ブロック線図のディジタル・シミュレーション・プログラム-BACS*

松浦卓丈** 外山守城**

Abstract

BACS is a problem-oriented language for performing dynamic simulation of complex control systems as an analysis and synthesis aid. The design considerations which entered into the development of BACS are the simplicity of usage, the flexibility of digital programming, the low cost in simulation performance and the logical balance among them. An input program to BACS is expressed as a description of a block-diagram and its object propram to be generated is automatically executed through BACS. The solutions are then printed out digitally and graphically. BACS program consists of BACS compiler and many routines. BACS language is made up of block oriented functions, only a few connection rules and FORTRAN and makes its programming so easy that control-oriented engineers who are unfamiliar with digital computer techniques can perform digital simulation. When special functions such as complex non-linear elements other than standard operators are required, users can define them in the form of algebraic functions coded with FORTRAN, use them with combination of standard operators and also utilize appropriate subroutines of FORTRAN library. Successive runs in the same model for various parameter values can be performed without recompilation.

Finally, the compilation time for BACS user program is negligible in comparison with the execution time. Therefore, the run cost in simulation with BACS is lower than with other languages.

要　約

制御系の解析とシンセシスの手段として，制御系の動特性をシミュレーションするための言語 BACS (Block diagram Analysis Compiler System) を開発した．BACSを開発するときの基本方針は，使用法の簡便さ，ディジタルプログラミングの融通性，シミュレーションコストの低廉，およびそれらをバランスさせることであった．

BACS へのインプットプログラムは，ブロック線図指向型であり，オブジェクトプログラムは，自動的に BACS により実行され，シミュレーション解は，印字形式とグラフ形式で与えられる．

BACS プログラムは，BACS コンパイラーと多くの実行ルーチンから構成されている．

BACS 言語は，ブロック線図の演算函数と，二，三の結合規則からなっており，ディジタル計算機の利用に不馴れな制御系技術者でも，容易に BACS インプットプログラムを作成することができる．

標準演算函数以外に，たとえば，複雑な非線型函数のような特殊函数が必要ならば，使用者は，FORTRAN で必要な代数函数を作成し，標準演算函数と結びつけて使用することができるし，また，FORTRAN の他のライブラリーを使用することもできる．

同一モデルの多様な条件下のシミュレーションが，再コンパイルをすることなく実行されるので，turn around time が相当減少される．

最後に，BACS によるコンパイル時間は，実行時間に比較して，無視できる程度であるから，シミュレーションにおけるランコストは，他の言語の場合よりも低い．

* BACS-A Block Diagram Oriented Digital Simulation Program, by Tsunetomo Matsuura and Moriki Toyama (Mitsubishi Electric Co.)

** 三菱電機株式会社

1. まえがき

ディジタルシミュレーションの必要性は，最近の制御系の規模の大きさ，複雑さ，要求される精度によって，ますます増大している．一方，ディジタル計算機の進歩，およびソフトウェアの発達により，ディジタル計算機のパフォーマンスは向上し，制御系の解析・設計の手段として，ディジタルシミュレーションは，急速に普及しつつある．現在は，大型高速の汎用計算機が各所に設置されているから，制御系のシミュレーションを行なう場合，ディジタル計算機固有のプログラミング技術を必要としない，制御系指向型シミュレーション言語を使用すれば，シミュレーションの総合的パフォーマンスを高めるのに非常に役立つ．このため各種のシミュレーション言語，たとえば，MIDAS, PACTOLUS, MIMIC, DES-1, DSL/90, CSMP などが開発された．

本論文では，ディジタルシミュレーションにおける筆者らの経験に基づいて，ディジタルシミュレーション言語が，十分実用的であるための条件をあげ，筆者らの開発した制御系用ディジタルシミュレーション言語 BACS について述べる．

BACS は，アナログ計算機の使用法に親しんだ制御系技術者が多数存在し，しかも，アナログ計算機はそれらの技術者の要求をみたしえないという現状を，開発の出発点としている．しかし，問題の性質としてアナログ計算機の解法に一度変換して解決するのではなく，直接微分方程式を考察の対象とする方がよい場合が多くある．そのため，BACS は，ブロック線図指向型のシミュレーションプログラムであると同時に，常微分差分方程式の数値解用の汎用プログラムとしても設計されている．

BACS は，当初 1964 年に IBM 7090 用に開発され[1]，筆者らの所属する企業の技術者によって，電力プラント，プロセスコントロール，航空などの分野で広く利用された．その後，IBMS/360 用に改良・拡張された．BACS は，ディジタル計算機の利用について全く経験をもたないか，あるいは，FORTRAN の数日コースを受講した程度の技術者によって，障害なく利用される．

BACS のソースプログラムの 70% は，FORTRAN で書かれ，BACS の最も重要な働きをする部分である残りの30%は，アセンブラ言語でコーディングされている．

BACS は，最低 97K バイトのコアメモリーを使用し，使用されるブロック線図の函数の増加に応じて，拡張可能である．オペレーションは，標準のオペレーティングシステム OS のもとで実行され FORTRAN コンパイラーの使用する I/O ユニットの他に，1 つのロジカルユニットを必要とする．

2. BACS の設計方針

シミュレーション言語のもつべき条件を考える前に，ディジタルシミュレーションに対して，制御系技術者から出される要求について考えてみる．

第 1 に，制御系シミュレーションの目的は，制御系の解析とシンセシスであるから，パラメーターはもちろん，回路の変更による繰り返し計算が，迅速に行なわれる必要がある．

第 2 に，ディジタル計算機の利用料金は，低下する傾向にあるが，計算量が多い場合，計算機使用料が問題となる．

第 3 に，制御系技術者が，簡便にディジタル計算機を利用できることが必要である．

以上の 3 項目以外に，記憶容量，計算速度が検討されるべきであるが，現在の大型計算機のそれらは，制御系技術者の要求をみたしているといってよい．

次に，アナログアプローチに比較して，ディジタルアプローチのもつ潜在的利点をあげておく．

1. 処理される問題の容量の大きいこと．
2. 使用される非線型函数の数および複雑さに対する制限がゆるい．
3. 解の精度を使用者の指定に従って高められる．
4. 時間および変数のスケーリングの必要がない．
5. むだ時間函数の作製が簡易である．
6. 特別な計算作業および機械保守の必要がない．
7. ドキュメンテーションの完全な自動化．
8. パラメーターのリセットとランコントロールの自動化．
9. プログラムの保存および後日の追加計算が簡単．

先に述べた要件を達成し，ディジタルアプローチの潜在的特徴をいかすために，シミュレーション言語 BACS の開発に際して考慮した事項を次にあげる．

(1) シミュレーション言語は，ブロック線図指向型であるべきである．ブロック線図は，現代の自動制御理論と密接な関連をもっており，シミュレーション言語がブロック線図指向型であることは，制御系の解

析とシンセシスに便利である．

（2） 制御系技術者が，簡単にシミュレーション言語を利用できなければならない．利用者に要求されるディジタルプログラミング技術は，可能な限り少なくするべきである．シミュレーションのインプットプログラムでは，ブロック線図，あるいは微分方程式との対応が，アナログ計算機プログラムにおけるよりも，もっと明確にみられることが望ましい．これによって，インプットプログラム中に現われる使用者の不注意な誤りを防ぐことができる．

（3） ディジタルシミュレーションの長所の1つは，複雑な非線型機能の扱いが簡単なことであるから，シミュレーション言語は，ディジタルプログラミングにおける融通性をもたなければならない．そして，シミュレーション言語では，ディジタルプログラミングにおける融通性が，使用の際の簡便さと両立しなければならない．DDC における計算機，最適化機構，各分野における固有の複雑な非線型函数などを含んだ系をシミュレーションする場合，シミュレーション言語に組み込まれている標準的函数だけでは，その制御系のシミュレーションのインプットプログラムを作成することはできないので，制御系技術者は，シミュレーションプログラム内で，FORTRAN 型コンパイラー言語を使用しなければならない．また，初期値，定数が複雑な計算で算出される場合には，使用者の作成した初期条件算出用のメインプログラムが，タイミングルーチンをもったシミュレーションプログラムをサブプログラムとして，呼びだせることが望ましい．融通性をもたないシミュレーション言語は，アナログ計算機の単なる代用となる可能性をもつ．

（4） シミュレーションを行なうとき，使用者のインプットプログラムに対するコンパイラなどの実行以前に使用される計算機使用時間が実行時間に比較して無視できる程度に短いことが必要である．ブロック線図の中には，パラメーターが多くあり，特に制御系のシンセシスでは，回路の変更がしばしば行なわれる．したがって，シミュレーション言語のコンパイラーを含めたオペレーティングシステムの効率は，シミュレーションのコストパフォーマンスに大きな影響を与える．シミュレーション言語のコンパイラー機能が悪い場合には，シミュレーション言語より FORTRAN 型コンパイラー言語を利用する方が，コストパフォーマンスが良いことがある．

（5） 使用者が，オンライン用遠隔インプットアウトプット装置を利用して，オンラインシミュレーションを行なえることが望ましい．バッチ処理計算機でのオンラインシミュレーションでは，多くのむだ時間が生じるため，シミュレーションコストが高くなるので，大型計算機によるタイムシェアリングシステムによって，オンラインシミュレーションが行なわれることが理想的である．

ディジタルシミュレーションプログラムを開発する場合，上に述べた各項目は，第1項と第5項を除いて，互いに矛盾するものである．使用者が簡単に使えるプログラムは，融通性を失ったり，あるいは，コンパイラーが複雑になって使用者プログラムのコンパイルに多量の時間を必要とする．また，融通性をもつプログラムは，複雑な使用規則と多くのコンパイル時間をしばしば要求する．

実用的ディジタルシミュレーションプログラムは，このように互いに両立しない要求をすべてみたし，それらの要求の間に，よいバランスをもつものでなければならない．

BACS は，上記の第1項から第4項までをみたすことを目的として作成されたブロック線図指向型のシミュレーションプログラムであり，その特徴は，使用法の簡単さ，ディジタルプログラミングの融通性，オペレーティングシステムの効率の良さ，およびそれらのバランスである．

次に，BACS の構造および使用例について述べる．

3. BACS のプログラム構造

BACS プログラムは，BACS コンパイラーと積分演算を実行するためのルーチン，たとえば，時間コントロール，積分，アウトプット，種々の伝達函数，インプットプログラム診断などのルーチンからなっている．

コンパイラーによって生成される実行プログラムはブロック線図の構造を表現するルーチンであり，あらかじめ確保された計算機のコアメモリ中に作成され，実行メインプログラムによって呼び出される．

BACS プログラムを作成するとき，筆者らの最も努力した点は，コンパイル，リンクエディットを含めた実行以前に使用される計算機時間を可能な限り減らし，ランコストを下げることであった．BACS は，インプットプログラムが一度 FORTRAN プログラムに変換され，それから実行プログラムにコンパイルされる，いわゆる two-pass コンパイラーではなく，1個

の BACS インプットステートメントが，直接数個の機械語命令に変換される one-pass コンパイラーである．したがって，BACSインプットプログラムのコンパイル時間は，実行時間に比較して無視できる程度に短い．使用者が特殊函数を FORTRAN で作成し，それを BACS 内で使用するときには，追加された FORTRAN サブプログラムのみが，FORTRAN トランスレーターとリンケージエディターによってプロセスされる．

計算のプロセスは，次のとおりである．まず，インプットカードが読み込まれる．もし，インプットプログラムに誤りがあるならば，エラーチェックルーチンがそれを指摘して，計算は中断される．エラーチェックルーチンを通過したインプットプログラムは，BACSコンパイラーによって，微分方程式の機械語プログラムに変換され，積分ルーチン，アウトプットルーチン，データーセットルーチンなどと結合されて実行される．**Fig. 1** は BACS のプログラム構成を示す．

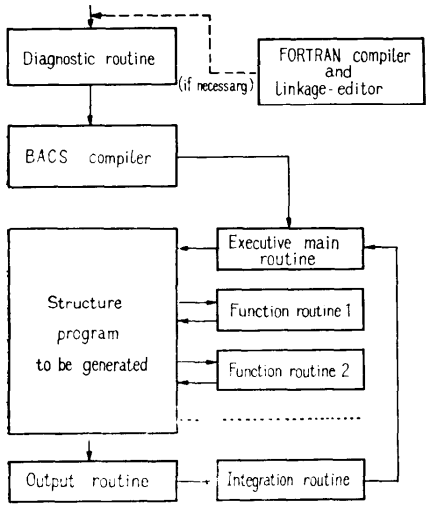


Fig. 1　Program structure of BACS

4. インプット言語

BACS インプットプログラムは，(1) ブロック線図の伝達函数とその接続を示す構造プログラム，(2) 使用者が作成する FORTRAN サブプログラム形式の使用者函数，(3) パラメーターに対するインプットデーターによって構成される．BACS は，**Table 1** に示されているように，アナログ計算機の要素をすべて含む多数の標準的函数をもっているが，これらの函数だけでは不十分な特殊なシミュレーションを行なう場合，使用者がそのうちで FORTRAN を自由に使うことができる函数を多くもっている．

ブロック線図の構造を表現する形式には，FORTRAN で用いられているように，変数名が英数字で表わされる言語的形式と，略式記号形式の２種類とがある．インプット量の減少，コーディングの簡単さおよびコンパイル時間の短縮の３つの理由から，後者が採用された．

BACS では，使用者は，ブロック線図を BACS 函数に分解し，各函数の出力側にランダムに数字で名前

Table 1　BACS functions in standard library

演算函数	使 用 例	BACS 構造ステートメント
積 分	5 → 1/S → 10	5 *I* = 10 0.0
一次遅れ	6 → 1/(1+ST) → 4	6 *F* *T* = 4 1.0
加 算	20, 30 → + → 45	20 + 30 = 45
減 算	3, 5 → + − → 10	3 − 5 = 10
乗 算	5 → M(8) → 10; 12 → 3.16 → 14	5 * 8 = 10 12 * 3.16 = 14
除 算	6 → D(5) → 8; 135 → 1/CON → 20	6 / 5 = 8 135 / CON = 20
無駄時間	10 → $e^{-1 \cdot S}$ → 12	10 *D* *L* = 12
飽 和	35 → 0.5 → 40, slope = *K*	35 *S* = 40 *K* 0.5
リミッター	3 → 4, slope = 1.5	3 *L* = 4 1.5
不感帯	8 → 0.5 → 10, slope = *C*	8 *Z* = 10 *C* 0.5
使用者函数*	4 → FUNC 15 → 18, parameters *A B*	4 (15) = 18 *A B*

その他に，ヒステリシス，バックラッシュ，ロジカルスイッチ，sin，cos，exp，log，arctan，$\sqrt{\ }$，絶対値，－様乱数，正規乱数，Quantization，ランプ，ステップ，サンプラー，＊は，標準ライブラリーには含まれない．
なお，飽和以下の要素は，すべて信号函数としても使用される．

をつける．次に，ブロック線図中の各函数の演算機能と入出力を，**Fig. 2** に示された形式でインプットシートに記入する．

たとえば，積分演算

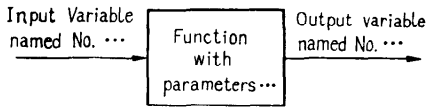
$$\int_0^t X_{15}(u)du + K = X_{20}(t)$$

は，次のような BACS 形式に書かれる．

15　*I*　= 20　*K*

No. of input variable	Function symbol	Parameter or Numerical value	=	No. of output variable	Parameter or Numerical ··· value

Structure statement format



Block diagram representation

Fig. 2 BACS structure statement

標準ライブラリーの他の函数の使用例を **Table 1** に示す.

微分方程式を解く場合，普通積分函数が使われ，一次遅れ函数は用いられない.

時間遅れのない函数の構造ステートメントは，積分および一次遅れ函数の出力から，連続的に配列され，各函数の構造ステートメントは，その函数の入力が定義されてから，配列されなければならない．また，ソーティングルーチンのそう入により，構造ステートメントのランダムな配列は，微分方程式の数値解のための正しい順序にソートされるが，ソーティングルーチンを使用した場合，使用者函数のための構造ステートメントにおけるインプットがふえること，および使用者が implicit 函数の扱いを誤りやすいこと，また，ソーティングルーチンを使用しなくても，その使用時に比較して使用者のインプットプログラム作成のための労力は変わらないこと，の3っの理由から，筆者らはソーティングルーチンを使用していない.

シミュレーションを実行するために，ブロック線図中の定数には，対応する構造ステートメントの中で，使用者は数値を与えるが，パラメーターには個有の変数名を与える．構造プログラムとパラメーターに対する数値データーが読み込まれると，実行が開始され解が得られる．使用者は，入出力線番，演算記号，アウトプット変数の指定以外のすべての変数，たとえば，積分のステップサイズ，時定数，ゲイン，グラフアウトプットのスケーリングなどをパラメーターとすることができる． パラメーターに対する数値は，DATA ステートメントによって与えられる.

5. 積分方法

積分ルーチンとしては，RUNGE-KUTTA-GILL 法と5次の予測修正子法が用意されていて使用者がそれを選択する．さらに，上記以外の積分ルーチンを用いたときには，使用者が作成した積分ルーチンを BACS にそう入する．積分計算のために，積分のステップサイズ，シミュレーションランの長さ，許容誤差を与えなければならない.

6. アウトプット

BACS 標準アウトプット形式には，数字印刷付グラフと数字印刷のみの2種類があり，使用者は両者を同時に使用することができる．アウトプットデータのフオーマットとして，使用者が標準形式を希望するならば，BACS インプットプログラムの構造ステートメントの出力線番の頭に，英数字の1字を記入する．そうすれば，指定された出力の数字印刷付グラフが，自動的に指定された文字によってプロットされる．また，使用者が構造ステートメントの出力線番の頭に＊印を記入すれば，BACS は指定された出力のデーターを標準形式で印刷する．グラフアウトプットのスケーリングは，自動と使用指定で行なわれる.

使用者がアウトプットデータを，特殊フォーマットで印刷させたいならば，使用者は使用者作成のアウトプットルーチンを BACS に入れればよい.

さらに，BACS の使用者函数によって，指定された変数の任意の時点における値が印刷されるが，これはインプットプログラムの構造上の誤りを検出するのに便利である.

先に述べたように，BACS のインプットプログラムの診断ルチンは，インプットプログラムの中の使用者の不注意による記入上の誤りを，インプットカードの読み込みが終了した時点で，診断メッセージの形式で印刷する．診断メッセージが印刷されると，積分計算は中止されるので，使用者はむだな計算を避けることができる.

7. 使用者函数とその応用

使用者は BACS の標準ライブラリーに含まれない函数を使用したいことがしばしばある．このような函数は，普通使用者の研究領域に固有なものであって，非常に種類が多いために，それらの函数をすべて標準ライブラリーに入れて準備しておくことは不可能である．BACS の使用者函数は代数函数であって，構造プログラムの中では，標準ライブラリーの代数函数と同様に扱われる．そして，使用者函数は FORTRAN サ

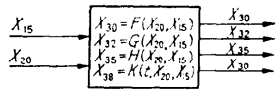
ブプログラム形式であり，そのなかでは FORTRAN ステートメントが自由に使用される．たとえば，インプットアウトプットステートメント，他のサブプログラムに対する CALL ステートメントなどを，使用者は必要に応じて使用することができる．

使用者函数の例としは次のものがある．すなわち，テーブル函数の座標データの読み込み，初期条件の計算，積分計算終了後のアウトプットデーターの処理計算，シミュレーションのランコントロール，implicit 函数や時間を explicit に含んだ函数，および多入力多出力の函数などの複雑な非線型函数の定義，構造プログラムのデバッグのための任意時点の変数値の印刷，最適化のための続行ランのパラメーター値の自動選択，および他のサブプログラムの呼び出しなのである．

使用者函数が用いられるシミュレーションでは，追加されたサブプログラムのみが，FORTRAN コンパイラーとリンケージエディターによって処理され，他のプログラムは，BACSコンパイラーによってプロセスされる．一般に，FORTRAN コンパイル時間は，BACSコンパイル時間に比して，はるかに長いため，インプットプログラムを実行するためのコンパイル時間は，追加された使用者函数が FORTRAN コンパイラーとリンケージエディターによって処理される時間にほぼ等しい．**Fig. 3** は使用者函数の例を示す．

```
SUBROUTINE FUNCn (TIME, X, P1, P2, ···)
DIMENSION X (800)
X(30) = ···
X(32) = ···
X(35) = ···
X(38) = ···
RETURN
END
```

FORTRAN program listing



Block diagram representation

Input	演算 (n)	Parameter	Output	Parameter
			=	P_1 P_2 ···

Structure statement format

Fig. 3 Illustration of BACS user defined function

Fig. 3 のパラメーター P 1, P 2, P 3 などは構造プログラムと使用者函数間の，およびある使用者函数と他の使用者函数間の，また，BACSをサブプログラムとした場合，メインプログラムとサブプログラム BACS 間の，データーの受け渡しのために利用される．

8. ランコントロール

シミュレーションは制御系の解析とシンセシスの手段であるから多くのパラメーター，たとえば，ゲイン，時定数，初期条件などをもった1つのモデルで，多くのパラメーター値に対する一連の計算が行なわれ，その計算のアウトプットが検討される．また，最適なパラメーター値が，プログラムによって自動的に求められることが一層望ましい．したがって，turn-around time を短縮するためにデーターの変更に関して，使用者のあらかじめ指定した順序による繰り返し計算が，1回のインプットで行なわれる必要がある．

BACS では DATA ステートメントと PROCON ステートメントによって，同一モデルによる繰り返し計算が再コンパイルの必要なく実行される．

DATA ステートメントは，独立変数である時間，積分函数の出力，および DATA ステートメントのもとに書かれたパラメーター値のリセットを行なう．

PROCON ステートメントは，時間および積分函数の出力をリセットし，パラメーター値の変更およびラン回数を，使用者函数の中で定義されるプログラムの制御のもとにおく．

次に，例を示そう．

```
DATA CASE 1
KA  0.351
C1  1.05
DATA CASE 2
KA  −6.485
DATAEND
```

第1回のラン "CASE 1" では，パラメーター KA, C 1 の初期値は，それぞれ 0.351 および 1.05 にセットされ，第2回のラン "CASE 2" では，KA の初期値は，−6.485 にセットされるが，C 1 の初期値は，第1回のランの最終値がセットされる，次に，PROCON の例をあげる．

```
DATA
KA  1.25
C 1  0.56
PROCON
```

多くあるパラメーターの中で，KA および C 1 の初期値は，それぞれ 1.25, 0.56 にセットされるが，その他のパラメーター値のセットおよびラン回数の制御は，使用者函数にゆだねられる．

シミュレーションの終了は，DATAEND，または

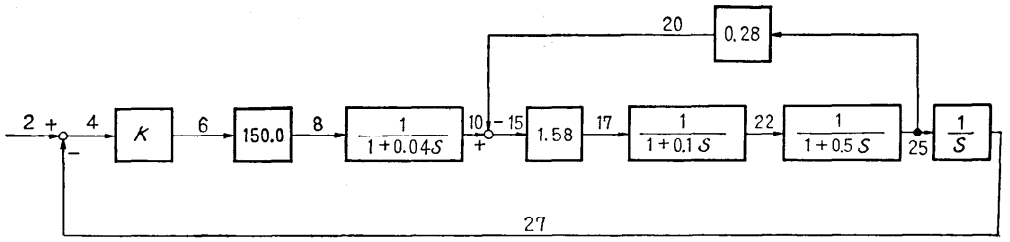


Fig. 4 Block diagram of Ward-Lonard drive system

```
*  EXAMPLE  WARD-LEONARD SYSTEM STUDY
*  SPECIFICATION OF METHOD, STEP SIZE , PRINT-PLOT INTERVAL
    RUNGE     0.01    5.0   10.0     AUTOSCALE
*  SPECIFICATION OF STRUCTURE PROGRAM
     STP           =      2     0.0      1.0
   2  -        27 =      4
   4  *         K = E    6
   6  *    150.0  =      8
   8  F    0.04   =     10     0.0
  25  *    0.28   =     20
  10  -        20 =     15
  15  *    1.58   =     17
  17  F    0.1    = A   22     0.0
  22  F    0.5    = V   25     0.0
  25  I           = T   27     0.0
END
DATA     CASE 1
K         0.015
DATA     CASE 2
K         0.02
DATAEND
```

Fig. 5 BACS user program listing of Ward-Leonard drive system

使用者函数の中で時間が負にセットされることにより行なわれる.

9. 例 1. ワードレオナード制御

簡単な例として, ワードレオナード方式による電動機の位置制御系をあげる. **Fig. 4** は本例のブロック線図を示す. この例におけるパラメーターは K である. 本例の BACS インプットプログラムを **Fig. 5** に示す. プログラムはフリーフォーマットで書いてよい. まず, 表題を記入し, 積分方法, ステップサイズ, 計算終了時間, 印刷間隔などの指定をする. 次に, 構造プログラムを記入し, DATA ステートメントによって, パラメーターに数値を与える. 使用する函数, アウトプットルーチン, ランコントロールは標準である. **Fig. 6** は計算結果を示す. 計算所要時間は IBM S/360-M75 で 2 秒程度である.

10. 例 2. パイロットの非常時における脱出

使用者函数を用いた例として, 航空機から脱出したパイロットの弾道軌道を考える. この問題は MIDAS, MIMIC, CSMP の説明の中で引用されているが[3], システムのモデル化には, シミュレーション言語の種々の機能が要求され, シミュレーション言語の比較のための好例であるので, 本論文でも考察する.

シミュレーションの目的は, 非常時に航空機から脱出したパイロットの弾道軌道を求め, パイロットが航空機の垂直尾翼と衝突するかしないかを確認することである. パイロットは航空機内では, レール上を定速 V_E, 後方に角度 θ_E で運動し, 座席が一度レールを離れると, その後は弾道軌道を飛行する. 航空機は定速 V_A で水平方向に飛行するものとする. **Fig. 7** はパイロットの脱出機構を示す.

パイロットの航空機に対する相対運動の方程式は,

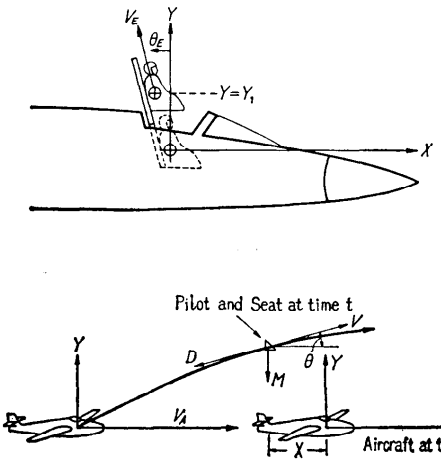
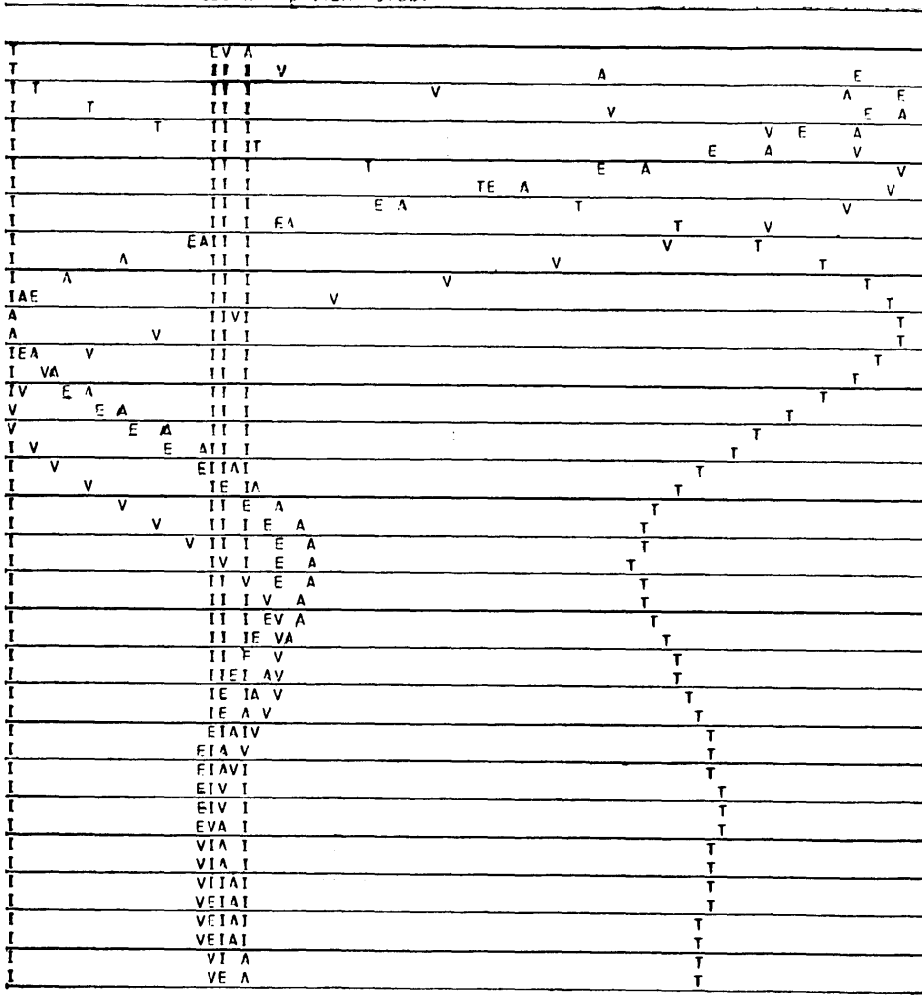


Fig. 7 Pilot ejection system

EXAMPLE WARD-LEONARD SYSTEM STUDY



T	E	A	V	T
0.0	0.0	0.0	0.0	0.0
0.100	2.06208	1.55548	0.12777	0.00377
0.200	2.18944	2.64759	0.50509	0.03424
0.300	2.07859	2.89576	0.92581	0.10609
0.400	1.86456	2.70665	1.26967	0.21676
0.500	1.57584	2.30059	1.49434	0.35601
0.600	1.23956	1.79059	1.59369	0.51143
0.700	0.88340	1.24460	1.57838	0.67092
0.800	0.53254	0.70957	1.46730	0.82391
0.900	0.20794	0.21958	1.28315	0.96194
1.000	-0.07452	-0.20139	1.04974	1.07888
1.100	-0.30425	-0.53830	0.79005	1.17098
1.200	-0.47581	-0.78416	0.52484	1.23668
1.300	-0.58827	-0.93894	0.27172	1.27632
1.400	-0.64463	-1.00837	0.04454	1.29185
1.500	-0.65088	-1.00259	-0.14682	1.28640
1.600	-0.61514	-0.93477	-0.29642	1.26385
1.700	-0.54679	-0.81970	-0.40195	1.22855
1.800	-0.45566	-0.67252	-0.46427	1.18487
1.900	-0.35136	-0.50824	-0.48680	1.13698
2.000	-0.24268	-0.33983	-0.47488	1.08862
2.100	-0.13720	-0.17865	-0.43512	1.04289
2.200	-0.04101	-0.03362	-0.37475	1.00224
2.300	0.04140	0.08888	-0.30106	0.96836
2.400	0.10720	0.18495	-0.22096	0.94224
2.500	0.15507	0.25306	-0.14058	0.92418
2.600	0.18506	0.29368	-0.06502	0.91396
2.700	0.19832	0.30894	0.00179	0.91088
2.800	0.19686	0.30218	0.05714	0.91393
2.900	0.18327	0.27757	0.09949	0.92187
3.000	0.16047	0.23969	0.12838	0.93337
3.100	0.13146	0.19319	0.14430	0.94711
3.200	0.09911	0.14244	0.14844	0.96184
3.300	0.06603	0.09136	0.14255	0.97646
3.400	0.03442	0.04322	0.12869	0.99008
3.500	0.00602	0.00054	0.10910	1.00200
3.600	-0.01791	-0.03490	0.08598	1.01177
3.700	-0.03664	-0.06211	0.06139	1.01913
3.800	-0.04987	-0.08077	0.03713	1.02403
3.900	-0.05771	-0.09118	0.01466	1.02659
3.999	-0.06059	-0.09407	-0.00490	1.02703
4.099	-0.05916	-0.09056	-0.02082	1.02570
4.199	-0.05424	-0.08194	-0.03272	1.02298
4.299	-0.04675	-0.06962	-0.04053	1.01927
4.399	-0.03758	-0.05501	-0.04446	1.01498
4.499	-0.02760	-0.03942	-0.04492	1.01047
4.599	-0.01757	-0.02399	-0.04247	1.00606
4.699	-0.00813	-0.00967	-0.03775	1.00202
4.799	0.00021	0.00284	-0.03146	0.99854
4.899	0.00712	0.01304	-0.02425	0.99575

Fig. 6. Time response of Ward Leonard system by BACS output routine

次のとおりである.

$\dot{X}=V\cos(\theta)-V_A$.

$\dot{Y}=V\sin(\theta)$.

$\dot{V}=0$, $(Y\leqq Y1)$.

$\dot{\theta}=0$, $(Y\leqq Y1)$.

$\dot{V}=-D/M-G\sin(\theta)$, $(Y>Y1)$.

$\dot{\theta}=-G\cos(\theta)/V$, $(Y>Y1)$.

$D=0.5\rho(H)C_DSV^2$.

ただし

$M=7$ slugs.

$G=32.2$ ft/sec^2.

$C_D=1$.

$S=10$ ft^2.

初期条件は

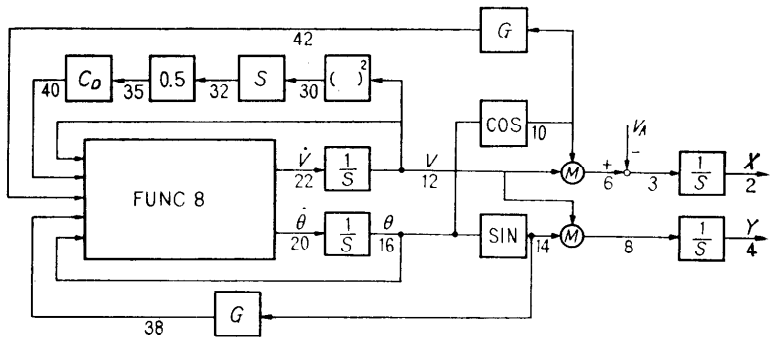


Fig. 8 Block diagram of pilot ejection system

```
*  EXAMPLE  PILOT  EJECTION  SYSTEM  STUDY
*  SPECIFICATION OF METHOD, STEP SIZE , FINAL TIME  ETC.
    RUNGE      DT     FMT
*  SPECIFICATION OF  STRUCTURE  PROGRAM
     (5)            =          VO       THO      VA       VE       THE
  16 COS            =      10
  10  *         12 =       6
   6  -         VA =       3
   3  I             =       2    0.0
  16 SIN            =      14
  14  *         12 =       8
   8  I             =       4    0.0
  12  *         12 =      30
  30  *          S =      32
  32  *       0.5   =      35
  35  *         CD =      40
  14  *          G =      38
     (8)            =          Y1       M        H        RI       HTR
  22  I             =      12   VO
  20  I             =      16   THO
    (10)            =           H       VA       FTM      HTR
END
DATA
DT          0.2
FTM         3.0
M           7.0
G          32.0
Y1          4.0
VE         40.0
THE        15.0
S          10.0
CD          1.0
H           0.0
VA        500.0
RI          1.0
HTR         1.0
PROCON
```

Fig. 9 BACS user strcuture program listing for pilot ejection system.

$V(0)=[[V_A-V_E\sin(\theta_E)]^2+[V_E\cos(\theta_E)]^2]^{1/2}$.

$\theta(0)=\arctan[V_E\cos(\theta_E)]/[V_A-V_E\sin(\theta_E)]$.

$V_E=40$ ft/sec.

$\theta_E=15$ degrees.

ここに，X, Y は座席の固定されてあった航空機内の位置を原点とする運動中のパイロットの相対位置を，V はパイロットの対地速度を，θ は V の水平方向に対する仰角を，M はパイロットの質量を，D はパイロットの受ける空気抵抗を，$\rho(H)$ は高度 H における空気密度を，S はパイロットの基準断面積を，G は重力加速度を，C_D は抵抗係数をそれぞれ示す．

垂直尾翼は航空機の座席の後方 30 ft のところにあり，12 ft の高さである．パイロットが垂直尾翼の上方 8 ft 以上の高さを通過すれば，すなわち，$X=-30$ のとき $Y\geqq 20$ ならば，脱出は安全であるとする．問題は，パイロットの弾道軌道それ自体ではなく，脱出の安全性である．したがって，脱出を安全にする速度 V_A および高度 H の範囲を $H\geqq f(V_A)$ なる形で表現することを考える．

速度 V_A の範囲は，500 ft/sec から，50 ft/sec ごと，1,000 ft/sec まで，高度 H の範囲は，海抜 0 ft より，1,000 ft ごと，60,000 ft までとする．ある速度および高度での計算結果が，安全な脱出とならなければ，

```
*  DEFINITION  OF FUNC5 FOR INITIAL VALUES
       SUBROUTINE FUNC5(TIME,X,V0,TH0,VA,VE,THE)
       DIMENSION  X(800)
       IF(TIME.GT.0.0)  GO TO 10
       V0 =SQRT((VA-VE*SIN(THE*0.017453))**2
      *  + (VE*COS(THE*0.017453))**2)
       TH0=ATAN((VE*COS(THE*0.017453))/
      *     (VA-VE*SIN(THE*0.017453)))
 10    RETURN
       END

*  DEFINITION  OF FUNC8  FOR VDOT , THEDOT
       SUBROUTINE  FUNC8(TIME,X,Y1,M,H,RI,HISTRY)
       DIMENSION  X(800),DATA(2,100)
       IF(RI.GT.1.0) GO TO 30
       READ(5,10)((DATA(1,I),DATA(2,I),I=1,15)
  10   FORMAT(6E12.4)
       RI=2.0
  30   IF(HISTRY.GT.1.0) GO TO 50
       IF(X(4).GT.Y1)    GO TO 50
       X(22)=0.0
       X(20)=0.0
       RETURN
  50   X(22)=-X(40)*CURVE(H,DATA)/M-X(38)
       X(20)=-X(42)/X(12)
       HISTRY=2.0
       RETURN
       END

*  DEFINITION OF FUNC10 FOR RUN CONTROL  AND
*  RESSETING  PARAMETERS
       SUBROUTINE FUNC10(TIME,X,H,VA,FTM,HISTRY)
       DIMENSION X(800)
       IF(X(4).GT.20.0) GOTO30
       IF(X(2).GT.-30.0)GOTO20
       H=H+1000.0
       IF(H.GE.60000.0) GOTO80
  10   TIME=FTM
       HISTRY=1.0
  20   RETURN
  30   CLEAR=X(4)-12.0
       WRITE(6,35) VA,H,CLEAR
  35   FORMAT(' VA= ',F10.3,' H= ',F10.3,' CLEAR BY ',F10.3)
       IF(VA.GE.1000.0) GO TO 80
       VA=VA+50.0
       GO TO  10
  80   TIME=-1.0
       RETURN
       END
```

Fig. 10　BACS user-coded functions program listing for pilot ejection system

安全となるまで 1,000 ft ずつあげていく．安全な高度が求まると，次に速度を 50 ft/sec あげて，同様の計算により安全限界高度を求める．パラメーター値の範囲が広いため，例1とは異なり，パラメーターのリセット，ランの順序，およびラン回数を自動的なプログラムコントロールのもとで行なう．**Fig. 8** は本例のブロック線図を，**Fig. 9** は構造プログラムを，**Fig. 10** は本例の中で使用される使用者函数のプログラムをそれぞれ示す．このインプットプログラムでは，標準アウトプットルーチンを使用しない．使用者函数 FUNC 5 では，V および θ の初期値 $V0$ および $TH0$ が，積分計算を開始する前に，各ランにつき一度だけ計算される．FUNC 8 では，パイロットの現在位置，すなわち，レール上か，弾道軌道上かに従って，$\dot{V}$ および $\dot{\theta}$ が計算される．変数 HISTRY は，パイロットが1度レールから離れたならば $Y<Y1$ となっても，パイロットは弾道軌道を飛行することを意味するためのものである．FUNC 8 が含む方程式の数は任意である．FUNC 10 はパイロットの描く軌道が安全かどうかを確認し，安全な条件を求めるためにパラメーターを新しい数値にリセットし，さらに，ランの終了を行なうルーチンである．

11. 結 言

ディジタル計算機が著しく向上している現在，制御系のディジタルシミュレーションは，制御系の解析とシンセシスにとって有力な手段であり，シミュレーション言語は，ますます重要となってきた．本論文で述べた BACS の特徴は，使用法の簡単さ，プログラミングの融通性およびコンパイラーを含めたオペレーティングシステムの効率の良さである．したがって，BACS は制御系のディジタルシミュレーションに適しており，十分実用的であると思われる．

筆者らの使用した計算機は，タイムシェアリング機構をもっていなかったために，オンラインシミュレーションは実現されなかったが，BACS に二，三のタイムシェアリングコマンドを追加すれば，BACS はオンライン用シミュレーションプログラムとなる．

制御系のディジタルシミュレーションの今後の課題は，各種制御系用の専用シミュレーション言語，およびタイムシェアリングシステムによるオンラインシミュレーションであろう．

オンラインシミュレーションでは，遠隔入出力端末，特に，ディスプレイ装置によるランコントロール，およびインプットされたブロック線図をダイレクトアクセス装置に記憶させ，ブロック線図の分割・組立てが可能なオペレーティングシステムが必要である．

特殊目的シミュレーション言語では，制御系個有の物理的対象と一般制御理論を調和させたインプット形式の開発が望ましい．

アナログシミュレーションに比較して，ディジタルシミュレーションのもつ欠点は，turn around time とコストであるが，上記の問題の発展がこの欠点を解決するであろう．

BACS は，その一般性と広い応用によって，科学技術計算用プログラムの中では，最も利用度の高いものの一つであり，将来のタイムシェアリング情報サービスを考えれば，公共性の強いものといえよう．

終わりにあたり，お世話になった三菱電機機械計算部吉江高明氏，海老名史道氏，桜井美和子氏，同社中央研究所林 重雄氏，ならびに同社の関係各位に謝意を表わす．

参考文献

1) 松浦卓丈，外山守城：ディジタル計算機によるブロックダイヤグラムシミュレーション，アナログ技術研究会資料，第4巻，第6号，1964.
2) R. Linebarger and R. D. Brennan: Digital Simulation for Control System Design, Instruments & Control Systems, Oct. 1965.
3) R. D. Brennan and M. Y. Silberberg: The System/360 Continuous System Modeling Program, Simulation, Dec. 1968.

(昭和44年3月14日受付)