FORTRAN77／SXにおける高速化技法

片山　博*，河原久美子**，大中幸三郎***

1. はじめに

ＳＸシステムは、システム全体の制御をつかさどる制御プロセッサ（以下ＣＰと略す）と、利用者プログラムの高速実行をつかさどる演算プロセッサ（以下ＡＰと略す）の二つのプロセッサから構成されています[1,2]。ＡＰは、科学技術計算の高速実行のために、ベクトルデータ間の演算を超高速で処理する命令（ベクトル命令）を多数備えています。ＡＰの性能を十分に引き出すためには、通常の命令（スカラ命令）のかわりに、できるだけベクトル命令を使用する、すなわちベクトル化する必要があります。ＳＸシステムのFORTRAN処理系には、ＣＰ用の目的プログラムを生成し、主としてデバッグのために用いられるFORTRAN77と、ＡＰ用の目的プログラムを生成し、ＳＸ本来の性能を引き出すためのFORTRAN77/SXの二つが用意されています。利用者のプログラムは、このFORTRAN77/SXコンパイラ[1]の自動ベクトル化機能により自動的にベクトル化され、高速実行されます。しかし、利用者が少しプログラムを手直しすることにより、ベクトル化の範囲が拡大され、あるいは、生成されるベクトル命令の効率が高められ、そのプログラムをさらに高速化できることがあります。

本稿では、まずFORTRAN77/SXコンパイラの自動ベクトル化機能について述べたあと、利用者が考慮すべきFORTRANプログラムの高速化技法について述べます。

2. 自動ベクトル化機能

コンパイラが原始プログラムを解析して、ベクトル命令で実行可能な部分を自動的に検出し、その部分をベクトル化することを、自動ベクトル化という。コンパイラは、プログラム中のDOループを対象に自動ベクトル化を行う。本節では、DOループがベクトル化されるための条件、およびベクトル化に関するいくつかの機能について述べる。

2.1　自動ベクトル化の対象

表1に、FORTRAN77/SXの自動ベクトル化の対象を示す。

＊　日本電気（株）基本ソフトウェア開発本部　　＊＊　関西日本電気ソフトウェア（株）
＊＊＊　大阪大学大型計算機センター研究開発部

表1　自動ベクトル化の対象

項　　目	ベクトル化の対象	備　　考
ベクトル化の 対象範囲	最深DOループ	拡張ベクトル化機能により 多重ループもベクトル化可 能（ **2.3**参照）
対象となる データの型	標準（４バイト）の整数型 実数型 倍精度実数型 複素数型 倍精度複素数型 標準（４バイト）の論理型	以下の型は対象外 ２バイトの整数型 ４倍精度実数型 ４倍精度複素数型 １バイトの論理型 文字型 日本語型
対象となる文	算術代入文 論理代入文 CONTINUE文 算術IF文 論理IF文 ブロックIF文 ELSE　IF文 ELSE文 END　IF文 単純GO　TO文 ただし、IFの入れ子は三重までが対象になる	以下の文は対象外 文字代入文 ホラリス代入文 日本語代入文 文番号代入文 （ASSIGN文） CALL文 計算形GO　TO文 割当て形GO　TO文 入出力文 STOP文 PAUSE文 RETURN文
対象となる演算	代入 加減乗除算 べき算 型変換 関係演算 論理演算 組込み関数の引用* 文関数の引用** マクロ演算	以下の演算は対象外 外部関数の引用 連結演算 ＊引数の型および結果の型が対象外のもの、および乱数発生の組込み関数は除く ＊＊右辺が条件を満たしていないものは除く
対象となるデータ	以下のデータが対象 定数および定数名 変数 配列要素	
対象となる ベクトル	連続・等間隔ベクトル 間接指標ベクトル 指標変数*	＊式の中に陽に現れたDO変数など

以下に、表1について補足を行う。

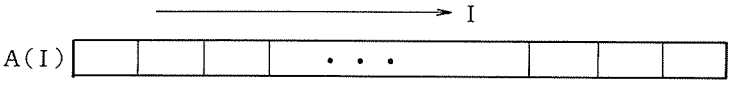
(1) 最深DOループ

その内部に他のDOループを含まないDOループ、すなわち、最内側のDOループを最深DOループという。

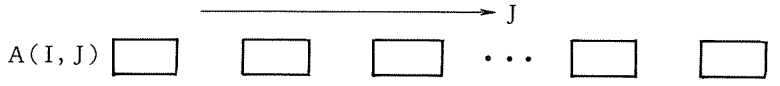
(2) 添字の形式とベクトルの種類

配列要素が線型添字あるいは非線型添字をもつとき、その配列要素はベクトルデータとして扱われる。また、指標変数も、ベクトルデータとして扱われる。ベクトルデータは、その要素の並び方により、連続ベクトル、等間隔ベクトルおよび間接指標ベクトルの三種類に分類される(図1参照)。

(a) 連続ベクトル



(b) 等間隔ベクトル



(c) 間接指標ベクトル

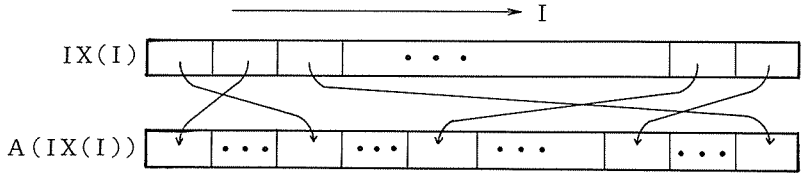


図1 ベクトルの種類

(a) 線型添字

添字式が次の条件をすべて満たすものを線型添字という。線型添字をもつベクトルは、連続ベクトルあるいは等間隔ベクトルとして扱われる。

- 整数型の式である
- 演算子として＋，－，＊および底が指標変数でなく指数が符号なし整定数のべき算のみを含む
- 括弧を含まない
- 変数および定数のみを含む。その変数は指標変数あるいは相対定数のいずれかである。た

だし、相対定数とは、次のいずれかのものをいう。

ループ中で引用のみの変数あるいは配列要素

実引数に定数あるいは相対定数のみを含む組込み関数

・指標変数は各次元に2個以上出現しない

(b) 非線型添字

上記の条件を満たさず、かつ添字の値がループの繰返しにより変化するものを非線型添字という。非線型添字をもつベクトルは、間接指標ベクトルとして扱われる。

(c) 指標変数

次のいずれかの条件を満たす整変数を指標変数という。

・ベクトル化対象DOループのDO変数

・ループの繰返しごとに一定値ずつ値が増加あるいは減少する整変数で、ループ内の一カ所でのみ定義されているもの

(3) マクロ演算

内積のように、複数の演算によって、より大きな演算を表わしているものを、マクロ演算という。コンパイラは、以下のマクロ演算を認識し、ベクトル化する。

総和	S＝S±A(I)
累積	S＝S＊A(I)
内積	S＝S±A(I)＊B(I)
最大値	S＝max(S, A(I))
最小値	S＝min(S, A(I))
漸化式	X(I)＝A(I)±X(I−1)＊B(I)
	X(I)＝(A(I)±X(I−1))＊B(I)
	X(I)＝A(I)±X(I−1)
	X(I)＝A(I)＊X(I−1)

ただし、S、A(I)、B(I)、X(I)は以下のものとする。

Sは変数名または配列要素名

A(I)、B(I)はベクトルである変数名、配列要素名または算術式

X(I)は配列要素名

2.2 ベクトル化の基本条件

以下に、DOループがベクトル化されるための基本的な条件を述べる。ただし、条件を満たさない場合でも、拡張ベクトル化機能(**2.3**参照)により、その一部あるいは全体がベクトル化される

ことがある。

(1) DOループが満たすべき条件

DOループは、表1のベクトル化の対象のもののみを含んでいなければならない。さらに、次の条件を満たしていなければならない。

- ループ外への飛び出しを含まない
- 逆方向分岐を含まない
- DO変数やDOのパラメータは整数型である
- 繰返し数をコンパイル時に確定できる場合、その値は5以上である

(2) 変数および配列要素が満たすべき条件

ベクトル化の対象となる変数および配列要素は、次の条件を満たしていなければならない。

(a) ベクトルデータとして扱われる配列要素は、その型に応じ、下記のバイト単位の境界に整列されていなければならない。

- 整数型、実数型、複素数型、論理型・・・4の倍数番地
- 倍精度実数型、倍精度複素数型　・・・8の倍数番地

なお、ALCオプションが有効な場合には、通常この条件は満たされている。

(b) 同一変数がループ中に二度以上出現する場合、いずれも引用のみであるか、あるいはループ内のすべての実行経路において、定義が引用に先行していなければならない。ただし、指標変数の場合、および、内積・総和などのマクロ演算のベクトル化が適用できる場合は例外である。また、EQUIVALENCE文で結合されているものが二度以上出現する場合には、いずれも引用のみでなければならない。

(c) 配列の定義・引用関係がベクトル化に適合していなければならない。

例　ベクトル化に適合しない例

```
      DO  10  I=1,3
        A(I)=3.0
10      B(I)=A(I+1)
```

ベクトル化しない場合の実行順序	ベクトル化した場合の実行順序
A(1)=3.0	A(1)=3.0
B(1)=A(2)	A(2)=3.0
A(2)=3.0	A(3)=3.0
B(2)=A(3)	B(1)=A(2)
A(3)=3.0	B(2)=A(3)
B(3)=A(4)	B(3)=A(4)

ベクトル化に適合するための条件は次のとおりである。

(イ) ループの中では引用のみ、すなわち、どこでも定義されていない。

(ロ) 定義と定義、あるいは定義と引用との間で要素の重なりがない。

```
例      DO  10  I=1, N, 2
                    =A(I)･･･A(1),A(3),A(5)
  10    A(I+1)=       ･･･A(2),A(4),A(6)
```

(ハ) 同一要素に対する定義と定義、あるいは定義と引用の原始プログラム上での出現順序と、ループの繰返しの進行における出現順序とが一致している。

```
例      DO  10  I=1, N          A(3)に着目すると、第1の文では2回目の、
                =A(I+1)         第2の文では3回目の繰返しにおいて、参照
  10    A(I) =                  されている。
```

この条件に関するコンパイラの判定方法は次のとおりである。

(i) 線型添字をもつ配列要素あるいは添字の値が一定の配列要素の場合には、以下の判定を行っている。

- 繰返しの進行に伴い添字の値が増加する配列要素同士の場合、繰返しの任意の回次において、先行する文に現われる配列要素の添字の値は、後続の文に現われる配列要素の添字の値と等しいか、それより大きい。ただし、代入文の右辺は、左辺よりも先行するとみなす。
- 添字の値が減少する配列要素同士の場合は、その逆である。
- 添字の値が増加する要素と減少する要素の間では、それらの大小関係が繰返しの進行により変化しない。

(ii) 非線型添字をもつ配列要素が定義あるいは引用の一方あるいは両方に含まれる場合、条件は満たされないものとする。

(iii) EQUIVALENCE文により結合された要素のいずれか一つが定義されているならば、条件は満たされないものとする。

(iv) 条件を満たすか否かをコンパイラが判断できない場合、条件を満たさないものとする。

なお、(ii)～(iv)の場合、後述のベクトル化指示行により、定義・引用関係がベクトル化に適合することをコンパイラに指示することができる(**2.5**参照)。また、条件を満たさない場合でも、漸化式命令が適用できる場合や、拡張ベクトル化機能により、ベクトル化することもある。

2.3 拡張ベクトル化機能

コンパイラは、前節で述べた基本条件を満たしていないDOループに対して、条件を満足しない

部分の前後でループを分割したり、条件を満足するようにループを変形したりすることにより、ループの一部分あるいは全体をベクトル化することがある。また、ベクトル化可能なDOループに対して、生成されるベクトル命令の効率を向上させるために、その一部を変形したりすることもある。これらの機能を拡張ベクトル化機能という。**表2**に、その概要を示す。

表2　拡張ベクトル化機能

項　　目	機能概要および例	備　　考

最大・最小要素を求めるループ	最大値・最小値あるいはその指標を求めるループを認識し、ベクトル化する
	DO 10 I=1, N
	IF(AMX.LT.A(I))THEN
	AMX=A(I)
	MXIDX=I
	END IF
	10 CONTINUE	

飛び出しを含むループ	探索ループなど、飛び出しを含むループも、条件を満たせばベクトル化する DO 10 I=1, N IF(IA(I).EQ.K)GO TO 20 10 CONTINUE 20 CONTINUE	[主な条件] ・飛び出し点は一カ所かつ繰返しに依存 ・飛び出しより前には配列要素の定義はない
ベクトル化を阻害する要素を含むループ	その前後でループを分割し、ベクトル化可能な部分をベクトル化する DO 10 I=1, N X=A(I)+B(I) Y=C(I)*D(I) WRITE(6,100) X,Y 10 CONTINUE ⇩ DO 101 I=1, N WX(I)=A(I)+B(I)　ベクトル化 WY(I)=C(I)*D(I) 101 CONTINUE DO 102 I=1, N WRITE(6,100)WX(I),WY(I) 102 CONTINUE	[主な条件] ・ループは次の文を含まない ループ外への分岐 STOP文 RETURN文 ・次の手続き呼び出しを含まない 外部関数 外部サブルーチン [注意] 分割点をまたがって定義・参照される変数は作業用ベクトル化される （例のWX, WY）

項　　目	機能概要および例	備　　考

多重ループ
〔この機能は、61年の秋頃利用可になる予定〕
	(1)　密な多重ループの一重化
	可能なら一重ループ化し、その後ベクトル化する
	`DO 10 J=1, N`
	`  DO 10 I=1, M`
	`10  A(I,J)=B(I,J)+C(I,J)`
	⇩
	`Ak = Bk + Ck`
	`(k=1, 2, ・・・, N*M)`
		〔主な条件〕
		・対象となるＤＯ変数は、配列要素の添字の左端から連続して同じ形式かつ順番で現われる
		・対象となるＤＯ変数を含む添字式は、対応する次元の端から端まで動く
		〔効果〕
		・ループ長の増加
		・連続ベクトル化による参照効率の向上

	(2)　密な多重ループの入れ換え
	有効と判断される場合、外側のいずれかのループと最深ループを入れ換える
	`DO 10 I=1, 1000`
	`  DO 10 J=1, 10`
	`10  A(I,J)=B(I,J)+C(I,J)`
	⇩
	`DO 10 J=1, 10`
	`  DO 10 I=1, 1000`
	`10  A(I,J)=B(I,J)+C(I,J)`
		〔主な条件〕
		・定義・引用関係が正しく保たれる
		〔効果〕
		・ベクトル化不可条件の除去
		・ループ長の増加
		・連続ベクトル化による参照効率の向上

	(3)　外側ループのベクトル化
	可能なら、最深ループの前後でループを分割し、それぞれの部分をベクトル化する
	`DO 10 I=1, N`
	`  A(I)=1.0`
	`  DO 20 J=1, M`
	`  X(J,I)=T*Z(J,I)`
	`20  CONTINUE`
	`10 CONTINUE`
	⇩
	`DO 101 I=1, N`
	`  A(I)=1.0`　　ベクトル化
	`101 CONTINUE`
	`DO 102 I=1, N`
	`  DO 20 J=1, M`　　密な多重ループとしてベクトル化
	`  X(J,I)=T*Z(J,I)`
	`20  CONTINUE`
	`102 CONTINUE`
		〔主な条件〕
		・定義・引用関係が正しく保たれる

項　　目	機能概要および例	備　　考
ベクトル化に適合しない配列の参照を含むループ 〔この機能は、61年の秋頃利用可になる予定〕	可能なら、文の入れ換えまたは作業用ベクトルの使用により、適合化する (a)　文の入れ換え DO 10 I=1, N A(I)=B(I)*C(I) 10 E(I)=1.0/A(I+1) ⇩ DO 10 I=1, N E(I)=1.0/A(I+1) 10 A(I)=B(I)*C(I) (b)　作業用ベクトルの使用 DO 10 I=1, N A(I)=B(I)*C(I) B(I)=T*B(I) 10 E(I)=B(I)+A(I+1) ⇩ DO 10 I=1, N W(I)=A(I+1)　　W(I): A(I)=B(I)*C(I)　作業用ベクトル B(I)=T*B(I) 10 E(I)=B(I)+W(I)	
外部関数の引用を含むループ	利用者の指示により疑似ベクトル関数化してベクトル化 *VDIR ELEMENTAL(PHI) DO 10 I=1, N 10 A(I)=B(I)+PHI(SQRT(X(I)) ここで、PHIは、渡された引数にのみ依存して関数値が定まり、かつ関数値を返すこと以外の副作用をもたないものとする	ベクトル化指示行による指示が必要

その他	可能なら、ループを変形し、ベクトル化の効率を向上させる
	DO 10 I=1, N
	IF(I.EQ.1)THEN
	A(I)=0.0
	ELSE
	A(I)=B(I)+T*C(I)
	END IF
	10 CONTINUE
	⇩
	A(I)=0.0
	DO 10 I=2, N
	A(I)=B(I)+T*C(I)
	10 CONTINUE	

2.4 VECTORオプション

自動ベクトル化機能は、翻訳時のオプションVECTOR（既定値）が指定されている場合に有効になる。ただし、NOVECTORが有効な場合でも、後述のベクトル化指示行によりベクトル化が指示されているループは、ベクトル化の対象となる。このオプションの形式およびパラメータの意味については、本号の別稿[4]およびマニュアル[5]を参照されたい。

2.5 ベクトル化指示行

コンパイラは、自動ベクトル化に際し、原始プログラムの解析結果を最大限に利用するが、最適なベクトル命令列の生成のためには、その情報だけでは必ずしも十分ではない。例えば、実際にはベクトル化可能であるにもかかわらず、コンパイラはそのことを判断できず、ベクトル化不能として処理してしまうことがある。ベクトル化指示行は、原始プログラムの解析によっては得ることができない情報を利用者が与えることにより、自動ベクトル化の効果を一層促進させるためのものである。その形式と意味については、本号の別稿[3,4]およびマニュアル[5]を参照されたい。

2.6 ベクトル化に関する出力リストと診断メッセージ

自動ベクトル化機能が有効な場合、コンパイラの各種出力リストにベクトル化に関する種々の情報が出力される。これらの情報は、プログラムのチューニング作業等において有効である。

(1) 編集リスト

その一部あるいは全体がベクトル化されたDOループに対しては、編集リスト中のループ構造を示す図に、そのことが表示される。図2に編集リストの例を示す。図において、"V"は、そのループがベクトル化されたことを示し、また、"S"は、その部分がループ分割によりスカラで実行されることを示す。

(2) ベクトル化診断メッセージおよびベクトル化情報リスト

VECTORオプションあるいはベクトル化指示行によりINFOMSGまたはFULLMSGが指定されている場合、コンパイラによる自動ベクトル化処理の対象となったループに対して、そのループがベクトル化されたか否かが表示される。また、FULLMSGが指定されている場合には、さらに、ベクトル化されなかったループあるいは部分に対して、その理由が表示される。これらのメッセージをベクトル化診断メッセージと呼ぶ。翻訳時オプションのMRGMSGが指定されている場合には、これらのメッセージは、原始プログラムリスト中に埋め込まれて表示される。FULLMSGが指定されている場合には、さらに、ベクトル化に関する詳細な情報を含むベクトル化情報リストが出力される。図3に、ベクトル化診断メッセージの例を、また図4にベクトル化情報リストの例を示す。

```
003100    310    326                              DO 13  JKL=1,10000
003110    311                    37---------->     |  CALL INITN(13)
003120    312    326             |                 |  DO 13 IP=1,128
003130    313                    | V--------->     |  |  I1=P(1,IP)
003140    314                    | |               |  |  J1=P(2,IP)
003150    315                    | |               |  |  P(3,IP)=P(3,IP)+B(I1,J1)
003160    316                    | |               |  |  P(4,IP)=P(4,IP)+C(I1,J1)
003170    317                    | |               |  |  P(1,IP)=P(1,IP)+P(3,IP)
003180    318                    | |               |  |  P(2,IP)=P(2,IP)+P(4,IP)
003190    319                    | |               |  |  I2=P(1,IP)
003200    320                    | |               |  |  J2=P(2,IP)
003210    321                    | |               |  |  P(1,IP)=P(1,IP)+Y(I2+32)
003220    322                    | |               |  |  P(2,IP)=P(2,IP)+Z(J2+32)
003230    323                    | |               |  |  I2=I2+E(I2+32)
003240    324                    | |               |  |  J2=I2+F(J2+32)
003250    325                    | |          S    |  |  H(I2,J2)=H(I2,J2)+1.0
003260    326                    37V----------S 13  CONTINUE
```

図2　編集リストの例

```
003100      310          DO 13  JKL=1,10000
003110      311          CALL INITN(13)
003120      312          DO 13 IP=1,128

            VEC   3 : DOループを部分的にベクトル化する．DO変数は IP
                      外部行番号 003130 - 003240 をベクトル化する．

003130      313          I1=P(1,IP)
003140      314          J1=P(2,IP)
003150      315          P(3,IP)=P(3,IP)+B(I1,J1)
003160      316          P(4,IP)=P(4,IP)+C(I1,J1)
003170      317          P(1,IP)=P(1,IP)+P(3,IP)
003180      318          P(2,IP)=P(2,IP)+P(4,IP)
003190      319          I2=P(1,IP)
003200      320          J2=P(2,IP)
003210      321          P(1,IP)=P(1,IP)+Y(I2+32)
003220      322          P(2,IP)=P(2,IP)+Z(J2+32)
003230      323          I2=I2+E(I2+32)
003240      324          J2=I2+F(J2+32)
003250      325          H(I2,J2)=H(I2,J2)+1.0
                         1
       1    VEC 137 : 参照関係不適とみなす．ベクトル添字を持つ配列要素が現れている : H 003250

003260      326     13   CONTINUE
```

図3　ベクトル化診断メッセージの例

（注）翻訳時オプションMRGMSGを指定した場合の例。なお、日本語によるメッセージの出力は、次回のバージョンアップ時より可能となる予定。

```
24   003120          13   DOループを部分的にベクトル化する．DO変数は IP
                          *  ベクトル化範囲          :  003130--003240
                          *  繰返し数                :  128
                          *  作業用ベクトル領域      :  1024 バイト
                          *  指標変数                :  IP

                     003250      137 : 参照関係不適とみなす．ベクトル添字を持つ配列要素が現れている : H 003250
```

図4　ベクトル化情報リストの例

（注）日本語によるメッセージの出力は、次回のバージョンアップ時より可能となる予定。

3. ベクトル化に関する高速化技法

ＳＸシステムの特徴の一つとして、多重並列パイプラインアーキテクチャ[1]とコンパイラの自動ベクトル化機能とにより、自然なプログラミングで、高いベクトル性能を引き出すことができること、および、高いスカラ性能とコンパイラの最適化機能とにより、ベクトル化率がそれほど高くないプログラムに対しても、一定レベル以上の性能が得られることが挙げられる。しかし、プログラムによっては、少しの手直しだけで、大幅な高速化が可能な場合もある。ここでは、そのようなＳＸシステム向きの高速化技法について、ベクトル化の観点からのべる。

ＳＸシステムにおける高速化のためには、次の２つが重要である。

- ベクトル化率を可能な限り高めること
- 生成されるベクトル命令の効率を可能な限り高めること

FORTRAN77/SXコンパイラは、このような利用者によるチューニング作業を支援するために、前述の編集リスト、ベクトル化診断メッセージやベクトル化情報リストを用意している。また、ANALYZER/SX, VECTORIZER/SX などの性能向上支援ツールも用意されている。なお、これらの支援ツールについては、次号で解説される予定である。

3.1 ベクトル化率を高めるための技法

プログラムをベクトル化せずに実行したときの全実行時間のうち、ベクトル化により、ベクトル命令で実行される部分の比率をベクトル化率という。プログラムの実効性能を高めるためには、ベクトル化率を高めることが極めて重要である[1]。

ベクトル化率を高めるためには、次の三つの方法がある。

- ベクトル化に適した計算法あるいはアルゴリズムを用いる
- ベクトル化不可原因を取り除く
- コンパイラによるベクトル化処理対象部分を増加させる

ここでは、プログラミング技法という観点に的を絞り、後二者について述べることとする。

(1) ベクトル化不可原因の除去

前述のように、ベクトル化不可原因は、ベクトル化診断メッセージにより表示される。以下では、いくつかの代表的な例について、その除去方法を示す。

(a) 定義・引用関係の適合性をコンパイラが判断できないループ

配列の定義・引用の関係がベクトル化に適合しているにもかかわらず、コンパイラがそのことを判断できないためにベクトル化されていない場合、ベクトル化指示行の NODEP パラメータを指定する。

例(1)
```
*VDIR  NODEP(A)          ← ベクトル化指示行
       DO  10  I=1, N
         A(I)=B(I)+T*C(I)
10       X(I)=A(I+L)*Y(I)
```

例(2)
```
*VDIR  NODEP             ← ベクトル化指示行
       DO  10  I=1, N
         IX(I)=IA(I)-IB(I)*IC(I)
10       H(IX(I))=H(IX(I))+1.0
```

たとえば、例(1)の場合には、$L \leqq 0$ あるいは $L \geqq N$ であれば、ベクトル化可能である。同様に、例(2)の場合には、IX(I)の値が全て異なっていれば、ベクトル化可能である。そのような場合には、例のように、ベクトル化指示行のNODEPパラメータを指定すればよい。

(b) 外部関数や外部サブルーチンの引用を含むループ

外部関数や外部サブルーチンを含むループは、そのままではベクトル化できないが、次の方法により、その一部あるいは全体をベクトル化できることがある。

(イ) その関数またはサブルーチンを直接組込む。

例
```
       DO  10  I=1, N
10       X(I)=Y(I)*FNORM(Z(I))
                  ↓
       FUNCTION  FNORM(X)
       FNORM=0.398*EXP(-0.5*X**2)
       RETURN
       END
                  ⇩
       DO  10  I=1, N
10       X(I)=Y(I)*0.398*EXP(-0.5*Z(I)**2)
```

なお、支援ツールのOPTIMIZERを使用すれば、組込みを効率よく行うことができる。

また、文関数に書き換え可能なら、そうすることにより、ベクトル化可能になることがある。

(ロ) ベクトル化指示行のELEMENTALまたはNORMALを使用することにより、そのループの一部あるいは全体をベクトル化できることがある。その使い方については、マニュアル[5]を参照されたい。

(c) その他の例

たとえば、以下の場合には、マクロ演算の漸化式の形に変形することによりベクトル化可能になる。

例

```
      DO  10  I=2, 998, 3
      X(I)=Z(I)*(Y(I)-X(I-1))
      X(I+1)=Z(I+1)*(Y(I+1)-X(I))
10    X(I+2)=Z(I+2)*(Y(I+2)-X(I+1))
                  ⇩
      DO  10  I=2, 1000
10    X(I)=Z(I)*(Y(I)-X(I-1))
```

(2) ベクトル化処理対象の拡大

IF文とGOTO文によるループをDOループに書き直したり、ループを分割し、外側のループを処理対象にすることにより、ベクトル化の処理対象を拡大することができる場合がある。詳細についてはマニュアル[5)]を参照されたい。

3.2 ベクトル命令の効率を高める技法

ベクトル化による高速化の程度は、生成されるベクトル命令の種類、ループの繰返し数（ループ長またはベクトル長）、配列の参照パターンなどによって大きく影響される場合がある。ここでは、生成されるベクトル命令の効率を可能な限り高めるためのプログラミング技法や注意事項について述べる。

(1) ループ長の拡大

ベクトル化の効果を十分に引き出すためには、ループ長を可能な限り長くする必要がある（図5参照）。

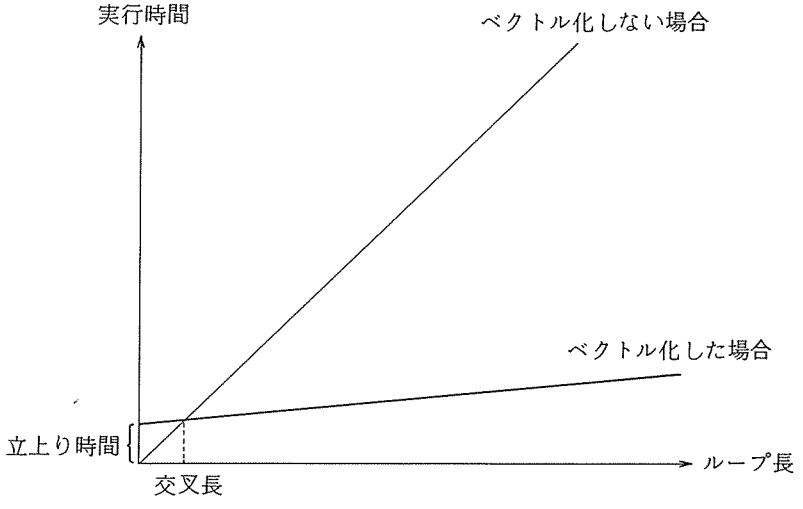


図5 ループ長と実行時間の関係

そのための方法としては、次のものがある。

- 行列の一次元化による多重ループの一重化
- 行と列の入れ換え

例

```
      DO 10 J=1, N
        DO 20 I=1, M
20        A(I, J)=X*B(I, J)+C(I, J)
10    CONTINUE
                  ⇩ N=10000, M=10とする
      DO 20 I=1, M
        DO 10 J=1, N
10        A(I, J)=X*B(I, J)+C(I, J)
20    CONTINUE
```

なお、ループ長をコンパイラが認識できる場合には、自動的に行うことがある（**2. 3**参照)。

(2) 配列の参照パターンの改善

ベクトルデータの記憶域からの取り出しや記憶域への格納に要する時間は、その参照パターンによって大きく影響される。すなわち、

(イ) 連続ベクトルおよび隣合う要素の間隔が奇数の場合には、最高速である。

(ロ) 隣合う要素の間隔が偶数の場合、特に2のn乗（4, 8, 16, . . .)で割り切れる場合には、nの値が大きくなるにしたがって効率が悪くなる。

(ハ) 間接指標ベクトルの場合の効率は、(イ)に比べて劣る。

したがって、次の方法により、高速化が可能である。

(a) 指標変数は、なるべく最初の次元に現れるようにする

例

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

(b) 指標変数がやむを得ず二次元目以降に現われる場合、繰返しごとの添字の値の変化が奇数になるようにする

例

```
      DIMENSION A(512,512),B(512,512)
                ↓
      DO 10 K=1,N
10      S=S+A(I,K)*B(K,J)
```

⇩

Aに対する配列宣言をA(513,512)に変更

(c) 間接指標ベクトルの引用頻度が高い場合、一時的に作業用の連続ベクトルに移す

例

```
      =A(IX(I))
      =A(IX(I))
```

⇩

```
      DO 10 I=1,N
10      WA(I)=A(IX(I))
```

```
      =WA(I)
      =WA(I)
```

(3) IF文の除去

IF文を含むループは、通常ハードウェアのマスク付きベクトル演算機能[1)]によりベクトル化される。しかし、その演算時間は、マスクの有無や"1"ビットの数によらず一定なため、実際に演算すべき要素数が少ない場合には、ベクトル化の効率が悪くなる。

したがって、次の方法により、高速化が可能である。

(a) IF文の単純な除去

例

```
      DO 10 I=1,N
        IF(A(I).NE.0.0)THEN
          D(I)=A(I)*(B(I)+C(I))
        ELSE
          D(I)=0.0
        END IF
10    CONTINUE
```

⇩

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

(b) ループ外での圧縮・伸長またはリストベクトルの使用

例

```
      DO 10 I=1,N
        IF(A(I).NE.0.0) THEN
          C(I)=A(I)*SIN(B(I))
        END IF
10    CONTINUE
            ⇩
      K=0
      DO 1 I=1,N                          ┐
        IF(A(I).NE.0.0)THEN               │
          K=K+1                           │
          IX(K)=I                         │
          AA(K)=A(I)                      ├ 圧縮
          BB(K)=B(I)                      │
        END IF                            │
1     CONTINUE                            ┘
      DO 10 I=1,K
        CC(I)=AA(I)*SIN(BB(I))
10    CONTINUE
      DO 2 I=1,K                          ┐
        C(IX(I))=CC(I)                    ├ 伸長
2     CONTINUE                            ┘
```

作業用ベクトルを使わず、以下のようにリストベクトルを直接添字に使用してもよいが、いずれの方法が高速かは、圧縮後の要素の参照回数に依存する。

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

(4) 内積と積和

行列積などにおいて、内積型、積和型のいずれによっても計算できる場合がある。SXでは、内積も積和も非常に高速であるが、ループ長が短い場合には、積和のほうが、より高速である。したがって、そのような場合には、積和型を用いたほうがよい。

内積型

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

積和型

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

(5) 遅い演算の回避

SXでは、漸化式演算もベクトル化されるが、他のベクトル命令に比べて低速なため、可能なら避けるほうが望ましい。

例

```
      DO  10  I=1, M
        DO  10  J=1, N
10        A(I, J+1)=X(I, J)-T*A(I, J)
                  ⇩
      DO  10  J=1, N
        DO  10  I=1, M
10        A(I, J+1)=X(I, J)-T*A(I, J)
```

また、ベクトル除算は、他の演算に比べて低速であるため、可能なら避けたほうがよい。

(6) ループ分割の回避あるいは分割数の削減

ループ分割によりベクトル化された場合、分割点ごとに分割による効率低下が生じる。したがって、可能なら、ベクトル化不可原因を除去したり、ベクトル化不可部分の集中化により、ループ分割の回避あるいは分割数の削減を図ることが望ましい。

(7) ループ展開(アンローリング)について

ループ本体をn倍にし、かわりに繰返し数を$1/n$にする処理をループ展開(アンローリング)という。SXでは、一般にアンローリングは、ベクトル化の効率を低下させるため、避けたほうが望ましい。

例(1)

```
      DO  10  I=1, 97, 3
10      S=S+A(I)*B(I)+A(I+1)*B(I+1)
     &        +A(I+2)*B(I+2)
                ⇩
      DO  10  I=1, 99
10      S=S+A(I)*B(I)
```

例(2)

```
      DO  10  I=1, 999, 2
        X(I)=Y(I)+Z(I)
10      X(I+1)=Y(I+1)+Z(I+1)
                ⇩
      DO  10  I=1, 1000
10      X(I)=Y(I)+Z(I)
```

ただし、外側のループについてのアンローリングで、かつ、記憶域の参照回数が減少するような場合には、大きな効果が得られることがある。

例
```
      DO  10  J=1, M
        DO  10  I=1, N
10        X(I)=X(I)+A(I, K)*B(K, J)
                  ⇩
      DO  10  J=1, M, 2
        DO  10  I=1, N
10        X(I)=X(I)+A(I, K)*B(K, J)
     &                  +A(I, K)*B(K, J+1)
```

4. おわりに

以上、SXシステムの高速性を十分に引き出すためのプログラミング技法について述べてきました。ただし、コンパイラの自動ベクトル化機能は、必ずしも固定されたものではなく、今後一層の強化が予定されています。したがって、現在は利用者による手直しが必要なものでも、将来はコンパイラによって自動的に処理されるようになるものもあるということをご承知おきください。

SX－1は、最大570MFLOPSという、高いベクトル性能をもつスーパーコンピュータです。また、スカラ性能の高さも、特長の一つです[2)]。しかし、全くベクトル化できないプログラムを実行させたとしたら、とても、その〝スーパー〟な性能を引き出すことはできません。そういう意味で、SXシステムが、スーパーコンピュータになるか、あるいはただの高速なコンピュータになるかは、ひとえに利用者の使い方次第であると言っても過言ではありません。

十分に説明しきれなかったところも多いと思いますが、本稿が、利用者の皆様のお役に立てば幸いです。

参 考 文 献

1) 渡辺，近藤，端山，大中，藤井：スーパーコンピュータSX－1の概要(1)，大阪大学大型計算機センター・ニュース，Vol.15，No.4(1986)．

2) 藤井：スーパーコンピュータSX－1の概要(2)，大阪大学大型計算機センター・ニュース，Vol.15，No.4(1986)．

3) 大中，後藤：SX FORTRAN77概要(1)，大阪大学大型計算機センター・ニュース，Vol.16，No.1(1986)．

4) 後藤，大中：SX FORTRAN77概要(2)，大阪大学大型計算機センター・ニュース，Vol.16，No.1(1986)．

5) GGB12－2 FORTRAN77，77/SXプログラミング手引書，日本電気(1986)．