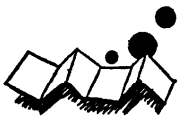
解　説



汎用マルチプロセッサシステムによる並列処理†

福　井　義　成††

1. はじめに

計算機の発達にともない，計算機による解析（シミュレーション）を行うことが多くなっている．その解析も次第に規模が大きくなり，現在では非常に大規模な計算が行われている．このような要求に応えるため出現したのがスーパコンピュータであり，計算機の解析能力を特に高めたものとなっている．スーパコンピュータの能力は大きいほどよいが，ある程度以上速い計算機は作りたくても物理的な限界がある．速い計算機を作る最もよい方法は速い素子を開発することであるが，現在作られている最も速い素子よりも 100～1,000 倍以上速い素子を作ることは容易でない．このような制約に対して，一つの計算で複数のプロセッサを同時に使うことにより，実際の処理能力を高めることができる．ここでは，まず科学技術計算の並列性について述べ，並列処理システム，汎用型マルチプロセッサでの並列処理，適用例について述べる．

2. 科学技術計算の並列性

自然現象は，いろいろな状態が常にバランスを保ったり，常に変化したりしている．自然現象をモデル化した方程式は，発展方程式またはその定常状態を表す方程式と表現されることが多い．自然現象を離散化し逐次処理の計算機で問題を解く場合，一度に計算可能なのは離散化された問題のある1点の状態だけである．しかし，その他の点での現象も同時に進行しており，並行して進行している二つ以上の点の状態を同時に計算するのは逐次処理の計算機では不可能である．いいかえると，自然現象がもっている並列性が逐次処理のプログラム（計算の手順を記述したものである）の中に埋没しているといえる．本来，自然現象は並列性を内在しているものが多く，偏微分方程式や常微分方程式としてモデル化できる問題はすべての点で同時に計算するほうが自然である[1)~3)]．

また，π の値を Machin の公式

$$\pi/4\ \arctan(1/5) - \arctan(1/239)$$

を用いて計算する場合なども arctan の各項は互いに独立であるため，並列計算に適している．

このように科学技術関係の問題ではもともとの問題自身が並列性を内在しているものが多く，うまくモデル化・離散化すれば並列計算がそのままで行える場合が多い．

3. 並列計算システム

現在，最も普及しているスーパコンピュータはパイプライン方式によるベクトル計算機である．ベクトル計算機は同じクラスの汎用機に比べて 10 倍以上の能力を発揮することもできるが，そのためにはベクトル化が必要である．現在，ベクトル化が可能なのは比較的単純な DO ループ，またはそれと同等なものに限られている．もともと簡単なループ構造をもたない問題や複雑すぎる条件分岐を含むループをベクトル化するのは困難である．これらをベクトル化するのも不可能ではないが，現在のベクトル計算機用にプログラムを変更するのは大変な労力を必要とする．ベクトル計算機は，ある条件（ベクトル計算機のベクトル化可能条件，以下では単にベクトル化条件と呼ぶ[4)]）を満足する場合のみベクトル化が可能で，ベクトル化できない場合は高速に処理できない．しかし，ベクトル化できない場合であっても，自然現象から派生した問題の中には並列処理可能となっているものが多い．そのような現象に対しては強引にベクトル化するよりも，複数のプロセッサを使用して並列処理を行うことのほうが容易である．5. で述べる回路解析の並列化は，基礎的な機能の調査を含めて1カ月ほどで行うことができた．しかし，現在開発を行っている回路解析のベクト

† Parallel Processing on the General Purpose Multi Processer System by Yoshinari FUKUI (Total Information & Systems Division, TOSHIBA CORPORATION).

†† 東芝総合情報システム部
現在，東芝 CAE システムズ

ル化は1年以上の期間を必要としている.

計算機を“命令”と“データ”の関係から分類すると以下のようになる.

SISD 方式 (Single-Instruction stream—Single-Data stream : 単一命令単一データ流れ)

SIMD 方式 (Single-Instruction stream—Multi-Data stream : 単一命令複数データ流れ)

MISD 方式 (Multi-Instruction stream—Single-Data stream : 複数命令単一データ流れ)

MIMD 方式 (Multi-Instruction stream—Multi-Data stream : 複数命令複数データ流れ)

SISD 方式は，普通の汎用計算機である．ILLIAC-IVやマクロに見た場合の多くのベクトル型スーパコンピュータがSIMD方式に当たる（マクロに見た場合，ベクトル型スーパコンピュータは一つのベクトル命令で複数のデータを処理している）．ミクロに見た場合，一つのデータの処理を複数の“段”に分けて演算しているので，パイプライン型の計算機は MISD 方式と考えられる．マルチタスクを適用した CRAY X-MP[5]，CRAY-2[5] や ALLIANT FX/8 (DPS 9000)[6]，PAX[2]，ADINA[7),8)] などの並列計算機が MIMD 方式にあたり，データフロー型計算機もこの中に含まれる．

複数のプロセッサを使って並列処理を行う場合，今度は複数プロセッサ間の同期が問題となる．同期のためのオーバヘッドの影響が実用になるレベルでなければならない．プロセッサ間の同期の方法にもプロセッサ間コミュニケーションレジスタを使う方法やバス接続を使う方法などいろいろあるが，どの方式がよいかは計算機のマシンサイクルなどの環境条件によって変わってくる．

現在，複数のプロセッサを使用した並列計算機にも大きく分けて

（1） 汎 用 型

（2） 専 用 型

の二つのタイプがある．汎用型は特定の問題に特化せずに，汎用性を目標にしたものである．CRAY X-MP, CRAY-2 などのように，スーパコンピュータ級の大型プロセッサを比較的少数使用したものと，ALLIANT FX/8 などのように比較的小さな能力のプロセッサを並列化したものなどがある[5),9)]．専用型の並列計算機は比較的小さな能力のプロセッサを大量に使用しており，PAX や ADINA などのように特定の問題を念頭に置いて作られているものである．PAX は偏微分方程式を陽的解法で解く場合を念頭に置いており，ADINA は ADI 法[10]を念頭に置いている（汎用型・専用型の分類は典型的な場合を示したものであり，この単純な分類には当てはまらないものも出現している．さらに，この分野は日々変化している）．

複数のプロセッサを結合する場合，プロセッサ相互およびメモリとの間の“ネットワーク”が最も重要である．この部分の設計が並列計算機の死命を制するといえる．専用型では問題を限定しているため，問題に向いたプロセッサネットワークを作りやすいので比較的多数のプロセッサを接続することが可能となっている．しかし，汎用型ではその使用目的をあまり限定しないため，専用機に比べて“ネットワーク”の設計が難しい．

4. 汎用型マルチプロセッサシステムでの並列処理

ここでは汎用型マルチプロセッサシステムの例として，CRAY X-MP について述べる．CRAY X-MP の場合，図-1 のように複数（m 台）のプロセッサで複数（n 個）のプログラム（ジョブ，タスク）を処理するようになっている．したがって，$m \leqq n$ の場合は実時間処理を必要とするプログラムでないかぎり，並列処理を行う意味はあまりない．しかし，

（1） 実時間処理を必要とする場合

（2） ターンアラウンド時間を短縮したい場合

（3） 大メモリを使うプログラムのため，使われないプロセッサができてしまう場合

などはプログラムを並列処理し，速く処理することが必要となる．また，$m > n$ の場合は使用するプロセッサの数が増えれば処理時間が短くなるので，並列処理を行う意味が大きい．

4.1 マルチタスキング

CRAY X-MP における並列処理はマルチタスキン

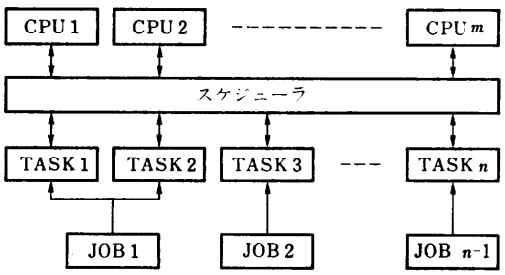


図-1 CRAY X-MP における CPU, TASK, JOB の関係

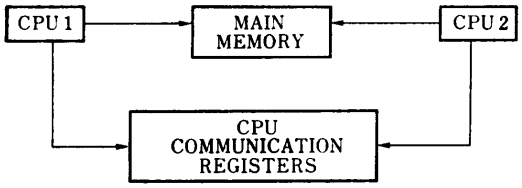


図-2 CRAY X-MP のハードウェア構成（2CPU の場合）

グと呼ばれ，次の二つの方法がある．

（1） マクロタスキング

（2） マイクロタスキング

マクロタスキングはプログラムを複数のタスクに分け，おのおののタスクを別々のプロセッサで処理することにより並列処理を行うものである．親タスクから子タスクを生成したり，タスク相互間で同期をとるためのライブラリが用意されている．マイクロタスキングは並列パイプライン型のスーパコンピュータ（同じ機能の複数のパイプラインをもつ）に近い機能も行い，FORTRAN とコンパイラディレクティブだけで並列処理が行える．

複数のプロセッサによる並列化は以下の５つのレベルに分類できる．

（1） ジョブのレベル

（2） ジョブステップのレベル

（3） サブルーチン相互間のレベル

（4） サブルーチン内のレベル

（5） DO ループのレベル

ジョブレベル，ジョブステップレベルの並列化は従来の汎用計算機でもすでに行われてる．マクロタスキングはサブルーチン相互間のレベルでの並列処理であり，マイクロタスキングはサブルーチン内のレベルと DO ループのレベルでの並列処理が行える．並列パイプライン型のスーパコンピュータは DO ループのレベルでの並列処理と解釈できる．

図-2 に二つのプロセッサを有す CRAY X-MP の概略を示す．メインメモリ上のマクロタスキングプログラムは二つのプロセッサを並行に使用する．

4.2 マイクロタスキング

マクロタスキングではタスクの管理をユーザが行うが，マイクロタスキングではタスクの管理はシステムが行ってくれるので，ユーザはプロセッサの数と並列処理をする部分をコンパイラディレクティブで指定するだけでよい．並列処理の指定には二つの方法があり，それは **図-3** の例のように “PROCESS～ALSO PROCESS～END PROCESS” で行う方法と “DO GLOBAL” で行う方法がある．“DO GLOBAL” が並列パイプライン型のスーパコンピュータに近い機能である．

```
      PROGRAM MICROT
        ・・・
CMIC$ GETCPUS(4)        ←―― 4個のＣＰＵを設定
        ・・・
      CALL SUB
        ・・・
CMIC$ RELCPUS           ←―― ＣＰＵを開放
        ・・・
      END
CMIC$ MULTI             ←―― 以下のサブルーチンを並列処理
      SUBROUTINE SUB
        ・・・
CMIC$ PROCESS           ←―― 並列処理部－Ａ　開始指定
        （並列処理部－Ａ：ＣＰＵ１）
CMIC$ ALSO PROCESS      ←―― 並列処理部－Ｂ　開始指定
        （並列処理部－Ｂ：ＣＰＵ２）
CMIC$ END PROCESS       ←―― 並列処理部　　終了指定
        ・・・
        ・・・
        ・・・
CMIC$ DO GLOBAL         ←―― ループの並列処理を指定
      DO 200 I=1,M
      DO 100 J=1,N
        ・・・          ――→ I= 1, 5, 9, ・・・ : CPU 1
        ・・・          ――→ I= 2, 6,10, ・・・ : CPU 2
        ・・・          ――→ I= 3, 7,11, ・・・ : CPU 3
        ・・・          ――→ I= 4, 8,12, ・・・ : CPU 4
  100 CONTINUE
  200 CONTINUE
      END
```

図-3 マイクロタスキングを適用した例

4.3 マクロタスキング

マクロタスキングはサブルーチン相互間のレベルでタスクが定義され，並列処理が行われる．タスクの発生，タスク間の同期などは全て FORTRAN プログラムからライブラリを呼ぶことにより行われる．マクロタスキングはスカラー処理とベクトル処理の双方に適用され，ベクトル化が最内側ループ周辺においてのみ適用されるのに対し，マクロタスキングはより外側のループに適用される．

タスクの発生消滅とタスク間の同期には二つのプロセッサ間の通信が必要となるが，これにはどのプロセッサからもアクセス可能なプロセッサ間コミュニケーションレジスタが使われる．これらのレジスタはタスクの発生・待ち・終了の情報交換に用いられる．

CRAY X-MP の FORTRAN コンパイラは，マクロタスキングのために複数のプロセッサで同一のサブルーチンを実行できる再入可能なコードを生成できる．実行形式の FORTRAN プログラムはコード領域，データ領域，さらにデータ領域は各ルーチンに局所的なローカルデータ領域，各ルーチンに共通なコモンデータ領域に分類できる．複数のプロセッサで同一のサブルーチンを実行する場合，ローカルデータ領域

表-1　マルチタスキングライブラリ

タスク制御ルーチン	CALL TSKSTART (TASKID, Subname, [Arg]) タスクを発生させる CALL TSKWAIT (TASKID) 他のタスクを待つ
イベント制御ルーチン	CALL EVPOST (EVENT) イベントを発生させる CALL EVWAIT (EVENT) イベントを待つ CALL EVCLEAR (EVENT) イベントを消す
ロック制御ルーチン	CALL LOCKON (LOCK) メモリへのアクセスをロックする CALL LOCKOFF (LOCK) メモリへのアクセスロックを解除する

はおのおののタスクで独立でなければならず，各ルーチンのローカルデータ領域はタスクの発生時に動的にメモリ内に割り付けられる．さらにコモンデータ領域はタスク間で共有するものと各タスク間で独立なものとに分類できるが，後者をサポートするのがタスクコモンである．通常のコモン領域がプログラムローディング時に割り付けられるのに対し，タスクコモンデータ領域はローカルデータ領域と同様にタスク発生時に動的にメモリ内に割り付けられ，おのおののタスク内でのみ有効である．マクロタスキング用 FORTRAN ライブラリルーチンは **表-1**，**図-4**～**図-6** のような機能をもっている．

図-7 に行列の積を求めるプログラムにマクロタスキングを適用した例を示す．この例では最外側ループを二つに分割し，二つのプロセッサで処理している．

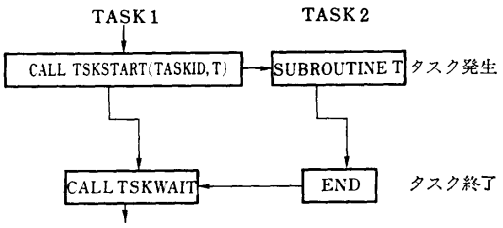


図-4　子タスクの発生と消滅

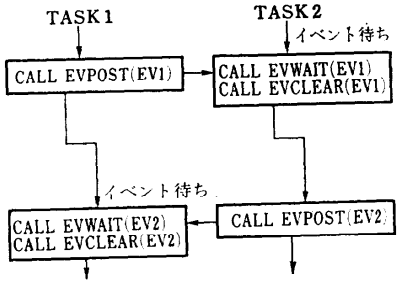


図-5　イベントによるタスク間の同期制御

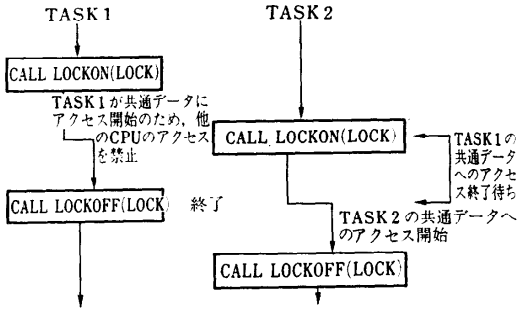


図-6　ロック制御によるタスク間共通データの処理

```
      PROGRAM  MACROT
      COMMON  A(100,100), B(100,100), C(100,100)
      EXTERNAL  T
                                  CALL TSKSTART( TASKID, T )
      DO 600 I=1,99,2
        DO 400 J=1,100
          C(I,J)=0.0
          DO 200 K=1,100
            C(I,J)=C(I,J)+A(I,K)*B(K,J)
  200     CONTINUE
  400   CONTINUE
  600 CONTINUE
                                  CALL TSKWAIT( TASKID )
      END
      SUBROUTINE T
      COMMON  A(100,100), B(100,100), C(100,100)
      DO 1600 I=2,100,2
        DO 1400 J=1,100
          C(I,J)=0.0
          DO 1200 K=1,100
            C(I,J)=C(I,J)+A(I,K)*B(K,J)
 1200     CONTINUE
 1400   CONTINUE
 1600 CONTINUE
      END
```

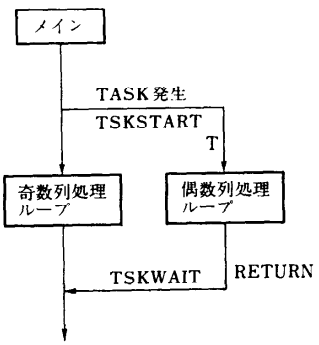


図-7　マクロタスキングを適用した例

CRAY X-MP のマルチプログラミング環境下では，スケジューラは各ジョブに対しプロセッサの割り付け・解放を行う．実在するプロセッサを物理的プロセッサと呼ぶ．タスク（親タスク，子タスク）が生成されると，タスクごとに一つの論理プロセッサが割り付けられる．論理プロセッサはタスクが消滅すると割り付けが解除される．スケジューラは論理プロセッサに物理的プロセッサを割り付けて実行する．マルチタスクジョブの場合は一つのジョブに複数の論理プロセッサが割り付けられ，マルチタスク化されていないジョブの場合は，一つのジョブに一つの論理プロセッサが割り付けられる．物理的プロセッサの論理プロセッサへのスケジューリングは，実際に存在する物理的プロセッサの数により異なる．すなわち，ユーザはマクロタスキングを行う場合，物理的プロセッサの数を意識する必要はないが，並列化の効果は使用可能な物理的プロセッサの数で決まる．

図-3, 図-7 の例でも分かるように，マイクロタスキングでは，一つのサブルーチンの内での並列処理はマクロタスキングより容易であるが，多くのサブルーチンにまたがるような並列処理はマクロタスキングのほうが向いている．回路解析で並列化を試みた部分は数千ステップで複数のサブルーチンにまたがっているので，マクロタスキングによる並列化を行った．

5. 適用例—回路解析の場合—

回路解析で行う解析には直流解析，交流解析，過渡解析などがあるが，なかでも過渡解析は計算時間が長くなるため，最も高速化を必要とする．過渡解析も次の三つの主な部分に分けて考えることができる．

（1） 回路行列の構成
（2） 回路行列計算（行列分析，求解）
（3） その他（入出力や全体の制御など）

表-2 コード生成による高速化の効果

	行列構成	行列計算		その他	全体
		分析	求解		
FORTRAN 版	275秒 (21.8%)	16秒 (1.3%)	942秒 (74.6%)	29秒 (2.3%)	1,262秒
		958秒 (75.9%)			
コード生成版	注) 257秒 (78.3秒)	16秒 (4.9%)	36秒 (11.0%)	注) 19秒 (5.8%)	328秒
		52秒 (15.9%)			

注）コード生成以外の変更も含む

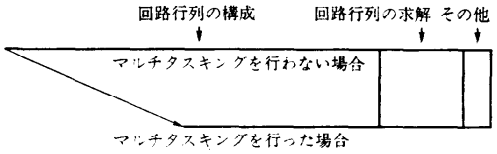


図-8 計算時間比率の変化

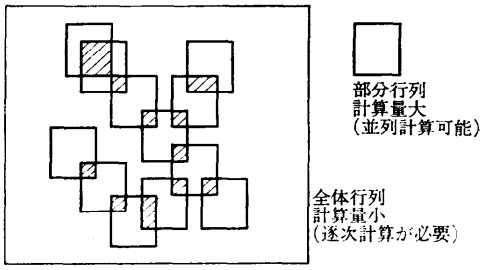


図-9 全体行列の構成

(2)の求解についてはコード生成を行い，942秒を36秒へと約26倍の高速化を実現した[11]．ある典型的な例では，(1)～(3)のプロセッサ使用時間の分布は **表-2** のようになっている．計算時間の変化を図示すると **図-8** のようになる．

回路解析に現れる行列はランダムスパース行列となり，スパース率も高い．そのため，SPICE[12]では回路行列を記憶するためのデータ構造としてリスト方式を採用している．その結果，(1)の回路行列を表現するリストをたどって初めてデータを得ることができる．このため必然的に数値的な計算よりも，条件判定とリスト構造をたどることに多くのプロセッサ時間を費やしている．当初使うことができた CRAY X-MP には，gather/scatter 機能（リストベクトルの機能）がなかったので，ベクトル化することは容易でなかった．そこで，この部分にマルチタスキングを適用して高速化を図ることにした．回路行列の構成の部分をさらに分けると

（1） 各行列要素の計算（部分行列）
（2） 計算された行列要素の全体行列への加算

となる．**図-9** において各部分行列は独立に計算可能であり，並列処理向きである．しかし，部分行列から全体行列を構成するとき，各部分行列が重なっているところ（斜線の部分）を並列計算してしまうと計算結果が異なってしまうので，逐次処理で計算しなければならない．幸いなことに，回路行列においては部分行列の計算が複雑なため，全体行列の

表-3 マルチタスキングの効果

ケース	経過時間（秒）			プロセッサ時間（秒）		
	単独	並列	比率	単独	並列注）	比率
1	373	253	0.68	371	418	1.13
2	354	231	0.65	354	398	1.12

注）二つのプロセッサの合計時間

構成に必要とされるプロセッサ時間は，部分行列の計算に必要とされるプロセッサ時間に比べて無視することができる．そのため，複数プロセッサによる並列処理の効果が大きい．

単独実行の環境でのマルチタスキングの効果を表-3に示す．プロセッサは二つである．この二つのケースのプログラム全体の経過時間の比率は 0.65〜0.68 倍と効果的である（マクロタスキングを適用した部分では，経過時間は約 0.55 倍になっている）．オーバヘッドによるプロセッサ時間の増加も 1 割程度である[13]．

6. ま と め

以上述べたように，マルチタスキングによる高速化は単一プロセッサの能力の限界を越えるのに効果的である．現在のスーパコンピュータで十分にベクトル化されたプログラムも，並列処理なしに現在の 100〜1,000 倍以上の性能を出すことは困難である．今後の方向として，複数のプロセッサによる並列処理が不可欠である．1 ユーザが使用できるプロセッサの台数が増えるに従い，並列処理が日常的なものになるであろう．

現在の計算機環境で FORTRAN を中心に巨大な計算をするという立場に立つと，プログラム変更に時間を必要とするベクトル化や並列処理よりも何もしないですむ速いスカラ計算機が望ましい．計算をすることが目的のユーザにとって，FORTRAN で書かれたプログラムをベクトル化や並列化することに時間をかけることは望ましくない．しかし，現在の逐次処理計算機を忘れてモデル化まで戻って考えると，科学技術計算では並列計算可能な問題が多い．それに対して，FORTRAN でプログラムを作成することは，人間が計算機に処理の手順を与えていることである．逐次処理計算機のために作られた FORTRAN プログラムをベクトル化・並列化することよりも，より抽象化されたレベルで何をすべきかということを計算機に示してやれば，ベクトル化・並列化もより容易にできる．たとえば，二つの正方行列の積を計算する場合，三重の DO ループ構造を与えるよりも行列定義と行列演算で記述すれば，使用している計算機ごとに能力を最大に発揮させることは容易になる．今後は，このような方面の研究が必要である[14),15)]．

並列計算のためには，次のことが大切である．

（1） 問題のもつ並列性を自然に表現できる手段
①モデル化の手法，②記述言語とデータ構造．

（2） 計算の再現性・検証のしやすさ

複数のプロセッサを使用すると，単独プロセッサではまったく問題とならないタイミングのズレなどにより計算の再現性が困難になる．

（3） アルゴリズムなどの評価の環境整備

単独プロセッサの並列計算のシミュレーションではプロセッサ間の競合などは起きえないため，並列処理の効果の評価は不可能であり，評価には複数プロセッサの環境が不可欠である．

今後のスーパコンピュータでは，ベクトル機能，マルチタスキングやデータフローなどの並列機能が利用可能になると考えられ，これらの機能の中で各問題に適した高速化をうまく使い分ける時代になるであろう．そして，将来はその使い分けも計算機が自動的に行うようになるべきである．

参 考 文 献

1) 物理学会編：スーパーコンピュータ，培風館 (1985).
2) 星野 力：PAX コンピュータ，オーム社 (1985).
3) 川合敏雄：シミュレーション的自然観と並列計算機，シミュレーション，Vol. 5, No. 2, pp. 64-70 (1986).
4) 唐木幸比古：スーパーコンピュータと行列計算，本号別章.
5) Hwang, K. and Briggs, F.: Computer Architecture and Parallel Processing, McGraw-Hill (1984).
6) 福田，佐々木：並列処理ミニ・スーパーコンピュータ DPS 9000，ファクトリ・オートーメーション，pp. 79-84 (Feb. 1987).
7) Nogi, T. and Kubo, M.: ADINA Computer I, The Memoirs of the Faculty of Engineering, Kyoto University, Vol. XL II, pp. 421-439 (1980).
8) Nogi, T.: ADINA Computer II, The Memoirs of the Faculty of Engineering, Kyoto University, Vol. XL III, pp. 124-144 (1981).
9) ボースック：並列処理コンピュータを情報ネットワークに活用，日経コンピュータ，Vol. 142, pp. 91-95 (1987).
10) Smith, G. D.: Numerical Solution of Partial

Differential Equations, 2nd ed., Oxford University Press, pp. 37-40 (1978).
11) 福井：問題の性質を利用した大規模スパース行列の高速解法，情報処理学会数値解析研究会資料 20-3 (1987).
12) 村田，小国，唐木：スーパーコンピュータ，丸善，pp. 180-185 (1985).
13) 福井，大吉，加藤，渡辺：マルチプロセッサ・システムにおける回路解析コードの並列処理，情報処理学会数値解析研究会資料，17-5 (1986).
14) 福井，佐々木，鈴木，佐藤：数値・数式融合計算のための FORTRAN と REDUCE の一結合方式，日本ソフトウェア科学会第3回大会論文集 D-6-2 (1986).
15) 梅谷他：数値シミュレーション用プログラム DEQSOL，情報処理，Vol. 26, No. 1, pp. 168-180 (1985).

(昭和62年7月2日受付)