (87) 12 (2)
FRIC	L-MERGE	156 (99)
MXCL	L-INDEX	24336 (15) 182 (12) 12 (72)

Table 2 The FORTUNE COST and the vectorization ratio of original version and vectorized one.

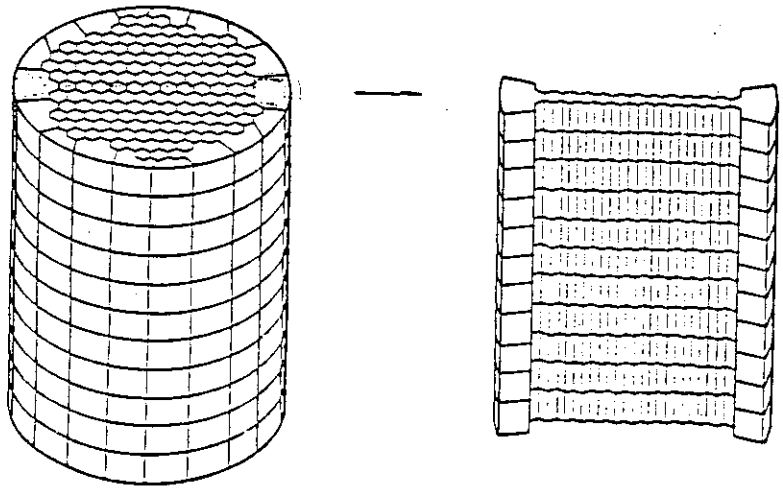
SUBROUTINE	ORIGINAL VERSION COST (%)	ORIGINAL VERSION V (%)	VECTORIZED VERSION COST (%)	VECTORIZED VERSION V (%)
JULY31	7.0	81.3	12.1	99.0
FUN	24.7	5.9	10.8	71.6
VSPRNG	12.7	0.0	17.3	99.1
DOWEL	16.3	0.0	22.3	97.3
MOMNT	4.6	20.9	5.5	97.3
BLOCK	28.8	0.0	16.9	85.9
FRIC	1.5	0.0	2.0	99.7
MXCL	1.6	80.2	2.1	99.8
SINCOS	---	---	7.4	99.9
ZERO	---	---	2.4	98.8
TOTAL	97.2	9.4	98.8	92.3

Table 3 The execution time t, their ratios r1 and r2 for original version and vectorized one.

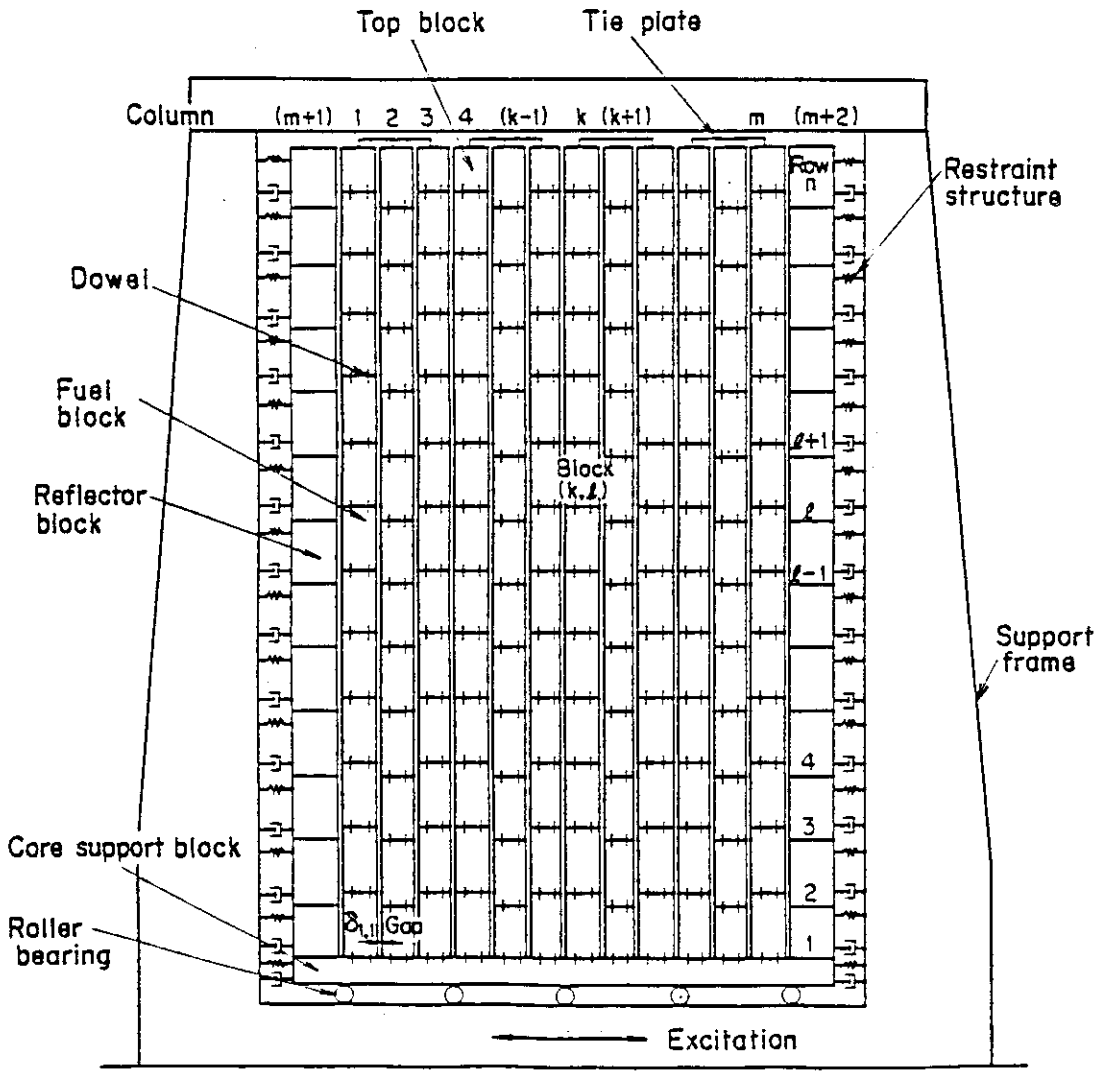
	ORIGINAL (scalar)	VECTORIZED (scalar)	VECTORIZED (vector)
t	106.12 sec	79.04 sec	17.09 sec
r1	1	0.74 (1/1.34)	0.16 (1/6.21)
r2	1.34 (1/0.74)	1	0.23 (1/4.62)

Table 4 The count of execution of subroutine for original version and vectorized one.

SUBROUTINE	ORIGINAL (NO)	VECTORIZED (NV)	RATIO (NO/NV)
BLOCK	632112	4052	156
DOWEL	632112	4052	156
FRIC	737464	4052	182
FUN (BDERIV)	4052	4052	1
(DERIV)	4003376	210704	19
JULY31	1013	1013	1
MOMNT	632112	4052	156
VSPRNG	632112	4052	156
SIN and COS	6343471	1281445	5

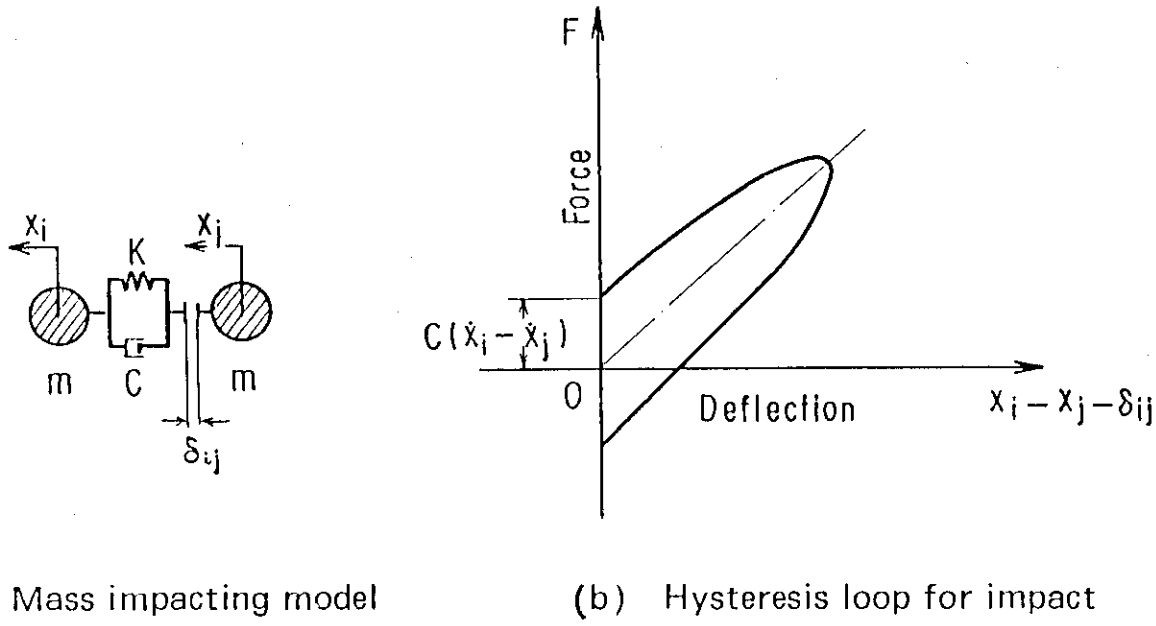


(a) Core (b) Model



(c) Calculation model of two-dimensional vertical slice core.

Fig. 1 Calculation model of the two-dimensional vertical slice core for the SONATINA-2V code.



(a) Mass impacting model (b) Hysteresis loop for impact

Fig. 2 Viscoelastic model with two impacting bodies.

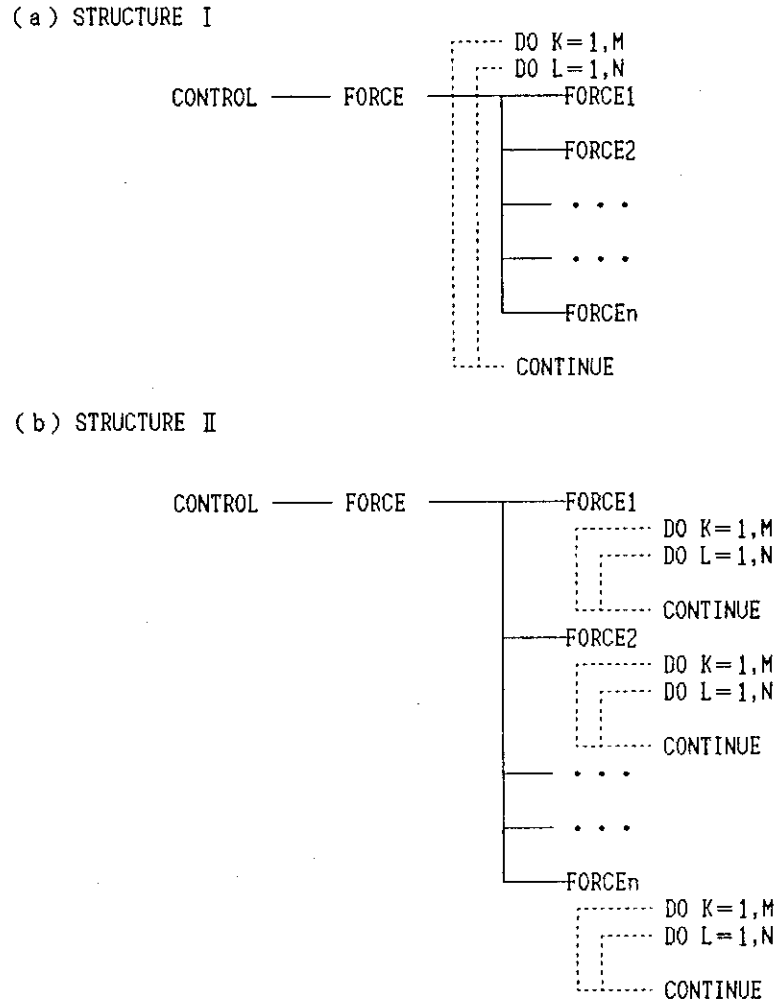


Fig. 3 The possible program structures for the SONATINA-2V code.

NO.	ROUTINE	LINES	EXECUTIONS	COST	%	0.....1.....2.....
0001	BLOCK	377	632112	360080193	28.8	**********
0002	FUN	722	1013	312790748	25.0	**********
	BDERIV		3039			
	DERIV		4003376			
0003	DOWEL	185	632112	204021992	16.3	**********
0004	VSPRNG	106	632112	158902461	12.7	**********
0005	JULY31	385	1013	87183845	7.0	**********
0006	MOMNT	104	632112	56982948	4.6	*******
0007	MXCL	115	681	19437102	1.6	**
0008	FUNBO	99	106365	15352015	1.2	**
	FUNB1		105352			
0009	FRIC	28	737464	13275602	1.1	*
0010	FUNWO	32	97248	12642240	1.0	*
	FUNW1		97248			
0011	ROC	753	1	6194611	0.5	
0012	DTAPR	447	1	1404397	0.1	
0013	MXPR	192	1	424395	0.0	
0014	SETARY	320	1	175437	0.0	
0015	CRDINN	754	1	108709	0.0	
0016	TITLE	9	14	14112	0.0	
0017	MAIN	96	1	1004	0.0	
0018	FMPR	88	0			
0019	FUNUO	98	0			
	FUNU1		0			
0020	FIT	11	0			
0021	XPR	81	0			
		5002		1248991811		

Fig. 4 The summary of execution cost measured by FORTUNE for subroutines of the original SONATINA-2V code.

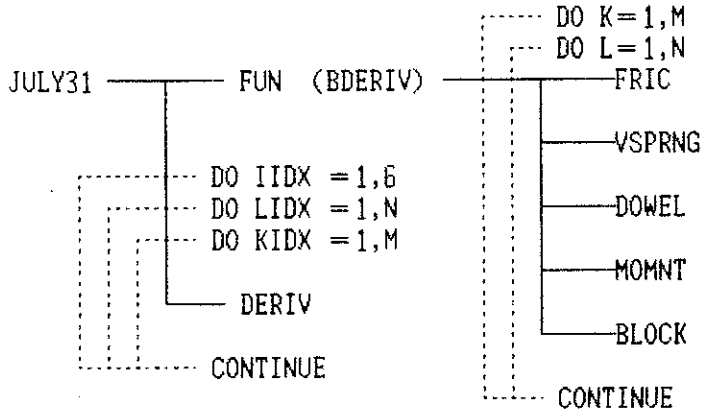


Fig. 5 The tree structure for the kernel part of the original SONATINA-2V code.

```
      DO 100  I = 1 , N
      X = A (I )
      Y = B (I )
      CALL EXAMPLE ( X , Y , Z  )
       C (I ) = Z
  100 CONTINUE
```

```
      SUBROUTINE EXAMPLE ( X , Y , Z  )
      Z = X + Y
      RETURN
      END
```

(a) Transfer of do-loop to bottom level subroutine EXAMPLE

```
      DO 100  I = 1 , N
       X¥ (I ) = A (I )
       Y¥ (I ) = B (I )
  100 CONTINUE

      CALL EXAMPLE ( X¥ , Y¥ , Z¥ , N  )

      DO 110  I = 1 , N
       C (I ) = Z¥ (I )
  110 CONTINUE
```

```
      SUBROUTINE EXAMPLE ( X , Y , Z  , N )
      DIMENSION  X (N) ,  Y (N) ,  Z (N)
      DO 100  I = 1 , N
       Z (I ) = X (I ) +  Y (I )
  100 CONTINUE
      RETURN
      END
```

(b) Inline expansion of bottom level subroutine EXAMPLE

```
      DO 100  I = 1 , N
      X = A (I )
      Y = B (I )
      Z = X + Y          ←      CALL EXAMPLE ( X , Y , Z  )
      C  (I ) = Z
  100 CONTINUE
```

Fig. 6 A simple example for the restructure of program.

```
(A)
  1-----------------DO 310 KIDX = 1 , M
  1 2---------------DO 310 LIDX = 1 , N
  1 2 3-------------DO 310 IIDX = 1 , 6
  1 2 3             T1(KIDX,LIDX,IIDX) = TIMSP * DERIV(TIMH,XN,KIDX,LIDX,IIDX)
  +-------------310 CONTINUE

(B)
  1----------S------DO 310 KIDX = 1 , M
  1 2--------S------DO 310 LIDX = 1 , N
  1 2 3-------------DO 310 IIDX = 1 , 6
  1 2 3             GO TO ( 3910 , 3920 , 3930 , 3940 , 3950 , 3960 ) ,IIDX
  1 2 3        3910 FUN =  XN(KIDX,LIDX,2)
  1 2 3             GO TO 3980
  1 2 3        3920 FUN = (  FIA(KIDX,LIDX,2)
  1 2 3            *    +    VRA(KIDX,LIDX,2)
  1 2 3            *    +    DWA(KIDX,LIDX,2)
  1 2 3            *    +    BKA(KIDX,LIDX,2)
  1 2 3            *    +    WGA(KIDX,LIDX,2)  ) / ROI(KIDX,LIDX)
  1 2 3             GO TO 3980
  1 2 3        3930 FUN =  XN(KIDX,LIDX,4)
  1 2 3             GO TO 3980
  1 2 3        3940 QQQQ = 0.D0
  1 2 3             IF( LIDX .EQ. N )     QQQQ = FTOP(KIDX)
  1 2 3             FUN = (  FIA(KIDX,LIDX,1)
  1 2 3            *    +    DWA(KIDX,LIDX,1)
  1 2 3            *    +    BKA(KIDX,LIDX,1) + QQQQ ) / AMS(KIDX,LIDX)
  1 2 3             GO TO 3980
  1 2 3        3950 FUN =  XN(KIDX,LIDX,6)
  1 2 3             GO TO 3980
  1 2 3        3960 FUN =-(  VRA(KIDX,LIDX,1)
  1 2 3            *    -    WGA(KIDX,LIDX,1)
  1 2 3            *    +    GSF(KIDX,LIDX)  + PPPP ) / AMS(KIDX,LIDX)
  1 2 3        3980 AC(KIDX,LIDX,IIDX) = FUN
  1 2 3             T1(KIDX,LIDX,IIDX) = TIMSP * FUN
  +-------------310 CONTINUE

(C)
  1----------S------DO 310 LIDX = 1 , N
  1 2--------V------DO 310 KIDX = 1 , M
  1 2        V        AC(KIDX,LIDX,1   ) = XN( KIDX , LIDX , 2 )
  1 2        V        AC(KIDX,LIDX,2   ) = ( FIA( KIDX , LIDX , 2 )
  1 2        V     *                        +VRA( KIDX , LIDX , 2 )
  1 2        V     *                        +DWA( KIDX , LIDX , 2 )
  1 2        V     *                        +BKA( KIDX , LIDX , 2 )
  1 2        V     *                        +WGA( KIDX , LIDX , 2 )
  1 2        V     *                         ) / ROI( KIDX , LIDX )
  1 2        V        AC(KIDX,LIDX,3   ) = XN( KIDX ,LIDX , 4 )
  1 2        V      QQQQ = 0.0D0
  1 2        V      IF( LIDX .EQ. N )  QQQQ = FTOP(KIDX)
  1 2        V        AC(KIDX,LIDX,4   ) = ( FIA( KIDX , LIDX , 1 )
  1 2        V     *                        +DWA( KIDX , LIDX , 1 )
  1 2        V     *                        +BKA( KIDX , LIDX , 1 )
  1 2        V     *         +QQQQ          ) / AMS( KIDX , LIDX )
  1 2        V        AC(KIDX,LIDX,5   ) = XN( KIDX ,LIDX , 6 )
  1 2        V        AC(KIDX,LIDX,6   ) = -( VRA( KIDX , LIDX , 1 )
  1 2        V     *                         -WGA( KIDX , LIDX , 1 )
  1 2        V     *                         +GSF( KIDX , LIDX )
  1 2        V     *          +PPPP          ) / AMS( KIDX , LIDX )
  1 2        V      T1(KIDX,LIDX, 1  ) = TIMSP*AC(KIDX,LIDX, 1  )
  1 2        V      T1(KIDX,LIDX, 2  ) = TIMSP*AC(KIDX,LIDX, 2  )
  1 2        V      T1(KIDX,LIDX, 3  ) = TIMSP*AC(KIDX,LIDX, 3  )
  1 2        V      T1(KIDX,LIDX, 4  ) = TIMSP*AC(KIDX,LIDX, 4  )
  1 2        V      T1(KIDX,LIDX, 5  ) = TIMSP*AC(KIDX,LIDX, 5  )
  1 2        V      T1(KIDX,LIDX, 6  ) = TIMSP*AC(KIDX,LIDX, 6  )
  +----------V--310 CONTINUE
```

Fig. 7 The inline expansion and the optimization of the function DERIV called by JULY 31.

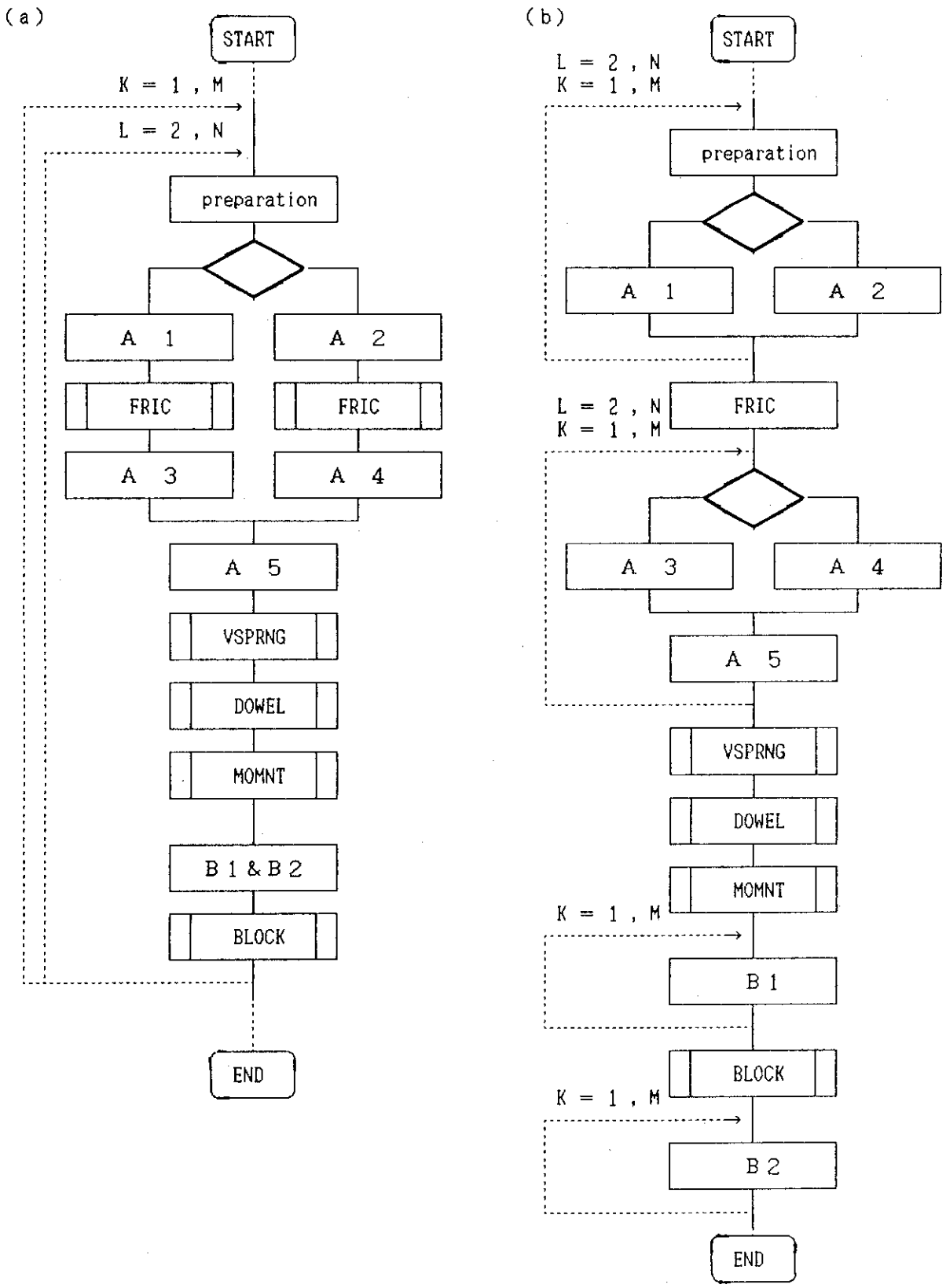


Fig. 8 The flow charts of the function FUN for original version in (a) and vectorized one in (b).
The velocity of block is calculated in A1 and A2. The force and moment acted on block are calculated in A3 and A4. At A5, the force at the boundary of block is accumulated. The collision between block and side fixed reflectors is calculated at B1 for the left reflector and B2 for the right one.

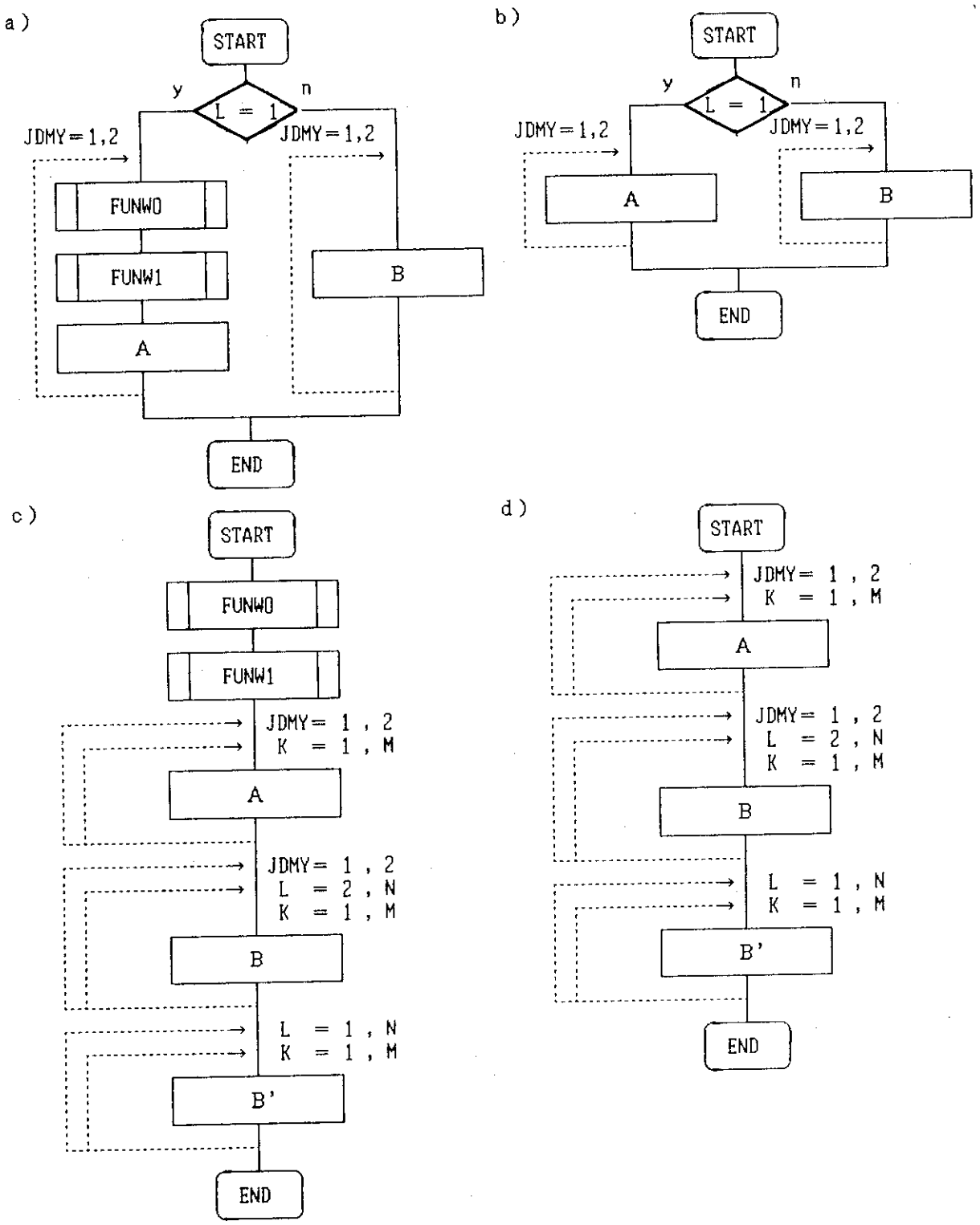


Fig. 9 The flow charts of the subroutines VSPRNG and DOWEL for original versions in (a) and (b) and vectorized ones in (c) and (d), respectively. The vertical force and its moment acted on the top and the bottom of a block are calculated in A for the bottom block and B for the others. The force and its moment acted on a block are summed up in B'.

```
(A)
 1------S------DO 1830 K = 1 , M
 1 2----V------DO 1830 L = 2 , N-1
 1 2    V      WGDMT¥(K,L) = 0.DO
 1 2           LDMY1        =  L + 1
 1 2 3--V------DO 1820 LDMY = LDMY1 , N
 1 2 3  V      WGDMT¥(K,L) =  WGDMT¥(K,L) + WG(K,LDMY)
 1 2 +--V-1820 CONTINUE
 1 2    V      WGDMB¥(K,L) =  WGDMT¥(K,L) + WG(K,L)
 1 2    V      WGA(K,L,1) =  WGDMT¥(K,L) - WGDMB¥(K,L)
 +------V-1830 CONTINUE


(B)
 1------V------DO 820 I = 1 , M*N
 1      V      K = MOD(I-1,M) + 1
 1      V      L = (I-1)/M + 1
 1      V      WGDMT¥(K,L) = 0.DO
 +------V--820 CONTINUE
 1------S------DO 830 L      =  1 , N-1
 1             LDMY1        =  L + 1
 1 2----S------DO  830 LDMY = LDMY1 , N
 1 2 3--V------DO  830 K    = 1 , M
 1 2 3  V      WGDMT¥(K,L) =  WGDMT¥(K,L) + WG(K,LDMY)
 +------V--830 CONTINUE


 1------V------DO 1835 I     = M+1 , M*N
 1      V      K = MOD(I-1,M) + 1
 1      V      L = (I-1)/M + 1
 1      V      WGDMB¥(K,L) =  WGDMT¥(K,L) + WG(K,L)
 1      V      WGA(K,L,1) =  WGDMT¥(K,L) - WGDMB¥(K,L)
 +------V-1835 CONTINUE
```

Fig. 10 An example of the optimal choice of the do-loop index for vectorization in the subroutine MOMNT.

```
(A)       ALPH          = ( ZRT - ZLT ) / ( ZRT - ZRB )
          IF(ALPH.LT.0.D0)                        GO TO 2880
          XI =((1.0D0-2.0D0*ALPH)*H(K,L)*SNMD + B(K,L)*CSMD - B(K,L))
         *             + X(K,L,3)
         *             - ( H(K+1,L)*SNR - B(K+1,L)*CSR + B(K+1,L))
         *             - X(K+1,L,3) - DLT(K,L)
    C
    C
    C
          IF(XI.LE.0.D0)                          GO TO 2880
          XIDF1=((1.0D0-2.0D0*ALPH)*H(K,L)*CSMD
         *       -B(K,L)*SNMD)*X(K,L,2)+X(K,L,4)
         *           -(H(K+1,L)*CSR+B(K+1,L)*SNR)*X(K+1,L,2)
         *           - X(K+1,L,4)
          SPKBK=SPKBKA(K,L,1)+SPKBKA(K,L,2)*XI+SPKBKA(K,L,3)*XI*XI
          SPCBK=SPCBKA(K,L,1)+SPCBKA(K,L,2)*XIDF1
         *                   +SPCBKA(K,L,3)*XIDF1*XIDF1
          BKFRCT = -( SPKBK*XI + SPCBK*XIDF1 )
          IF( BKFRCT .GE. 0.0D0 )                         GO TO 2880
          BKMRCT = BKFRCT * ( (1.0D0-2.0D0*ALPH)*H(K,L)*CSMD - B(K,L)*SNMD)
          CFA(K+1,L,5) = CFA(K+1,L,5) - BKFRCT
          BKA(K+1,L,1) = BKA(K+1,L,1) - BKFRCT
          BKA(K+1,L,2) = BKA(K+1,L,2) - BKFRCT
         1      *(H(K+1,L)*CSR+B(K+1,L)*SNR)
          GO TO 2890
     2880 CONTINUE
          BKFRCT = 0.0D0
          BKMRCT = 0.0D0
     2890 CONTINUE
          CFA(K,L,4)     = CFA(K,L,4) + BKFRCT
          BKA(K,L,1)     = BKA(K,L,1) + BKFRCT
          BKA(K,L,2)     = BKA(K,L,2) + BKMRCT
     2800 CONTINUE
```

```
(B) 1---------------IF( M .NE. 1 )      THEN
    1          *VOCL LOOP,NOVREC
    1 2------V------DO 3000 KL = 1 , (M-1)*N
    1 2      V      K = MOD( KL-1,M-1 ) + 1
    1 2      V      L = (KL-1)/(M-1) + 1
    1 2      V            .
    1 2      V            .
    1 2      V            .
    1 2 3----V------IF(  (ZRT.GE.ZLT .AND. ZRT.GT.ZRB)
    1 2 3    V     & .OR.(ZRT.LE.ZLT .AND. ZRT.LT.ZRB) ) THEN
    1 2 3    V         IFLG¥2(K,L) = 1
    1 2 3    V         ALPH = ( ZRT - ZLT ) / ( ZRT - ZRB )
    1 2 3    V         ALPH¥2(K,L) = ALPH
    1 2 3    V         XI¥2(K,L)=((1.0D0-2.0D0*ALPH)*H(K,L)*SNMD
    1 2 3    V     *             + B(K,L)*CSMD - B(K,L)) + X(K,L,3)
    1 2 3    V     *            - ( H(K+1,L)*SNR - B(K+1,L)*CSR   + B(K+1,L))
    1 2 3    V     *            - X(K+1,L,3) - DLT(K,L)
    1 2 +-----------ENDIF
    1 2      V            .
    1 2      V            .
    1 2      V            .
    1 +------V-3000 CONTINUE
    1          *VOCL LOOP,SCALAR
    1 2-------------DO 8000 K = 1 , M-1
    1 2 3-----------DO 8000 L = 1 , N
    1 2 3                 .
    1 2 3                 .
    1 2 3                 .
    1 2 3 4-------------IF( IFLG¥2(K,L) .EQ. 1 ) THEN
    1 2 3 4               XI = XI¥2(K,L)
    1 2 3 4 5-------------IF(XI.GT.0.D0) THEN
    1 2 3 4 5                 ALPH = ALPH¥2(K,L)
    1 2 3 4 5                 XIDF1=((1.0D0-2.0D0*ALPH)*H(K,L)*CST¥(K,L)
    1 2 3 4 5      *                          -B(K,L)*SNT¥(K,L))*X(K,L,2)+X(K,L,4)
    1 2 3 4 5      *           -(H(K+1,L)*CST¥(K+1,L)+B(K+1,L)*SNT¥(K+1,L))*X(K+1,L,2)
    1 2 3 4 5      *           - X(K+1,L,4)
    1 2 3 4 5                 SPKBK=SPKBKA(K,L,1)+SPKBKA(K,L,2)*XI+SPKBKA(K,L,3)*XI*XI
    1 2 3 4 5                 SPCBK=SPCBKA(K,L,1)+SPCBKA(K,L,2)*XIDF1
    1 2 3 4 5      *                        +SPCBKA(K,L,3)*XIDF1*XIDF1
    1 2 3 4 5                 BKFRCT = -( SPKBK*XI + SPCBK*XIDF1 )
    1 2 3 4 5 6---------------IF( BKFRCT .LT. 0.0D0 )  THEN
    1 2 3 4 5 6                  BKMRCT=BKFRCT*((1.0D0-2.0D0*ALPH)*H(K,L)*CST¥(K,L)
    1 2 3 4 5 6    *                                        -B(K,L)*SNT¥(K,L))
    1 2 3 4 5 6                  CFA(K+1,L,5) = CFA(K+1,L,5) - BKFRCT
    1 2 3 4 5 6                  BKA(K+1,L,1) = BKA(K+1,L,1) - BKFRCT
    1 2 3 4 5 6                  BKA(K+1,L,2) = BKA(K+1,L,2) - BKFRCT
    1 2 3 4 5 6    *                      *(H(K+1,L)*CST¥(K+1,L)+B(K+1,L)*SNT¥(K+1,L))
    1 2 3 4 5 6                  CFA(K,L,4) = CFA(K,L,4) + BKFRCT
    1 2 3 4 5 6                  BKA(K,L,1) = BKA(K,L,1) + BKFRCT
    1 2 3 4 5 6                  BKA(K,L,2) = BKA(K,L,2) + BKMRCT
    1 2 3 4 5 +---------------ENDIF
    1 2 3 4 +--------------ENDIF
    1 2 3 +-------------ENDIF
    1 2 3                 .
    1 2 3                 .
    1 2 3                 .
    1 +--------8000 CONTINUE
    +-------------------ENDIF
```

Fig. 11 An example of removing the part with short vector length from the vectorized do-loop in the subroutine BLOCK.

```
(A)       SUBROUTINE JULY31(
         *  WG          ,AMS        ,ROI        ,H          ,B
         * ,D           ,A          ,DLT        ,DLTDWL     ,DLTDWR
         * ,SPKVRA      ,SPCVRA     ,SPKDWA     ,SPCDWA     ,SPKBKA
         * ,SPCBKA      ,GSF        ,CXX        ,FI1        ,FI2
         * ,FIA         ,VRA        ,DWA        ,BKA        ,WGA
         * ,X           ,XN         ,AC         ,Q          ,T1
         * ,Z           ,M          ,N          ,MDMY1      ,MDMY2
         * ,TIM         ,AUX        ,IAG        ,IVSG       ,IFSG
         * ,IMG980      ,CFA        ,DWF
         * ,SNT¥        ,CST¥       ,ALF¥       ,V¥         ,FIFT¥
         * ,CKSU¥       ,FIMB¥      ,CXCX¥      ,FIFB¥      ,FIMT¥
         * ,WORK¥       ,IWORK¥     )
                          .
                          .
                          .
    C¥....WORK AREA
          DIMENSION
         *  SNT¥(M,N)   ,CST¥(M,N)   ,ALF¥(M,N)   ,V¥(M,N)    ,FIFT¥(M,N)
         * ,CKSU¥(M,N)  ,FIMB¥(M,N)  ,CXCX¥(M,N)  ,FIFB¥(M,N) ,FIMT¥(M,N)
         * ,WORK¥(16*M*N)     ,IWORK¥(4*M*N)
                          .
                          .
                          .
          RETURN
          END

(B)       SUBROUTINE JULY31(
         *  WG          ,AMS        ,ROI        ,H          ,B
         * ,D           ,A          ,DLT        ,DLTDWL     ,DLTDWR
         * ,SPKVRA      ,SPCVRA     ,SPKDWA     ,SPCDWA     ,SPKBKA
         * ,SPCBKA      ,GSF        ,CXX        ,FI1        ,FI2
         * ,FIA         ,VRA        ,DWA        ,BKA        ,WGA
         * ,X           ,XN         ,AC         ,Q          ,T1
         * ,Z           ,M          ,N          ,MDMY1      ,MDMY2
         * ,TIM         ,AUX        ,IAG        ,IVSG       ,IFSG
         * ,IMG980      ,CFA        ,DWF        )
                          .
                          .
                          .
    C¥....WORK AREA
          DIMENSION
         *  SNT¥(M,N)   ,CST¥(M,N)   ,ALF¥(M,N)   ,V¥(M,N)    ,FIFT¥(M,N)
         * ,CKSU¥(M,N)  ,FIMB¥(M,N)  ,CXCX¥(M,N)  ,FIFB¥(M,N) ,FIMT¥(M,N)
         * ,WORK¥(16*M*N)     ,IWORK¥(4*M*N)
                          .
                          .
                          .
          RETURN
          ENTRY SETJLY(
         *  SNT¥        ,CST¥       ,ALF¥       ,V¥         ,FIFT¥
         * ,CKSU¥       ,FIMB¥      ,CXCX¥      ,FIFB¥      ,FIMT¥
         * ,WORK¥       ,IWORK¥     ,M    ,N )
          DUMMY = SETFUN(
         *  SNT¥        ,CST¥       ,ALF¥       ,V¥         ,FIFT¥
         * ,CKSU¥       ,FIMB¥      ,CXCX¥      ,FIFB¥      ,FIMT¥
         * ,WORK¥       ,IWORK¥     ,M    ,N)
          RETURN
          END
```

Fig. 12 An example of the reduction of dummy arguments for the work area in the subroutine JULY31.

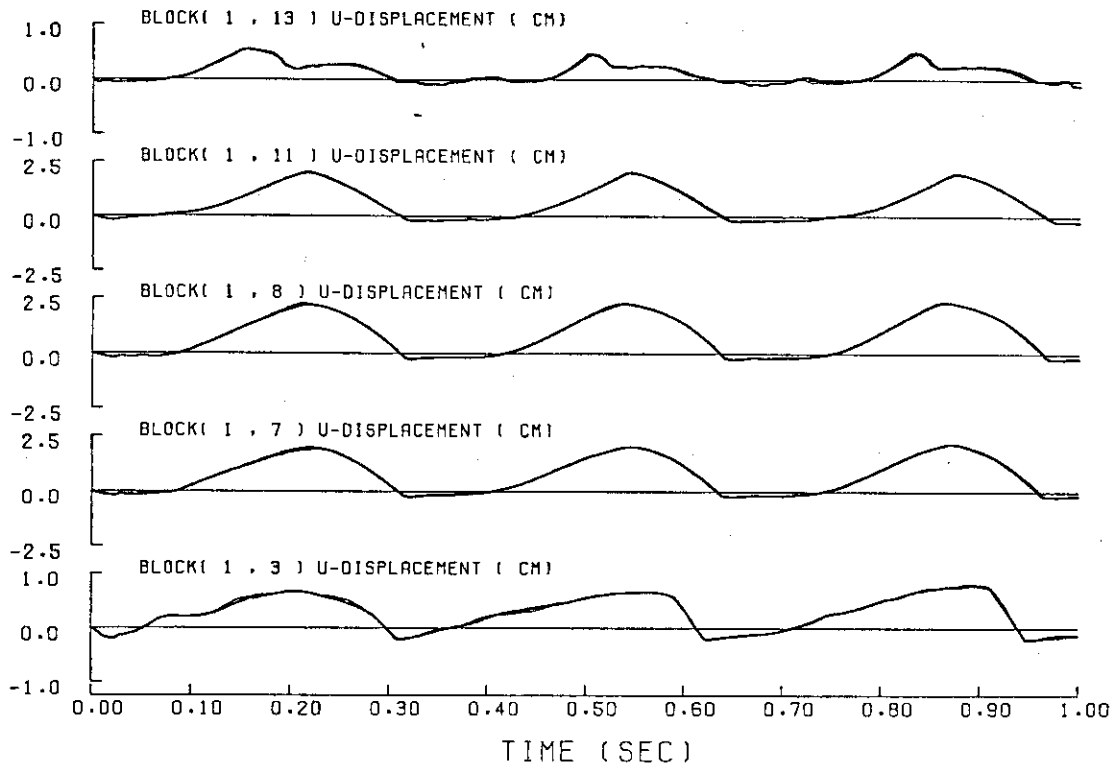
```
(A)                SNT     = DSIN( X(K,1,1) )
                   CST     = DCOS( X(K,1,1) )
                           .
                           .
                           .
                   ALF   = X(K,L,1)
                   IF(IGO.EQ.0.AND.ALF.LT.0.D0)          IAG(K,L) = 1
                   IF(IAG(K,L).EQ.1)                     GO TO 240
                   V = X(K,1,4) - ( H(K,1)*CST + B(K,1)*SNT )*X(K,1,2)
                 *     - XBP(2)
                           .
                           .
                           .
(B)                CALL SINCOS( SNT¥  ,CST¥  ,X      ,M      ,N  ,MDMY2)
   1----V------DO 5000 K = 1 , M
   1    V         ALF¥(K) = X(K,L,1)
   1    V       IF( IGO.EQ.0 .AND. ALF¥( K ).LT.0 )      IAG( K ) = 1
   1 2--V------IF( IAG( K ) .NE. 1 )                     THEN
   1 2  V       V¥(K) = X(K,1,4)-(H(K,1)*CST¥(K,1)+B(K,1)*SNT¥(K,1))*X(K,1,2)
   1 2  V      *           - XBP(2)
   1 2  V               .
   1 2  V               .
   1 2  V               .
   1 +--V------ENDIF
   1    V               .
   1    V               .
   1    V               .
   +----V-5000 CONTINUE
```

Fig. 13 An example of the reduction of the function SIN and COS in function FUN.

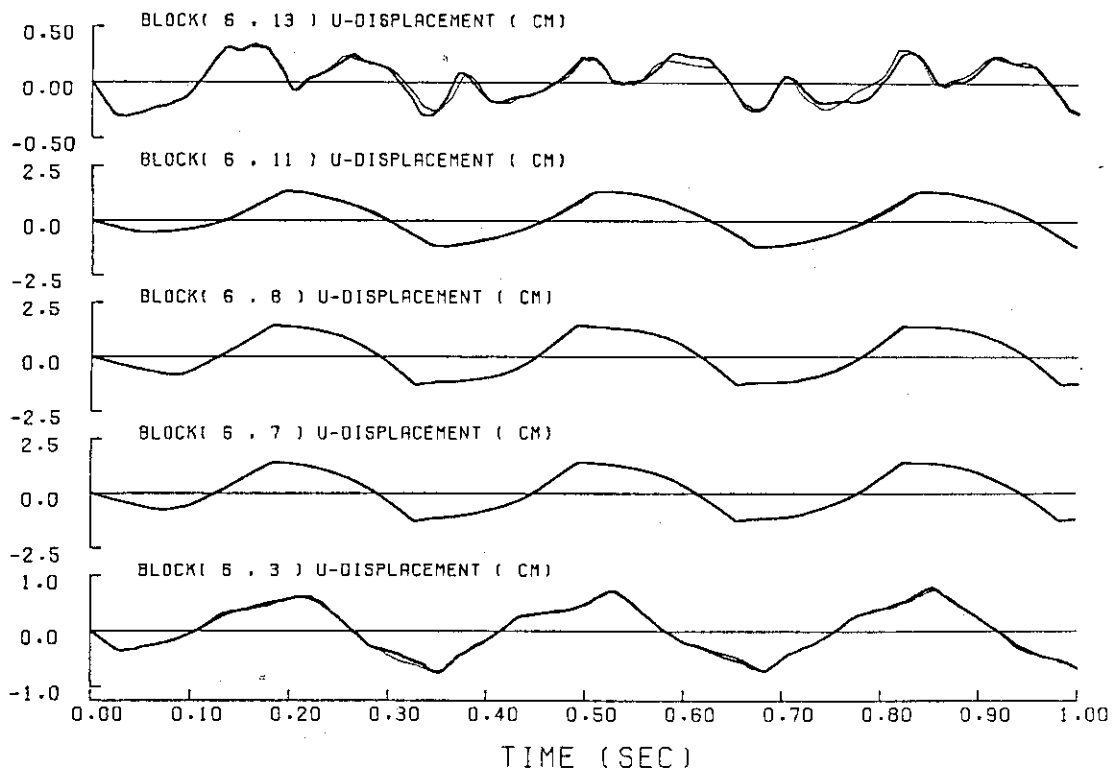
```
(A)                SUBROUTINE SINCOS( SNT¥ , CST¥ , X , M , N , MDMY2 )
                   IMPLICIT REAL*8 ( A-H , O-Z )
                   DIMENSION
                 D    SNT¥( M,N ),     CST¥( M,N ) ,      X( MDMY2,N,6 )
   1----S------DO 100 J = 1 , N
   12---V------DO 100 I = 1 , M
   12   V         SNT¥( I,J ) = DSIN( X( I,J,1 ) )
   12   V         CST¥( I,J ) = DCOS( X( I,J,1 ) )
   +----V--100 CONTINUE
                   RETURN
                   END

(B)                SUBROUTINE SINCOS( SNT¥ , CST¥ , X , M , N , MDMY2 )
                   IMPLICIT REAL*8 ( A-H , O-Z )
                   DIMENSION
                 D    SNT¥( M*N ),     CST¥( M*N ) ,      X( MDMY2,N,6 )
   1----V------DO 100 K = 1 , M*N
   1    V         I = MOD( K-1,M )+1
   1    V         J = (K-1)/M + 1
   1    V         SNT¥( K ) = DSIN( X( I,J,1 ) )
   1    V         CST¥( K ) = DCOS( X( I,J,1 ) )
   +----V--100 CONTINUE
                   RETURN
                   END
```

Fig. 14 An example of the modification of nested do-loops to single do-loop in subroutine SINCOS.

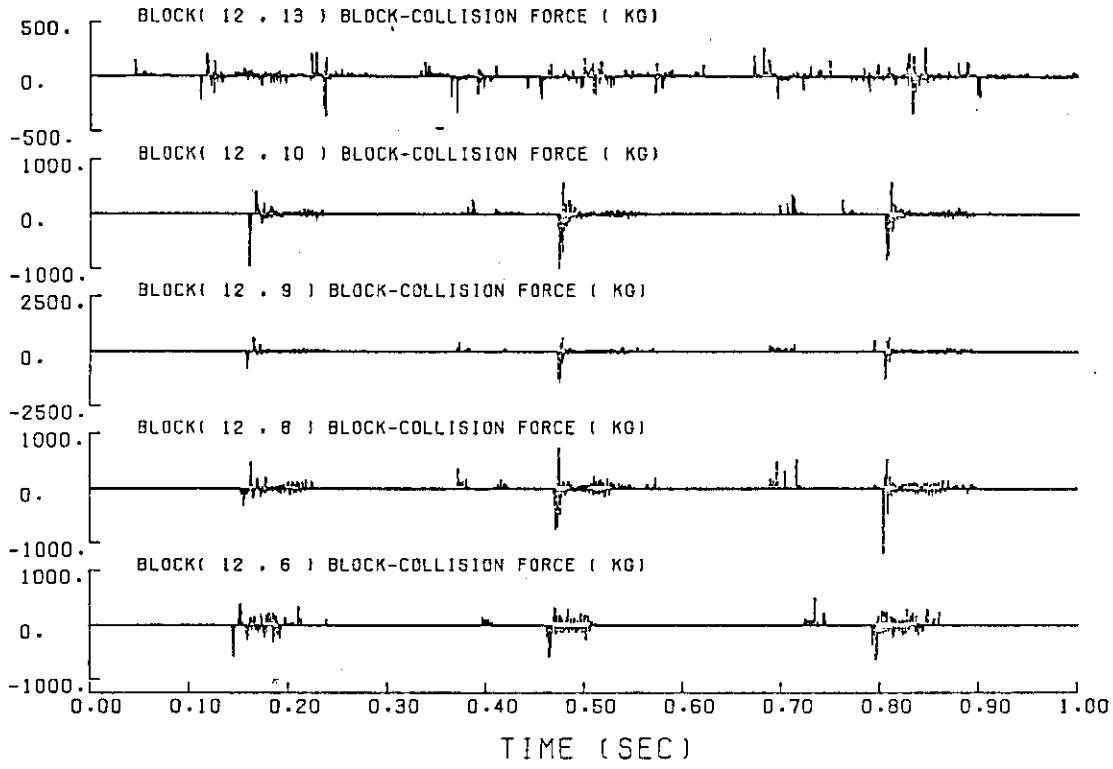


(a) Displacement response curve (1)

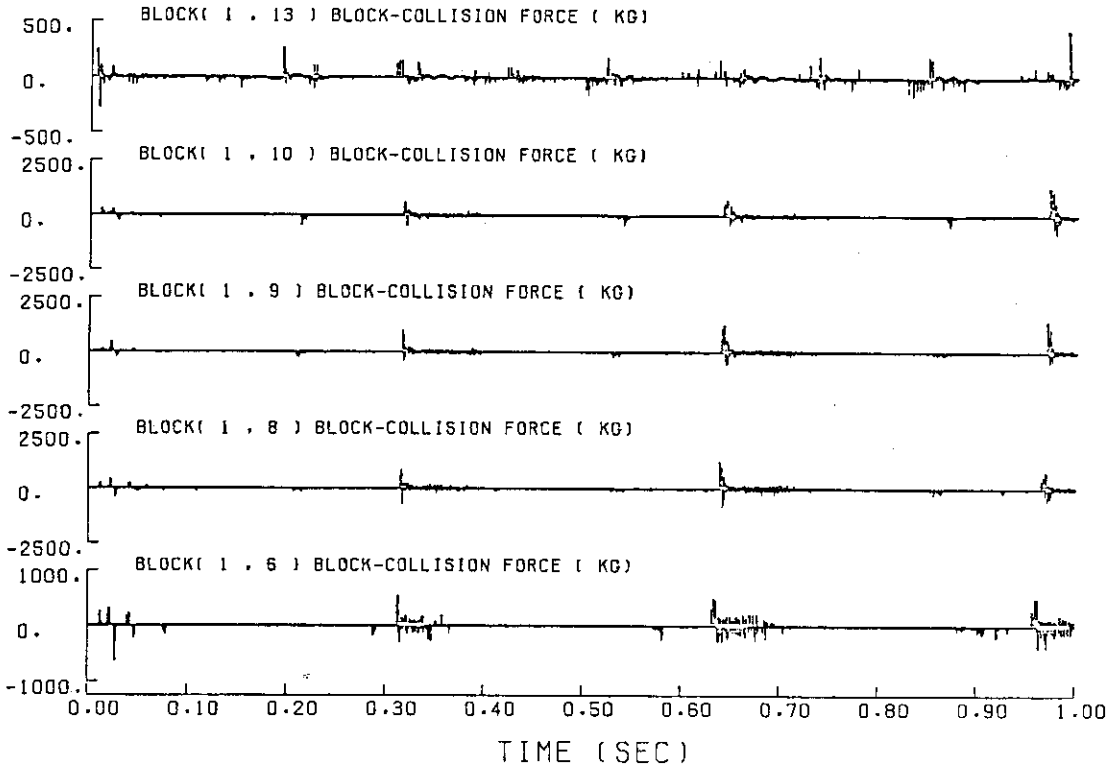


(b) Displacement response curve (2)

Fig. 15 Comparison between calculated results in the scalar mode and in the vector mode. This displacement response curves in (a) and (b) and the force response curves in (c) and (d) are depicted.

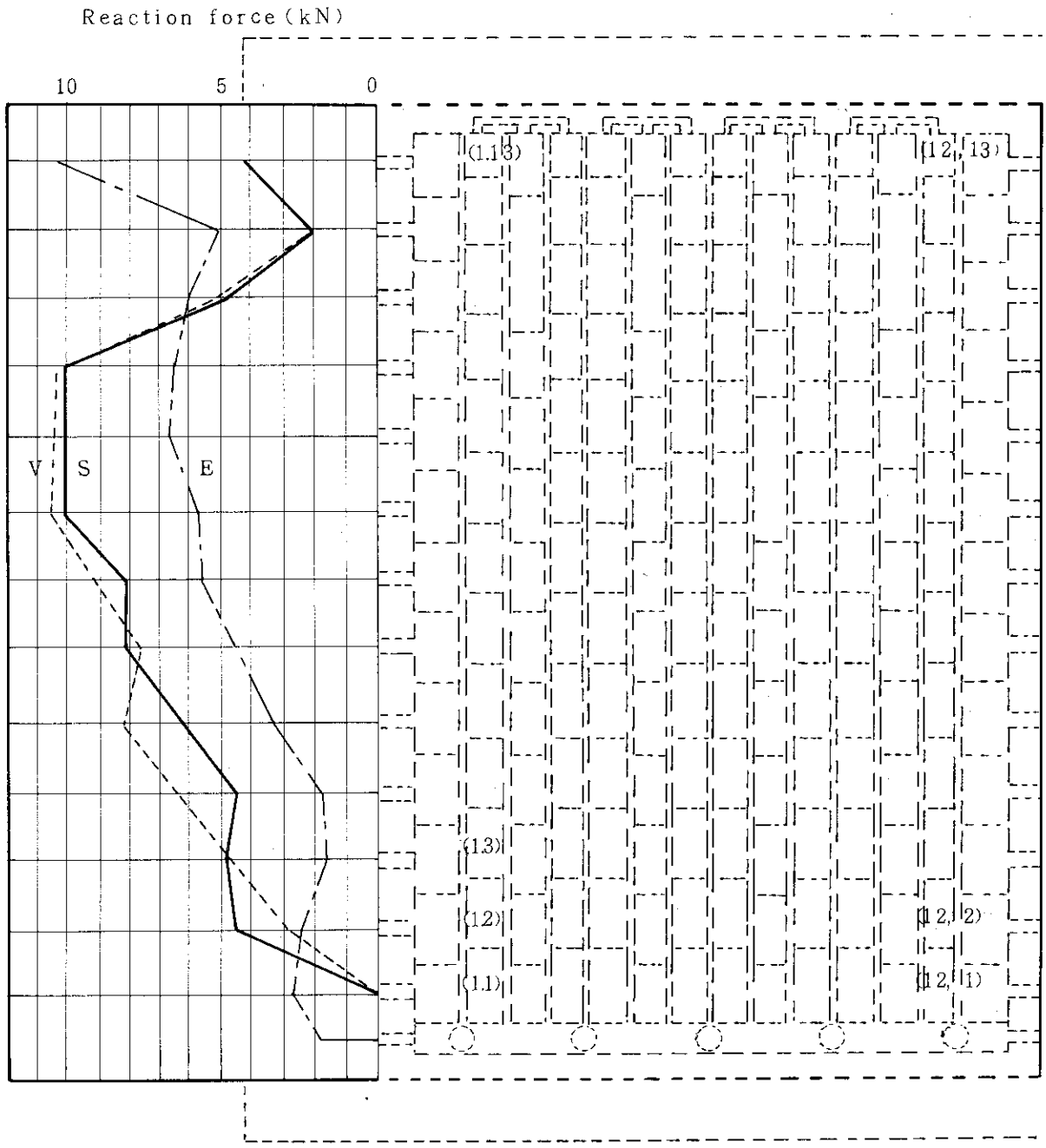


(c) Force response curve (1)



(d) Force response curve (2)

Fig. 15 Comparison between calculated results in the scalar mode and in the vector mode. This displacement response curves in (a) and (b) and the force response curves in (c) and (d) are depicted.



E : Experiment

V : Vector processor

S : Scalar processor

Fig. 16 Comparison among experimental reaction force, calculated one in the scalar mode and calculated one in the vector mode.