NAL TR-895

ISSN 0389-4010
UDC 621. 438:
681. 3.02

航空宇宙技術研究所報告
TECHNICAL REPORT OF NATIONAL AEROSPACE LABORATORY

TR-895

高効率ガスタービンの制御に関する研究(1)
制御システム解析装置およびその性能

杉山七契・越沼　威・西尾健二

1985年12月

航空宇宙技術研究所
NATIONAL AEROSPACE LABORATORY

目　　次

高効率ガスタービンの制御に関する研究(1)
制御システム解析装置およびその性能*

杉山七契** 越沼威** 西尾健二**

"A Study on Control System of High Efficiency Gas Turbine AGTJ100 (I) — Control System Development Facilities and its Characteristics"

by
Nanahisa SUGIYAMA, Takeshi KOSHINUMA and Kenji NISHIO

ABSTRACT

A national R&D program on "High Efficiency Gas Turbine" is under way. The target of the program is to develop an intercool-reheat type gas turbine engine for 100-MW-class power generation with very high thermal efficiency, resulting in energy conservation. The requirement for engine control is severe due to complexity of engine cycle and geometry. Therefore, development of an engine controller, inherently sophisticated multivariable controller, is considered to be one of the most important items for success of the program.

This paper describes the basic plan, fabrication and characteristics of control system development facilities for the "High Efficiency Gas Turbine", installed at NAL. The facilities consist of a high speed digital simulator (AD10) for real-time gas turbine simulation, a peripheral array processor (MAP300) for matrix operation and time series analysis, and a general purpose super mini-computer (VAX11/750). These can be utilized effectively in various stages of development of a control system, such as design, evaluation and hardware test.

1. まえがき

本報告は，通商産業省・工業技術院の省エネルギー技術開発プロジェクト『高効率ガスタービンの研究開発』の一環として当研究所に設置した『高効率ガスタービン制御システム解析装置』の基本計画，ハードウェア，ソフトウェア，および予備試験について述べる。

『高効率ガスタービンの研究開発』は，最新の高効率化の技術をとりいれたガスタービンを中核とし，その排熱を蒸気タービンに供給する複合発電システムを構成し，総合熱効率55%（ＬＨＶ：低位発熱量換算）を達成しようというものである。効率の向上のため，高効率ガスタービンは，圧縮過程で中間冷却，膨脹過程で再熱を行ない，最高圧力5.5ＭＰａ，最高温度1300℃（第一次試作プラント），ガスタービン単体出力100MW，単体熱効率39%，という従来にみられない複雑かつ高度な熱サイクルを有している。また，高効率ガスタービンを制御対象としてみた場合，状態変数および制御変数に制限のついた非線形多変数系となっている。制御変数の数が多く（主要なものだけでも６変数），状態量および制御

* 昭和60年10月28日受付
** 原動機部

量に種々の制限が設定されており，作動特性は強い非線形性を有している，等，従来のガスタービンの制御に比べて格段に多くの制約条件が課せられている。高効率ガスタービンを効率よく，しかも安全に作動させる制御システムの開発は，この研究開発プロジェクトの中でも最も重要な研究課題の一つと考えられている。

ここで述べる『高効率ガスタービン制御システム解析装置』は，高効率ガスタービンの制御システムの開発のための tool であって，制御システムの設計，評価，動作テストなど開発のあらゆる phase で有効に利用されるよう計画されている。

2. 基本計画

2.1 制御システム開発手順

ガスタービンは種々の負荷の状況に応じた広範な出力状態において，効率よく安全に運転されなければならない。また，ガスタービンの起動，停止，負荷の急激な変動，等の過渡状態においても安全かつ安定に作動するものでなければならない。これらはガスタービン各部に取付けられた検出器の信号をもとにして，供給燃料，可動操作部を操作するガスタービン制御システムによって達成される。

試作されたガスタービンを最初に運転する時から制御システムは必要である。しかし制御システムの設計，製作の時点では，ガスタービン本体は存在せず，運転試験によって設計データを得ることができない。このため，ガスタービン本体の設計データに基づいて，数学的モデルを作り，計算機上でその静的，動的挙動を忠実に模擬する装置，すなわち，ガスタービン・シミュレータを用いて，制御システムの設計，評価，動作テストなどの一連の問題を解決するという手法をとる。ガスタービン・シミュレータを利用して制御システムを確立する手順は次の様に要約される（図１）。

(1) ガスタービン本体の設計データに基づいて数学的モデルを作成し，シミュレータに組込む（プログラミング）。この数学的モデルやデータは，ガスタービン本体の開発の進展にともない逐次，改訂されていく。

(2) ガスタービンの静特性，動特性データをシミュレータより収集し，これを基にして制御システムの設計を行なう。

(3) 設計された制御システムのモデルをシミュレータ内部，あるいは外部制御用計算機に作成し，(1)で得られたガスタービン部と結合して，制御システムの評価を行なう。

(4) 種々の制御システムについて(2)～(4)を繰り返し，安全性，効率の観点から最も良い制御システムを求める。

(5) 最終的に決定された最良の制御システムのハードウェアを製作し，ガスタービン・シミュレータと結合して，閉ループ動作テストを行なう。必要に応じて(2)～(5)を繰り返し，実機のガスタービンに装着しうる制御システムを得る。

(1)～(3)の段階ではシミュレータの演算速度は大きな

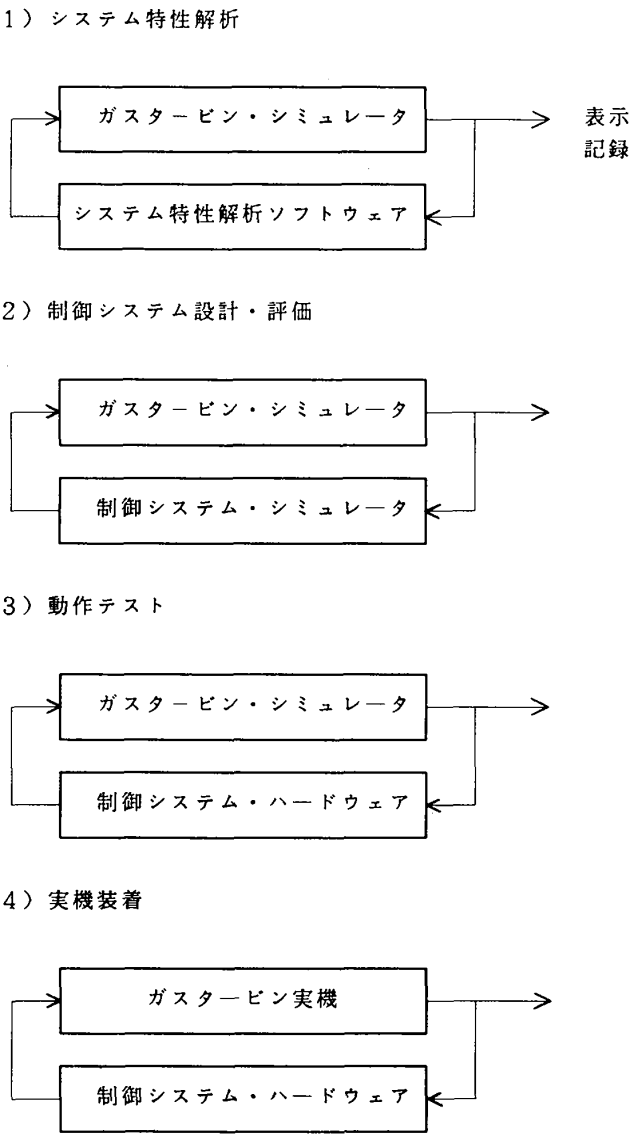


図１ 制御システムの開発手順

問題とはならないが，(5)の段階では演算が実際の現象と同じ速度で進行すること，すなわち，実時間動作をすることが最も重要な条件となる。

上記の(1)～(3)の段階をより詳細に記述するとおよそ次の様になる（図2参照）。

(1) 全作動範囲にわたって忠実に実機をシミュレートする詳細モデルを作成し，プログラム化する。

(2) 詳細モデルから線形モデルを導出する。詳細モデルの非線形性のため，種々の代表的作動点について多くの線形モデルが得られる。また，この線形モデルの次元は制御システムの設計に対して必要以上に高次である場合が多く，適当な次元低下法で低次元線形モデルとする。

(3) 低次元線形モデルについて制御システムの設計を行なう。制御対象が線形化されているので古典あるいは現代制御理論によって制御システムは設計され得る。

(4) 設計された制御システムと線形モデルとを結合して，制御システムの評価を行なう。制御要求を満たし，良好な制御システムが得られるまで(3)～(4)を繰り返す。

(5) (4)で得られた制御システムと非線形詳細モデルとを結合して，制御システムの評価を行なう。この場合，線形化により設計された制御システムを非線形系に適応するために何等かの手段が必要になることが多い。全作動範囲で良好な制御性を示すシステムが得られるまで(2)～(5)を繰り返す。

ここではシミュレーションの演算の他に，行列演算，多量のデータ処理演算，結果の図形出力などの演算処理が必要となる。

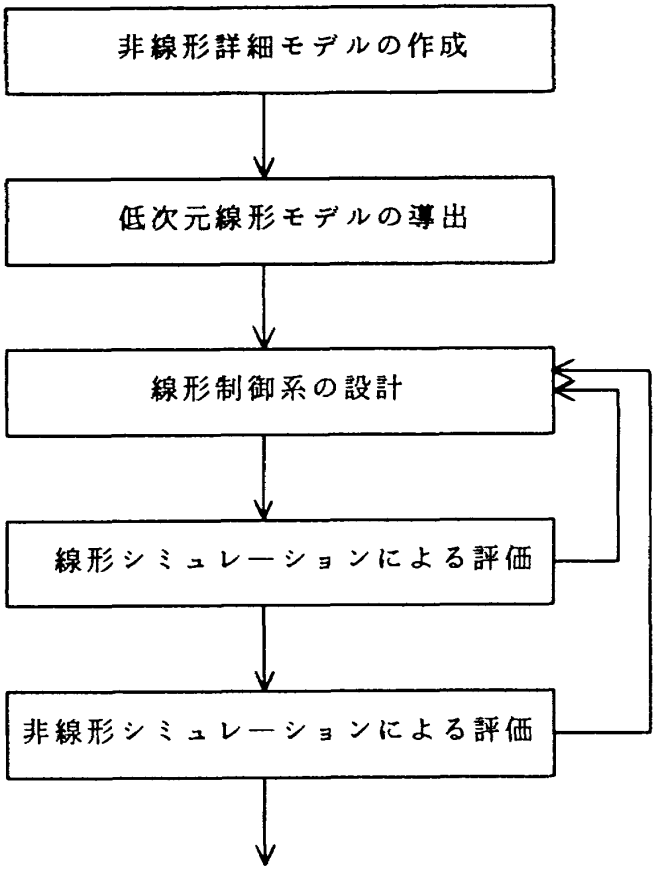


図2　制御システムの設計手順

2.2　高効率ガスタービン[1,2]

図3は高効率ガスタービンの構成ブロックおよびヒート・バランスを示す。ロータは低圧軸，高圧軸の2軸構成で，高圧軸には起動モータ，低圧軸には起動モータおよび100 MW級同期発電機が結合される。空気取り入れ口を通過した空気は低圧軸上の10段軸流低圧コンプレッサによって昇圧される。低圧コンプレッサは全段可変静翼で空気流量制御を行なう。低圧コンプレッサを通過した空気はダクトで中間冷却器に導かれる。このダクトには負荷遮断時の過回転をさけるための放風弁が設けられている。中間冷却器では水噴射により空気温度を下げ，次段の高圧コンプレッサの駆動パワーの低減に寄与する。水噴射は中間冷却器出口で相対湿度90%を目やすにして行われ，冷却された空気はダクトで高圧軸上の16段軸流高圧コンプレッサに導かれる。高圧コンプレッサの入口案内翼は可変翼で起動時の流入角制御を行う。また，6段，8段，11段および最終段で抽気を行い，タービン冷却，起動時ブリード，バランス・エアとして利用される。高圧コンプレッサで昇圧された空気は缶型主燃焼器に導かれ，天然ガスの燃焼により，設計点で1300℃，5.5 MPaの燃焼ガスとなる。このガスは高圧軸上の2段軸流高圧タービンを通過し高圧軸の回転トルクを発生する。さらにガスはダクトを経て低圧軸上の2段軸流中圧タービンを通過し缶型再燃焼器に導かれ，再び天然ガスの燃焼により設計点で1200℃，0.88 MPaの燃焼ガスとなる。再燃焼は次段の低圧タービンの発生パワーの増加に寄与する。燃焼ガスは低圧軸上の4段軸流低圧タービンを通過し，中圧 タービンとと

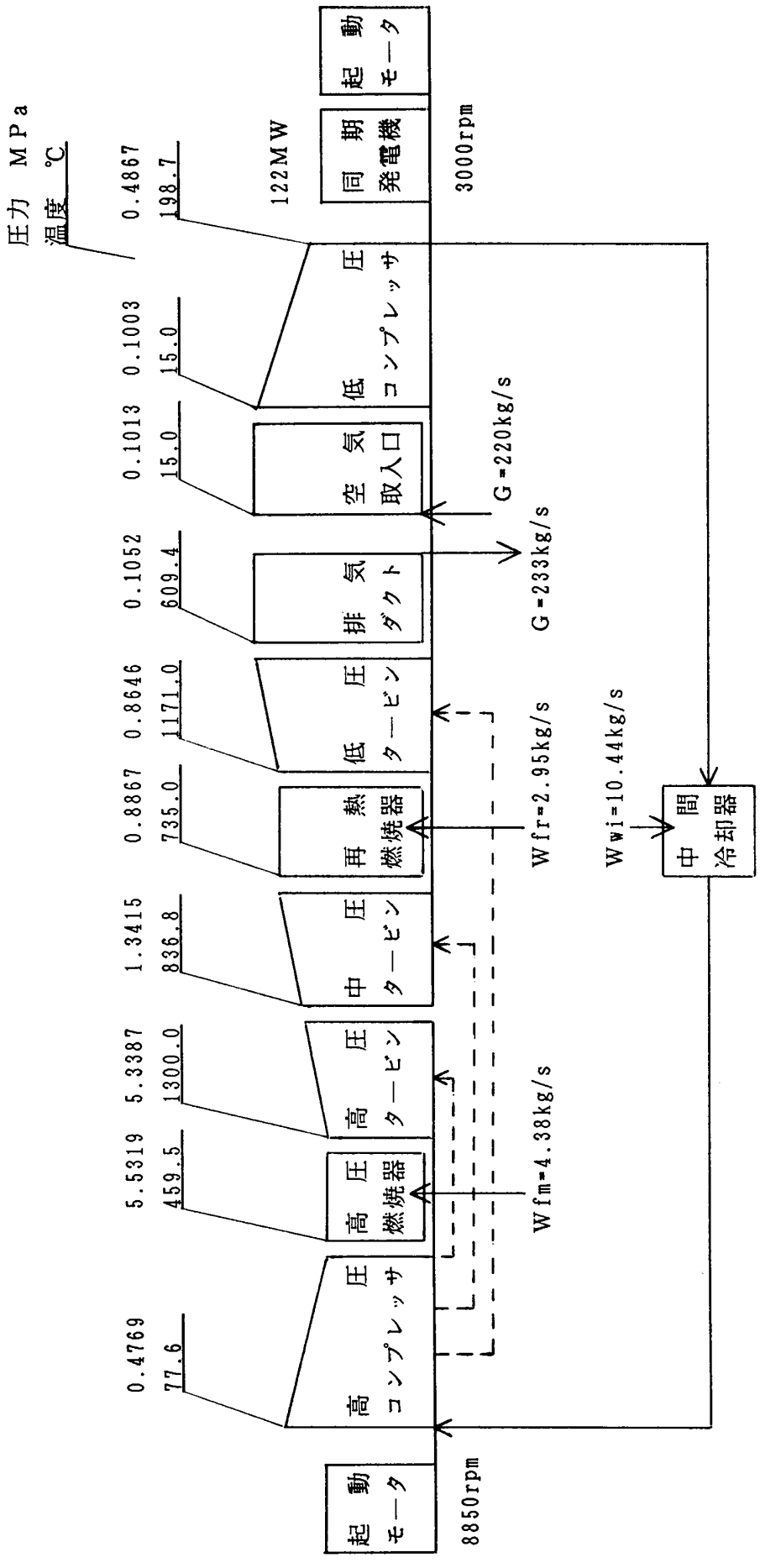


図3 高効率ガスタービンの構成ブロックおよびヒートバランス

もに低圧軸の回転トルクを発生する。低圧タービン出口でガス温度はボトミング・サイクル（蒸気タービン・サイクル）を考慮して609℃と高く設計されている。

この様な熱サイクルの採用により，『高効率ガスタービン』は，ⅰ）定格点の熱効率，ⅱ）部分負荷時の熱効率，ⅲ）起動時間，ⅳ）その他，冷却水，設置面積，等の点で従来の火力発電プラントに比べ格段に有利であるとされている。

2.3　ガスタービン・シミュレーション

上述の様に高効率ガスタービンは極めて複雑な機械的構造，熱サイクルを有しており，その数学的モデルは，多数の非線形多変数関数，高次の非線形連立微分方程式で構成される。通常，ガスタービン・エンジンはエンジン要素特性，熱力的空力的関係式といくつかの保存法則に基づいてモデル化される。図3に示した構成のガスタービンをシミュレーション計算モシュールで表わすと図4の様になる。ここで計算モジュールはコンプレッサ，タービン等のエンジン要素に対応している。ただし，高圧コンプレッサは4つのコンプレッサを直列に結合した構成としてあるが，これは中間段の抽気の取り扱いを容易にするためである。図4には計算モジュールの使用回数，制御変数が一覧してある。各計算モジュールの演算内容はおよそ表1に示す様であるから，高効率ガスタービン・シミュレーションの全体規模は表2に示す様であると考えられる。このシミュレーション規模は航空用2軸ファン・エンジンの場合の約2倍であり，ガスタービンのシミュレーションとして極めて大規模なものとなる。このシミュレーショ

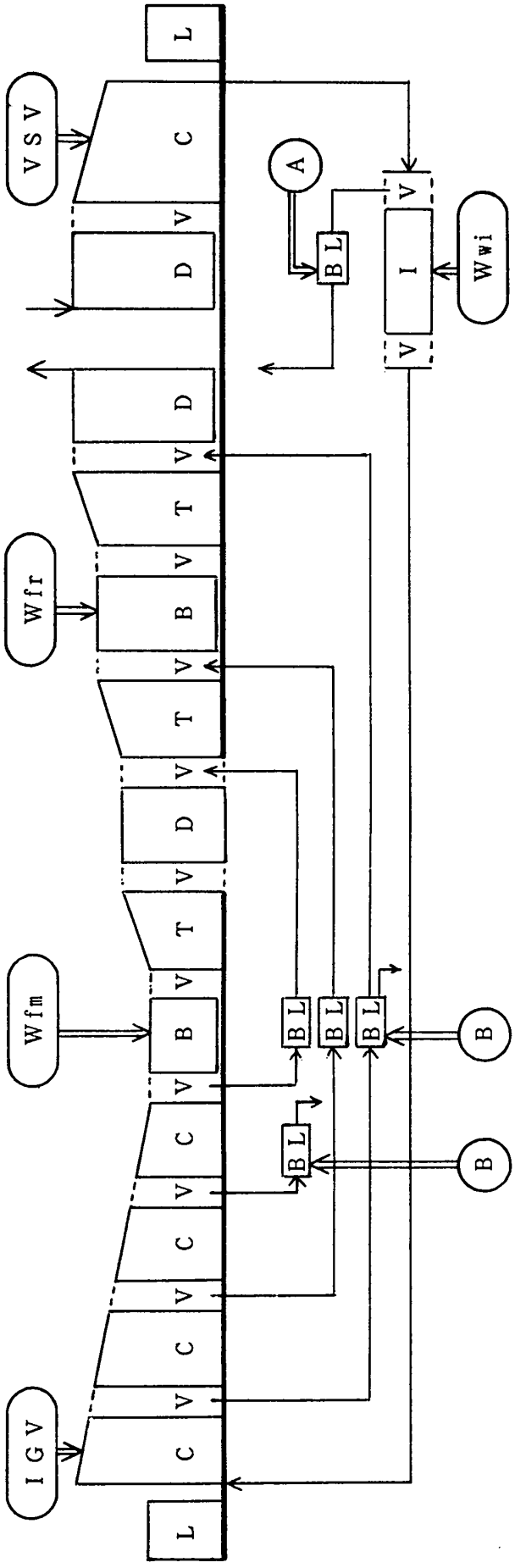
表1　計算モジュールの演算内容

計算モジュール	関数発生の回数 1変数関数	2変数関数	3変数関数	積分演算の回数	変数の数
コンプレッサ	2	0	5	0	9
タービン	2	0	5	0	9
燃焼器	1	0	1	0	6
ダクト	1	0	1	0	4
中間冷却器	1	1	1	0	6
ブリード	2	3	3	0	6
ロータ	1	0	0	1	2
容積	4	1	2	2	8
負荷，モータ	1	0	0	0	2

表2　高効率ガスタービンのシミュレーション予測規模

積分演算回数	28
変数の数	244
関数発生回数	
1変数関数	89
2変数関数	28
3変数関数	87
関数発生テーブル数	
1変数関数	7
2変数関数	3
3変数関数	22
積分きざみ幅（msec）	2

主要な関数発生テーブルのデータ数	
低圧コンプレッサ	4K
高圧コンプレッサ1～6段	2K
高圧コンプレッサ7～8段	1K
高圧コンプレッサ9～11段	1K
高圧コンプレッサ12～16段	1K
高圧タービン	1K
中圧タービン	1K
低圧タービン	1K
ガス・テーブル	5K
合計	17K



制御変数	
Wfm	：高圧燃焼器燃料流量
Wfr	：再熱燃焼器燃料流量
VSV	：低圧コンプレッサ静翼角度
IGV	：高圧コンプレッサ入口案内翼角度
Wwi	：中間冷却器水噴射量
A	：放風弁開度
B	：ブリード面積

計算モジュール	使用回数
C：コンプレッサ	5
T：タービン	3
B：燃焼器	2
D：ダクト	3
I：中間冷却器	1
BL：ブリード	5
R：ロータ	2
V：容積	13
L：負荷、モータ	2

図4 シミュレーション計算モジュール

ンの特徴として，ｉ）大規模（数十の状態変数と数百の従属変数，ii）スティッフ（広い周波数分布，iii）非線型（多数の実験的及び解析的多変数関数），があげられる。これは航空宇宙や原子力等の最近の先端技術分野のシミュレーションがもっているのと同じ特徴である。

2.1～2.2に述べた事柄をも勘案して，高効率ガスタービンのシミュレーションには次の事項が必要とされる。

(1) 全作動領域での忠実なシミュレーション……起動から全負荷まで，予測される負荷条件，大気条件，燃料特性変動範囲，制御変数変化範囲において，ガスタービンの静特性および動特性を高精度で忠実にシミュレートすることが必要である。

(2) 実時間動作……シミュレータ外部の機器との接続を現実的なものとするため，シミュレータは実時間で動作しなければならない。制御問題を取扱う場合必要となる周波数帯域は0～10Hzと考えられるので，動的過程での演算精度を維持するためにディジタル演算の周期（frame time）は数msec程度が必要である。また，システムはスティッフであるため高い積分精度を必要とする。

(3) 外部の機器との接続……シミュレータは制御器や記録器，ディスプレイ，等の外部機器と接続できることが必要である。これは制御器の閉ループ動作テストを可能にし，シミュレーション過程のモニタや結果の記録に利用される。接続方法はアナログ信号によるものが最も容易である。

(4) 要素特性の変更……ガスタービンの要素特性や各種定数は，開発の各 phase で逐次改訂されていく。これを即刻シミュレーションに反映させるため，全シミュレーションに影響を及ぼすことなく，容易に改訂がなされなければならない。

現在のシミュレーション技術はほとんどハイブリッド方式か全ディジタル方式かに限定されている。演算請度，信頼性，融通性，プログラミング法，等の点で，全ディジタル方式が有利であるが，演算速度が遅いという欠点をもっている。このため，実時間シミュレーションの様に演算の高速性を要求される場合，全ディジタル方式では実現困難であり，ハイブリッド方式を採用することが多い。しかし，ガスタービンのシミュレーションの様に演算の大部分が多変数関数発生（表2参照）である場合，ハイブリッド方式でもそれはディジタル演算部で行なわざるをえず，ハイブリッド方式の高速性を有利に利用できない。従って，高効率ガスタービンのシミュレータとして最適なものは超高速全ディジタル方式であると考えられる。筆者らの経験によると，この場合，一般の汎用ミニ・コンピュータの約100倍の演算速度が必要となるが，近年のディジタル技術の進歩はこれを実現している。4.に詳細を述べるAD10システムがそれで，高精度ディジタル演算，超高速の多変数関数発生，高精度数値積分，高速多チャンネル・アナグロ／ディジタル入出力，ストアド・プログラム，等の機能を有し，上に述べた要求事項を全て満足する唯一のシステムである。

2.4　高速演算機能

制御システムの開発の各 phase において，シミュレーション演算以外に次の様な特殊演算を行なうべきケースは少なくない。

(1) 行列演算……ベクトル演算，逆行列，行列式，固有値，連立方程式

(2) 信号処理演算……相関，たたみこみ，フィルタリング，分散

(3) フーリエ変換……実数変換，複素数変換，逆変換

高効率ガスタービンの場合，状態変数の数は28（表2参照）であるので，センサ，アクチュエータのダイナミックスを考慮しても，システムの次元は50次元を越すことはないと考えられ，行列演算の規模は50×50行列を目やすとする。また，信号処理およびフーリエ変換のデータ点数は最大 4096 点程度であると考えられる。

これらの演算は汎用コンピュータで実行可能であり，多くのソフトウェアも発表されている。しかし，制御システムの開発の際に生ずる大量のデータ処理を考える場合，演算時間，メモリ容量の点で負荷が大きく効率的でないため，専用演算装置の設置がのぞましい。これはペリフェラル・アレイプロセサ（peripheral array processor）と称する装置で実現される。ペリフェラル・アレイプロセサは種類が多く，その特徴も一様ではないがおよそ次の

ことが言える。

(1) ホスト計算機（大部分はミニ・コンピュータ）に接続された周辺機器（peripheral）であり，ホスト計算機からのコマンドで動作する。

(2) 高速の行列演算，FFT演算を行なう。この演算は高度に構造化されるため，並列処理やパイプライン処理の採用により演算の高速化をはかっている。

(3) 装置固有の言語（アセンブラ言語）をもっているが，通常その知識は必要でない。ユーザはホスト計算機上にFORTRAN等の高級言語で処理目的にあったプログラムを自由に作成することができる。

現在，この種のペリフェラル・アレイプロセサとして十数機種が発表されている。ホスト計算機との接続，演算機能，演算精度（演算方式，語長），演算速度，価格，等を考慮すると利用可能な機能は多くなく，MAP300（CSPI社）およびAP120B（Floating Point社）の2機種が選択される。両機種のコスト・パフォーマンスを比較すると前者の方が若干良いため（図6参照），MAP300システムを採用することとした。これにより前記の特殊演算は，ホスト計算機のみで実行した場合の50～100倍の処理速度を有することになり，制御システム開発の効率化に大きく寄与すると考えられる。

3. 全体構成

前章で述べた基本計画にそって構成した『高効率ガスタービン制御システム解析装置』の全体機器構成を図5に示す。ホスト計算機（VAX 11/750）および基本的周辺機器（ディスク，磁気テープ，プリンタ・プロッタ，ターミナル・ライン）は『原動機データ処理装置』として既設のものを利用している。この原動機データ処理装置は，各種オンライン実験計測，データ処理，技術計算，等のために約40名のユーザが使用しているスーパー・ミニ・コンピュータ・システムで，マルチユーザ，マルチタスク，インタラクティブTSSシステムとして稼動している。

『高効率ガスタービン制御システム解析装置』の主要部は，ガスタービン・シミュレータ（AD10システム）と高速演算装置(MAP300システム）で，それぞれ単独では動作することはできず，共にホスト計算機のサポートを必要とする。つまり，これらのシステムのプログラム作成，データ作成，実行制御，診断，等は全てホスト計算機（VAX11/750）からのコマンドでなされる。ホスト計算機のオペレーティング・システム（OS）は，マルチユーザ，TSSシステムであるので，これらのシステムとホスト計算機が交信中は，他のユーザのプログラム実行速度は若干低下するが，交信時間は短いので，決定的なシステム欠陥とはならない。むしろ，ホスト計算機およびその基本的周辺機器の膨大なソフトウェア，ハードウェアを有効に利用しうる利点の方がはるかに大きい。

AD10システムとMAP 300システムとは前記の様に演算機能を異にするが，次の様なハードウェア構成上の類似点をもっている。

(1) 周辺装置……ホスト計算機VAX11/750の周辺装置として接続される。

(2) マルチ・プロセッサ……独立した複数の専用高速プロセサ（インターフェイス・プロセサ，制御プロセサ，演算プロセサ，入出力プロセサ，等）を内蔵しており，並列処理が行われる。各プロセサはそれ自身のプログラム・メモリとともに，中間処理結果を一時的に格納するための高速メモリをもっている。

(3) 高速演算……高速ディジタル回路を採用してサイクル・タイムが極めて速く（AD10は100 nsec MAP 300は200 nsec），さらにパイプライン処理を組込み，演算の高速化を図っている。

(4) データ・メモリ……大容量の低速メモリをもっており，入出力データや演算結果の格納を行う（AD10の最大容量は2Mバイト，MAP 300の最大容量は768Kバイト，メモリサイクル・タイムは共に500 nsec）。

この様な類似のハードウェア構成をもつ装置はペリフェラル・プロセサ（peripheral processor）と呼ばれ，比較的小規模なホスト計算機の性能向上のために用いられる。ホスト計算機（VAX11/750），ガスタービン・シミュレータ（AD10）および高速演算装置（MAP 300）のコスト・パフォーマンスは図6に示す様に位置付けられている[3]。これらの機

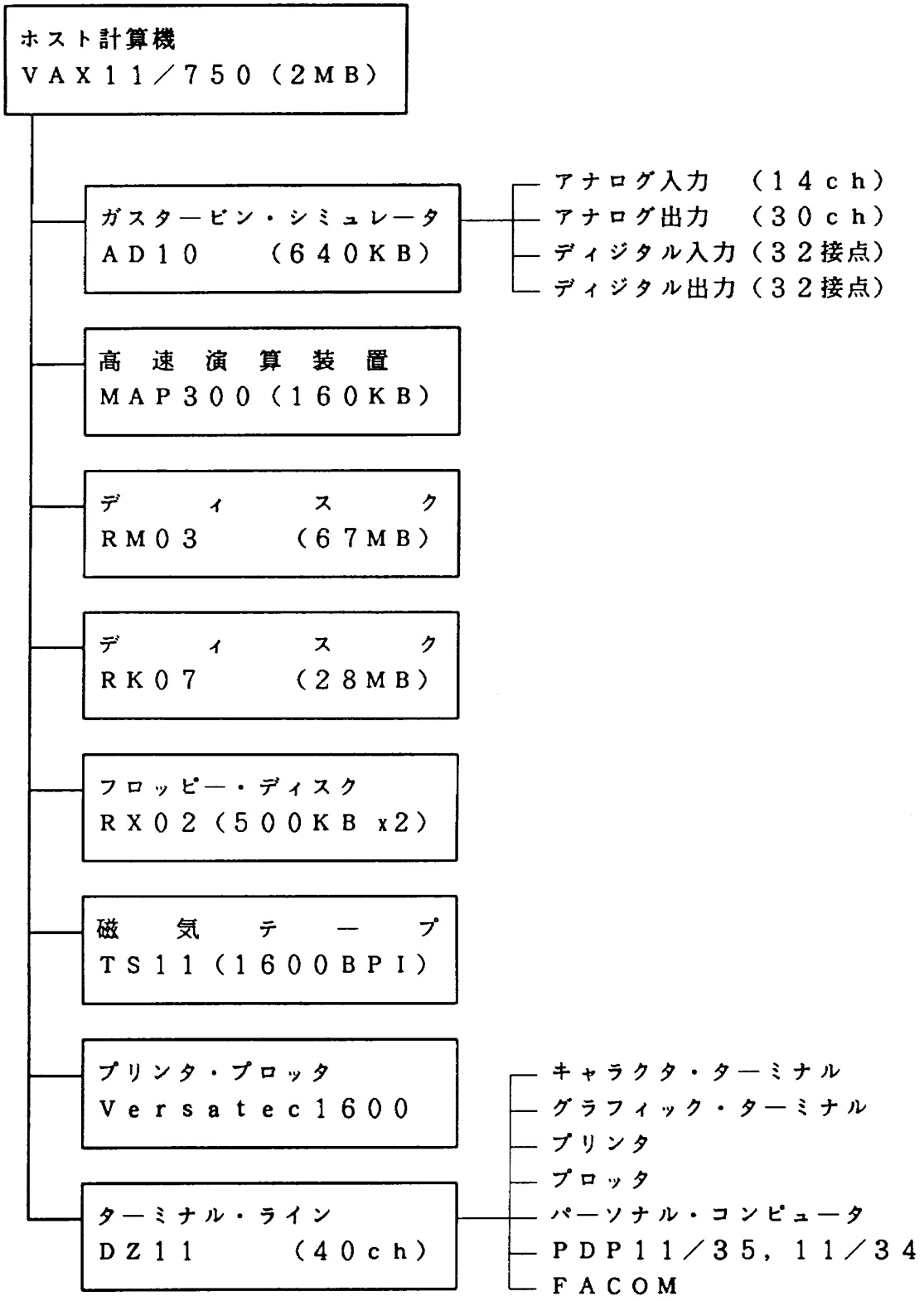


図５　システム構成

器の組合せにより，大型計算機並の演算処理能力をもちながら，小規模・低価格のシステムを構成することができる。

4.～5.において上記のＡＤ10システムおよびＭＡＰ300システムについて，その機能，性能，ハードウェア，ソフトウェアの詳細を述べる。これらは数年度にわたって逐次設置されたものであり，機能拡大のための増設が繰り返されてきている。また，将来にも，ハードウェア，ソフトウェアの増強が必要になる可能性もあるため，システム構成はかならず

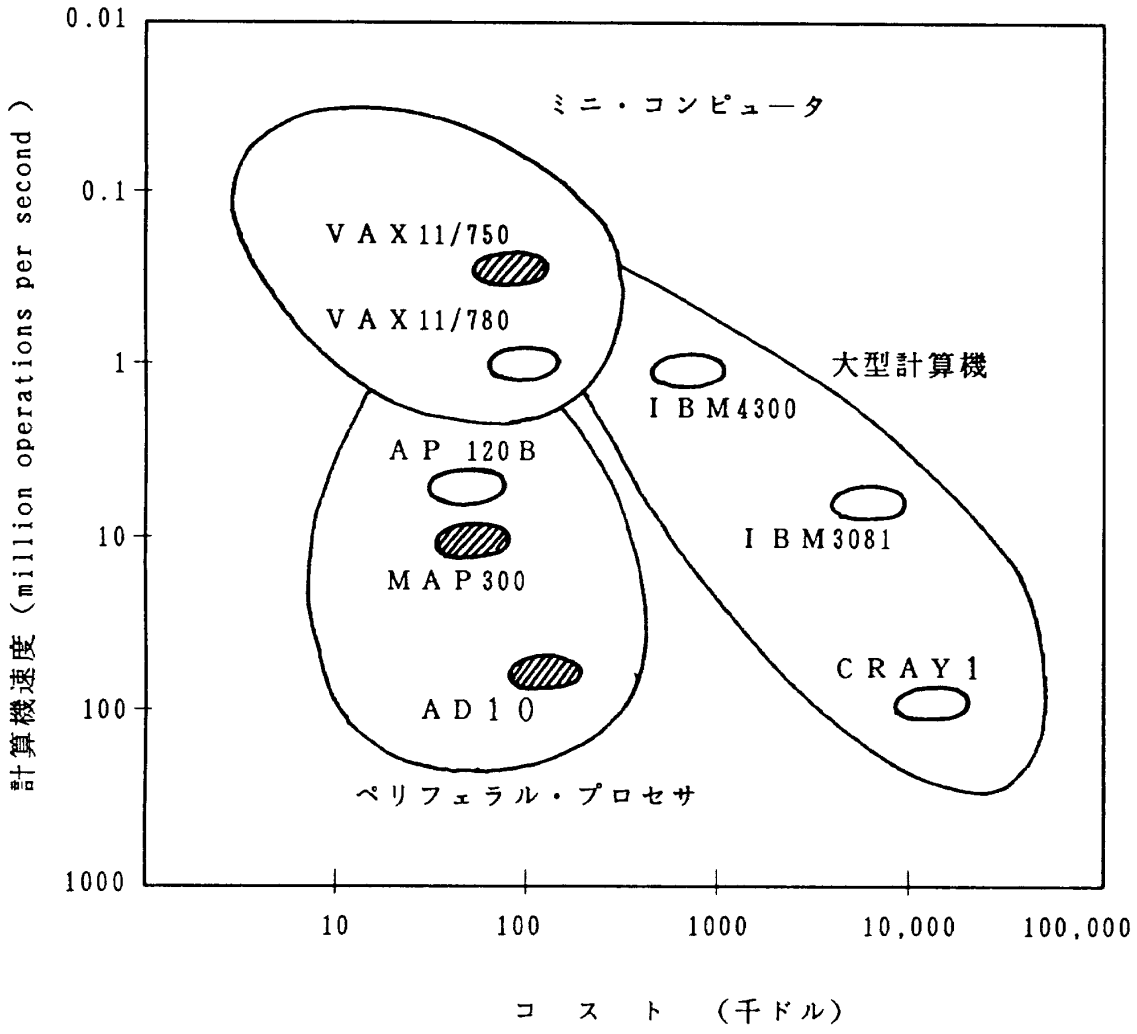


図6 各種計算機のコスト・パフォーマンス[3]

しも一定ではない。以下の説明は，現状のシステム構成にそって進める。

4. ガスタービン・シミュレータ（AD10）

AD10システム[4~9]は高速ディジタル回路，並列演算処理，パイプライン，等の新しいディジタル処理技術を導入し，ディジタル・シミュレーションの唯一の欠点である演算速度を飛躍的に向上させたシミュレーション用特殊ディジタル計算機である。シミュレーションにおいて必要となる演算機能は，i）多変数関数発生，ii）積分，iii）入出力，iv）その他の一般計算，等であるが，AD10にはこれ等の機能が適切に具備されており，大規模システムの実時間シミュレーション，あるいは実時間以上の高速シミュレーションが可能になっている。

AD10の機能の内，特に顕著なものは，多変数関数発生の高速性で，例えば，3組の3変数関数発生（3組の異なった関数を同時に発生すると，パイプラインは最も効率的に作動する）は9.6μ secで完了する。この関数発生の高速性および大容量低価格データメモリの存在は，シミュレーション手法に若干の変革をもたらしてくる[5]。例えば，sin(x)，$\sqrt{x}$等の演算は従来のべき級数近似による計算よりも，前もってデータ・テーブルを用意しておき，シミュレーション実行時には関数発生のみによって計算した方が効率的となる。また，2～4変数の複雑な計算式は，同様に多変数関数のデータ・テーブルとして用意しておき関数発生によって計算した方が効率的となる。この様に，AD10では，シミュレーションに現われる演算はできるだけ関数発生（あるいは，テーブル・ルックアップ）に置き換えることにより，より効率的なシミュレーションが可能になる。

4.1　システム構成

図7はAD10のシステム構成図である。マルチバスと呼ばれる高速データ/コントロール・バスを中心に，次の5種の独立した専用プロセサ，2種の専用コントローラ，および大容量データメモリが並列に接続されている。

(1) 制御プロセサ（COP）……AD10内の全てのプロセサ，コントローラの制御を行う。

(2) メモリ番地プロセサ（MAP）……データメモリへの書込み読出しを行う。データメモリのフィジカル番地は，直接アドレッシングおよびインデックス・アドレッシングのいずれかのモードで指定される。

(3) 論理判定プロセサ（DEP）……データの比較，判定を行い，MAPのインデックス・アドレッシングのためのインデックス・データを発生する。これは関数発生におけるバイナリ・サーチ（二分探索）を極めて効率化する。

(4) 演算プロセサ（ARP）……16ビット固定小数点型の数学的演算を高速で行う。特に，線形内挿の基本式である　±（A±B）×C±D　の形の演算は175 nsec で完了し，パイプライン構造のために，この演算結果は 100 nsec 毎に次々得られる。つまり，1μsecに20の加算と10の乗算を行う。

(5) 数値積分プロセサ（NIP）……48ビットの精度で数値積分を行う。積分方式はオイラー法をはじめルンゲ・クッタ法，アダムス・バシュフォース法，等17種であり，実行時間は0.3～2.6 μsecである。

(6) 入出力チャンネル・コントローラ（IOCC）……12ビット・バイポーラAD変換，DA変換およびディジタル（接点）入出力（16接点/チャンネル）を行う。AD変換時間は10μsec，DA変換のセット時間は3μsec である。

(7) ホスト・インターフェイス・コントローラ（HIC）……ホスト計算機VAX11/750とAD 10との間のデータ，命令の授受を行う。

(8) データメモリ（DM）……関数発生のためのデータ・テーブル，一般データ，解データ，等の格納を行う。16 KW単位（ページ）で1 MWまで増設可能。AD10の高速データバスの転送速度にみあう高速バイポーラRAM（サイクルタイム 50 nsec）は価格，消費電力，実装方法の点で不利であるため，低速MOS・RAM（サイクルタイム 500 nsec)を採用している。図7に示す様に，データメモリはマルチバスに対して，ページ毎に独立しており，データメモリへの書込み読出しをオーバラップさせることができ，これで低速性はカバーすることができる。

5つのプロセサ(1)～(5)は，演算時間向上のために，シミュレーション作業の内，種々の異なった部分的作業を分担し，並列に動作する。各プロセサはそれぞれ自分自身のプログラム・メモリとプログラム・カウンタを持っており，同期信号に同期してそれぞれのプログラムを10 MHz（100 nsec）のレートで進行させる。命令語長（プログラム・メモリのビット長）はプロセサによって32～80ビットと異なっているが，全プロセサを合計すると 272 ビットに及んでいる。1インストラクションの実行時間は 100 nsec であるが，各プロセサ内部では，この時間幅をさらにスライスして，25 nsec単位で命令の実行，データの移動が行われる。

演算の高速化の観点から，AD10内の全プロセサは同期方式で結合されており，プロセサ間のデータ授受，通信におけるシェイクハンド手続きは不用となっている。これは非同期方式のゆるい結合に対して，きつい結合と呼ばれているが，演算が高速という利点とソフトウェアが複雑という欠点を合わせ持っている。一方，AD10外部とはホスト・インターフェイス・コントローラ（HIC）を介して非同期方式で結合されている。

4.2　高速関数発生の方法

多くのシミュレーションにおいて，関数発生を必要とする部分は極めて多い。例えば，ガスタービンのシミュレーションでは，関数発生演算は全シミュレーション演算の70～80%に及んでいる。また，この関数の多くは，実験データに基づく1～4変数の多変数関数（データ・テーブル）である。このため，関数発生の高速化はシミュレーションの高速化に直接寄与し，実時間シミュレーションや実時間以上の

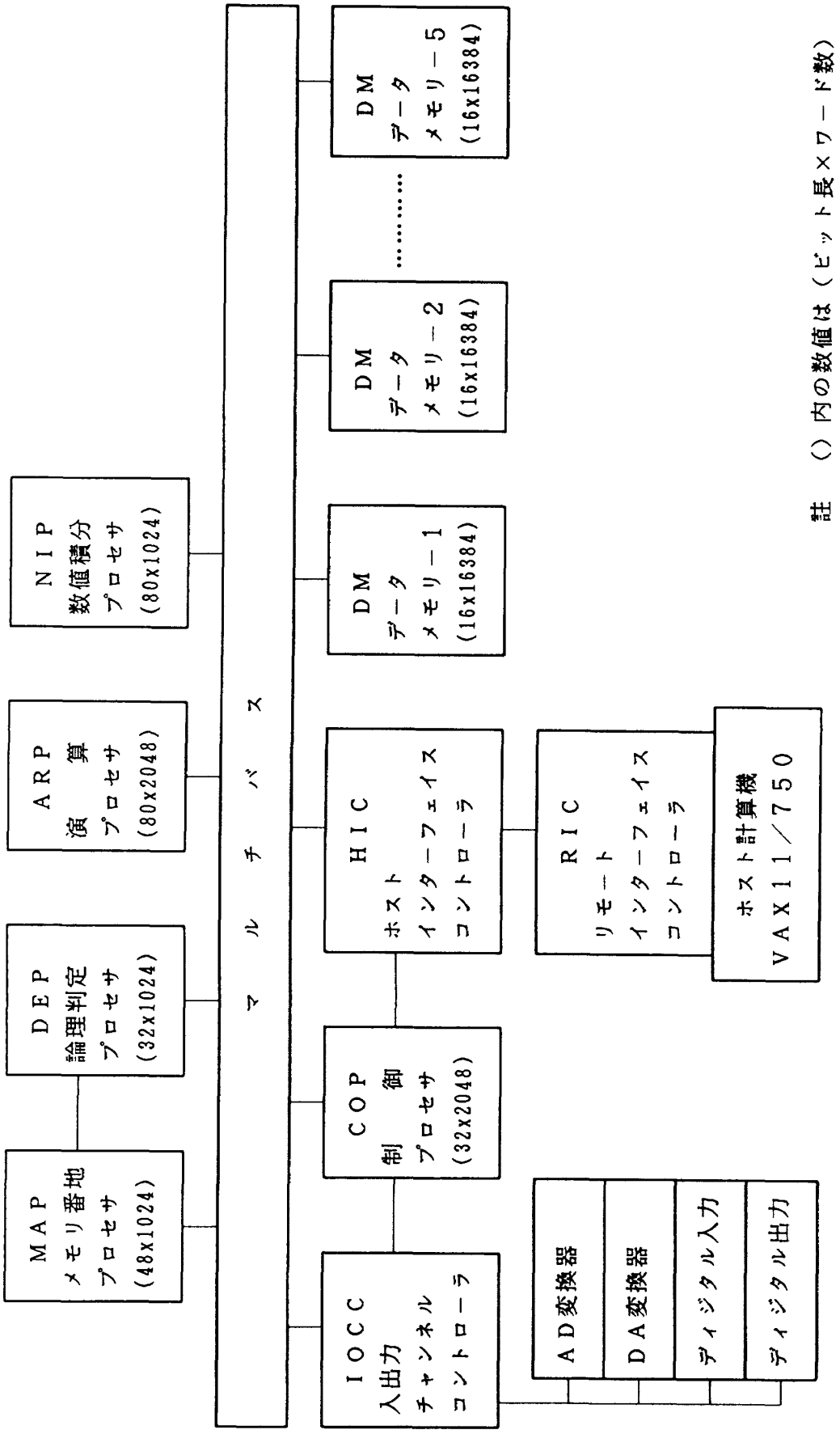


図7 AD10システム構成図[8]

高速シミュレーションを可能にするばかりでなく，より複雑で忠実なシミュレーション・モデルの採用を可能にし，シミュレーションの周波数帯域を拡大する。

関数発生のアルゴリズムは，ｉ）関数データを与えた点（ブレーク・ポイント）ではそのデータに一致させ，ii）それ以外の点では近傍のデータを用いた線形内挿で求める様にする。簡単のため３変数関数の場合についてアルゴリズムを説明するが，ｎ変数関数の場合への拡張は容易である。関数 $f(x,y,z)$ はブレーク・ポイント（x_i，y_j，z_k）で関数値 $f(x_i, y_j, z_k)$ が与えられているとする。

$$f_{000} = f(x_i, y_j, z_k)$$
$$f_{100} = f(x_{i+1}, y_j, z_k)$$
$$\vdots$$
$$f_{111} = f(x_{i+1}, y_{j+1}, z_{k+1}) \quad (1)$$
$$X = (x-x_i)/(x_{i+1}-x_i)$$
$$Y = (y-y_j)/(y_{j+1}-y_j)$$
$$Z = (z-z_k)/(z_{k+1}-z_k) \quad (2)$$

ただし，

$$x_i \leqq x \leqq x_{i+1},\ y_j \leqq y \leqq y_{j+1},\ z_k \leqq z \leqq z_{k+1} \quad (3)$$

ここで，式(2)の（Ｘ，Ｙ，Ｚ）をデルタ値，式(3)を満足する（ｉ，ｊ，ｋ）をインデックス値と呼ぶことにする。任意の点（ｘ，ｙ，ｚ）における関数値は線形内挿により，

$$\begin{aligned} f(x,y,z) = {} & f_{000}(1-X)(1-Y)(1-Z) + f_{100}X(1-Y)(1-Z) \\ & + f_{010}(1-X)Y(1-Z) + f_{001}(1-X)(1-Y)Z \\ & + f_{110}XY(1-Z) + f_{101}X(1-Y)Z \\ & + f_{011}(1-X)YZ + f_{111}XYZ \end{aligned} \quad (4)$$

と表わされる。式(4)の計算は，項の並び替えで種々の計算手順を生ずる。計算回数，計算構造，計算精度，等を総合的に評価して，次のアルゴリズムが最も有利である[6]。

$$g_{ij} = f_{ij0} + (f_{ij1} - f_{ij0})X \quad (5)$$
$$h_i = g_{i0} + (g_{i1} - g_{i0})Y \quad (6)$$
$$h = h_0 + (h_1 - h_0)Z \quad (7)$$

計算手順は，ｉ）独立変数（ｘ，ｙ，ｚ）の入力，ii）式(3)を満足するインデックス値（ｉ，ｊ，ｋ）の探索，iii）式(2)でデルタ値（Ｘ，Ｙ，Ｚ）を計算，iv）式(5)で４つのｇを計算，ｖ）式(6)で２つのｈを計算，vi）式(7)でｆを計算，vii）結果の出力，となる。この計算の内，ii）において，入力された（ｘ,ｙ,ｚ）についてブレーク・ポイント・テーブルをサーチして，式(3)を満たすインデックス値（ｉ，ｊ,ｋ）を探索する演算と，iv）～vi）において，±（Ａ±Ｂ）×Ｃ±Ｄの形の演算とが線形内挿演算の主要部になる。

インデックス値（ｉ，ｊ,ｋ）の探索は，ブレーク・ポイント・テーブルの作り方によりいくつかの方法が考えられる。

(1) ブレーク・ポイントが等間隔（equally spaced）である場合。

$$x_{i+1} - x_i = S = \text{constant},\ (i=0,1,2,\cdots\cdots N-1) \quad (8)$$

従って

$$x_i - x_0 = iS, \qquad (i=1,2,3,\cdots\cdots N) \quad (9)$$

となる。任意の $x(x_i \leqq x \leqq x_{i+1})$ について，

$$\begin{aligned} (x-x_0)/S &= (x-x_i)/S + (x_i-x_0)/S \\ &= \Delta + i \end{aligned} \quad (10)$$

ここで，$\Delta = (x-x_i)/S < 1$

つまりｘからｘ。（データ・テーブルの基準値）を引いて，定数（１／Ｓ）を掛けた時，整数部がインデックス値となり，小数部がデルタ値となる。

(2) ブレーク・ポイントが等間隔で，しかもその間隔が２のべき乗である場合。

(1)において，

$S = 2^m$

とすると，定数（１／Ｓ）を掛けることは，ｘをｍビットだけシフトさせることと等価であるので，より単純な操作でインデックス値とデルタ値が求まる。

(3) ブレーク・ポイントが不等間隔である場合。

これが最も一般的な場合で，関数値変化が急激な部分ではブレーク・ポイント間隔を細かく，また緩慢な部分では粗くとる様にすると，ブレーク・ポイントの点数を増加させずに線形内挿の精度が向上する。この場合，インデックス値の探索アル

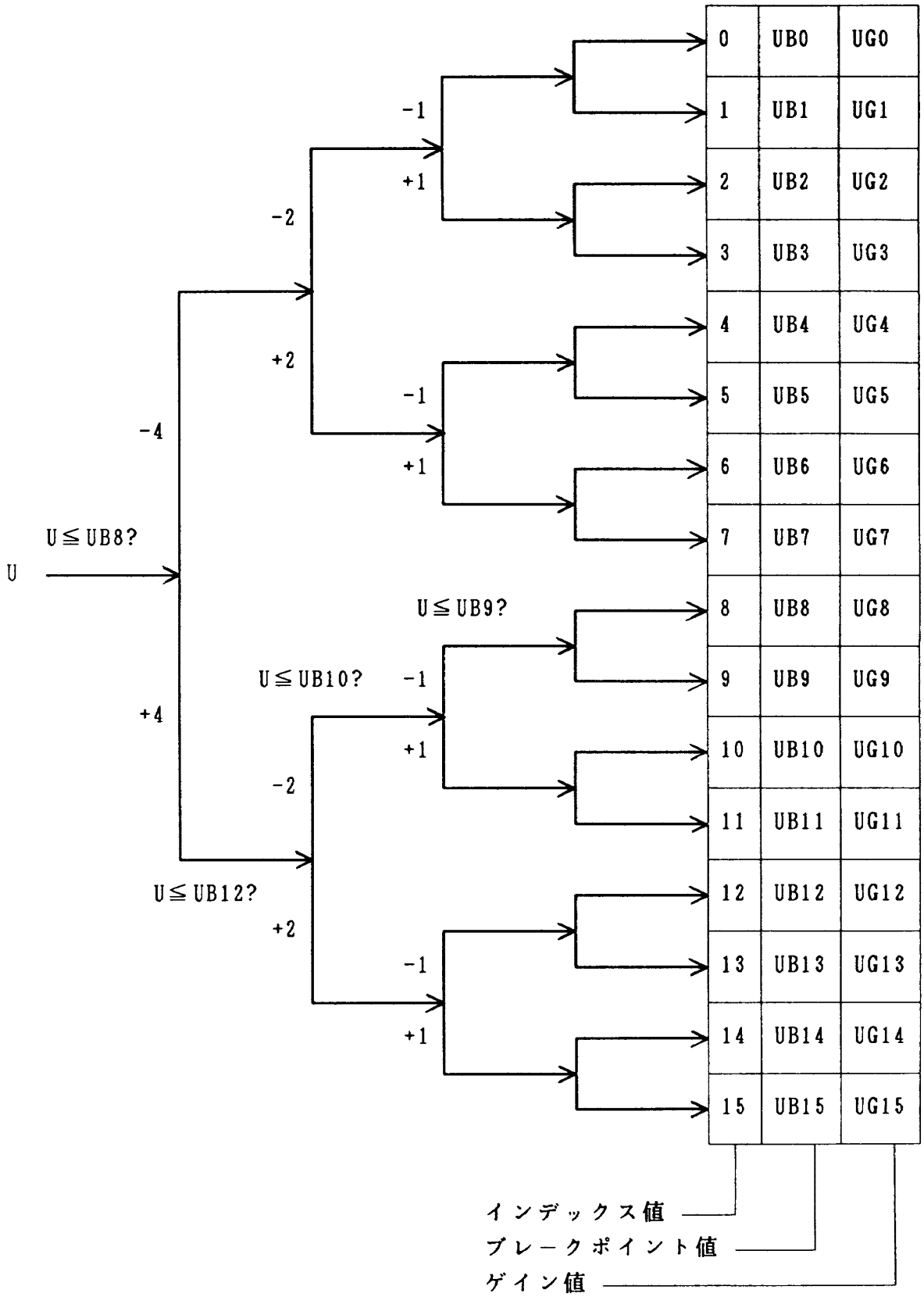


図 8　バイナリ・サーチ

ゴリズムは種々あるが，極めて効率的な方法がバイナリ・サーチ法（二分法）である。図8はブレーク・ポイントが16点の場合の手順を示しているが，その探索スキームは図から明らかであろう。デルタ値は式(2)によって一々求めてもよいが，式(2)の内，既知の部分 $1/(x_{i+1}-x_i)$ の値（これをゲイン値と呼ぶ）をテーブルにしておき，インデックス値で参照すれば，除算が避けられ，演算は速くなる。

AD10のメモリ番地プロセサ（MAP）と論理判定プロセサ（DEP）は，このバイナリ・サーチを行うに適した構造をもっている。DEPのインストラクションの内，CMM（Compare and Modify）命令は2つのデータの大小比較を行い，その結果によってインデックス・レジスタの内容を指定したオフセットだけ増減させることができる。つまり，

```
IF UB≦U
    THEN(INDEX)←(INDEX)+OFFSET
    ELSE IF OFFSET≠0
        THEN(INDEX)←(INDEX)−OFFSET
        ELSE(INDEX)←(INDEX)−1
```

という操作を 100 nsec で行う。また，メモリ番地プロセサ（MAP）は，このインデックス・レジスタの指定するデータメモリ番地の内容（関数値データ）を読み出す。図8の例では，始めにインデックスは8，オフセットは4に指定されており，最初の比較操作でインデックスは4か12に，オフセットは2に変更される。次の比較操作でインデックスは2，6，10，14のいずれかに，オフセットは1に変更される。この比較操作を4回行うと線形内挿のインデックス値が求まる。一般に，ブレーク・ポイントの数が 2^L ならば，比較操作の回数はL回である。

次に，式(5)～(7)の線形内挿演算の基本式である±（A±B）×C±Dの形の演算について述べる。この演算は演算プロセサ（ARP）で行われる。図9に示す様に，まず，式(5)～(7)の関数値(f)，中間結果（g，h），あるいはデルタ値（X，Y，Z），等がレジスタA，B，C，Dにセットされると，次に（A＋B）の加算，（A＋B）×Cの乗算，（A＋B）×C＋Eの加算，と順に演算が進行する。ここで，パイプライン構造のため，レジスタA，B，C，Dにセットされたデータが次の加算のステップに移行すれば，直ちに別のデータをレジスタA，B，C，Dにセットしてかまわない。±（A±B）×C±Dの形の演算の所要時間は 175 nsec であるが，パイプライン構造により，この演算結果は 100 nsec 毎に次々得られることになる。

以上に述べた多変数関数発生法をAD10に組込んだ場合，演算時間は表3に示す様になる。ここで，1組あるいは2組の関数発生の演算時間は，3組の関数発生の演算時間と同じである。従って，3組の

表3　多変数関数発生の演算時間

	等間隔ブレーク・ポイント	不等間隔（32点）ブレーク・ポイント
3組の1変数関数	3．4 μsec	7．1 μsec
3組の2変数関数	5．5	9．2
3組の3変数関数	9．6	17．0
3組の4変数関数	12．6	20．0
3組の5変数関数	20．3	31．4

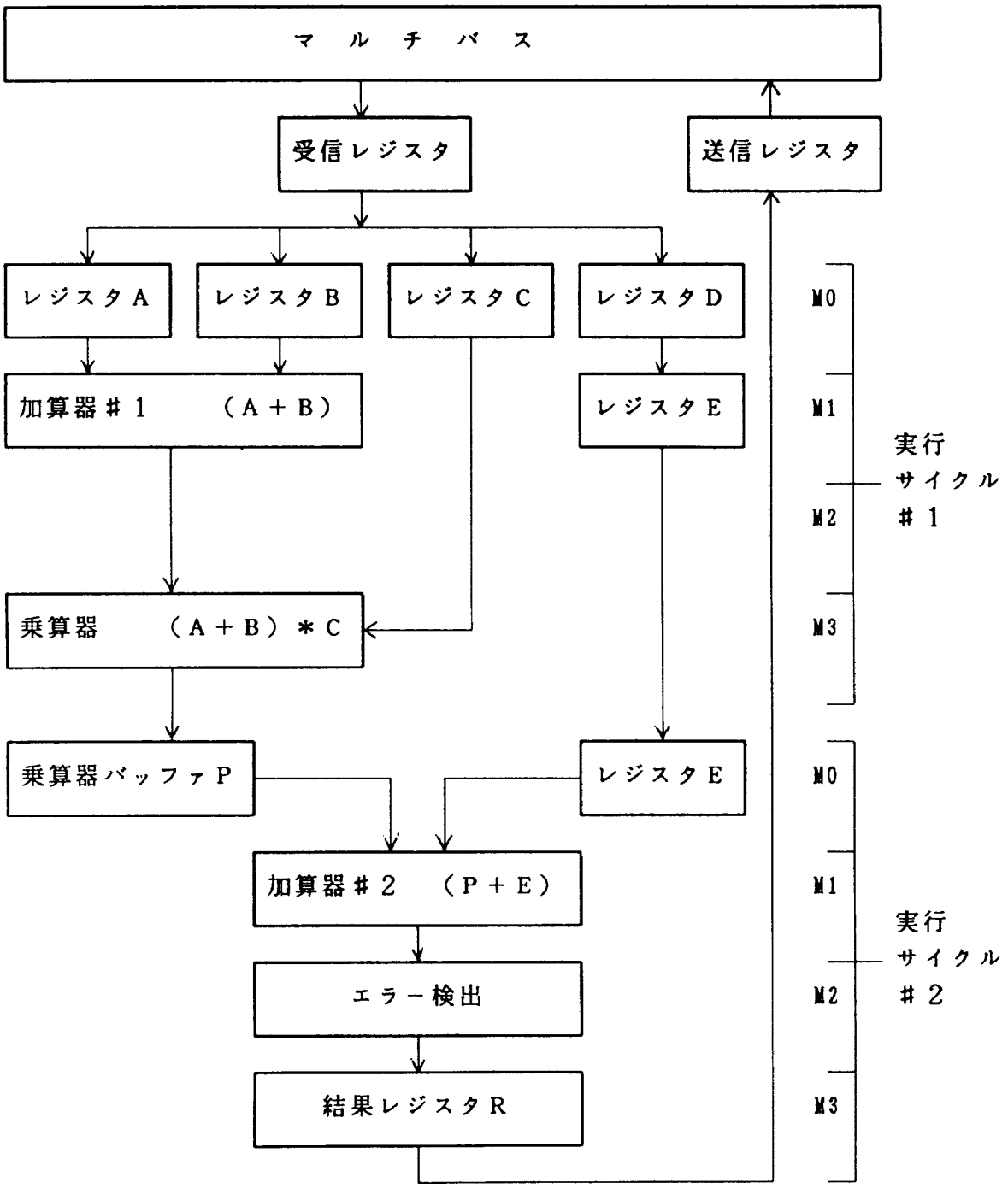


図9 演算プロセサ（ARP）のデータ・フロー[8]

関数発生を行わせたとき演算効率は最高になる。表3には5変数関数まで示したが，原理的にはより多くの入力変数をもつ関数発生も可能であるし，ソフトウェアの大きさも，演算時間も妥当な範囲にある。ただし，データメモリに準備すべき関数値の数は，入力変数の数の増加とともに急激に増大するため，5変数関数程度が実用上の限界であろう。

4.3 数値積分

表4はディジタル・シミュレーションで用いられる代表的な数値積分の演算公式を示している。演算公式の選択は，演算精度，安定性，メモリ容量，等を考慮してなされるが，2.3で述べた様にシミュレータと外部機器ハードウェアとを結合して実時間シミュレーションを行う場合，積分公式の実時間性をも考慮しなければならない。

例えば，微分方程式

表4　数値積分の演算公式

アダムス・バシュフォース法（予測子法）

1次　$y_{n+1}=y_n+hf_n$　（オイラー法）

2次　$y_{n+1}=y_n+(h/2)(3f_n-f_{n-1})$

3次　$y_{n+1}=y_n+(h/12)(23f_n-16f_{n-1}+5f_{n-2})$

4次　$y_{n+1}=y_n+(h/24)(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3})$

アダムス・モルトン法（予測子－修正子法）

1次　$y_{n1}=y_n+hf_n \rightarrow y_{n+1}=y_n+hf_{n1}$

2次　$y_{n1}=y_n+(h/2)(3f_n-f_{n-1}) \rightarrow y_{n+1}=y_n+(h/2)(f_{n1}+f_n)$

3次　$y_{n1}=y_n+(h/12)(23f_n-16f_{n-1}+5f_{n-2})$
$\rightarrow y_{n+1}=y_n+(h/12)(5f_{n1}+8f_n-f_{n-1})$

4次　$y_{n1}=y_n+(h/24)(55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3})$
$\rightarrow y_{n+1}=y_n+(h/24)(9f_{n1}+19f_n-5f_{n-1}+f_{n-2})$

ルンゲ・クッタ法

2次　$y_{n+1}=y_n+(h/2)(f_{n0}+f_{n1})$
$f_{n0}=f[t_n,y_n,r(t_n)]$
$f_{n1}=f[t_n+h,y_n+(h)f_{n0},r(t_n+h)]$

2次実時間　$y_{n+1}=y_n+(h)(f_{n1})$
$f_{n0}=f[t_n,y_n,r(t_n)]$
$f_{n1}=f[t_n+h/2,y_n+(h/2)f_{n0},r(t_n+h/2)]$

3次 (Nystrom Form)　$y_{n+1}=y_n+(h/8)(2f_{n0}+3f_{n1}+3f_{n2})$
$f_{n0}=f[t_n,y_n,r(t_n)]$
$f_{n1}=f[t_n+2h/3,y_n+(2h/3)f_{n0},r(t_n+2h/3)]$
$f_{n2}=f[t_n+2h/3,y_n+(2h/3)f_{n1},r(t_n+2h/3)]$

3次実時間 (Heun Form)　$y_{n+1}=y_n+(h/4)(f_{n0}+3f_{n2})$
$f_{n0}=f[t_n,y_n,r(t_n)]$
$f_{n1}=f[t_n+h/3,y_n+(h/3)f_{n0},r(t_n+h/3)]$
$f_{n2}=f[t_n+2h/3,y_n+(2h/3)f_{n1},r(t_n+2h/3)]$

4次 (Classical Form)　$y_{n+1}=y_n+(h/6)(f_{n0}+2f_{n1}+2f_{n2}+f_{n3})$
$f_{n0}=f[t_n,y_n,r(t_n)]$
$f_{n1}=f[t_n+h/2,y_n+(h/2)f_{n0},r(t_n+h/2)]$
$f_{n2}=f[t_n+h/2,y_n+(h/2)f_{n1},r(t_n+h/2)]$
$f_{n3}=f[t_n+h,y_n+(h)f_{n2},r(t_n+h)]$

4次実時間 (Bernstein Form)　$y_{n+1}=y_n+(h/24)(-f_{n0}+15f_{n1}-5f_{n2}+5f_{n3}+10f_{n4})$
$f_{n0}=f[t_n,y_n,r(t_n)]$
$f_{n1}=f[t_n+h/5,y_n+(h/5)f_{n0},r(t_n+h/5)]$
$f_{n2}=f[t_n+2h/5,y_n+(2h/5)f_{n0},r(t_n+2h/5)]$
$f_{n3}=f[t_n+3h/5,y_n-(2h/5)f_{n0}+(h)f_{n1},r(t_n+3h/5)]$
$f_{n4}=f[t_n+4h/5,y_n+(3h/10)f_{n0}+(h/2)f_{n3},r(t_n+4h/5)]$

$$\dot{y}(t)=f[t, y(t), r(t)] \quad (11)$$

（y：状態変数，r：入力変数，t：時間）

を表4のルンゲ・クッタ法4次（Classical Form）で，演算する場合，時刻 t_n から時刻 t_{n+1}（$=t_n+h$）までの1ステップの積分を行うために，異なったt，y，r について式(11)の右辺の計算を4回行い，f_{n0}，f_{n1}，f_{n2}，f_{n3} を求める必要がある。この過程をサブ・ステップと呼ぶことにする。サブ・ステップの計算は時刻 t_n，$t_n+h/4$，$t_n+h/2$，$t_n+3h/4$ の様に（$h/4$）刻みで進行する。表4の公式を吟味すると，時刻 $t_n+h/4$，$t_n+3h/4$ に開始される f_{n1}・f_{n3} の計算は，時刻 $t_n+h/2$，t_n+hにおける入力変数r(t)の値（未来値）を必要としている。入力変数の時間変化が前もって指定されている場合は別として，一般に，入力変数 r(t) の未来値は外挿計算で推定することになる。外挿は計算時間を増加させるだけでなく，演算精度の低下をもたらし好ましくない。この様なケースは表4のアダムス・モルトン法でも生ずる。

入力変数の未来値を用いない積分公式，つまり，表4のアダムス・バシュフォース法と実時間型のルンゲ・クッタ法が実時間シミュレーションに適しているといえる。ここで，実時間型のルンゲ・クッタ法とは，ルンゲ・クッタ法の導出過程におけるパラメータの任意性に着目して，入力変数の未来値を使用しない様に導出したものである[7]。この実時間型のルンゲ・クッタ法は，入力変数の未来値を用いないという利点の他に，サブ・ステップで計算される状態変数 y の値はその時刻におけるよい近似値になっているという利点も持っている。つまり，時刻 t_n から時刻 t_{n+1}（$=t_n+h$）までの1ステップの積分を行う時，全サブ・ステップの完了後に状態変数 $y(t_n+h)$ が得られるばかりでなく，サブ・ステップの途中でも状態変数 $y(t_n+h/5)$，$y(t_n+2h/5)$……がよい精度で得られ，シミュレーション出力として

表5　数値積分の演算時間およびタイプ

積　分　法	記　号	タイプ	ステップ数	演算時間
アダムス・バシュフォース法				μsec
1次	AB1	0	1	0．3
2次	AB2	0	1	0．7
3次	AB3	0	1	1．6
4次	AB4	0	1	2．6
アダムス・モルトン法				μsec
1次	AM1	1	2	0．6
2次	AM2	1	2	0．8
3次	AM3	1	2	1．6
4次	AM4	1	2	2．6
ルンゲ・クッタ法				μsec
2次 (Improved Euler)	RK2	1	2	0．8
2次 (Improved Polygon)	RKRT2	0	2	0．6
3次 (Nystrom)	RK3	1	3	1．1
3次 (Heun)	RKRT3	0	3	0．8
4次 (Classical)	RK4	1	4	0．9
4次 (Bernstein)	RKRT4	0	5	2．1

使用することができる。

表5は表4の積分公式をAD10にプログラミングした時の積分演算時間を示している。タイプ0は実時間型の積分，タイプ1は入力変数の未来値を必要とする積分である。また，ステップ数は微分方程式の右辺（式(11)のf［t，y，r］）の計算回数，演算時間はサブ・ステップ当たりの演算時間である。1ステップの積分演算時間は，（サブ・ステップ数）×（サブ・ステップ演算時間）となるが，上述の様に実時間型（タイプ0）のものはサブ・ステップ毎に状態変数を出力することができるため，演数時間は表5に示したものとなる。

数値積分においては，加算項の桁落ち誤差に注意しなければならない[4]。特に2.3で述べた広い周波数分布を持つスティフなシステムでは，この誤差が顕著になる。式(11)をオイラー法で解く場合を考えると，公式により，

$$y_{n+1} = y_n + h \cdot f$$
$$= y_n + h \cdot K \cdot f' \qquad (12)$$

ここで，K；スケーリング・ファクタ

f'；1に規準化されたf

となる。このKの逆数はモード時定数と等価である。多変数システムでは状態変数の数だけ式(12)のKの値が定まるが，その最大値，最小値をK_{max}，K_{min}とすると，ガスタービン・システムでは，（K_{max}／K_{min}）の値は1000に達することも考えられる。また，過渡時の演算精度を維持するために，積分時間間隔（h）を最小時定数（$1/K_{max}$）の1/1000程度とすると，最大時定数を有する式は，

$$y_{n+1} = y_n + h \cdot K_{min} \cdot f'$$
$$= y_n + (h \cdot K_{max})(K_{min} / K_{max}) \cdot f'$$
$$= y_n + 10^{-6} f'$$
$$= y_n + 2^{-20} f' \qquad (13)$$

となる。つまり，f'が16ビットの精度で計算されるならば，yには36（＝16＋20）ビット以上の精度が要求される。一般技術計算では積分過程のみを倍精度指定して計算するのが普通であるが，上述のことがその理由である。AD10では，この加算項の演算を48ビットで行なっており，桁落ち誤差は極めて小さいと言える。

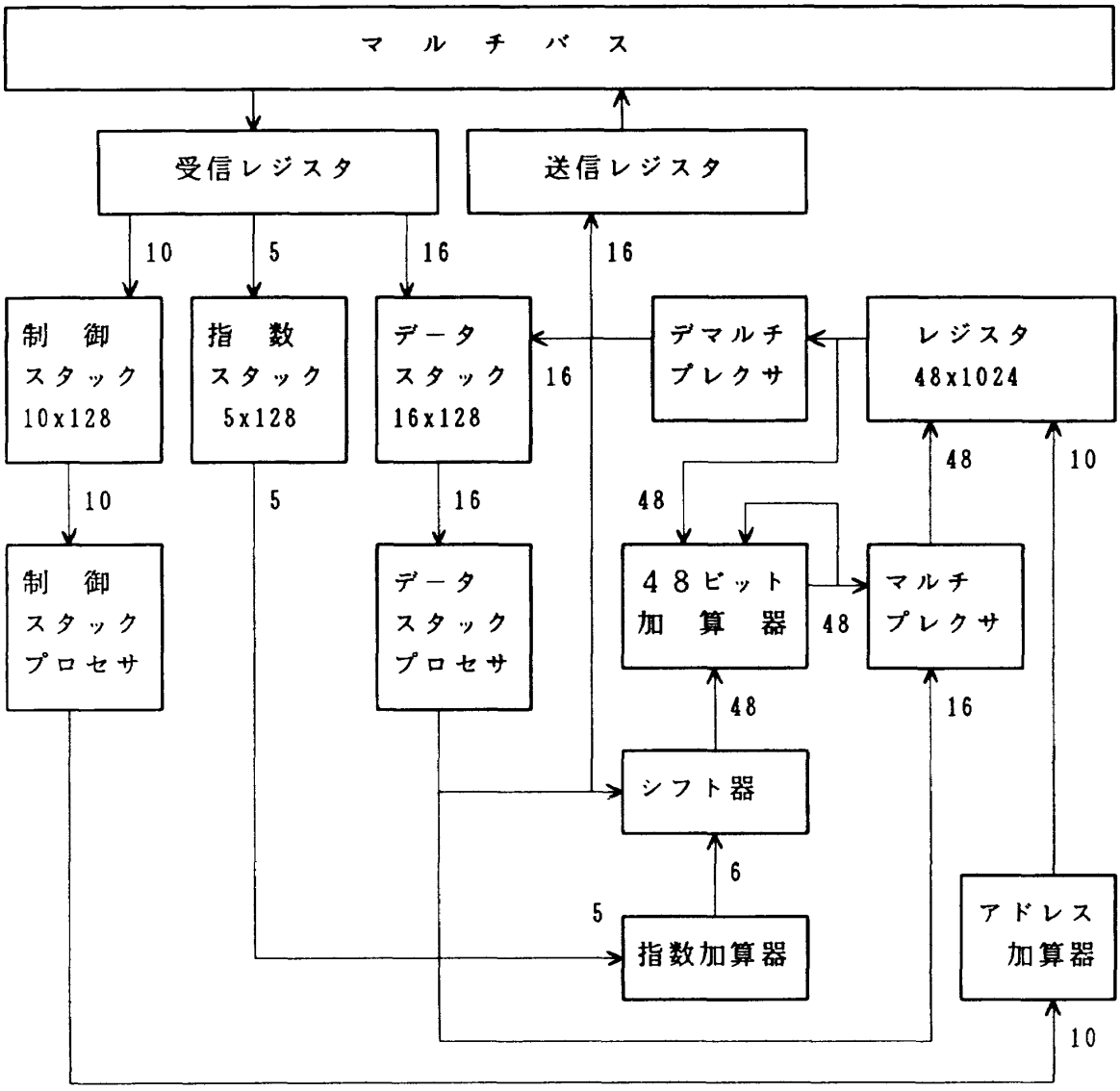
以上述べた数値積分演算は，AD10内の数値積分プロセサ（NIP）で行なわれる。図10はNIPにおけるデータ・フローを示している。演算プロセサ（ARP）で計算された微分方程式の右辺は，16ビットのデータと5ビットの指数データとして，マルチバスを経由してNIPに取り込まれる。NIPの中心部は48ビットのシフト器と加算器で，積分公式に従って加算項の蓄積を行ない，積分結果をレジスタに格納したり，マルチバスに送信したりする。NIPの命令語長（プログラム・メモリのビット長）は80ビットあり，図10のデータ・フローの制御や演算制御は，かなりの部分で同時進行（マイクロ・プログラミング）が可能である。

4.4　入出力装置

シミュレータは2.1，2.3で説明した様に，各種の外部機器との接続が可能でなければならない。この接続は多くの場合，アナグロ信号でなされ，ハードウェアとしてはAD変換器，DA変換器が用いられる。高速ディジタル・シミュレーションにおいては，入出力装置もそれに見合った速度と精度をもつ必要があるが，これはAD／DA変換器の変換速度とビット長の選択で決定される。また，アナログ信号の多チャンネル同時入力，演算結果の多チャンネル同時出力も重要で，入出力信号がチャンネル毎に時間的ずれ（スキューイング）をもたない様にする必要もある。

アナログ信号による外部機器との接続のほかに，ディジタル信号による接続が必要となる場合もある。オンオフ的な信号の入出力に用いる接点入出力装置はよく用いられるが，これ以外のディジタル接続はインターフェイスの複雑さから，あまり採用されない。

AD10のAD／DA変換器の仕様を表6に示すが，これは本『高効率ガスタービン制御システム解析装置』に採用したAD／DA変換器の仕様であって，別の変換速度とビット長をもつ変換器も用意されている。ディジタル（接点）入出力装置の仕様もあわせて表6に示す。AD10のAD／DA変換器はチャンネル毎に完全に独立（マルチプレクサ方式でない）



註 矢印線わきの数値はそのラインのビット数を示している

図10 数値積分プロセサ（ＮＩＰ）のデータ・フロー[8]

であり，各チャンネル毎に中間データ・バッファを持っている。このため，1命令で多チャンネル同時入力，同時出力が可能であり，スキューイングは全く考慮しなくてよい。

4.5 ソフトウェア

シミュレータの総合性能は，単にハードウェアの性能だけでなく，ソフトウェアの能力によっても左右される。いかに高性能なハードウェアを装備していても，使いにくいシステムの価値は低いと言うべきである。ＡＤ10のソフトウェアとして表7に示すものが準備されており，プログラムの開発，シミュレーションの実行，シミュレータの維持管理等，全てのフェーズにおいて必要なソフトウェアをカバーしている。これ等のソフトウェアの全ては，ホスト計算機のオペレーティング・システム（ＶＡＸ／ＶＭＳ）のもとに運用される。表7の内，一般のユーザに最も利用されるのは，高級シミュレーション言語（ＭＰＳ10……Modular Programming System ）である。これは，汎用シミュレーション言語（ＣＳＳＬ，ＣＳＰＬ等）に類似したものであるが，特殊なハードウェア構成を有するＡＤ10の機能が最大限に発揮され得るようにコーディングされたプログラム・モジュールで構成されている。

表6　入出力機器の仕様

ＡＤ変換器	分　解　能 レ　ン　ジ 直　線　性 変換時間 素　　子 チャンネル数	１１ビット＋符号ビット ±１０Ｖ ±1/2 ＬＳＢ １０μsec ＢＢ－ＡＤＣ８４ＫＧ－１２ １４
ＤＡ変換器	分　解　能 レ　ン　ジ 直　線　性 変換時間 素　　子 チャンネル数	１１ビット＋符号ビット ±１０Ｖ ±1/2 ＬＳＢ ３μsec ＢＢ－ＤＡＣ８０ＣＢＩ－Ｖ ３０
ディジタル入力	変換時間 接　　続 チャンネル数	１００ nsec ＴＴＬレベル ２（３２接点）
ディジタル出力	変換時間 接　　続 チャンネル数	１００ nsec ＴＴＬレベル ２（３２接点）

表7　AD10ソフトウェア・システム

基本ソフトウェア	ＨＬＩＢ……ホスト・インターフェイス・ライブラリ ＡＤＸ……対話型エクゼクティブ ＡＳＭ……クロス・アセンブラ ＭＦＬＩＢ…マクロ・ライブラリ ＡＤＳＭ……ＡＤ１０シミュレータ ＤＩＡＧ……診断プログラム
高級言語ソフトウェア	ＭＰＳ１０…シミュレーション言語

ＭＰＳ10[10]のユーザ・プログラムはシミュレーション・モデルの数学的表現に極めて近く，例えば，2階微分方程式

$$a\ddot{x}+b\dot{x}+cx=u \tag{14}$$

をＭＰＳ10でコーディングすることを考える。まず，式(14)を1階連立微分方程式に書き換えて，

$$\dot{x}=y$$
$$\dot{y}=-(b/a)y-(c/a)x+u \tag{15}$$

とし，これから直接，ＭＰＳ10言語では，

```
INT  M1, X=XI  $ Y
INT  M1, Y=YI  $ [-B/A] *Y+[-C/A] *X+U
```
(16)

表8 ＭＰＳ10の主な演算モジュール

演算モジュール	モジュール名	機　　能
積分	ＩＮＴＬ	加算積分
	ＩＮＴＱ	乗算を含む加算積分
加算	ＳＵＭＬ	加算
	ＳＵＭＱ	乗算を含む加算
乗算	ＰＲＯＤ	乗算
除算	ＤＩＶ	除算
一般演算	ＡＲＰＡＬＧ	演算プロセサで可能な任意の計算
	ＦＡＳＴＡＬＧ	ＡＲＰＡＬＧの高速版
差分	ＤＩＦＦＬ	加算差分
	ＤＩＦＦＱ	乗算を含む加算差分
ベクトル積	ＣＲＯＳＳＡＤＤ	ベクトル積およびベクトル和
関数発生	ＦＣＮＧ	１～２変数関数発生
関数内挿	ＦＣＮＩ	１～４変数不等間隔ブレーク・ポイント関数の内挿計算
関数ルックアップ	ＦＣＮＬ	１～４変数不等間隔ブレーク・ポイント関数のテーブル・ルックアップ
インデックス・デルタ	ＩＸＤＥＬ	インデックス・デルタ値の計算
座標変換	ＣＯＯＲＤ１	１軸の回転
	ＣＯＯＲＤ２	２軸の回転
	ＣＯＯＲＤ３	３軸の回転
	ＩＮＶＲＥＳ	直角座標－極座標変換
入出力	ＣＩＯＣＣ	入出力コマンド
	ＲＩＯＣＣ	入力
	ＷＩＯＣＣ	出力
定数	ＣＯＮＳＴＡＮＴ	定数の設定
サーキュラ・バッファ	ＣＩＲＢＵＦ	シミュレーション結果の格納
比較スイッチ	ＣＭＰＲＳＷ	データの比較、論理設定
論理	ＬＣＯＭＰＡＣＴ	16の論理値を１ワードへ圧縮
	ＬＥＸＰＡＮＤ	16の論理値に復元
最大値検出	ＭＡＸＶＡＬ	指定した変数の最大値を保持
データ転送	ＴＲＡＮＳ	モジュール間のデータ授受
ループ・カウンタ	ＬＣＯＵＮＴ	プログラム・ループのカウント
ポウズ	ＰＡＵＳＥ	演算の一時停止
無条件ホルト	ＨＡＬＴ	演算の停止
条件ホルト	ＨＡＬＴＦＣ	Ｆ論理で演算停止
	ＨＡＬＴＴＣ	Ｔ論理で演算停止

表9　ＭＰＳ10の主な対話機能

コ　マ　ン　ド	機　　能
ＡＴＴＡＣＨ	ＡＤ１０とユーザ・プログラムの接続
ＤＥＴＡＣＨ	ＡＤ１０とユーザ・プログラムの接続解除
ＬＯＡＤ、ＬＯＡＤＤＡ	データ・エリア・ファイルのＡＤ１０へのロード
ＬＯＡＤＳＹＳ	演算モジュールのＡＤ１０へのロード
ＩＮＩＴ	初期化
ＧＯ	初期化の後、演算の実行
ＨＡＬＴ	演算の一時停止
ＣＯＮＴＩＮＵＥ	演算の継続
ＢＲＥＡＫ	任意のモジュールにブレーク・ポイントを設定
ＡＥＲＲＯＲ	エラー検出のオン・オフ
ＨＥＲＲＯＲ	エラー検出がオンならば演算の一時停止
ＳＴＯＰ	演算の停止
ＳＴＡＴＵＳ	ＡＤ１０のステータスの表示
ＷＡＩＴ	ＡＤ１０がホルトするまで待機
ＲＵＮＳＰＥＣＳ	実行形式の表示あるいは変更
ＦＲＡＭＥＴＩＭＥ	フレームタイムの計測あるいは設定
ＯＵＴＰＵＴＳ	出力変数の数値表示あるいは仕様変更
ＰＡＲＡＭＥＴＥＲＳ	パラメータの数値表示あるいは変更
ＭＥＴＨＯＤＳ	積分方式の表示あるいは変更
ＳＣＡＬＥ	スケーリング値の表示あるいは変更
ＷＩＮＤＯＷ	ディスプレイ表示範囲の表示あるいは変更
ＩＯＣＣ、ＩＯＣＳ	入出力指定の表示あるいは変更
ＡＤＩＳＰＬＡＹ	演算プロセサ（ＡＲＰ）のレジスタ内容の表示
ＡＭＯＤＩＦＹ	演算プロセサ（ＡＲＰ）のレジスタ内容の変更
ＦＩＬＥＳ	関数データ・ファイルのファイル名表示あるいはＡＤ１０へのロード
ＩＣＤＥＲ	状態変数の初期２ステップにおける変化率の表示
ＬＩＮＫ	ユーザ指定のリンク
ＬＯＣＫ	任意のコマンドにおける表示あるいは変更の禁止
ＵＮＬＯＣＫ	ＬＯＣＫの解除
ＬＯＧ	コマンド・ロギングのオン・オフ
ＭＡＸＶ	ＭＡＸＶＡＬモジュールで指定した変数の最大値およびオーバフローの表示
ＲＥＱＵＥＳＴ	ユーザのＦＯＲＴＲＡＮプログラムの結合
ＤＡ	データエリア、セグメントの表示

と書く。ここで，ＩＮＴは積分，Ｍ１はデータエリア名，ＸＩ，ＹＩは初期値を表わしている。その他の部分の表記は式(15)との対比から明らかであろう。

式(16)におけるＩＮＴ（積分）をＭＰＳ10では演算モジュールと呼んでいる。積分の他に，加算，乗算，関数発生等のシミュレーションにおいて必要となる種々の演算モジュールが用意されている（表8参照）。これ等の演算モジュールはあたかもアナログ計算機の演算要素の様であり，MPS10によるプログラミングは，演算要素をパッチコードで次々と接続していくアナログ計算機の方法と類似している。

ユーザとシミュレータとの対話は，プログラムの開発時，シミュレーションの実行時に必要となるが，その対話機能の良否はプログラム開発期間やマンパワー，シミュレーションの能率に大きな影響を与える。MPS10においては，対話は表9に示すコマンドによってなされるが，ｉ）プログラムの実行制御，ii）値や仕様の表示，変更，iii）入出力の制御，iv）ファイルのロード，セーブ，ｖ）デッバギング・トゥール，等を含み，使い易いコマンド・システムとなっている。

5. 高速演算装置(MAP 300)

ＭＡＰ 300は信号処理，データ解析，科学技術計算において生ずるＦＦＴ，逆行列，たたみ込み，相関，ベクトル操作，等の演算を高速に処理する様に設計されたペリフェラル・プロセサである。これらの演算は高度に構造化されうるもので，大量のデータに比較的単純な演算操作（乗算，加算）をくり返し行うことにより達成される。ＭＡＰ 300において，演算は32ビット浮動小数点（有効数字６桁，ダイナミックス・レンジ$10^{\pm 76}$ で行われ，通常のミニ・コンピュータの10～1000倍（12ＭＦＬＯＰＳ）の処理速度をもっている。

高速処理を可能にしている因子は後述する様に，ｉ）複数プロセサの並列処理，ii）複数データバスによる並列データ転送，iii）高速ディジタル回路の採用，等である。通常の計算機では，プログラムの処理はシリアルに進行し，１時点では１つの命令を実行しているにすぎない。これに対し，並列処理型の計算機では，複数プロセサ上で命令が同時に処理される。例えば，データの入出力処理と算術演算処理が，別々のプロセサで進行する様に構成すると，データの入出力時に算術演算を中止する必要がなく，処理速度の向上をもたらすことになる。

ＭＡＰ 300のユーザ・ソフトウェアは，効率的にコーディングされたＦＯＲＴＲＡＮ言語でＣＡＬＬできるライブラリ（ＳＮＡＰ）を利用して作成され，ハードウェアの知識がなくても，目的とする計算を効率的に行うことができる。

5.1 システム構成

図11はＭＡＰ 300のシステム構成図である。３本のバスを中心に，次の５種の独立した専用プロセサと３つの独立したデータ・メモリが接続されている（ただし，現時点では下記(4)(5)は設置してない）。

(1) ホスト・インターフェイス・モジュール（ＨＩＭ）……ホスト計算機ＶＡＸ 11/750とMAP300との間の命令の授受，データの転送，および転送中のデータ・フォーマット変換を行う。

(2) 制御管理プロセサ（ＣＳＰＵ）……ＭＡＰ 300内の全てのプロセサの制御を行う。これ自体が１つのミニ・コンピュータで，常時オペレーティング・システム（ＳＮＡＰ）が動いており，演算や入出力の待ち行列管理，プロセサの制御等を行っている。

(3) 演算プロセサ（ＡＰ）……乗算，加算，論理演算，等の数学的演算を行う。

(4) アナログ入出力（ＡＤＡＭ，ＡＯＭ）……アナログ信号の入出力を行う。

(5) ディジタル入出力（ＩＯＳ）……ディジタル信号の入出力を行う。

これらのプロセサは，それぞれのプログラムを格納するプログラム・メモリとプログラム・カウンタおよびクロックを別個にもっている。従って，プロセサ間の結合は，ＡＤ10と異なり非同期方式のゆるい結合（4.1参照）となっている。この場合，プロセサ間のデータ授受，通信にはシェークハンド手続きが必要となるが，後述する様にデータ駆動型(Data Driven)のハードウェア構造を持つため，オーバーヘッドは非常に小さくなっている。

一方，データ・メモリとして，入出力データの格

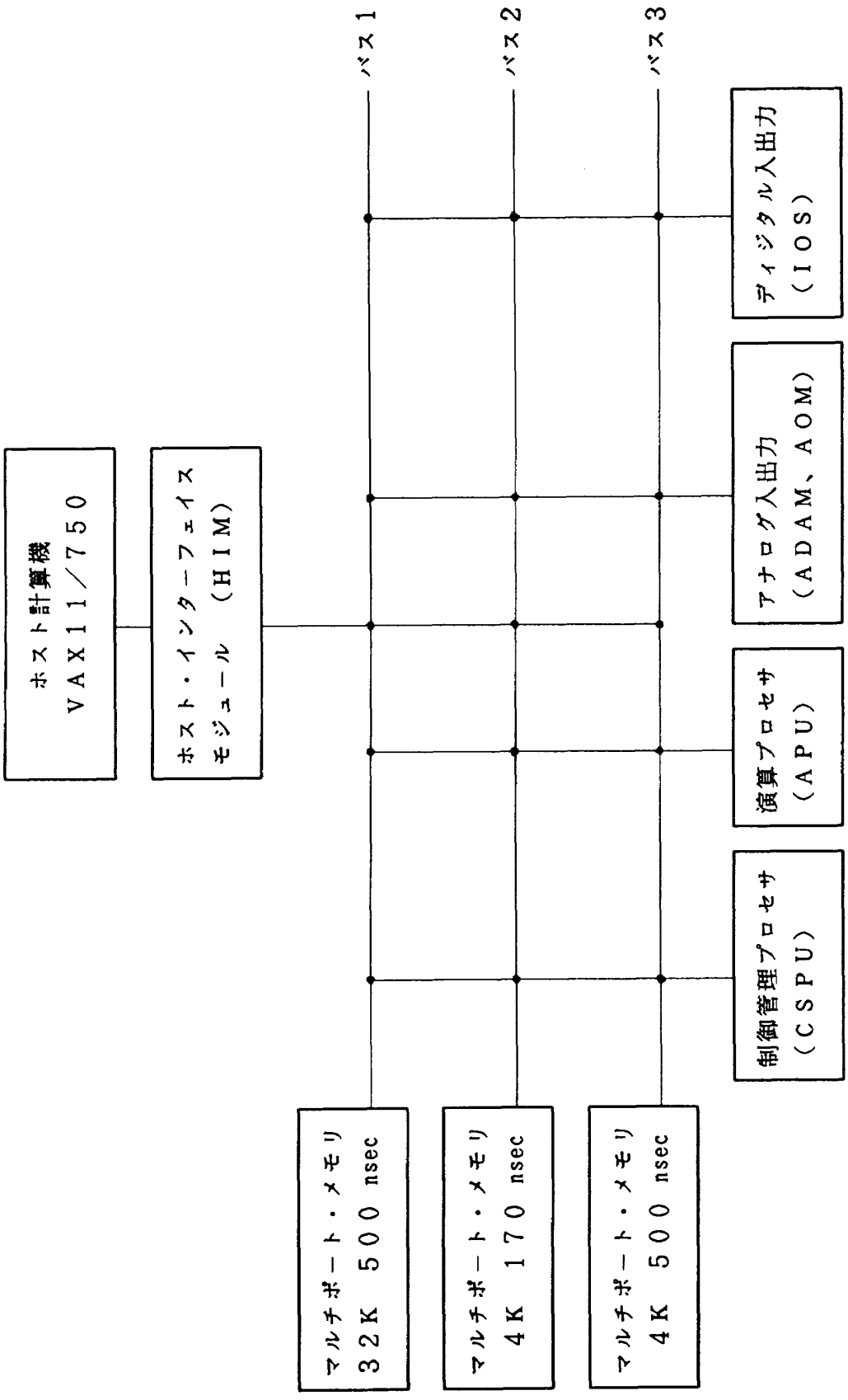


図11　ＭＡＰ３００システム構成[13]

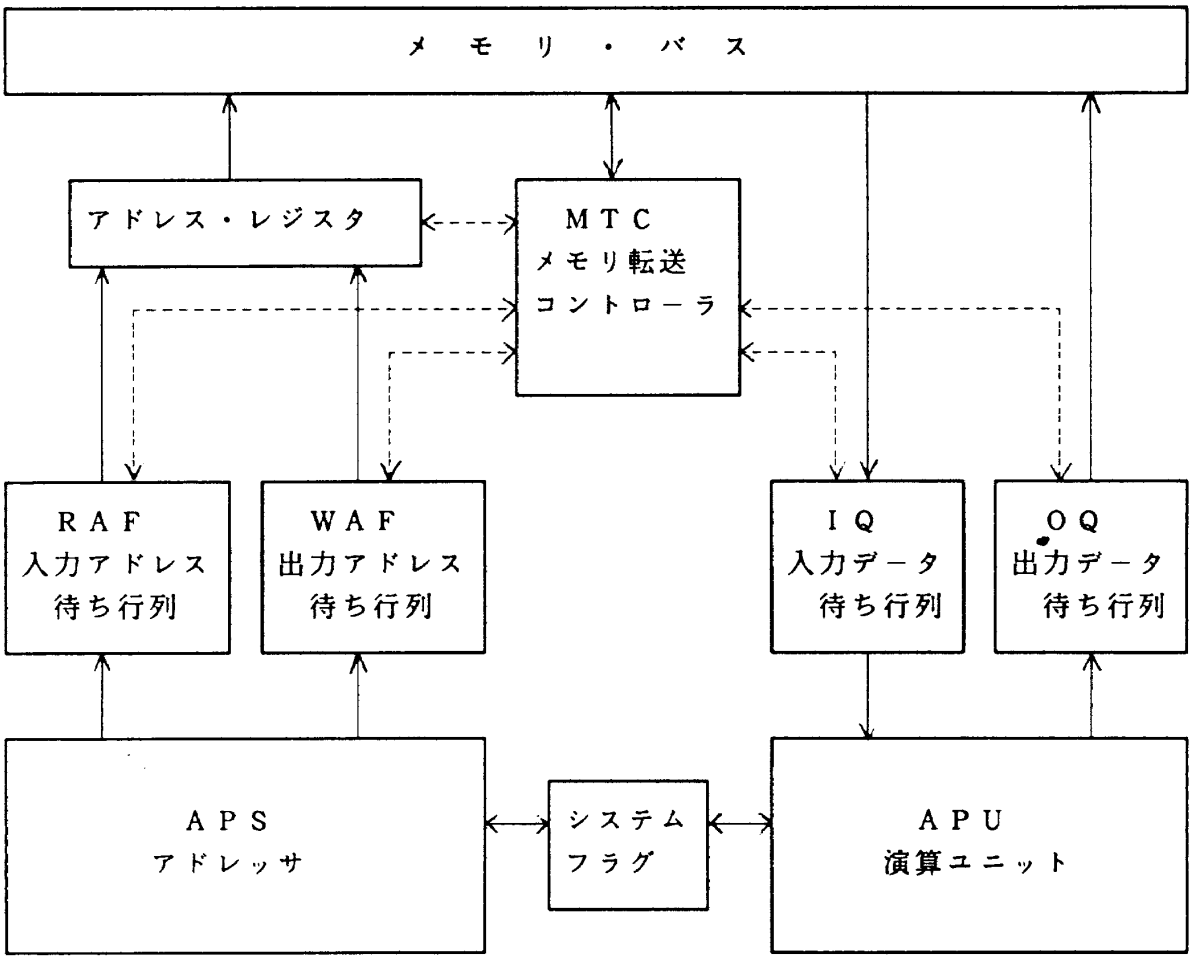
納のための大容量，低速メモリ（ＭＯＳメモリ，アクセス・タイム 500 nsec ），中間処理結果の一時的格納のための高速メモリ（バイポーラＩＣメモリ，アクセス・タイム 170 nsec ）が必要となる。これらのデータ・メモリと各プロセサは独立した３本のバスで結合されており，任意のプロセサあるいはメモリ間で並列高速データ転送が行われる。複数本のバスはプロセサの待ち時間を減らし処理速度を向上させる（ＡＤ10ではバスは１本）。メモリ容量は処理するデータ点数を規定することになるが，図11に示した４Ｋワードは，時系列演算やベクトル演算ならば 4096 点迄，行列演算ならば64×64迄の処理が可能なメモリ容量である。

5.2 高速演算の方法

高速演算はバランスのとれた機能分割により並列ハードウェアを構成することで実現される[11,12]。通常の計算機ではシリアルに進行する機能，例えば，メモリからのデータの読み出し，加算演算，乗算演算，レジスタ間のデータ転送，メモリへのデータの書き込み，等の機能の分割並列化が考えられる。ＭＡＰ 300ではネックになる所がない様にバランスを考慮して，次の様に機能分割を行っている。

【浮動小数点演算】
- 乗算演算
- 加算演算およびその他の若干の演算
- レジスタ間のデータ移動
- 入力データ待ち行列
- 出力データ待ち行列

【アドレス計算およびループ・カウント】
- 計算および計数
- メモリ転送



註　実線矢印はデータ・フロー、破線矢印はコントロール・フローを示している

図12　演算プロセサ（ＡＰ）ブロック図[13]

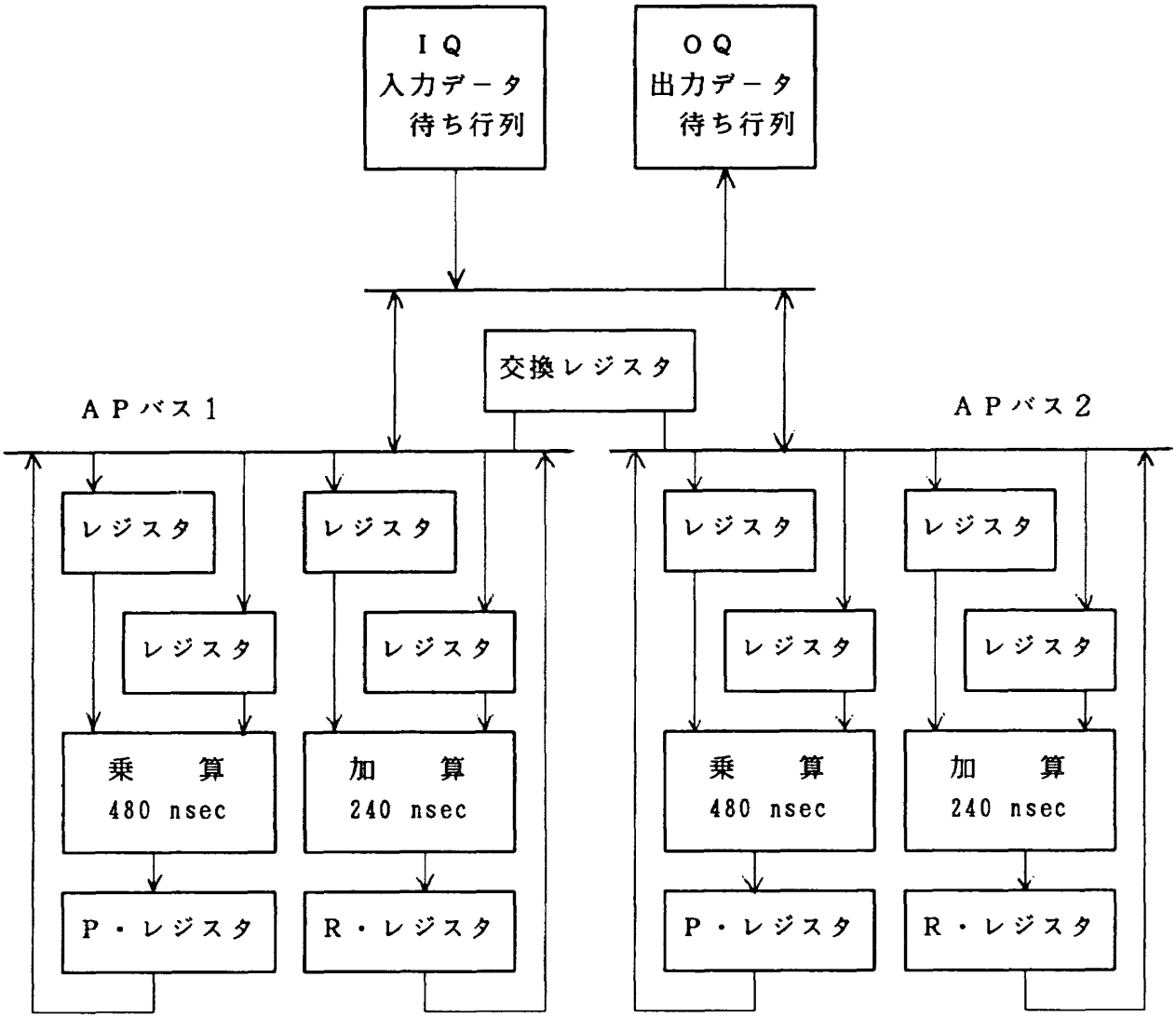


図13　演算ユニット（ＡＰＵ）ブロック図[13]

・入力アドレス待ち行列
・出力アドレス待ち行列

図12は演算プロセサ（ＡＰ）のブロック図，さらに，図13はＡＰ内の演算ユニット（ＡＰＵ）のブロック図であるが，この機能分割並列化がみられる。ここで，待ち行列はＦＩＦＯ（First-In-First-Out，古いデータから順に出力される）方式で駆動される。

図12に示す様に，演算プロセサは，アドレッサ（ＡＰＳ），メモリ転送コントローラ（ＭＴＣ），演算ユニット（ＡＰＵ）の３つのユニットで構成されており，これ等は４つの待ち行列，即ち，入力アドレス待ち行列（ＲＡＦ），出力アドレス待ち行列（ＷＡＦ），入力データ待ち行列（ＩＱ），出力データ待ち行列（ＯＱ）で接続されている。まず，アドレッサ（ＡＰＳ）はデータ・アドレスを計算してＲＡＦとＷＡＦに送り込むことのみを行う。次に，メモリ転送コントローラ（ＭＴＣ）は，ＲＡＦにあるアドレスに従ってメモリからデータを読み出し，それをＩＱに送り込む，あるいは，ＯＱにあるデータをＷＡＦにあるアドレスに従ってメモリに書き込むことのみを行う。また，演算ユニット（ＡＰＵ）は，ＩＱにあるデータに演算をほどこして結果をＯＱに送り込むことのみを行う。ＡＰＳ，ＭＴＣ，ＡＰＵ，はそれぞれ固有の速度で独立に動作しているが，４つの待ち行列がそれらを互いに結びつけている。つまり，入力待ち行列にデータが来ていれば処理を行うが，来てなければ来るまで待つ，また，出力待ち行列が空いていれば処理データを送り出して処理を進めるが，空いてなければ空くまで待つ，という論理で作動している（これをデータ駆動方式と呼ぶ）。

上記のことを具体的に説明するために，次の様なスカラーとベクトルの積の計算を考えることにする。

Y(I)=A*X(I), I=1, 2, 3, …… 1000

この計算のユーザ・プログラムは後述する様に，FORTRANで書かれ，

CALL VSML(Y, A)

とすればよい。ここで，サブルーチンVSML はMAP 300の用意されたライブラリ・ルーチンの1つで，ベクトルとスカラーの積を計算する。この時，MAP 300の内部では，およそ次の様な2つのプログラムが実行されている（実際には，より効率的なプログラミングが可能であるが，簡単のため下記のプログラムで説明する）。

【演算ユニット（APU）】

A+0：MOV（IQA, M0） →IQの先頭からAを入力。IQを1つ前進させる。

A+1：MOV（IQA, M4） →IQの先頭からX(I)を入力。IQを1つ前進させる。

A+2：MUL（M0, M4） →乗算A*X(I)を実行。

A+3：MOV（P, OQ） →結果をOQへ出力。

A+4：JUMPC(A+1, FI) →IQにデータがあるならA+1：へ（A+1～A+4 を1000回くり返す）。

A+5：CLEAR（RA） →APUホルト。

【アドレッサ（APS）】

B+0：LOAD（BR0, AB, TF） →Aの番地をRAFへ。

B+1：LOAD（BR0, XB） →Xのベース番地をレジスタBR0へ。

B+2：LOAD（BW4, YB） →Yのベース番地をレジスタBW4へ。

B+3：LOAD（BR1, 999） →データ数をレジスタBR1へ。

B+4：SET（RA） →計算開始。

B+5：ADD（BR0, XS, TF）→次のXの番地をRAFへ。

B+6：ADD（BW0, YS, TF）→次のYの番地をWAFへ。

B+7：SUBL（BR1, 1），JUMPP（B+5） → B+5，B+6 の操作回数がデータ数未満ならB+5：へ（B+5～B+6を 1000 回くり返す）

B+8：CLEAR（RI） →APSホルト。

上の説明文でプログラムの意味は明らかであろう。演算ユニット（APU）におけるMOV（IQA, M0）命令は，入力データ待ち行列（IQ）にあるデータの内，1番古いデータを読み出して乗算レジスタM0におく（図13参照）。もしIQが空ならば，データがセットされる迄，命令の実行を待つ。また，MOV（P, OQ）命令は，乗算の結果がセットされているPレジスタの内容を，出力データ待ち行列（OQ）に書き込む。もしOQが一杯ならば，空きがでる迄，命令の実行を待つ。一方，アドレッサ（APS）において，引数にTF（転送指定フラグ）を伴う命令（B+0, B+5, B+6）はRAF, WAFにメモリ・アドレスを書き込む。もしRAF, WAFが一杯ならば，空きがでる迄，命令の実行を待つ。上の例ではRAFにはA, X(1), X(2), X(3), ……, X(1000) のアドレスが，またWAFにはY(1), Y(2), Y(3), ……, Y(1000) のアドレスが順にセットされることになる。そしてメモリ転送コントローラ（MTC）はRAF, WAFにセットされたアドレスに従って，IQを満たし，OQを空にすることのみを行い，プログラミングは必要がない。

図12, 図13, および上例から明らかな様に，MAP 300では，2つの加算，2つの乗算，レジスタ間のデータ移動，アドレス計算，メモリ・データの読み出し書き込み，等の機能が同時進行する。演算効率を向上させるため，これらの並列演算ユニット間のバランスは重要で，例えば，加算と乗算の計算速度比，待ち行列の深さ（FIFO depth，蓄積するデータの個数）等は適切なものでなければならない。ネックになる所がなく，バランスのとれた分割により並列ハードウェア化することで，FFT，行列演算，ベクトル操作等の演算は飛躍的に高速化される。代表的なアルゴリズムの計算時間を表10に示すが，これは通常のミニ・コンピュータの10～1000 倍の処理速度となっている。

5.3 ソフトウェア

表11はMAP 300のソフトウェア・システムを

表10　ＭＡＰ300の代表的アルゴリズムの計算時間
（ 170 nsec 高速メモリを使用した時 ）

演　　算	演　算　時　間
ベクトル加算	０．６μsec/point
ベクトル乗算	０．６μsec/point
ベクトル除算	１．６μsec/point
ベクトル内積	０．６μsec/point
ベクトル逆正接	７．５μsec/point
ベクトル・クリップ	０．６μsec/point
１０２４点実ＦＦＴ	３．２ msec
１０２４点複素ＦＦＴ	５．３ msec
４０×４０行列積	３８．４ msec
４０×４０逆行列	６５．６ msec

表11　ＭＡＰ300のソフトウェア・システム

システム・ソフトウェア	ＳＮＡＰ３……ＭＡＰ常駐エクゼクティブ ＭＰＤＲＩＶＥＲ…ＭＡＰドライバー ＨＬＤＳ……診断プログラム
開発ソフトウェア	ＭＡＰＡＳＭ……クロス・アセンブラ ＭＡＰＳＩＭ……ＭＡＰ３００シミュレータ ＬＯＯＫ……デバッガ
応用ソフトウェア	ＳＮＡＰ……アレイプロセサ・ライブラリ

示している。AD10と同様に，これ等のソフトウェアの全ては，ホスト計算機のオペレーティング・システム（ VAX／VMS)の基に運用される。表11の内，一般のユーザに最も利用されるのは，応用ソフトウェア（ ＳＮＡＰ，Symbolic Notation for Array Processing）である。これは，ＭＡＰ300の機能が最大限に発揮され得るようにコーディングされたサブルーチンで構成されている。サブルーチンの数は約200にものぼり，多くの数学演算ルーチン，入出力ルーチン，コントロール・ルーチンを含んでいる。数学演算ルーチンの代表的なものは，その演算時間とあわせて，表12に示す。

応用ソフトウェア（ ＳＮＡＰ）内のサブルーチンは全てＦＯＲＴＲＡＮでＣＡＬＬすることができる。従って，ユーザの任意の処理プログラムは，ＳＮＡＰのサブルーチンを次々とＣＡＬＬしていくことで完成する。例えば，40×40次元の行列の逆行列を求めるユーザ・プログラムは次の様になる。

```
1  DIMENSION X(40, 40)
2  INTEGER BUS2, REAL4, CNV
3  DATA MAPM, N, BASE, CNV/1, 40,
   0.0, 1/
4  DATA BUS2, REAL4, INC, INCH/
   2, 4, 2, 4/
```

演　　算	計　算　式	コ　マ　ン　ド	演　算　時　間	
			170 nsec 高速メモリ	500 nsec 低速メモリ
			μsec/point	μsec/point
行列コピー	$Y_{ij}=U_{ij}$	MMOV(Y,U)	0．5	1．0
行列ベクトル積	$Y_i=\Sigma U_{ij}*V_j$	MVMUL(Y,U,V)	0．7	1．1
行列スカラー積	$Y_{ij}=SA*U_{ij}+SB$	MSMA1(Y,A,U,B)	0．6	1．0
行列スカラー積	$Y_{ij}=SA*U_{ij}+SB*V_{ij}+SC$	MSMA2(Y,A,U,B,V,C)	1．1	1．5
行列積	$Y_{ij}=\Sigma U_{ik}*V_{kj}$	MMUL(Y,U,V)	24．0	40．0
行列積（複素数）	$Y_{ij}=\Sigma U_{ik}*V_{kj}$	CMMUL(Y,U,V)	12．0	20．0
行列積（第一行列転値）	$Y_{ij}=\Sigma U_{ki}*V_{kj}$	MMLT1(Y,U,V)	24．5	40．7
行列積（第二行列転値）	$Y_{ij}=\Sigma U_{ik}*V_{jk}$	MMLT2(Y,U,V)	24．5	40．7
逆行列（ピボットなし）	$Y_{ij}=1/(Y_{ij})$	MINV(Y,A,B)	41．0	67．0
逆行列（列ピボット）	$Y_{ij}=1/(Y_{ij})$	MINVC(Y,A,B)	45．0	72．0
逆行列（複素数、ピボットなし）	$Y_{ij}=1/(Y_{ij})$	CMINV(Y,A,B)	22．0	37．0
逆行列（ウィーナ・レビンソン）	$Y_{ij}=1/(Y_{ij})$	MWLD(Y,A,U,V)	15．5	18．4
一般行列ＬＵ分解（ピボットなし）		MFF(Y,A)	11．1	15．4
一般行列ＬＵ分解（列ピボット）		MFFP(Y,A,U)	12．6	17．5
行列方程式の解（ピボットなし）		MEF(Y,A,U,V)	1．1	1．4
行列方程式の解（列ピボット）		MEFP(Y,A,U,V,W)	1．3	1．6
行列方程式（ウィーナ・レビンソン）		MEWL(Y,IA,U,B,V,W)	23．0	30．0

表12－1　ＭＡＰ 300の主な行列演算ルーチンと演算時間[13]

演　算	計　算　式	コ　マ　ン　ド	演　算　時　間	
			170 nsec 高速メモリ	500 nsec 低速メモリ
			μsec/point	μsec/point
複素ＦＦＴ	$Y_k = \Sigma U_n * \exp(-j2\pi kn/N)$	FFTN(Y,M,U,V,W)	5．2	10．5
複素逆ＦＦＴ	$Y_n = \Sigma U_k * \exp(+j2\pi kn/N)$	FFTIN(Y,M,U,V,W)	5．2	10．5
実ＦＦＴ	$Y_k = \Sigma U_n * \exp(-j2\pi kn/N)$	FFTNR(Y,1,U,V,W)	4．5	7．5
実逆ＦＦＴ	$Y_n = \Sigma U_k * \exp(+j2\pi kn/N)$	FFINR(Y,1,U,V,W)	4．5	7．5
２次元複素ＦＦＴ	$Y_{k1} = \Sigma\Sigma\ U_{mn} * \exp(-j2\pi(mk/NR+n1/NC))$	FFT2D(Y,NR,U,NC,V,W)	7．2	13．3
列複素ＦＦＴ	$Y_{1k} = \Sigma\ U_{1i} * \exp(-j2\pi ki/NC)$	FFROW(Y,NR,U,NC,V,W)	5．5	－
行複素ＦＦＴ	$Y_{k1} = \Sigma\ U_{i1} * \exp(-j2\pi ki/NR)$	FFCOL(Y,NR,U,NC,V,W)	5．5	－
微分	$Y_k = U_k - U_{k-1}$	DDIFF(Y,A,U,B)	0．5	1．0
積分	$Y_k = U_k + Yk-1$	DINTG(Y,A,U,B)	0．6	1．0
２次フィルタ	$Y_k = U_k + a_0 * U_{k-1} + a_1 * U_{k-2} - a_2 * Y_{k-1} - a_3 * Y_{k-2}$	DFL22(Y,A,U,B)	1．3	1．3
重みつきフィルタ	$Y_k = V_k * (U_k - a * U_{k-1})$	DPRE(Y,A,U,B,V)	1．0	1．5
たたみこみ	$Y_k = \Sigma\ U_{kL+b-m-1} * V_m$	DCVM(Y,L,U,V)	4．9	6．8
相関	$Y_k = \Sigma\ U_{k-a+j} * V_k$	DCORZ(Y,IA,U,V)	4．4	4．4

表12－２　ＭＡＰ 300の主なフーリェ解析・時系列処理ルーチンと演算時間[13]

演　算	計　算　式	コ　マ　ン　ド	演　算　時　間	
			170 nsec 高速メモリ	500 nsec 低速メモリ
			μsec/point	μsec/point
加算	$Y_k=Y_k+U_k$	VAD(Y,U)	0．6	1．5
減算	$Y_k=Y_k-U_k$	VSB(Y,U)	0．6	1．5
乗算	$Y_k=Y_k*U_k$	VML(Y,U)	0．6	1．5
除算	$Y_k=Y_k/U_k$	VDV(Y,U)	1．6	1．8
スカラー倍（1）	$Y_k=a*U_k+b$	VSMA1(Y,A,U,B)	0．6	1．5
スカラー倍（2）	$Y_k=a*U_k+b*V_k+c$	VSMA2(Y,A,U,B,V,C)	0．6	1．5
スカラー倍（3）	$Y_k=a*U_k+b*V_k+c*W_k+d$	VSMA3(Y,A,U,B,V,C,W,D)	0．9	1．9
ベクトル・クリア	$Y_k=0$	VCLR(Y)	0．4	1．0
ベクトル・コピー	$Y_k=U_k$	VMOV(Y,U)	0．4	1．0
絶対値	$Y_k=ABS[Y_k]$	VA(Y)	0．4	1．0
符号反転	$Y_k=-Y_k$	VNEG(Y)	0．4	1．0
2乗	$Y_k=Y_k*Y_k$	VS(Y)	0．4	1．0
平方根	$Y_k=a*SQRT[ABS[U_k]$	VSQRT(Y,A,U)	5．2	5．2
逆正接	$Y_k=a*ARCTAN[ABS[V_k/U_k]$	VATN2(Y,A,U,V)	7．5	7．5
マグニチュード	$Y_k=SQRT[U_k*U_k+V_k*V_k]$	VMG(Y,U,V)	3．7	3．7
パワー	$Y_k=UR_k*UR_k+UI_k*UI_k+a*V_k$	VPOW(Y,A,U,V)	1．0	1．9
内積	$a=b*\Sigma U_k*V_k$	SDOT(A,U,B,V)	0．6	1．0
最大値	$Y_k=MAX[Y_k,U_k]$	VMAX(Y,U)	0．6	1．5

表12－3　ＭＡＰ300の主なベクトル演算ルーチンと演算時間[13]

```
5   :
6   :
7   CALL MPCLM(BUS2, MAPM, BASE,
    N, N, REAL4, INC, 0)
8   CALL MPWDB(MAPM, X(1, 1),
    INCH, CNV, X(N, N))
9   CALL MINV(MAPM, MSA, MSB)
10  CALL MPRDB(MAPM, X(1, 1),
    INCH, CNV, X(N, N))
11  :
12  :
```

1行目はホスト計算機の行列データ・エリアの宣言である。2～4行目はMAPの制御コード(データのベース・アドレス，バス番号，データ型等)の値を指定している。7行目はMAPへ制御コードを転送して処理の準備をさせる。この時点で，X(40，40)にはデータがセットされているものとする。8行目はホスト側のX(40, 40)にあるデータをMAPへ書き込む。9行目はMAPに逆行例の計算をさせる。そして10行目はMAPから計算結果をホスト側に読み出す。これで，ホストのX(40, 40)に逆行列が得られる。

6. 予備試験

前章までに述べた『高効率ガスタービン制御システム解析装置』の基本的処理能力を確認するため，ⅰ)多変数関数発生(AD10)，ⅱ)数値積分(AD10)，ⅲ)逆行列計算(MAP 300)，ⅳ)FFT計算(MAP 300)，にいての予備試験を行った。この4つの処理機能は，高効率ガスタービンのシミュレーション，制御システムの設計，評価等の各種の段階で，最も多用される重要な機能である。予備試験では主に処理速度と処理精度を評価検討することとする。特に，ディジタル演算の公称処理速度(表3，表12等)とは理想的条件の時に達成される速度(いわゆる瞬間風速)であって，現実の処理速度はユーザ・プログラム上の制約，ホスト計算機からの制約等により，これを下回るのが普通である。

6.1 多変数関数発生 (AD 10)

4.2で述べた様に，不等間隔ブレーク・ポイントのバイナリ・サーチ(二分法)による非線形多変数発生が最も一般的な関数発生法である。4.5で述べた高級シミュレーション言語(MPS10)を用いて，この関数発生を行うには，例えば，2つの3変数関数F(X，Y，Z)，G(U，V，W)の場合，

```
1：IXDEL DA1 XBSG X 33
2：IXDEL DA1 YBSG Y 33
3：IXDEL DA1 ZBSG Z 33
4：IXDEL DA1 UBSG U 33
5：IXDEL DA1 VBSG V 33
6：IXDEL DA1 WBSG W 33
7：FCNI DA2 F 0 /XBSG YBSG
   ZBSG/
8：FCNI DA2 G 0 /UBSG VBSG
   WBSG/
```

と書かれる。表8に示した様に，IXDELはインデックス・デルタ値を計算し，FCNIは関数内挿計算をする演算モジュールである。最初の1～6行目で入力変数X，Y，ZおよびU，V，Wのインデックス値，デルタ値を求め，7～8行目でその結果を入力として2つの3変数関数発生を行ってF，Gを求めている。ここで，演算モジュールはパイプライン構造のため，同一の処理を同時に進行させることができる。上の例では，DA1と名付けられた1つのインデックス・デルタ・モジュール(IXDEL)で，6つの変数について処理が同時進行しており，DA2と名付けられた1つの関数内挿モジュール(FCNI)で，2つの関数についての処理が同時進行している。IXDELやFCNIの演算時間は，その同時に進行している処理の個数に単純比例はしない。演算時間と同時処理個数の関係をAD10を用いて実測すると，表13および図14が得られる。この表から，例えば，1変数関数発生を行う場合，1つの関数発生のみならば11.2+7.9=19.1μsecを要するが，12の関数発生を同時に処理するならば，24.7+13.7=38.4μsec，つまり，1関数当り 38.4/12=3.2μsec でよいことがわかる。また，インデックス・デルタ・モジュール(IXDEL)の演算時間は，関数内挿モジュール(FCNI)の演算時間とほぼ同程度となっている。一方，図14から，同時処理個数が増加すると，1処理当りの演算時間は反比例的に減少し，その演算モ

表13 インデックス・デルタ（ＩＸＤＥＬ）および
関数内挿（ＦＣＮＩ）演算モジュールの演算時間

同時処理個数	インデックス・デルタ ＩＸＤＥＬ	関数内挿 ＦＣＮＩ			
		１変数	２変数	３変数	４変数
個	μsec	μsec	μsec	μsec	μsec
1	11.2	7.9	9.2	10.1	14.1
2	11.8	8.2	9.8	11.5	15.3
3	12.5	9.0	10.9	13.4	17.0
4	13.1	9.3	11.5	−	−
5	13.8	10.1	12.6	−	−
6	14.4	10.4	13.2	−	−
7	21.5	11.2	−	−	−
8	22.1	11.5	−	−	−
9	22.8	12.3	−	−	−
10	23.4	12.6	−	−	−
11	24.1	13.4	−	−	−
12	24.7	13.7	−	−	−

＊註 この表から得られる多変数関数発生の演算時間は、表３の数値より大きいものとなる。これはＭＰＳ１０言語における手続き処理の為である。

ジュールに許されている最大値と等しい時に最小（演算効率は最大）となる。

表13を用いて，高効率ガスタービンのシミュレーションにおける全関数発生（表２参照，89回の１変数関数発生，28回の２変数関数発生，87回の３変数関数発生）の予測演算時間は次の様に算定される。まず，最悪のケースとして同時処理が全く無い場合を考えると，

１変数関数……（11.2＋7.9)＊89＝1699.9μsec
２変数関数……（11.8＋9.2)＊28＝ 588.0μsec
３変数関数……（12.5＋10.1)＊87＝1966.2μsec
（合計）＝4254.1μsec

となる。次に最良のケースとして同時処理が最大限に実限された場合を考えると，

１変数関数……（24.7＋13.7）＊84/12
＋(13.8＋10.1)＝292.7μsec
２変数関数……（24.7＋13.2）＊24/6
＋(22.1＋11.5)＝185.2μsec
３変数関数……（22.8＋13.4)＊87/3＝1049.8μsec
（合計）＝1527.7μsec

となる。従って，予測される演算時間は，1.53～4.25msecである。上の計算では，各関数の入力変数とブレーク・ポイント・テーブルは全て異なると仮定している。もし，これ等に共通のものがあると，インデックス・デルタ・モジュールの処理回遂が減り，演算時間はさらに減少する。

同時処理（並列処理）をはばむものは，直列性が要求される計算プロセスである。例えば，ある関数発生の結果が，次の関数発生の入力になっている場合，この２つを同時処理させることはできない。ガスタービンのシミュレーションでは，ガスタービン要素（コンプレッサ，タービン，燃焼器等，図４参照）の各計算モジュール内にこの様な直列計算プロセスが存在する。しかし，各直列計算プロセスは互いに独立であるため，図15に示す様に，いくつかの直列計算プロセス内の同じ処理をグルーピングして

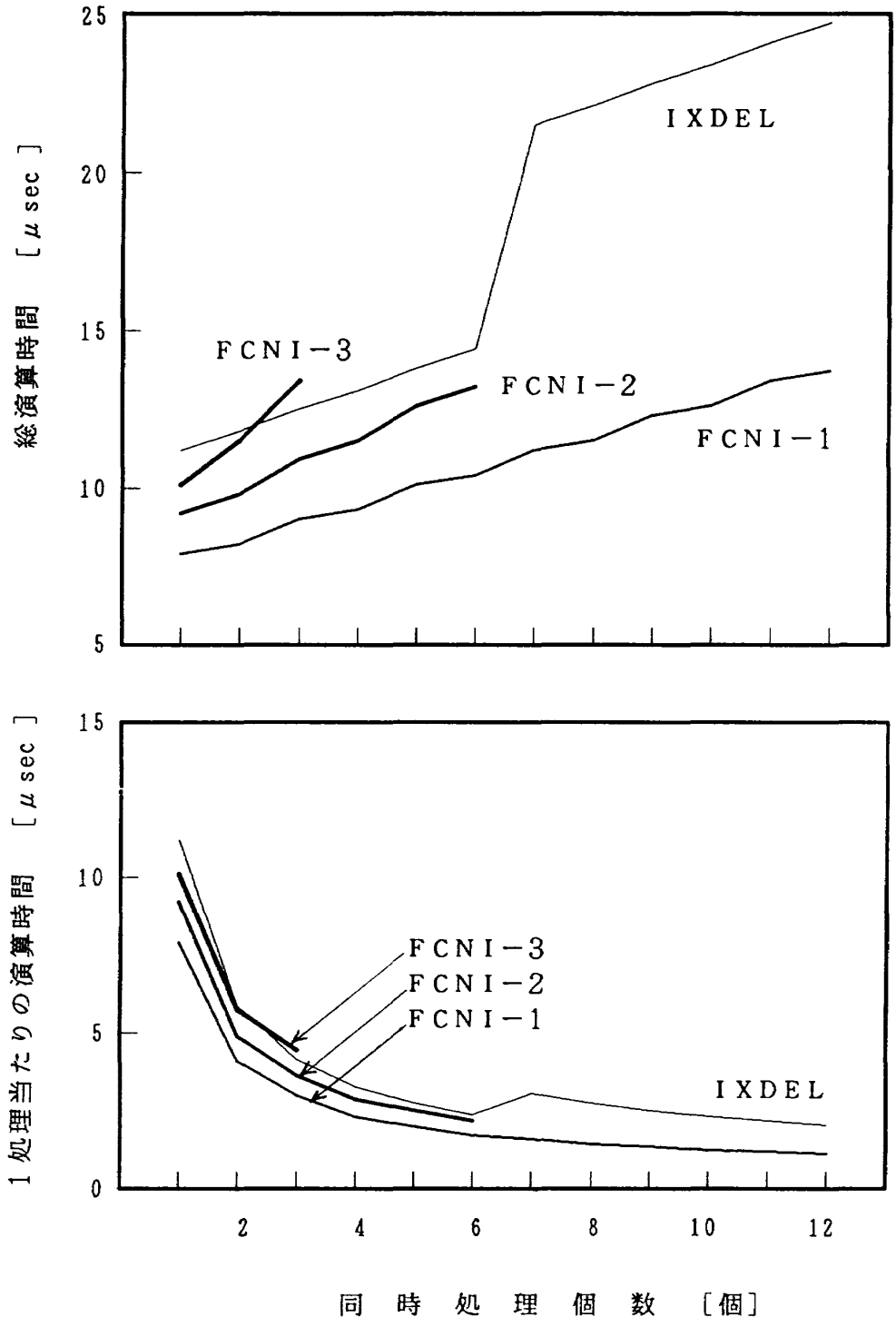
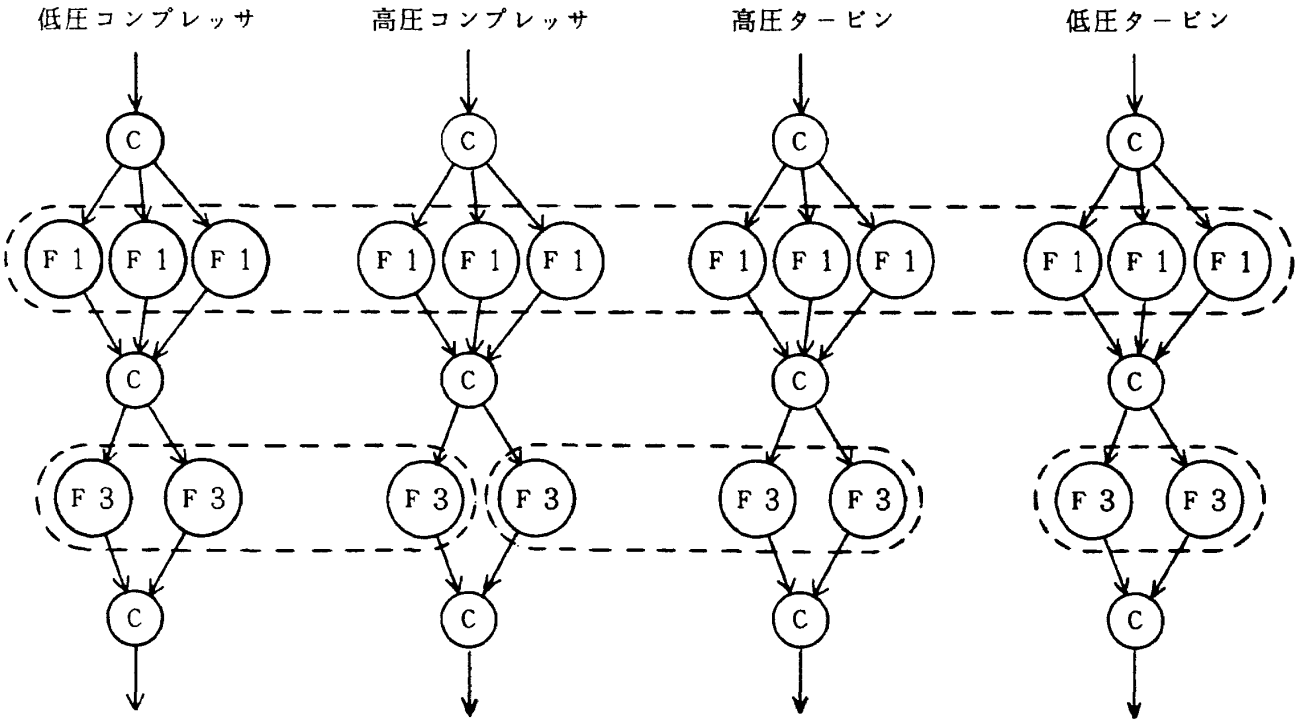


図14　インデックス・デルタ（ＩXDEL）および関数内挿（FCNI）
演算モジュールの総演算時間と１処理当たりの演算時間

同時処理をさせることができる。従って，前述の同時処理が全く不能な最悪のケースは無く，グルーピングの工夫で最良のケースに近づけることができる。この点に留意してプログラミングをすれば，上の高効率ガスタービンのシミュレーションにおける全関数発生の演算時間は２～３ msec程度と予測される。全シミュレーション演算の70～80%は関数発生演算であることから，この演算時間は実時間シミュレーションを実現するに十分であると考えられる。

次に多変数関数発生の演算精度について述べる。ＡＤ10の演算方式は16ビット固定小数点演算であるので，プログラム中の全変数にはスケーリングが必



註 F 1 : 1変数関数発生
F 3 : 3変数関数発生
C : 一般計算

図15 直列計算プロセスのグルーピング

要となる。このスケーリングの適不適は演算精度に大きな影響を及ぼすが，関数発生の場合，関数データ，入力データともにその値は既知であるので，スケーリングは適切に行い得る。従って，関数発生の演算精度は16ビット（±0.003％）に近い精度が期待できると考えられる。例として，ガスタービンのシミュレーションで用いる燃焼ガスの比熱比の関数発生（3変数非線形関数），

$$\kappa = f(T, f/a, s/a) \quad (17)$$

ここで，比熱比：κ ，温度：T

燃空比：f/a，蒸気空気比：s/a

をAD10およびホスト計算機（VAX11/750，32ビット浮動小数点）の両方で，同じアルゴリズムで行い，演算精度を比較する。図16はこの結果を示している。実線は，f/a=0，s/a=0 の時の，比熱比（κ）と温度（T）の関係で，AD10の結果とVAX11の結果はほとんど同一線となる。AD10による関数値（κ_{AD10}）とVAX11による関数値（κ_{VAX}）との差のフルスケール（$\kappa_{FULL} = 1.4016$ ）に対する割合，

$$100(\kappa_{AD10} - \kappa_{VAX})/\kappa_{FULL} \quad [\%]$$

は図中の記号で示してある。これによると，フルスケールに対する演算誤差は，ほぼ±0.006％以下となっている。

6.2 数値積分（AD 10）

高級シミュレーション言語（MPS10）での数値積分は，積分演算モジュール（INTL）で行われる。1つの積分演算モジュールは，最大24個の数値積分を同時進行させることができる。最大入力項数（微分方程式右辺の合計項数）は48項で，積分方式は表4に示したものを含め17種がある。積分演算モジュール（INTL）の演算時間をAD10で実測すると，図17の様になる。図17(A)は積分の同時処理個数からきまる処理時間，図17(B)は入力項数からきまる処理時間で，全演算時間は両者の和である。積分方式によって演算時間は変らない。これは全てのアルゴリズムを演算時間の一番長い実時間型ルンゲ・クッタ4次の演算時間に一致する様してあるためである。（単純なアルゴリズムの積分方式では時間をロスす

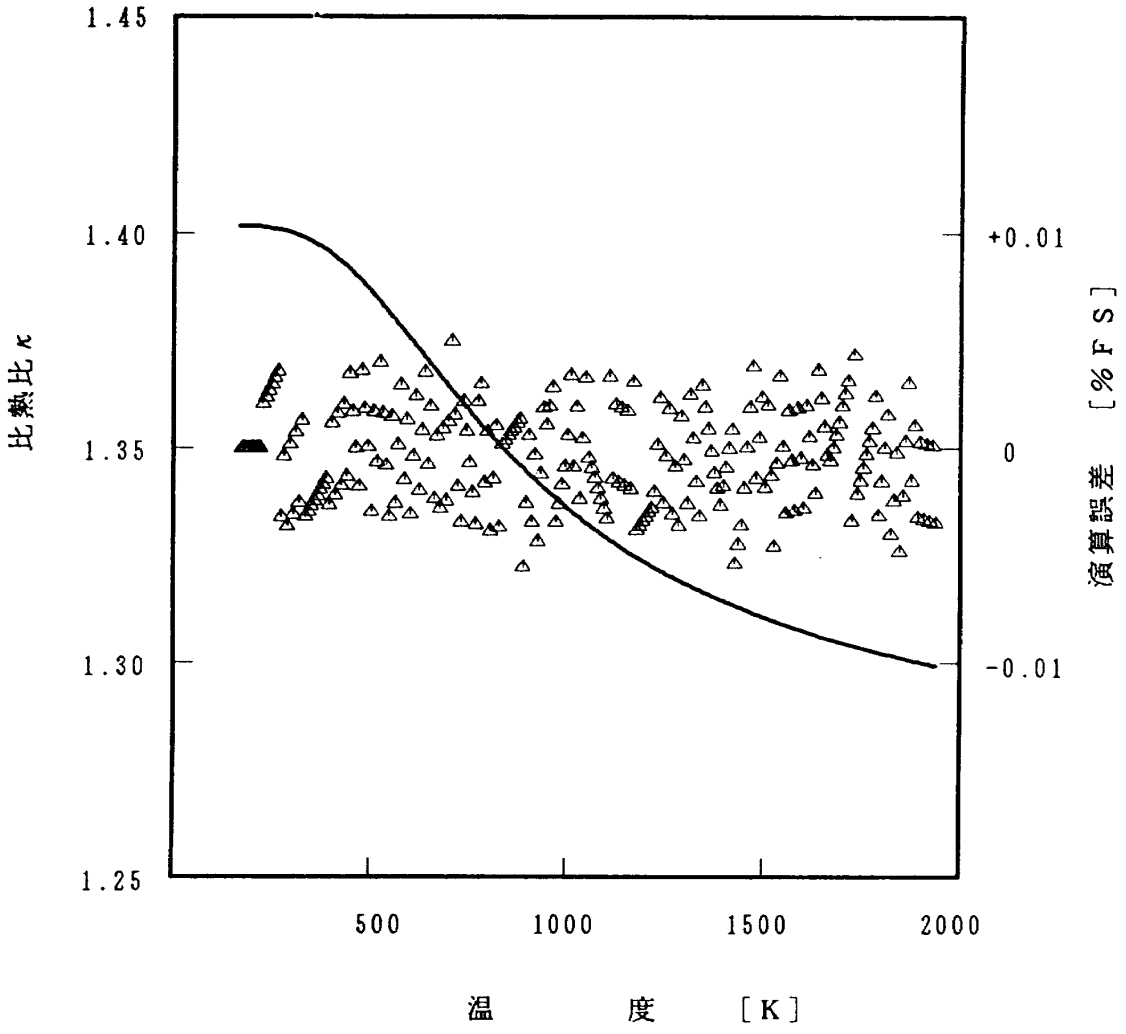


図16　燃焼ガスの比熱比 $\kappa = f(T, f/a, s/a)$ の
関数発生におけるAD10の演算誤差

ることになるが，積分方式の変更が容易となる）。図17から，48の入力項数で，24個の積分を同時に行う場合，全演算時間は112.9 μsec である。高効率ガスタービンのシミュレーションでは，表2の様に積分演算回数は28であるが，制御器，センサー，アクチュエータ等の組込みを考慮しても，48個以上の積分，つまり，2つ以上の積分演算モジュール（INTL）を必要とすることは無い。従って，数値積分の全演算時間はたかだか225.8 μsec であり，前述の全関数発生の演算時間2～3 msecに比較して問題にならない。

各種の数値積分方式の演算精度，安定性については多くの研究（例えば［14］）がある。ここでは，AD10に特有な，実時間型ルンゲ・クッタ法の演算精度について検討する。4.3で述べた様に，実時間型ルンゲ・クッタ法には，i）入力変数の未来値を使用しない，ii）サブ・ステップ毎によい精度で状態変数が得られる，iii）全サブ・ステップの完了後には，同じ次数の古典的ルンゲ・クッタ法と同程度の精度で状態変数が得られる，という特徴がある。以下，4.5で述べた2階微分方程式を用いて，これ等の特徴を評価することにする。

数学的モデルは，

$$a\ddot{x} + b\dot{x} + cx = u$$

計算条件は，

初期条件：$x(0) = \dot{x}(0) = 0$

係　　数：$a = b = c = 1$

外　　力：$u = 0.5$

積分きざみ幅：$\Delta t = 0.0625$ (sec)

とする。このモデルは，理論解をもつため，シミュレーション結果の精度の評価ができる。AD10（MPS10言語）のプログラムの主要部は，

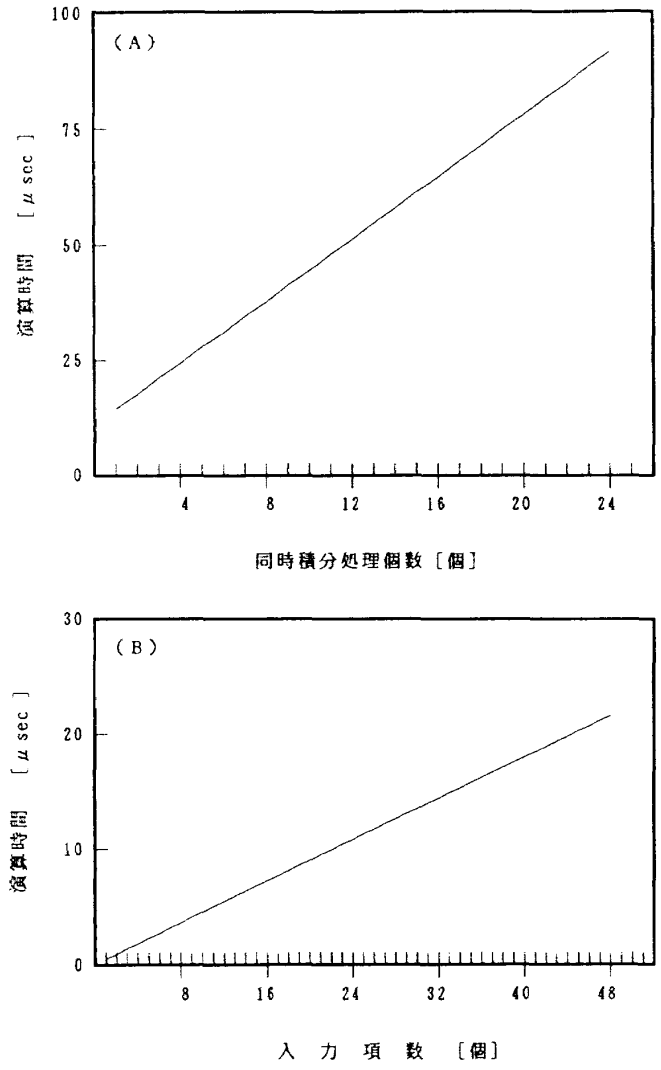


図17 積分演算モジュール（INTL）の，(A)同時積分処理個数からきまる演算時間，および，(B)入力項数からきまる演算時間

INT M1, X=XI $ Y

INT M1, Y=YI $ [-B/A]*Y+[-C/A]*X+U

と書くことができ，これは式(16)と全く同じである。初期条件（XI，YI），係数（A，B，C），外力（U），積分きざみ幅，積分方式等はシミュレーション実行時に指定される。表4に示した積分方式の内，6種類の方式について，このプログラムをAD10で実行し，得られた応答波形 $x(t)$ の理論応答波形 $x(t)_{th}$ に対する誤差を求めると，図18の様になる。ここで，誤差E(t)は次式で定義する。

$$E(t)=100\{x(t)-x(t)_{th}\}/x(\infty) \quad [\%]$$

$$x(\infty)=u=0.5$$

図18において，(A)は理論応答波形 $x(t)_{th}$ である。(B)，(C)はそれぞれAB1，AB2（アダムス・バシュフォース法1次，2次）による。応答誤差の時間変化である。(D)，(E)，(F)はそれぞれRKRT2，RKRT3，RKRT4（実時間型ルンゲ・クッタ法2次，3次，4次）による，また，(G)はRK4（古典的ルンゲ・クッタ法4次）による応答誤差の時間変化である。図18(B)～(G)において，細線はサブ・ステップの結果を結んだもの。太線はサブ・ステップの完了時点のみを結んだものである。(B)，(C)のアダム・バュフォース法は，サブ・ステップをもたないため，細線太線の区別はない。

図18の6種類の積分法の内，サブ・ステップ完了時点での演算語差が最も小さいのはRK4法によるものであり，その誤差は±0.05％以下となっている。一方，演算誤差が最も大きいのはAB1法（オイラー法と等価）によるものであり，その誤差は±1.2％となっている。RK4法のサブ・ステップの完了時点での演算精度は，極めて良好であるが，図18(G)に示す様に，サブ・ステップの途中結果は精度が悪く，シミュレーション出力としては使えない。また，サブ・ステップの途中で入力変数が変化する場合（実際のシミュレーションではこれが普通である）に演算精度低下を生ずる，入力変数の未来値を必要とする，等の欠点がある。これを改良したのが，実時間型ルンゲ・クッタ法であるが，図18(D)，(E)，(F)みられる様に，サブ・ステップの途中結果の精度が向上しており，これをシミュレーション出力として使うことができる。しかし，出力波形は滑らかにならず，ノイズ的な変動を含んだものとなっている。

図18の2次系モデルの例では，AB2，RKRT2，RKRT3が実時間シミュレーションの積分方式として適当と考えられるが，他のモデルにも適当であるとは限らない。積分方式の選択は，実時間型積分方式の上記の様な特性をも考慮して行うべきである。

6.3 逆行列の計算（MAP 300）

次の手順でMAP 300の逆行列計算の計算速度および計算精度を評価する。

(1) 乱数（−1～+1）を用いて，（n×n）正方行列Aを作る。ここで，MAP 300のメモリ容量の制限から，$2 \leqq n \leqq 64$。

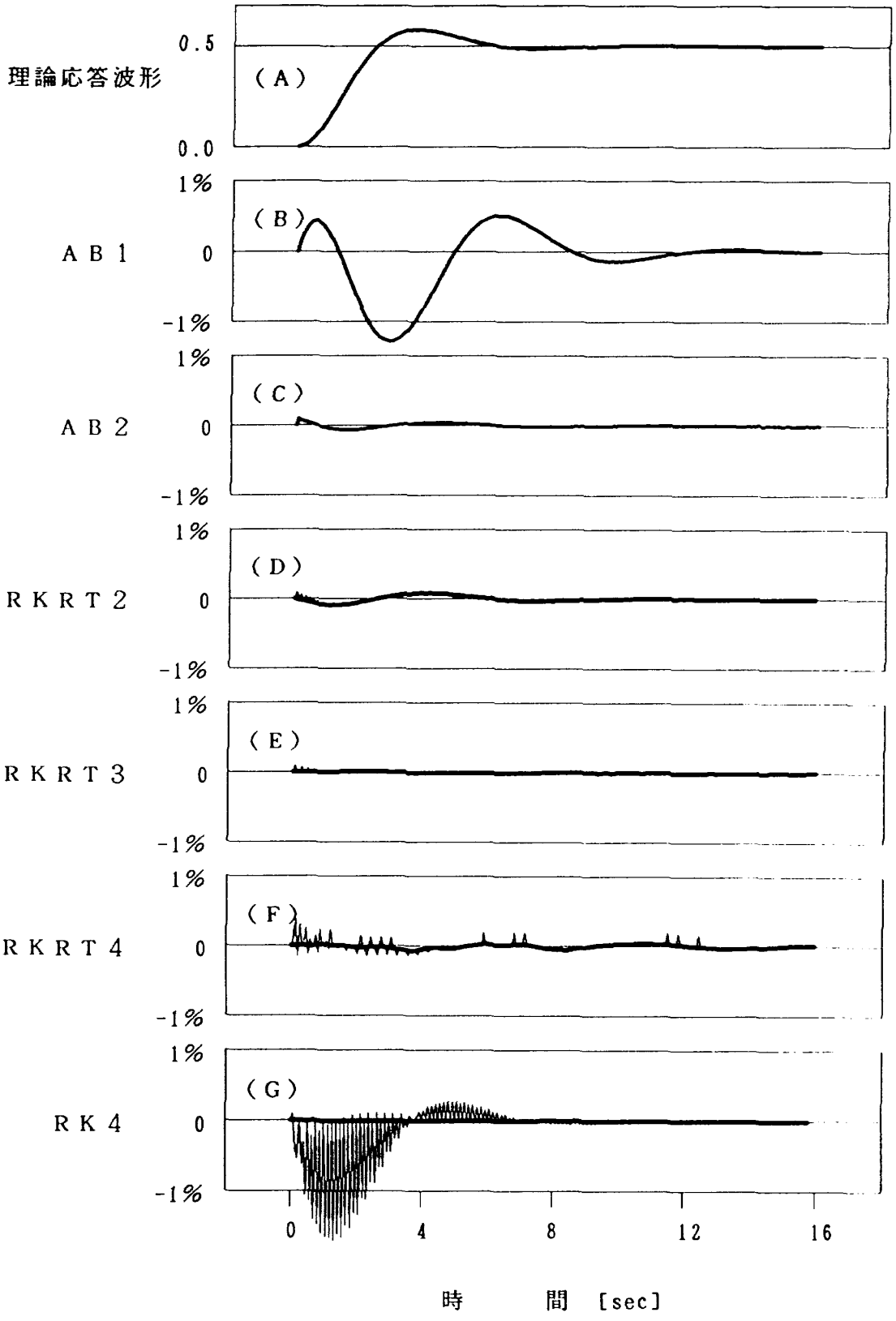


図18　2次系の過渡応答における各積分方式の演算誤差

(2) ＭＡＰ300およびホスト計算機（VAX11/750）上で，ｉ）A^{-1}（逆行列）の計算，ii）AA^{-1}（行列積）の計算，iii）$AA^{-1}-I$（行列差，Ｉは単位行列）の計算，を行い計算時間を計測する。

(3) $B=AA^{-1}-I$は0行列であるべきであるが，数値計算では計算誤差のため０行列にはならない。行列Ｂの要素 b_{ij}（ｉ，ｊ＝1，2，3，……ｎ）の絶対値の最大値，$\max|b_{ij}|$ を計算精度の指標として求める。この指標が０に近い程，計算精度が高いといえる。

この手順を実行するプログラムの主要部は，図19の様に書かれる。プログラムはホスト計算機のオペレーティング・システムの管理下で進行するため，計算時間の計測はオペレーティング・システムの状態により若干のばらつきは避けられない。図20は，(4×4)～(64×64)次元の正方行列について求めたＭＡＰ300の計算速度，ホスト計算機との計算速度比，計算精度を示している。

図20(A)はＭＡＰ300における逆行列計算の所要時間（△記号），入出力（初期行列のＭＡＰ300への転送および結果行列のホスト計算機への転送）の所要時間（◇記号）である。■記号は公称速度（40×40次元，72 msec）であるが，実測値はほぼこれに近い。64×64 次元の正方行列において，逆行列計算の所要時間は約 215msec，入出力の所要時間は約 87 msec であり，入出力にかなりの時間を要することが分る。従って，プログラミングはＭＡＰ300上のみで出来るだけ多くの一連の処理をさせ，ホスト計算機との入出力回数を減らす様にすべきである。図20(B)はＭＡＰ300とホスト計算機との計算速度比で，ＭＡＰ300への入出力時間を含めた場合（◇記号）と含めない場合（△記号）を示している。これによると，ＭＡＰ300はホスト計算機に比べて約65倍速く，入出力時間を考慮しても約47倍速い。図20(C)は計算精度の指標で，ＭＡＰ300による $\max|b_{ij}|_{MAP}$（△記号）とホスト計算機による $\max|b_{ij}|_{VAX}$（×記号），および両者の比（実線）を示している。ＭＡＰ300はホスト計算機に比べ，1桁計算精度が悪いが，これは逆行列の計算アルゴリズムの違いによる。共にガウス・ジョルダン法であるが，ホスト計算機では行列ピボッティングを行なうのに対し，ＭＡＰ300では列ピボッティングしか行なわないものである。

6.4 ＦＦＴの計算（ＭＡＰ300）

次の手順でＭＡＰ300の高速フーリェ変換（ＦＦＴ）の計算速度および計算精度を評価する。

(1) 乱数（−1～＋1）を用いて，2^n 点からなる乱数時系列 x(t) を作る。ここで，ＭＡＰ300のメモリ容量の制限から，$2 \leqq n \leqq 12$。

(2) ＭＡＰ300およびホスト計算機（VAX11/750，32ビット浮動小数演算）上で，同一の乱数時系列 x(t) に対して複素フーリェ変換を行い，計算時間を計測する。

(3) ＭＡＰ300による変換結果 $X(f)_{MAP}$，ホスト計算機による変換結果 $X(f)_{VAX32}$，および，ホスト計算機の倍精度演算（64ビット浮動小数演算）による変換結果 $X(f)_{VAX64}$ を用いて計算精度の評価を行う。

この手順を実行するプログラムの主要部は，図21の様に書かれる。ＭＡＰ300において，フーリェ変換のためのサブルーチンは，データ・タイプ，作業エリアの要不要，次元，等により10以上の種類がある（表12－２参照）。ここでは，使用頻度の高い１次元複素フーリェ変換（ＦＦＴＮ）を用いる。一方，ホスト計算機用のＦＦＴプログラムも多数あるが，文献[15]にあるものを用いることにする。図22は，16～2048点からなる乱数系列のフーリェ変換における，ＭＡＰ300の計算速度，ホスト計算機との計算速度比，計算精度を示している。

図22(A)はＭＡＰ300におけるＦＦＴ計算の所要時間（◇記号），入出力（時系列データのＭＡＰ300への転送および変換結果のホスト計算機への転送）の所要時間（△記号）である。■記号は公称速度（1024点，5.3msec，170nsec高速メモリ使用）で，実測値はほぼこれに近い。図22(A)から，ＭＡＰ300によるＦＦＴ処理では，処理時間の大半は入出力の所要時間（1.2 Ｍ－byte/sec）であって，ＦＦＴ計算時間はその1/5前後となっている。このことは，図22(B)でも明らかで，MAP 300 とホスト計算機とのＦＦＴ計算速度比（◇記号）は200倍にも達するが，入出力所要時間をも考慮した速度比（△記

```
C Matrix Inverse Test
      REAL A(8192),B(8192),C(8192),D(100),E(100)
      INTEGER BUS1,BUS2,BUS3,LG,RL,CT,MA,MB,MC,MD
      DATA    BUS1,BUS2,BUS3,LG,RL,CT,MA,MB,MC,MD
     +       /   1,   2,   3, 0, 0, 1, 1, 2, 3, 4/
      DO 10 N = 4,64                                    ! N = Dimension of Matrix
      CALL FILL(A,N)                                    ! Fill Matrix (A) with Random Number
      CALL MPOPN(3)                                     ! Open MAP
C-----Configure Buffers and Transfer Data to Map-----
      CALL MPCLM(BUS2,MA,0.0,N,N,RL,CT,LG)              ! Configure MAP Buffer <MA>
      CALL MPCLM(BUS3,MB,0.0,N,N,RL,CT,LG)              ! Configure MAP Buffer <MB>
      CALL MPCLM(BUS3,MC,FLOAT(2*N*N),N,N,RL,CT,LG)     ! Configure MAP Buffer <MC>
      CALL MPCMB(BUS1,MD,50000.0,FLOAT(N),RL,CT,LG)     ! Configure MAP Buffer <MD>
           IT0 = TIMER(0)                               ! Current Time
      CALL MPWDB(MA,A(1),4,1,A(N*N))                    ! Transfer Matrix Data to MAP (A)-->&lt;MA&gt;
           IT0 = TIMER(IT0)                             ! Elapse Time
C-----Execute Inverse, Multiply in MAP-----
      CALL MMOV (MB,MA)                                 ! Move Data <MA>--><MB>
           IT1 = TIMER(0)                               ! Current Time
      CALL MINVC(MA,50,MD,52)                           ! Matrix Inverse 1/<MA>--><MA>
           IT1 = TIMER(IT1)                             ! Elapse Time
      CALL MMUL (MC,MA,MB)                              ! Matrix Multiply <MA>*<MB>--><MC>
C-----Execute Same in Host-----
      CALL MCPY (A,B,N,N,0)                             ! Move Data (A)-->(B)
           IT2 = TIMER(0)                               ! Current Time
      CALL MINV (A,N,DET,D,E)                           ! Matrix Inverse 1/(A)-->(A)
           IT2 = TIMER(IT2)                             ! Elapse Time
      CALL GMPRD(A,B,C,N,N,N)                           ! Matrix Multiply (A)*(B)-->(C)
C-----Read Answers Back from MAP-----
           IT3 = TIMER(0)                               ! Current Time
      CALL MPRDB(MC,A(1),4,1,A(N*N))                    ! Transfer Result Data to Host <MC>--><A)
           IT3 = TIMER(IT3)                             ! Elapse Time
      CALL MPCLS(0)                                     ! Close MAP
       :
       : (compare answers)
       : (calculate timing results)
       :
10    CONTINUE
```

図19　逆行列計算テスト・プログラム主要部

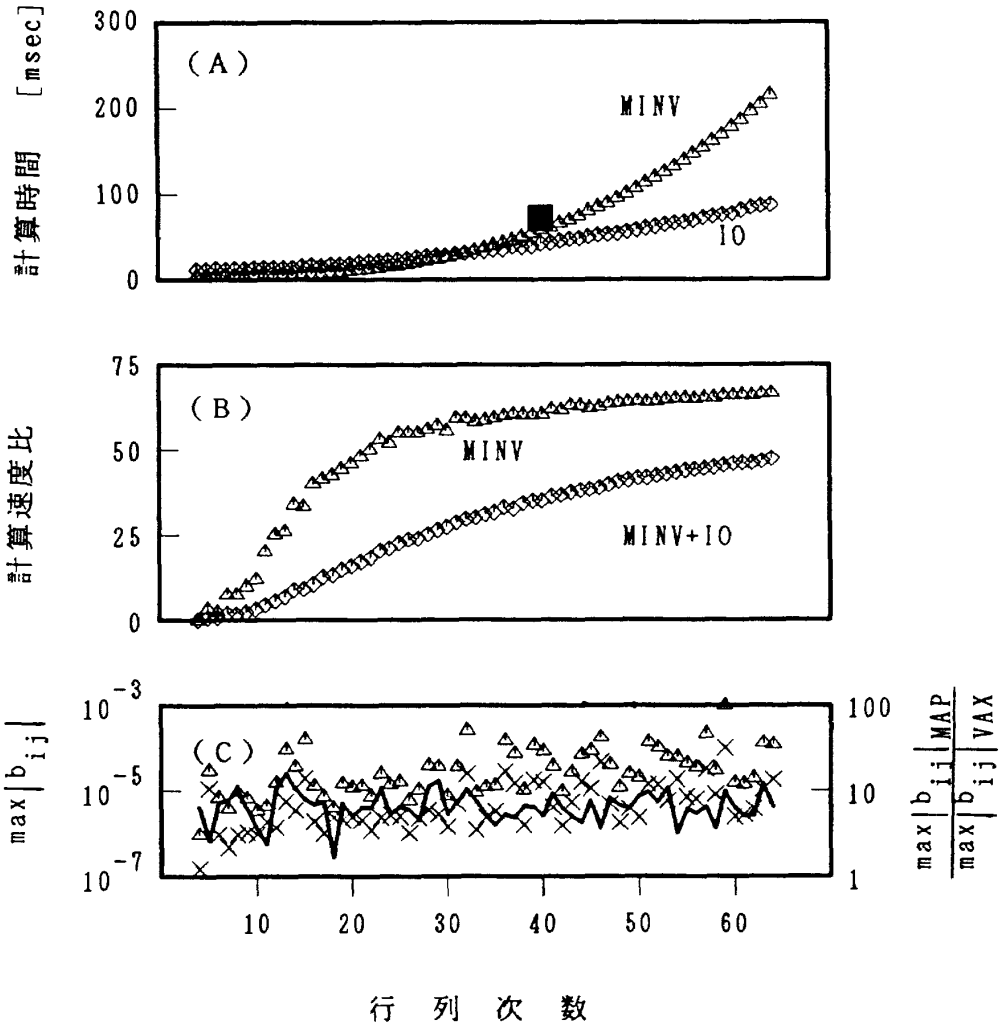


図20 MAP300とVAX11/750による逆行列計算の
計算時間(A), 計算速度比(B), 計算精度(C)

号)は, 30倍程度である。従って 6.3 で述べた逆行列の計算の場合と同様に, プログラミングはMAP 300上のみで出来るだけ多くの一連の処理をさせ, ホスト計算機との入出力回数を減らす様にすべきである。実際のFFT処理では, FFT計算に引き続き, パワー計算, アベレージング計算, 逆FFT計算, 等が行われることが多いため, 全体処理時間に対する入出力時間の比率を低下させることができる。また, 長い時系列データや多チャンネルの時系列データを次々とFFT処理していく場合には, 複数のデータ・バッファを用意して, 入力, 計算, 出力オーバーラップさせて処理を進行させる(これをダブル・バッファリングと呼ぶ)ことによりMAP 300の計算効率を向上させることもできる。

図22(C)は, ホスト計算機倍精度演算による計算結果を基準とした, MAP 300計算誤差(△記号)およびホスト計算機単精度演算の計算誤差(×記号)をプロットしたものである。FFTのアルゴリズムは多種類あるが, その違いは主に, データ・タイプ(実数, 複素数), 作業エリアの要不要, データ並び替えの要不要, COSテーブルの作り方, 次元, 等に起因するもので本質的には変わらない。従って, データのビット長が同じであれば, どのFFTアルゴリズムによっても, ほぼ同じ結果が得られると考えられる。図22(C)によると, 計算誤差はMAP 300, ホスト計算機ともに10^{-7}前後と同じになっている。若干の差はデータ・フォーマットの違いによるもので, MAP300は指数部7ビット, 仮数部25ビット, IBMフォーマットであるのに対して, ホスト計算機は指数部8ビット, 仮数部24ビットのDECフォ

```
C      FFT Test
       COMPLEX*8  A(2048),B(2048)
       COMPLEX*16 D(2048)
       INTEGER BUS1,BUS2,BUS3,LG,RL,CT,CPX,ONE,COST,MB,MA
       DATA     BUS1,BUS2,BUS3,LG,RL,CT,CPX,ONE,COST,MB,MA
      +       /   1,   2,   3, 0, 0, 1,   1,   6,   3, 4, 5/
       DO 10 NN = 4,11
       N = 2**NN                                          ! N = Number of Data
       CALL FILL (A,D,N)                                  ! Fill Time Series (A),(D) with Random Number
       CALL MPOPN(3)                                      ! Open MAP
C-----Configure Buffers, Create Cosine Table and Transfer Data to Map-----
       CALL MPCMB(BUS3,MA,0.,FLOAT(N),CPX,CT,LG)          ! Configure MAP Buffer <MA>
       CALL MPCMB(BUS2,MB,0.,FLOAT(N),CPX,CT,LG)          ! Configure MAP Buffer <MB>
       CALL MPDCV(ONE,1,FLOAT(N),RL)                      ! Define Constant Vector for Use by VCOS
       CALL MPWST(50,1.0/FLOAT(N),1,1)                    ! Define Scalar Value
       CALL MPCMB(BUS1,COST,53192.,FLOAT(N),RL,CT,LG)     ! Configure MAP Buffer <COST>
       CALL VCOS (COST,0,ONE,50,ONE,0)                    ! Create Cosine Table for FFT
            IT0 = TIMER(0)                                ! Current Time
       CALL MPWDB(MA,A(1),8,1,A(N))                       ! Transfer Time Series Data to MAP, (A)--><MA>
            IT0 = TIMER(IT0)                              ! Elapse Time
C-----Execute FFT in MAP-----
            IT1 = TIMER(0)                                ! Current Time
       CALL FFTN(MB,1,MA,COST,MB)                         ! MAP FFT <MA>--><MB>
            IT1 = TIMER(IT1)                              ! Elapse Time
C-----Execute Same in Host-----
            IT2 = TIMER(0)                                ! Current Time
       CALL FFT (A,NN)                                    ! Single Precision FFT (A)-->(A)
            IT2 = TIMER(IT2)                              ! Elapse Time
       CALL FFTD(D,NN)                                    ! Double Precision FFT (D)-->(D)
C-----Read Answers Back from MAP-----
            IT3 = TIMER(0)                                ! Current Time
       CALL MPRDB(MB,B(1),8,1,B(N))                       ! Transfer Result Data to Host, <MB>-->(B)
            IT3 = TIMER(IT3)                              ! Elapse Time
       CALL MPCLS(0)                                      ! Close MAP
        :
        : (compare answers)
        : (calculate timing results)
        :
10     CONTINUE
```

図21　ＦＦＴ計算テスト・プログラム主要部

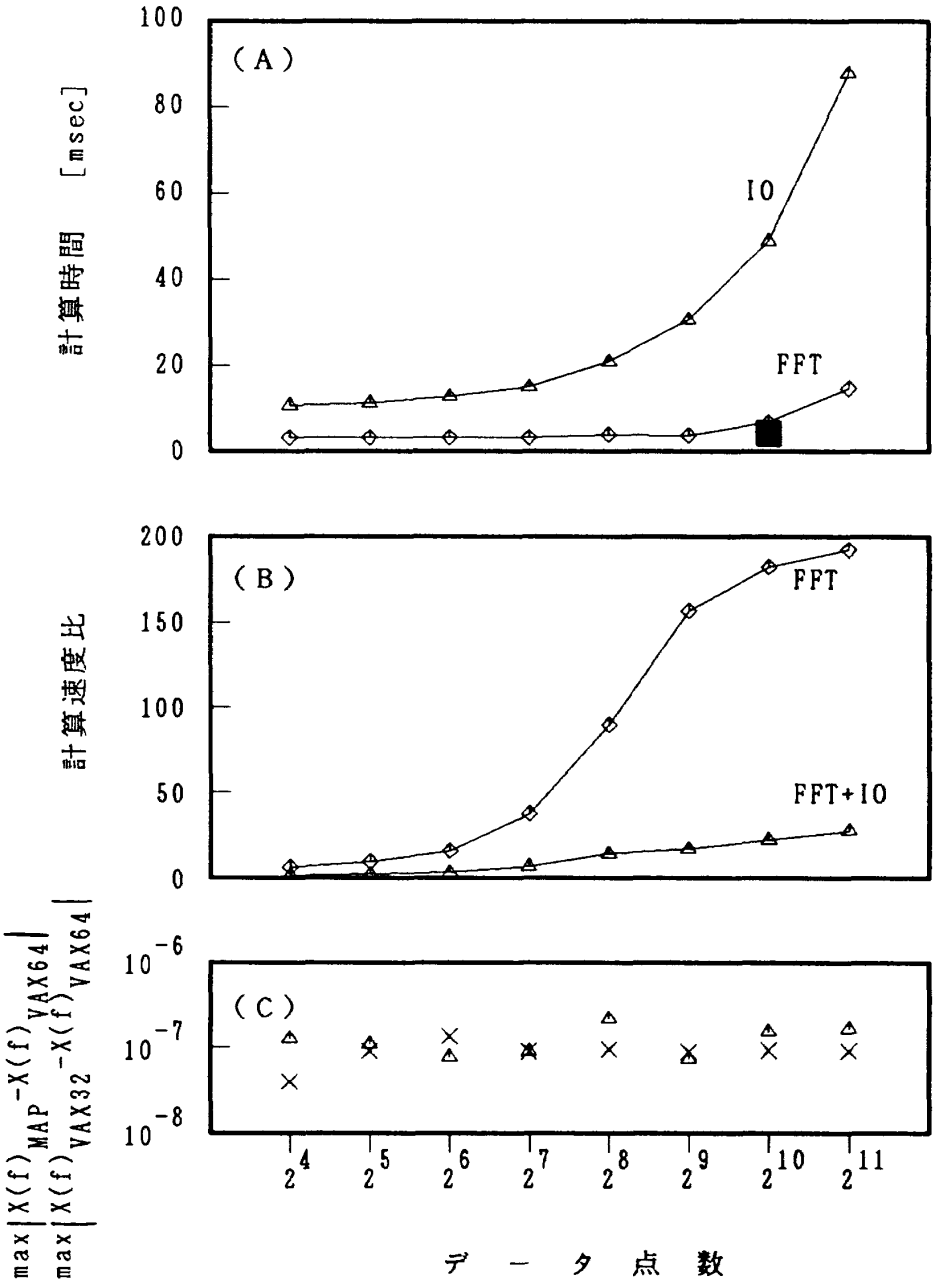


図22 MAP300とVAX11/750によるFFT計算の
計算時間(A)，計算速度比(B)，計算精度(C)

ーマットが採用されている。

7. むすび

『高効率ガスタービンの研究開発』における制御システム技術の研究開発に用いる『高効率ガスタービン制御システム解析装置』の基本計画，ハードウェア，ソフトウェア，および予備試験について述べた。この装置は超高速ディジタル・シミュレータ(AD10)と高速演算装置(MAP300)とを，汎用ミニコン(VAX11)に接続したもので，比較的小規模ながら高い演算処理能力をもち，基本計画を完全に満足するシステムとなっている。この装置を用いて，i)高効率ガスタービンのシミュレーションの研究，ii)高効率ガスタービンの動特性の研究，iii)高効率ガスタービンの制御方式の研究，等が引き続き行われている(これ等の研究については，逐次報告していく予定である)。

一方，高効率ガスタービンのパイロット・プラントの工場試運転試験(高砂)およびサイト運転試験(袖ケ浦)に先立ち，実機制御システム(高効率ガ

スタービン技術研究組合製作）は，二度，当研究所に搬入され，本『高効率ガスタービン制御システム解析装置』との接続による，閉ループ・シミュレーション・テストが行われた。これは，起動制御，定常制御，負荷制御，異常制御等，実際のプラントが遭遇する種々の制御モードをカバーする大規模なシミュレーション・テストで，延べ1000時間にわたって行われた。この結果，パイロット・プラントの運転に際し，制御システムは良好に作動したと報告されている。

文　献

(1) Hori, A., and Takeya, K., "Outline of Plan for Advanced Reheat Gas Turbine". ASME paper 81-GT-28, 1981.

(2) Takeya, K., and Oteki, Y., "Technical Problems on Advanced Reheat Gas Turbine Under the Moonlight Project", 1983 Tokyo International Gas Turbine Congress, 83-TOKYO-IGTC-117, 1983.

(3) Karplus, W. J., "Peripheral Array Processors : Selection and Evaluation", Proc. Conference on Peripheral Array Processors, San Diego, Oct. 21-22, 1982, pp. 1-12.

(4) Gilbert, E. O., and Howe, R. M., "Design Considerations in a Multiprocessor Computer for Continuous System Simulation", AFIPS-Conference Proceedings Vol. 47, 1978, pp. 385-393.

(5) Gilbert, E. G., and Howe, R. M., "An Expanded Role for Function Generation in Dynamic System Simulation", Proc. 1977 Summer Computer Simulation Conference, Chicago, July 18-20, 1977, pp. 305-308.

(6) Gilbert, E. O., "A Special Digital Computer for High Speed Function Generation in Hybrid and Digital Simulation", Proc. 8th AICA Congress, Delft, August 23-28, 1976, North-Holland Publishing Co., Amsterdam, pp. 471-482.

(7) Bernstein, D. S., "The Treatment of Inputs in Real-Time Digital Simulation", SIMULATION, August, 1979, pp. 65-68.

(8) Applied Dynamics International, "AD10 Hardware Reference Manual".

(9) Applied Dynamics International, "AD10 Software Reference Manual".

(10) Applied Dynamics International, "MPS10 Reference Manual".

(11) Cohler E. U. and Storer J. E., "Functionally Parallel Architecture For Array Processors" I.E.E.E. COMPUTER magazine, September 1981, pp. 28-35.

(12) Cohler J. C., "Array Processors Enhance Mini Computer Performance", Mini-Micro Systems, April 1982.

(13) CSPI, "MAP-200/300 Specifications".

(14) 渡辺，"飛行制御・飛行シミュレーションにおける実時間ディジタル積分と積分きざみ幅"，航技研報告TR－743，1982

(15) 日本機械学会，"技術資料・流体計測法"，1985，pp254

航空宇宙技術研究所報告895号
昭和60年12月発行
発行所 航空宇宙技術研究所
東京都調布市深大寺東町7丁目44番地1
電話武蔵野三鷹(0422)47-5911(大代表)〒182
印刷所 株式会社 実業公報社
東京都千代田区九段南4－2－12

Printed in Japan