48－S 002

コンバインドシミュレータの研究開発

昭和49年3月

JIPDEC

財団法人　日本情報処理開発センター

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助金を受けて昭和48年度に実施した「コンバインドシミュレータの研究開発」の成果をとりまとめたものであります。

序

当財団は情報処理に関する研究開発の一環として，各種のソフトウエアの開発をおこなっておりますが，本書は「オンラインコンバインドシミュレーション言語」の開発成果を報告するものであります。

一般に利用されているシミュレーション言語には，連続系シミュレーション用と離散系シミュレーション用の２つのタイプがあります。しかし，シミュレーションの対象が広がるにつれて，一方のタイプの言語だけでは扱いきれない場合も多くなってまいりました。この為，両者の機能を兼備したコンバインドシミュレーションシステムの開発が望まれております。

当財団では，我が国で最初のオンライン利用によるコンバインドシミュレーションシステムの開発を進めております。本年度はその初年度にあたり，システム開発の為の基礎調査とシステムの基本設計を実施いたしました。

本報告書が，この方面に興味を持つ方々に広く利用され，情報処理技術の発展の一助として寄与できることを念願いたす次第であります。

昭和49年3月

財団法人　日本情報処理開発センター
会長　　難　　波　　捷　　吾

コンバインドシミュレータの研究，開発

目　　次

ま　え　が　き

現在までに数多くの汎用シミュレーション言語が開発されている。

これらは連続系シミュレーション用と離散系シミュレーション用の２つに大別する事ができ，それぞれの系を専用に扱う目的で作られている。コンピュータシミュレーションが広く普及し，対象とする領域も広がり，内容も複雑になってくるにつれ，連続系と離散系の両方の要素を含んだ問題も多くなって来た。この様なシミュレーションに対しては，従来，一方の言語で扱える様にシステムを簡略化したり省略したりしていたが，満足の行かない場合も多かった。そこで連続系と離散系が同時に扱える，シミュレーション言語の開発が望まれるようになった。

我々が開発を意図しているコンバインドシミュレータは，この様な領域を対象とした汎用シミュレーション言語であり，離散系のシミュレーション・システムＳＩＭＢＯＬと同様，インタラクティブにシミュレーションが行なえる点で特徴がある。

なお，このプロジェクトは２年計画で開発することになっており，今年度は調査および基本設計を中心におこない，具体的なシステム開発は次年度の予定である。

従って本報告書は調査結果を主体にまとめてあり，次の様な内容からなっている。

第１章ではコンバインドシミュレーションについて概説し，第２章では連続系シミュレーション言語と離散系シミュレーション言語の歴史について述べ，第３章でコンバインドシミュレータＳＩＭＢＯＬ－Ⅱと，この開発に先だって，予備調査を行なう目的で作成した実験用の会話型コンバインドシミュレータＳＰＡＣＥについて述べている。又，第４章にはＳＰＡＣＥを用いて行なった，簡単な交通流シミュレーションモデルについて述べ，第５章および第６章には連続系シミュレーション言語の調査として，国内のデータセンターを利用しＣＳＭＰ－Ⅲ，ＣＳＳＬ－Ⅲ，ＭＩＭＩＣの比較を行なったので，この結果をとりまとめた。

1．コンバインドシミュレータの概要

1. コンバインド・シミュレータの概要

1.1 はじめに

コンピュータによるシステム・シミュレーションはその対象となるシステムの時間経過に伴う状態変化として捉えたモデルによる数値実験である。

現在，コンピュータ・シミュレーションの多くは汎用のシミュレーション言語によって行なわれるが，汎用シミュレーション言語は，連続系シミュレーション言語，離散系シミュレーション言語の2つに分けることができる。連続系シミュレーション言語によるシステム・シミュレーションは，状態の変化が連続的であると見なせる現象を解析するのに適しており，力学系や自動制御系の運動方程式，微分方程式で表現されるモデルによって行なわれる。この様な連続系シミュレーションに用いられる代表的な言語としてはMIMIC，CSSL-3，CSMP-3などがある。これに対し，離散系のシミュレーション言語によるシステム・シミュレーションは，本来連続的なシステムのダイナミックスを，離散的に生起する現象として取り扱い，システムを全体的に把握できる様なモデルを用いて行なわれる。従って，離散系シミュレーションのモデルは連続的な変化を省略したり，或は離散的な変化に置き換えたりして構成される。離散系シミュレーションはジョブ・ショップ問題，在庫管理システム，エレベータの制御システム等のシミュレーションに応用されており，その代表的な言語としてはGPSS，SIMSCRIPT，SIMULAなどがある。

汎用のシミュレーション言語によってシミュレーションを行なう際，そのシミュレータのワールド・ビューに沿ってモデルを構成しなければならない。つまり連続系のシミュレーションではシステムの変化を連続的に，又離散系のシミュレーションではシステムの変化を離散的に捉えなければならないのであるが，汎用のシミュレーション言語によるシミュレーションでは対象となるシステムの全ての要素が組み込まれる必要はなく，シミュレーションの目的に見合った範囲で，システムの要素が組み込まれていれば，充分であり，しかも一般的にはなるべく余分な部分を省略する方向でモデルを構成する。

しかしながら，コンピュータの規模が大きくなり，高速化されるにつれ，シミュレーションの対象とする範囲も広がり，連続的なモデル或は離散的なモデルでは取り扱いにくい要素，つまり連続的な要素と離散的な要素とがからみ合った様な要素を含むシステムのシミュレーションに対する要求が起ってきた。この要求に対し1968年頃から連続・離散両系をカバーし，しかもこの両系をインターアクトさせるタイプのシミュレーション言語 ── コンバインド・タイプ・シミュレーション言語（Combined type simulation language）── が開発され始めた。このコンバインドタイプのシミュレーション言語ははとんど既存の離散系言語と連続系言語とを，結合した形で作られ

ている。しかしながらコンピュータシミュレーション，特に離散系のシミュレーションに於いては，シミュレーション・ランのＣＰＵタイム，占有するメモリースペースが問題とされるのが常であり，どこまで大きなシミュレーション・モデルが組めるかが，シミュレーションの使用価値を決めることがあり，既存の離散系言語に連続系言語のあらゆる機能をつけ加えることはいたずらにシミュレータの容量を小さくしてしまうことになる。従ってコンバインド・タイプのシミュレーション言語は連続・離散両系の全ての機能を持ち合わせたものではなく，またこれらの両シミュレーション言語に取って変わるべき性格のものでもない。コンバインド・タイプのシミュレーション言語は，むしろ連続系言語或は離散系言語単独ではカバーし切れなかった隙間をうめるべき性質のものというべきであろう。

1.2　コンバインド・タイプ・シミュレーション言語に要求される機能

コンバインド・タイプ・シミュレーション言語は連続・離散両系のシミュレーション言語単独では取り扱えないタイプのシステム・シミュレーションに対し，自然に起った要求であることは，先に述べた通りである。従って，コンバインド・タイプ・シミュレーション言語は，離散・連続両系シミュレーション言語単独の機能，即ち，離散系言語の様にテンポラリーな要素 ── シミュレーション進行中にシステムに一時的に存在する様な要素，例えば，GPSSではトランザクション，SIMSCRIPTではテンポラリー・エンティティ，SIMULAではプロセス──を取り扱うことができ，システムの各要素に対する統計処理ができること，又，連続系シミュレーション言語の様に微分方程式で表現されているモデルをシミュレートできること，の外に，次の様な機能がコンバインド・タイプ・シミュレーション言語として要求される。

(1)　連続的な要素と離散的な要素とが同一のモデルに組み込めしかもこれらの要素をお互いに関連付けができること。

(2)　両要素が時間に従ってパラレルに動作できること。

具体的な例として化学工場におけるバッチ反応器の例を挙げる。各バッチ反応器における化学反応プロセスは連続的にシミュレートする必要があり，微分方程式の形でモデル化される。一方バッチの到着する様子は離散系のモデルによって構成しなければならない。しかも，バッチの到着によりバッチ反応器のスイッチをオンにしたり，プロセスの終了によってオフにしたりしなければならない。つまり，コンバインド・タイプのシミュレータでは同機の取られた２つのシミュレーション・クロックがパラレルに進行し連続系から離散系の事象を引き起こしたり，或は，離散系の事象によって連続系のモデル（プロセス）をスタートさせたり，ストップさせたりすることができる様になっていなければならない。

2．シミュレーション言語の歴史

2. シミュレーション言語の歴史

第1章では，コンバインド・タイプのシミュレーション言語の概要を示し，これが離散系シミュレーション言語と連続系シミュレーション言語の中間を行くものであることを述べた。ここでは連続系シミュレーション言語及び離散系シミュレーション言語の発展を歴史的に捉え，各々がどの様な機能をもつ様になってきたかを探り，コンバインド・シミュレータに備えつけられるべき機能を探り出す手がかりとしたい。

2.1 アナログ・コンピュータ，ハイブリッド・コンピュータによるシミュレーションの歴史

2.1.1 アナログ・シミュレーション

アナログ・コンピュータはブロック・ダイヤグラムで表現されたシステムを，積分器，加算器，掛算器等の演算子を用いて，システムとのanalogue（電圧等の連続的な物理量）を作り出し，システムを解析してゆくものである。

1950年代Digital Computerがまだ真空管式で精度・信頼性において，充分でなかった頃アナログ・コンピュータはコンピュータ・シミュレーションの主流であった。アナログ・シミュレーションはパッチボード上に各演算子をパッチ・コードで接続し，ポテンシャル・メータ等をsetすることにより行なわれる。

アナログ・シミュレーションの長所

(1) パラレル演算を行なえるため計算時間が非常に短い。

(2) 入出力信号が連続的であるため物理現象の相対関係を捉えやすい。

このことは，アナログ・コンピュータがマン・マシーン的性格をもつことを意味する。

(3) プログラムや操作が簡単である。

(4) 部品点数が比較的少ないので，信頼性が高い。

アナログ・シミュレーションの短所

(1) フルスケールに対し，精度が0.1～0.01％と低い。

(2) スケーリングの必要があり，データ処理性に乏しい。

2.1.2 ハイブリッド・シミュレーション

アナログ・コンピュータの欠点をカバーし，アナログ・コンピュータとディジタル・コンピュー

タとをinteractさせることにより，メモリー機能をつけ加え，ロジック処理を可能にしたのがHybrid Computerである。ハイブリッド・シミュレーションはアナログ・シミュレーションと同様，連立の非線型微分方程式であらわされる連続系モデルやフィードバックループをもつ制御系のシミュレーション・プログラムに威力を発揮する。

1964年，Wisconsin大学，Colorado大学において，Continuous System Simulator，COBLOC，HYBLOC，MADBLOCが次々に開発され，一時期Continuous System Simulation Languageの中核を成した。

現在では，Hybrid SimulatorはMIMICなどのContinuous System Simulation Languageでかかれたプログラムをディジィタル・インプットすれば，パッチ・ボードがスイッチングでなされ，エキスキューションを行ない，アナログ・アウトプットが得られるものやT.S.Sによるエキスキューション可能なものが登場するまでに発展してきている。又chip-size L.S.I circuitなど回路素子の開発が進めば，アナログ・コンピュータのパラレル・コンピューテーションという特徴を生かした，ハイブリッド・コンピュータが現在のContinuous System Simulation LanguageによるDigital Simulationと取って変わり，Simulationの中心的手法となる可能性も大きい。

2.2 連続系シミュレーション言語

Analog Computerで大規模なシステムのシミュレーションを行なう際salingの煩雑さやpatchingを正確に行ない得たか否かは重要な問題であり，何らかの方法を用いてAnalog Solutionをcheckする必要がある。

1960年代前半，Digital Computerは真空管式からsolid state式へと移行し，その信頼性も飛躍的に向上した。又プログラミング・ランゲージもFORTRAN-IIからFORTRAN-IVへと脱皮しプログラミングも容易になった。ディジィタル・コンピュータのこのような発展を背景にして，Continuous System Simulation Language（以下C.S.S.Lと略す）の開発がさかんに行なわれる様になってきた。

C.S.S.Lの歴史を全体的に見渡すと，MIDAS以前とMIMIC以降とでは，ある種の区分けができると思われる。その理由はMIDAS以前のC.S.S.Lでは，アナログ・コンピュータの機能をディジィタル・コンピュータによって模倣しようとしていたのに対し，MIMIC以降では，Digital Computerの機能を積極的に生かそうとしていることに依る。具体的にはMIDAS以前のC.S.S.Lではブロック・ダイヤグラムがプログラムの中核であったのに対しMIMIC以降のC.S.S.Lでは，微分方程式をdirectにプログラムできる様になった。

Continuous System Simulation Languageを二つに分類し ―― 即ち，MIDAS以前のC.

S.S.LをBlock Oriented C.S.S.L(Digital Analog Simulator), MIMIC以降のC.S.S.LをEquation Oriented C.S.S.L ―以下の二節で，それぞれの歴史的流れ，及び言語の特徴を紹介する。

2.2.1 アナログ・コンピュータ・タイプ連続系シミュレーション言語

アナログ・コンピュータ・タイプ連続系シミュレーション言語はアナログ・コンピュータ機能をディジィタル・コンピュータによって模倣しようとしたもので，アナログ素子一つ一つに，ブロック・ファンクションを対応させている。

1. Selfridge

Selfridgeのプログラムは第一世代のコンピュータIBM701のもとで1955年に開発された。当時は今日の様にプログラミング・エイドであるコンパイラー(FORTRAN, ALGOL等)はおろか，シンプルなアセンブラーさえも備えつけられてもいなかったため，セルフリッジのプログラムは実現され得なかった。

彼は，Analog素子一つ一つを，Digital Computerで一つのblockとを対応させ，そのブロックのタイプ，入力要素を決め，それらを結合することによってブロック・ダイヤグラムで記述された系をシミュレートできるという様な提案をした。セルフリッジの提案は，彼のプログラムが実現されなかったにしても，すばらしいものであり，以後開発される全てのC.S.S.Lの原点となるものである。

2. DEPI

1957年，カリフォルニア工科大学，ジェット・エンジン研究所のF.Leshはセルフリッジの，研究を受け継ぎ，DEPIを開発した。DEPIはDATATRON.204のもとで，インタープリターとして書かれたものであるが，完成するには到らなかった。後にHurleyによりIBM704にコンバートされ，DEPI-4へと発展して行く。

3. ASTRAL

ASTRALは，1958年Convair AstronauticsのMartin L.Stein, Jack Rose, D.B. Parkerらにより，IBM704のもとに開発された。

ASTRALは，プログラムの構造上から分類すればFORTRANのプリ・コンパイラであり，アナログ・コンピュータ・PACEに良く似た要素により成り立っている。従って，アナログ・ユーザーが最小限の修正でアナログ・コンピュータ用のプログラムをコンパートすることができ，アナログ・シミュレーションの解のチェックにも役立つ。又ディジィタル・コンピュータであるからパラメータの変更がアナログ・コンピュータに比べ容易に行なえるので，アナログ・コンピュータで得られた"興味深い領域"で解の精度を高めたり，パラメータの影響度などをシミュレーション・スタディすることができる。

ASTRALの最大の特徴は，sorting機能を持っているということである。ユーザーのプログラムはブロック・ダイヤグラムをコード化したものであるが，このブロック・ダイヤグラムを見ると1時点における状態変数間の関係がはっきり分り，しかもブロックへの入力，出力をたどるとどの変数が先に計算されていなければならないか，という順序が分かる。sorting機能はこの様な順序関係を見つけユーザーのプログラムのステートメントを自動的に並べ変えるものであり，ユーザーはProceduralにステートメントを書く必要がなく，あたかもAnalog Computerで，Patchingを行なっているかのようにプログラムを組むことができる。

4. DYSAC

1961年，Wisconsin大学のJ. J. Skiles, V. C. Rideout両教授のもとにJ. R. Husleyにより開発された。DYSACは基本的にはCDC 1604のDEPI-4と同じであるが，アルファニュメリック・シンボルや演算子を入力として受け入れられる点で異なっている。又積分ルーチンをDEPI-4と同じ4 th-order Runge Kutta Gillであるが，指・対数関数variable time delay機能がつけ加えられ，後にtan, arctan, arcsin, arccos, 等もつけ加えられ，空気動力学のシミュレーションも可能になった。

DYSACはMagnetic Tapeを用いて，オフライン・プロッティングやCALCOMP 570でのプロッティングを可能にした最初のプログラムである。

5. PARTNER

1962年，Stover, KnundstonによりIBM 650用に開発された。PARTNERはINT, ADD, MULT, DIV等のニューモニックな命令で表わされるブロックにより成り立っている。PARTNERは27個の別々のファンクション・ブロックを持ち，各ブロックは機械語のサブルーチンに対応している。つまり，PARTNERはサブルーチン群から成り立っている言語であると言える。

6. DAS

C.S.S.Lの中で，その言語構造とインプリメンテーションの両面から見て，最も興味深いプログラムの一つとしてDASが挙げられるであろう。DASはGaskill, Harris, Mcknightらにより，IBM 7090のもとに1963年に開発された。

DASはプログラム・ストラクチャーから分類するとコンパイラーに属するが，DYSACが中間言語としてFORTRANをつくり出したのに対し，DASは各ブロック・ステートメントをFAPオープン・マクロ・ルーチンに等価変換を行なう。又DASはASTRAL, DYSACと異なり，ブロックの出力にニューモニックな変数をつけることができ，プログラムを全てFAPのマクロアーギュメントリストに登録させているのでデコーディングタイムを大いに減少させることに成功している。一連のDASステートメントはひとたびFAPマクロ・ルーチンに変換されれば後はDASプログラムがIBM 7090 FAPルーチンを呼び出すだけで，プログラムのアセンブル，エキ

スキューションが行なわれるのである。

しかし，DASはソーティング機能を備えつけていないので，積分のブロックはステートメントリストの最後に置かなければならず，プログラムをプロスィデュラルに組み立てなければならないのは不便な点である。又，積分ルーチンにはRectanguler Integration Method が用いられ，積分誤差が比較的大きい。

ここでC.S.S.Lの心臓部である積分ルーチンについてふれておきたい。積分法には簡単で演算時間の比較的短いものとして，シンプソン則，オイラー法，Rectanguler法他と，比較的精度の高いものとして，Runge-Kutta Gill法，Milne法，Molton法などがある。C.S.S.Lを単にAnalog Computerの解のcheckとして用いるなら，前者の様なもので充分であるし，C.S.S.Lにより正確な解を必要とするなら，後者の様に精度の高いものが必要であろう。従って，理想的なC.S.S.Lの積分ルーチンは，いくつかの積分法を用意しておいて，ユーザーにその使用目的により選択させるという形式を取るべきであろうと思う。

7．JANIS

JANISは1963年ベル・テレフォン研究所のE.R.Byrneにより開発され，IBM 7090 FAPシンボリックコードで書かれている。JANIS programでは一つ一つのblockが，サブルーチンで書かれており，ユーザーはFORTRANの"call"文を用いてモデルを定義する。従ってユーザーが自分で定義したサブルーチンをblockとして使うこともできるし，call文とcall文との間にlogic condition，functional relationship，parameter変更など，多くのalgebraic手法をプログラム中にもり込めるという利点をもっている。JANISのシステム・プログラムは各時刻において，ユーザーの用意したモデル，即ちサブルーチン群を計算し，必要なvariable outputやplottingを行ない，simulation clockを進めてゆくのである。

8．MIDAS

MIDASはDASの思想を受け継いだものであるが，次の様な点でDASと異なっている。

(1) 積分ルーチンにvariable-step，5th-order，predicter-correctorを用いている。

(2) DASがcompilerであったのに対し，MIDAS processorはinterpreterである。

(3) ASTRAL以後無視されていたsorting機能の追加。

(4) C.R.T. plottingが可能になった。

(5) Implicit Functionが記述できる。

MIDASの紹介から多少はずれるが，C.S.S.L processorのことについてふれておきたい。processorとしては，interpreterとcompiler（translator）があるが，両者とも各々利点をもっている。compilerはプログラムをいったんコンパイルしてexcutionの段階に入ると実行スピードの点で何回のランも行なえるし，パラメーターの変更も容易である。一方interpreterはプログラムが小さい場合やモデルの構造がたびたび変更される場合には非常に有利で

ある。又会話型にして，C.S.S.LにMan-Machine的性質を与えようとする時interpreterであった方が都合が良い。しかし，今日ではコンパイル時間が非常に短かくなってきているという理由で，compilerタイプのMan-Machine言語も現われてきている。processorとしてcompilerが良いかinterpreterが良いかという問題は，ユーザーがその応用分野にどの様なシミュレーション・スタディを行なうかにより判断されるべきものである。

9. PACTOLUS

PACTOLUSは比較的小規模なコンピュータIBM 1620用につくられた科学用digital analog simulatorである。PACTOLUSの特徴はタイプライターを用いて，シミュレーション実行中にパラメーター変更初期条件の変更あるいはモデルの構造を変えることのできるようになっているシミュレータである。このようなon-line的性格をもったプログラムは以前にはOL-DASだけであるが，シミュレーションの実行形式としては理想的なものであり以後CSMP-Ⅲなどに受け継がれている。

PACTOLUSは二次のルンゲ・クッタ法を積分ルーチンに持ち，陰関数を書くことができる。又9つのダミー用特殊elementをもっていて，ユーザーが指定すればblock functionとして用いられ，プログラムにflexibilityを与えている。

後にIBM 7090にconvertされon-line，off-line両用になった。

2.2.2 方程式タイプ連続系シミュレーション言語

Block-Oriented C.S.S.Lでは，システムのBlock Diagramを描き，そのAnalog素子を対応するBlock Functionに置き換えプログラムを作成するという煩雑さがあったが，1965年に発表されたMIMICでは微分方程式をdirectにプログラムすることが可能になりDigital Simulation Languageは大きな前進を遂げた。

MIMICがinterpreterであったのに対し，MIMICの半年後に発表されたDSL／90はpre-compiler，即ち，DSL／90 processorはプログラム中のDSLステートメントをFORTRANステートメントにおとし，それをFORTRAN compilerに渡すという形式，でありユーザーはプログラム中にFORTRANステートメントをかくことができる様になった。このDSL／90のpre-compilerであるという特徴は以後開発される，CSMP／360，CSSL，CSMP-Ⅲ，CSSL-Ⅲなどに受け継がれ，今日の全てのC.S.S.Lの基盤となるのである。

巻末の付録にMIMIC，CSSL-Ⅲ，CSMP-Ⅲの詳細な言語比較をつけ加えてあるので，この節では簡単な言語紹介を行なうにとどめることにする。

1. MIMIC

MIMICはMIDASの作成者Petersonにより開発されたBlock-Orientedな言語であるが，functionのnestingが許されAlgebraic expressionが可能であるという点でEquation-

Orientedな C.S.S.Lと言えよう。

積分ルーチンは，variable-stepの4 th-order Runge Kutta法により行なわれ，プロッターによる出力も可能である。しかし，MIMIC processorはinterpreterであり，FORTRANとのinterfaceが許されず，そのためproceduralなステートメントを書けないという欠点を持っている。後にMIMIC processor内にダミー用のサブルーチンを用意し，FORTRANとのinterfaceを可能にしたが，それとても本質的にはinterpreterであるという点をカバーしきれないに終ってしまった。

2. DSL／90

DSL／90は先に述べたようにpre-compilerであるという特徴をもっているがさらに次の様な点を長所として挙げられる。

(1) 6つの積分ルーチンを持ち任意に選択できる。

(2) ユーザーの定義したサブルーチンをライブラリーにつけ加えられる。

(3) Sorting，Nosorting，PROCEDURAL機能を持っている。

(4) Graffic出力可能。

(5) Macro-Generatorをもっている。

3. CSSL

CSSLはSimulation CouncilのSimulation Software Commiteeにより1967年10月に出版されている。CSSLはシミュレーションの手段としての言語に柔軟性と強力な威力を持たせ，DisplayやT.S.Sなどによりシミュレーション機能を拡張してゆくことを目的として作られた。

4. CSMP／360，CSMP-Ⅲ

CSMP／360とCSMP-Ⅲは入出力関係が多少違っているだけで本質的には同一のものであり，ともにDSL／90，DSL／360の流れを組むものである。

表 1－1　連続系シミュレーション言語の開発経過

NAME	SOURCE OF NAME	DATE	AUTHOR	COMPUTER
Selfridge	-----	1955	Selfridge	IBM 701
DEPI	Diff. Eq. Psuedo Code Interpreter	1957	F.Lesh	Burroughs 204
ASTRAL	Analog Schematic TRanslator to Algebraic Language	1958	Stein,Rose and Parker	IBM 704
DEPI-4	DEPI for IBM 704	1959	Hurley	IBM 704
DYSAC	DigitallY Simulated Analog Computer	1961	J.J.Skiles and J.R.Hurley	CDC 1604
DAS	Digital Analog Simulator	1963	R.F.Gaskill	IBM 7090
OLDAS	On-Line DAS	1963		
JANIS	-----	1963	R.G.Byrne	IBM 7090
PACTOLUS		1964		IBM1620
MIDAS	Modified Integration DAS	1964	Sanson and Peterson	IBM 7090
MIMIC	-----	1965	Sanson and Peterson	IBM 7090
DSL/90	Digital Simulation Language for IBM 7090 class computer	1965	Syn and Wyman	IBM 7090
EASL	Engineering Analysis snd Simulation Language	1965	L.Sashkin and S.Schlesinger	IBM 7090
DSL/360	DSL for IBM/360	1967		IBM 360
CSMP/360	Continuous System Modeling Program	1967		IBM360
CSSL	Continuous System Simulation Language	1967		
CSMP-3	Continuous System Modeling Program	1971		

2.3 離散系シミュレーション言語

Discrete Event Simulation Language（以下DESLと略記する）は，先に述べたCSSLが時間を連続軸として全てのvariableをplotできるものであったのに対し，時間経過とともにシステムの状態のDiscreteな変化を数値的・論理的に取り扱うタイプのシミュレーション用の言語である。このタイプのシミュレーションはフローチャートやLogic Diagramなどにより，シンボリックにフォーミレートされ，DESLにコード化された後，Digital Computerによりimplementされるものである。

2.3.1 離散系シミュレーション言語の歴史的背景

Discrete Digital Simulationの起源は，モンテ・カルロ法の確立にあると言われる。モンテ・カルロ法は，システムを数学的・論理的かつ確立的に表現することにより，物理系の動的なふるまい，特に工業，軍事関係の問題の解法に応用できる実験的手法であり，1950年代初期から実用に給されている。モンテ・カルロ法の計算方法はディジタル・コンピュータ・テクニックの発展により強力なpowerを与えられ，その利用範囲も急速に広がって行った。

Discrete Event Simulationが始められた当初，シミュレーション・プログラムはマシーン・コード或はアセンブリー言語により書かれていたが，FORTRAN等のcompilerが開発され，シミュレーションに携わる人々は，このコーディングの容易な言語を用いる様になった。しかしながら，シミュレーション・アナリストたちはモデルをこの様に汎用言語を用いてコーディングするのはあまり賢明な方法でないのに気づいた。なぜなら，シミュレーションとはモデルと理論をiterativeに開発してゆくものであり，いろいろな状況のもとで何回ものコンピュータ・ランを必要とするものであるからである。従ってシミュレーションには，シミュレーション・モデルを容易にコード化でき，いろいろな状況をつくり出せる言語が必要なのである。

シミュレーション・モデルをつくり上げる労力を少しでも緩和しようとして行なわれた最初のステップは，特殊な問題を取り扱える様なシミュレーション・プログラムの開発であった。即ち，ジョブ・ショップ・シミュレータ，在庫管理シミュレータ，軍事作戦シミュレータなどの開発である。これらのシミュレータの設計意図は，ある特殊な問題分野において，様々な状況に適応できる様なモデルを作ることにより，その分野における多くのシミュレーション・プログラムを取り扱おうとしているところにある。これらのプログラムはその設計者にとっては大いに価値のあるものとなりはしたが，設計者以外の人々には自分のプログラムをあるモデルに当てはめなければならないという不便さがあり，この汎用モデルというアイデアは功を奏しなかった。

このような汎用モデルの開発と平行し，プログラミング・エイドの発展の力を借り，シミュレーション用の言語の開発も盛んに行なわれていた。最初のシミュレーション言語は1959年イギリス

で開発されたMontecode interpretive languageである。1959年から1960年初めにかけて，この様な特殊な言語がイギリスなどで数多く出版された。

2.3.2 第一世代のシミュレーション・プログラミング言語

DESLの歴史は，シミュレーション理論，モデリング哲学の発展と非常に興味深いものを示唆してくれる。ここでは1964年以前に開発された言語について簡単な解説を加える。

第一世代のDESLはそのプログラミングの方法により分類すると，GPSSタイプとSIMSCRIPTタイプとに分けることができる。前者はフローチャート・シンボルを用いてモデルを記述する（Block-oriented）のに対し，後者はステートメント言語（FORTRANなど）を用いてモデルを作る（Equation-oriented）Simulation languageである。

1. GPSSタイプ

GPSSは1964年IBMのGordonらにより開発された汎用シミュレーション言語であり，GPSS，GPSSⅡ，GPSSⅢ，GPSS／360という様な順に改良されて来ている。

GPSSは時間の経過とともにシステムを流れるtransaction（Xact）を注目し，これがシステムにどの様な変化をもたらすかを記述したプログラムを作り，モデルを構成する。GPSSはこの様なWorld ViewはXact-orientedと言われる。GPSSではシステムの構成要素をfacility，storage，logic switch，Xact等により細分化し，具体化された形で記述する。従って後で述べるSIMSCRIPTに比べて実用的，即物的なシミュレータである。

GPSSは基本命令はblockと称されるマクロ命令で，各々block-type subroutineに対応していて，プログラムはこれらのブロック・タイプ・サブルーチンを通してインタープリティーブに実行される。

2. SIMSCRIPTタイプ

SIMSCRIPTタイプのシミュレータはイベント中心の言語と呼ばれる。イベント中心の言語はシミュレーション・モデルをイベントと呼ばれるサブモデルに分割する。イベントとは，それが起った時，システムのステータスがどの様に変化するかを記述してあるプログラムである。

典型的な例として窓口システムについて言えば，モデルは客の到着というイベント，サービス終了というイベント，それに不必要かも知れないが待行列が長すぎたため客が立ち去るというイベントという風なサブ・モデルに分けられる。イベントを記述するというのは，イベントが未来のある時刻に起りそれがシステムのステータスをどの様に変化させるかをScheduleするLogicをプログラムすることである。例えば，窓口システムの例で，窓口が使用可能であり，客が到着したというイベントが起った時，窓口のステータスを使用中にし，未来のある時刻に使用可能になるというイベントをスケジュールする様なプログラムとして到着イベントを作るのである。一般的には未来の時刻はある確率分布から，ランダム・サンプリングすることにより得られる。イ

ベントが未来のある時刻にスケジュールされるとそれは，future events calenderに置かれ，シミュレーション言語のexcutive programにより保持される。そして一つのイベントが完了するとカレンダーからスケジュール・タイムに従って，一つのイベントがシミュレーション実行ルーチンにより取り出され，それに相当するプログラムがcallされ，一つのイベントを引き起こすのである。

SIMSCRIPT（original version）は1963年，RAND Corp.のMarkwitz，Haunsner，Karrらにより開発され，IBM7090，Univac 1107などによりインプリメントされている。又SIMSCRIPTファミリーとしてはSIMTRAN，GASP，MONTCODE，SEAL，SIMP-ACなどが挙げられる。

2.3.3 第二世代のシミュレーション言語

GPSS，SIMSCRIPTが出現した後，この両者の利点を抽出してまとめた形の言語SOLが発表された。SOLはBurroughs社で開発されたものであるが，SIMSCRIPTのモデル作成に見られる柔軟性，出力の多様性と，GPSSの具体的なモデル構成概念を兼ね備えたものであると言える。SIMSCRIPTとGPSSの結合という新しい試みはCDCのSIMULAに受け継がれ，プロセスという概念を生み出した。プロセスとはある時間において特殊な動作パターンをもつ動的エンティティであると定義づけられる。即ちXactが移動するパターンをプロセスと言う。プロセスはイベントとトランザクションの概念を持ち合わせており，この両者と非常に似寄った性質を持っている。例えばプロセスにおけるreaction pointの概念はそのままXactの位置に，アトリビュートは，Xactのパラメータに又イベントはプロセスのなかの一時点に対応しているということによりうかがえるであろう。SIMULAはSOLと同じ様にALGOLをベースとし，ALGOLのもつ演算機能を生かしたprocessorを持つcompiler言語である。

SIMULAのプロセスという概念は，IBMのMSS，SIMPL／I，CPL／Iベース）などに受け継がれ，プロセスタイプの言語の基礎を作った。プロセスタイプの言語はGPSS同様非常に使いやすく，SIMSCRIPTの様な柔軟性を兼ね備えた言語である。

第二世代の言語としては，プロセスタイプの言語以外にGPSSⅢ，GPSS／360或はSIMS-CRIPT 1.5，SIMSCRIPT 2などがある。これらは，従来のGPSS，SIMSCRIPTの機能を拡張したものであり，World Viewなど，本質的な部分の変更は成されてはいない。SIMSC-RIPT 1.5のプロセッサーは，第一世代のそれがpre-compilerであり，FORTRANを中間言語として作り出していたのに対し，object codeをdirectに作り出す形式に変更された。

第二世代の言語の重要な特徴として，シミュレーション言語がよりユーザー・オリエントなものになってきたことが挙げられる。これはシミュレーション言語によるモデルの表現能力が拡張され，同時にプログラミングの煩雑さが取り除かれ規則的なルールに従ってモデル作りができる様になっ

てきたことに起因している。このためシミュレーション言語はシミュレーションの手段としてはもちろん，シミュレーションそのものの定義というべきものへと変身してきている。この傾向はシミュレーション・アナリシストの関心を以前よりモデルそのものへ向けることに成功したと言える。

2.3.4 次世代のシミュレーション言語

次世代のシミュレーション言語は今日なされている様々な研究，アイデアを応用したものになるであろう。

M. I. T.シミュレーション・システムOPS-3ではタイム・シェアリングが可能になっているが，シミュレータを会話型とし，Man-Machine的性格をもたせることは，シミュレーション・アナリストに大きな力となるであろう。

又シミュレータの新しい方向としてDESLとCSSLの結合型がある。これについては次節で詳しく述べるが，DESLによりマクロな世界そしてCSSLによりミクロな世界をカバーする新しいシミュレータはシミュレーションという解析方法にuniversalなpowerを与えるであろう。

3．コンバインドシミュレータの開発

3. コンバインドシミュレータの開発

まえがき

この章では2つの会話型コンバインドシミュレータ，SPACEとSIMBOL－Ⅱについて説明する。このうち，SPACEはSIMBOL－Ⅱの開発に先き立ち，いわば実験用システムとして作成したシステムである。システム自身は小規模であるが，コンバインドシミュレータ，インプリメント上の技術的な問題点や会話型システムとして備えるべき機能の検討などの実験目的は充分に果たしてくれた。

一方，SIMBOL－Ⅱは実験システムから得た成果をもとにして，離散系シミュレーションシステムSIMBOLをベースに連続系シミュレーション機能を加える形で設計した。

3.1 コンバインド型シミュレーション言語"SPACE"

3.1.1 概　要

"SPACE"（Simulation Program Aided Combined Modeling）は連続系機能と離散系機能を含わせ持ったFORTRANをベースとしたコンバインド型のシミュレーション言語であり，しかも会話型としてインプリメントされており，シミュレーションをインターアクティブに行なう事ができる実験システムである。

コンバインドシミュレーション言語は離散系シミュレーション言語と連続系シミュレーション言語とを結合する事により構成されるが，SPACEでは，イベント中心のGASPタイプの離散系言語に連続モデルをシミュレートするのに必要と思われる最小限の機能を付け加える事により構成されている。

具体的にはSPACEは離散系のスケジューリング機能と連続系の積分機能を組み合わせたタイミングルーチンにより，積分を行ないながら時刻を進めて行き，イベントがスケジュールされている時点に達したら，スケジュールされていたイベントを発生させ，その処理が終ると再び積分を行なうといった様に積分を進めて行く。

また連続系変数の値により離散系のイベントをスケジュールしたり，あるいは離散系イベントにより連続系をコントロールする事がモデルの作成上必要となるが，その点はSPACEが管理していないので，ユーザーは自分でそれを行なえる様に，プログラムしなければならない。

3.1.2　SPACEによるシミュレーションの実行

SPACEはTSS-FORTRANにより作られており，このTSS-FORTRAN の機能によりシミュレーションをインターアクティブに行なう事ができるのであるが，SPACEシステムはサブルーチン群によって構成されている為，ユーザーはシステムの持っているサブルーチンを呼び出す形でモデルを記述し，端末からシミュレーションに必要なデータを与えなければならない。

本節では，SPACEの具体的な動作についての説明及び実行にあたってのモデルの記述方法とデータの与え方を述べる。

(1)　SPACEの構造

(i)　TSS-FORTRANの構造

SPACEのベースになっているTSS-FORTRANは，FACOM 230／60，Monitor V の管理下で動く会話型Fortran である。その構造は図3－1で示す通りである。

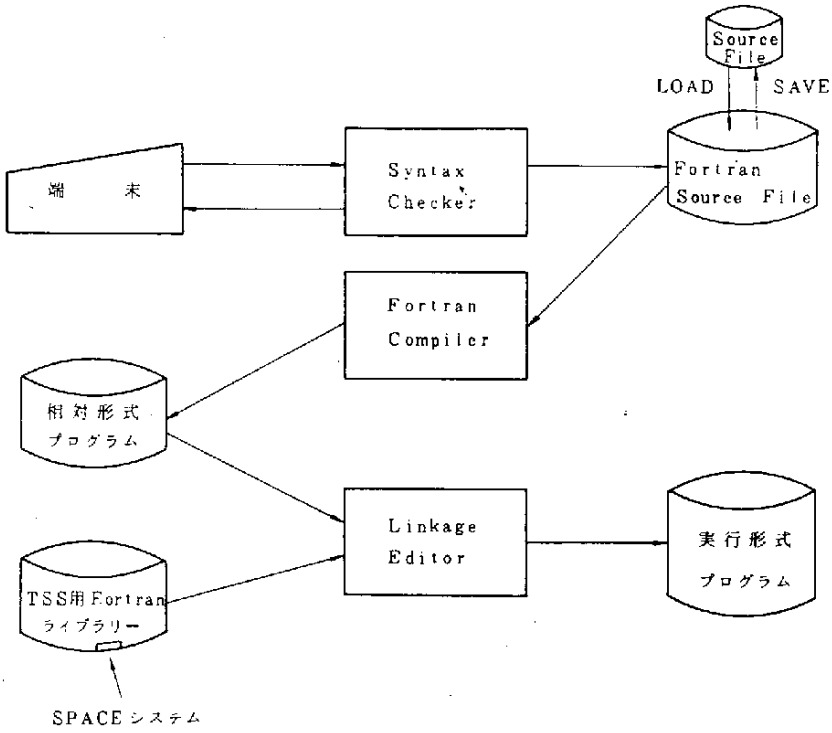


図3－1　TSS Fortran 概略図

ユーザーが端末より入力したプログラムは，シンタックスチェッカーによりFORTRAN の文法をチェックした上でFortran リースファイルにいれられる。全てのプログラムの入力を終るとユーザーがRUNのコマンドを出す事により，リースプログラムはコンパイルされて相

対形式のプログラムが作られる。コンパイルが終るとリンケージエディターが起動され，TSS用のFortranライブラリーとリンクして実行形式のプログラムを作成して，これを実行する。TSS用Fortranライブラリーは一般のFortranライブラリーに，入出力部分等の修正を加えたものである。

また実行途中で，プログラムの修正を行なった場合にも，再び全体のCompileからやり直さなければならない。

(ii) TSS-FORTRANの機能

1) プログラムの作成，変更，実行などがユーザーのコマンドにより任意に行なう事ができる。

2) プログラムを1ステートメント入力するごとに，Fortranの手法エラーをチェックする事ができる。

3) 入出力をフリーフォーマットにより行なう事ができる。

4) プログラムをファイルとして保存する事ができる。

以上の様な特徴がある会話型Fortranであり，Fortranとしては，JIS FORTRAN 水準7000にごくわずかの制限を加えたものである。

プログラムの入力，修正等の為に次のコマンドが用意されている。

¥ INPUT	ソースプログラムの入力
¥ UPDATE	すでに入力したソースプログラムの入力
¥ LIST	ソースプログラムのリストを端末に
¥ DELETE	ソースプログラムの削除
¥ RESEQ	入力の終ったソースプログラムのLine Numberの
¥ REPLACE	ソースプログラムの一部の文字列の置換
1 Line 修正コマンド	ソースプログラム入力中において1行の修正，追加

その他実行コマンドとか，プログラム保存に関するコマンドなど次のものがある。

¥ RUN	入力したプログラムの実行を指示する。
¥ SAVE	入力したソースプログラムをファイルに保存する。
¥ PURGE	SAVEコマンドによって保存されているソースプログラムを保存ファイル上から消去する。
¥ LOAD	保存ファイル上にあるソースプログラムを今回のJOBで使う為にソースファイルに取り出す。
¥ NAMELIST	保存ファイルにあるプログラムを端末に出力する。
¥ STOP	JOBの終了を指示する。

その他の機能としてプログラムの実行中に一時停止をさせる事ができる。これをクイット機能と呼ぶ。一時停止後，実行を継続する時はRESTART，実行を中止する時はENDと入力す

ればよい。

(iii) SPACEにおける機能

SPACEシステムのサブプログラムは全てコンパイルされて，FORTRANライブラリーの中にいれられていて，ユーザーのプログラムとリンクされてシミュレーションを行なう。

(2) モデルの記述法

SPACEによってシミュレーションを行なう場合，ユーザーは次のルーチンを記述しなければならない。

① MAINルーチン

② Subroutine START

③ イベントルーチン

④ Subroutine CONTRL

⑤ Subroutine CMODEL

⑥ Subroutine SCOND

⑦ Subroutine ENDRUN

CMODELおよびSCONDは連続系のモデルについて記述するルーチンであり，イベントルーチンは離散系の状態の変化をイベントごとに記述するルーチンである。以下各々のルーチンの記述法について説明する。

(i) MAINルーチン

一般のFortranのプログラムでその中でサブルーチンSPACEをコールする。このルーチンの中でユーザーはSPACEがシミュレーション実行中にファイルとして使用するエリアを宣言しなければならない。そのエリアは二次の配列が必要で，配列の行数がファイルに一時にストアできるエレメントの個数，列数－2がエレメントの最大属性値個数となるので，ユーザーは必要に応じた大きさの配列を宣言すればよい。下はMAINルーチンの一般的な記述例である。この場合，一時にファイルにストアされるエレメントの個数は100個，最大属性値個数は28個となる。

```
DIMENSION SET ( 100, 30 )
ISET = 100
JSET = 30
CALL SPACE ( SET, ISET, JSET )
STOP
END
```

(ii) Subroutine START

ユーザーが定義した変数についての初期値設定等を行なう為に用意されている。その他，状

態変数の初期値設定，イベントのスケジュール，任意変数の読み込み等，ユーザーが自由に行なう事ができる。

(iii) イベントルーチン

離散系の状態の変化を記述するルーチンであり，各々のイベントごとに記述される。ただし対象とするシステムは同一でも，モデル化の方法，あるいはプログラムの方法によりイベントルーチンの数，内容は異なる事がある。

(v) Subroutine CMODEL

連続系部分の中心となるサブルーチンで，微分方程式を記述する。記述法には制約があり，次の様に書かなければならない。

微分方程式により表わされる状態変数をS(I)と置き，その導関数をD(I)と置いて，方程式をD(I)について書かなければならない。例えば，$\frac{dx}{dt} = x$という式では，xをS(1)と置き，その導関数$\frac{dx}{dt}$をD(1)と置く事により次の様に表わされる。

```
D(1) = S(1)
```

この式をCMODELの中に書く事により，S(1)の値は次々とSPACEに含まれるRunge-Kutta-Gillの積分ルーチンにより求められていく。ただしS(I)の初期値はデーターあるいはサブルーチンSTARTの中で指定しなければならない。

また同種の微分方程式が複数個ある場合にはDO，LOOPにより表わす事ができる。例えば$\frac{dx_1}{dt} = C_1 * x_1$，$\frac{dx_2}{dt} = C_2 * x_2$の二式がある時，$x_1$，$x_2$をS(1)，S(2)と置き，$C_1$，$C_2$をC(1)，C(2)にストアしておく事により，次の様に記述すればよい。

```
      DO  100  I = 1, 2
100   D(I) = C(I) * S(I)
```

また高階の微分方程式は一階の連立微分方程式に変換した上で記述すればよい。

例えば，$a\ddot{x} + b\dot{x} + cx = 0$という微分方程式では，まず次の様に変換する。

$$\frac{dx}{dt} = \dot{x}$$

$$\frac{d\dot{x}}{dt} = \ddot{x} = -\frac{1}{a}(b\dot{x} + cx)$$

ここでxをS(1)，$\dot{x}$をS(2)と置くと

$$\frac{dS(1)}{dt} = S(2)$$

$$\frac{dS(2)}{dt} = -\frac{1}{a}(bS(2) + cS(1))$$

そして $\frac{dS(1)}{dt}$ をD(1)，$\frac{dS(2)}{dt}$ をD(2)と置いて，次の式をCMODEL の中で記述すればよい。

$$D(1) = S(2)$$

$$D(2) = -\frac{1}{a}\ (\ b * S(2) + c * S(1)\ )$$

このようにして記述されたD(I)についての方程式の順序関数は特に意味を持たない。例えば上の例ではD(1)についての方程式が先にあっても，D(2)についての方程式が先にあってもかまわない。これは積分の計算が，一式づつ行なわれていくのではなく，全ての導関数の値を求めてから行なわれる事による。

(vi) Subroutine SCOND (SET, ISET, JSET)

連続系の変数に対する論理的な判断を行なって，離散系の事象をスケジュールしたり，統計量の収集等を行なう為のサブルーチンである。

(vii) Subroutine ENDRUN (SET, ISET, JSET)

標準レポート出力後にサブルーチンREPORTの中でコールされるので，ユーザーは必要に応じて任意の変数の値を出力する事ができる。

今まで述べた7種類のルーチンのうち，イベントルーチン以外のルーチンは必要のない場合でも必ず記述しなければならない。

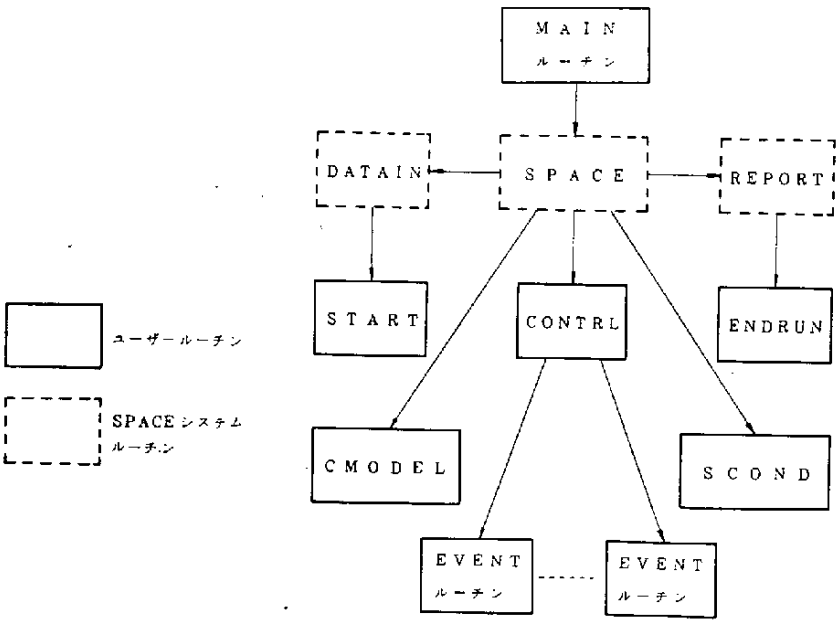


図3－2　SPACEのシミュレーションモデルの構造

```
    * * * S P A C E * * *

 INPUT THE START TIME
  DATA 0
 DO YOU INPUT THE END TIME AS A DATUM?
  DATA Y
 INPUT THE END TIME
  DATA 200
 INPUT THE NUMBER OF FILE
  DATA 1
 INPUT THE COLUM TO DECIDE THE PRIORITY OF FILE   1
  DATA 1 0
 INPUT THE RULE TO DECIDE THE PRIORITY OF FILE   1
  DATA 0 0
 DO YOU USE THE HISTOGRAM?
  DATA N
 DO YOU USE THE RANDOM NUMBER?
  DATA Y
 INPUT THE PARAMETER OF RANDOM NUMBER  (NO./PARAM.1/PARAM.2/PARAM.3/PARAM.4)
  DATA 1 0.4 0.8 0 0
 IS THAT ALL?
  DATA N
  DATA 2 0.03 0.15 0 0
 IS THAT ALL?
  DATA N
  DATA 3 0.2 0.1 0 0
 IS THAT ALL?
  DATA N
  DATA 3 0.02 0.1 0 0
 IS THAT ALL?
  DATA Y
 DO YOU USE THE SELECT FUNCTION?
  DATA Y
 SPECIFY A SELECT FUNCTION  (NO./AMOUNT OF POINTS)
  DATA 1 4
 INPUT THE VALUE OF SELECT FUNCTION  NO= 1   POINT= 4   (X1,Y1,X2,Y2,....)
  DATA 0 0 1 3.1 3 0.65 5 0.8
 IS THAT ALL?
  DATA Y
 INPUT THE AMOUNT OF STATE VARIABLES TO BE USED
  DATA 5
 INPUT THE STEP SIZE OF INTEGRATION
  DATA 0.5
 DO YOU SPECIFY THE INITIAL CONDITION OF STATE VARIABLE?
  DATA Y
 SPECIFY THE INITIAL CONDITION OF STATE VARIABLE  (X1,X2,...)
  DATA 10 10 10 10 10
 DO YOU INPUT THE SCHEDULING DATA?
  DATA Y
 INPUT A SCHEDULING DATA  (FILE NO./ATT1/ATT2/.....)  ICOL= 3
  DATA 1 5 1 0
 IS THAT ALL?
  DATA N
  DATA 1 10 2 0
 IS THAT ALL?
  DATA N
  DATA 1 15 3 0
 IS THAT ALL?
```

図3－3　シミュレーション開始時のパラメータ設定の例（その１）

```
 DATA N
 DATA 1 20 4 0
IS THAT ALL?
 DATA N
 DATA 1 25 5 0
IS THAT ALL?
 DATA N
 DATA 1 0 200 5
IS THAT ALL?
 DATA Y
DO YOU INPUT THE TRACE DATA?
 DATA Y
INPUT THE TRACE DATA  (AMOUNT/STATE VARIABLE NO.....)  (END--1E0F)
 DATA 5 1 2 3 4 5
 DATA 1E0F
DO YOU WANT TO PLOT THE STATE VARIABLES?
 DATA Y
DO YOU SPECIFY THE UPPER LIMIT AND LOWER LIMIT OF STATE VARIABLES?
        IF YOU DO NOT SPECIFY   U.L.=100.  L.L.=0.
 DATA Y
INPUT  (STATE VARIABLE NO./UPPER LIMIT/LOWER LIMIT)
 DATA 1 60 0
IS THAT ALL?
 DATA N
 DATA 2 60 0
IS THAT ALL?
 DATA N
 DATA 3 60 0
IS THAT ALL?
 DATA N
 DATA 4 60 0
IS THAT ALL?
 DATA N
 DATA 5 60 0
IS THAT ALL?
 DATA Y
DO YOU START THE SIMULATION?
 DATA Y
DO YOU WANT TO DUMP THE QUEUE AT THE STARTING POINT?
 DATA Y
```

図3－3　シミュレーション開始時のパラメータ設定の例（その２）

(3) シミュレーションデーターの与え方

SPACEによりシミュレーションを行なう場合，シミュレーションの実行に必要なデーターは，SPACEシステムが一つ一つコメントを出して要求して来るので，ユーザーはそのコメントに従がったデーターを入力すればよい。

SPACEではシミュレーションの1回のランが終了した後パラメーター等の値を変更してランを繰返す事ができるが，その時の為にデーターを次の3種類に分類している。

① シミュレーションのランを行なうたびに入力させる必要があるもの。（例えば終了時刻など）

② シミュレーションのランのごとに変更する可能性のあるもの。（例えばパラメーターなど）

③ 途中で変更する可能性のないもの。（例えば使用するファイルの数など）

そして1回目のランを行なう前には①から③まで全てのデーターを入力させるが，2回目以降のランでは③については入力させない。また②については値を変更するかどうかを尋ね，変更する場合についてのみ新しい値を入力させる。

図3－3は第4章で述べた交通モデルについて実行を行なった際のデーター入力部分である。（但し数値は記載されている結果を出した時のものではない。）

3.1.3 言語仕様の概略

SPACEシステムはFORTRANのサブプログラム群により構成されているが，主要なサブプログラムを機能により分類すると次の様になる。

① SPACEシステム全体の管理等

Subroutine SPACE

② テンポラリーエレメントを扱う為にSPACEではファイルと呼ばれるチェイン構造を持ち，このファイルを操作，管理する為に次のサブルーチンがある。

Subroutine INSERT ：ファイルへエレメントを挿入する

Subroutine REMOVE ：ファイルからエレメントを取り出す

Subroutine SFIND ：ファイルから特定の条件を満足するエレメントを検索する

Subroutine SETUP ：ファイルの管理を行なう

③ 統計量の収集機能

Subroutine STATIS ：統計表

Subroutine TMSTAT ：時間統計表

Subroutine HSTGRM ：度数分布表

④ 連続系機能

Function SWICH ：スイッチ

Function DELAY ：おくれ時間

Function　IEQUL　：条件一致の判定

Function　CMPAR　：比較器

Function　RAMP　：ランプ入力

Function　STEP　：ステップ入力

Function　DESPA　：デッドスペース

連続系機能の中心となるのは積分機能であるが，積分機能はサブプログラムの形をとらずに，サブルーチンSPACEの中に含まれ，微分方程式はサブルーチンCMODELの中に特別の記述法に従がって記述する事により積分が行なわれる。

⑤　乱数発生機能

Function　RNORML：正規乱数

Function　RANDOM：一様乱数

Function　NPOISN：ポアソン乱数

Function　ERLANG：アーラン乱数

Function　RLOGNM：対数正規乱数

⑥　任意の分布に従がう関数の発生機能

Subroutine　CSELCT：連続数値関数

Subroutine　DSELCT：離散数値関数

⑦　出力機能

Subroutine　QDUMP　：待行列ダンプ

Subroutine　QPRINT：待行列統計表作成

Subroutine　LPPLOT：時系列グラフ

Subroutine　TRACER：トレース

Subroutine　HSTGRP：度数分布表のグラフ

その他の出力機能として次のものがある。

- 連続系状態変数の時系列のプリント
- 連続系状態変数の時系列のグラフの作成機能

(1)　サブルーチン　SPACE

サブルーチンSPACEはSPACEシステムの中核をなす部分であり，連続系モデルにおける微分方程式を積分しながら時間を進める機能，スケジュール表に従がってイベントを発生させる機能および，SPACEシステム全体をコントロールする機能を持っている。サブルーチンSPACEの流れの概略は図3－5に示す通りである。

サブルーチンSPACEはユーザーのメインプログラムの中でコールされる事により実行を開始すると，まずデーターの入力ルーチンをコールして必要なデーターの読み込み，および初期値設

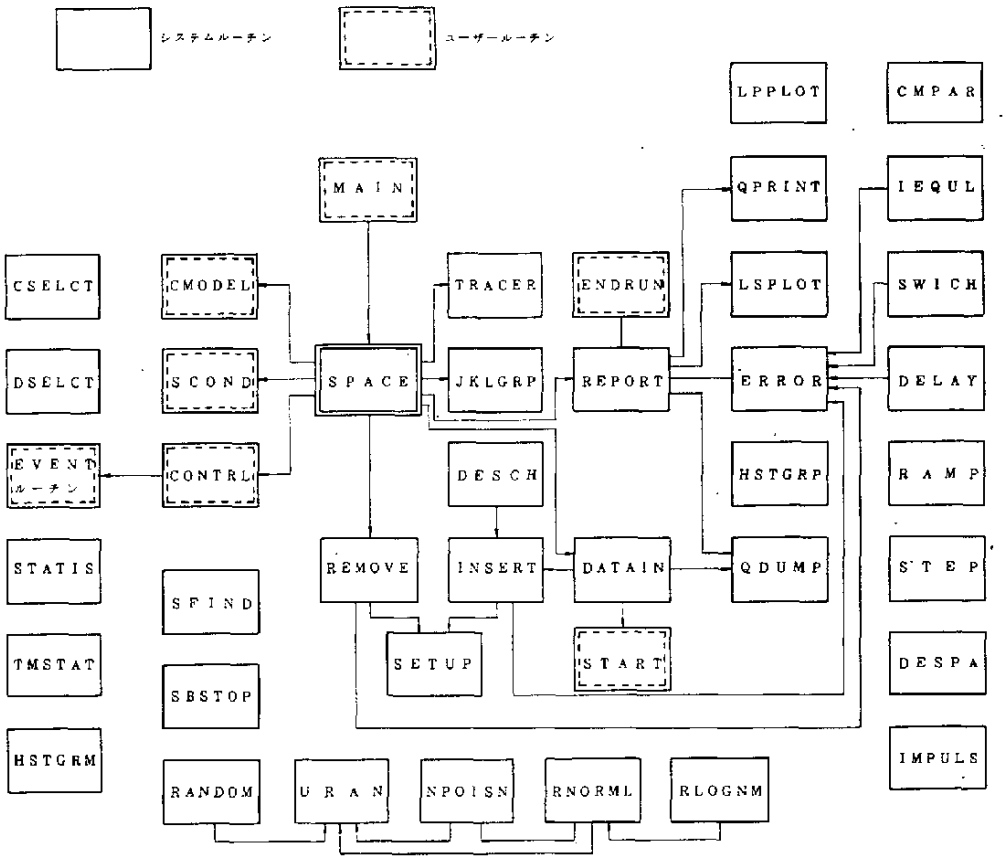


図3－4　SPACEシステムルーチンとユーザールーチン構成図

定を行なう。その後シミュレーションを開始し，離散系のイベントの発生と，連続系の積分のタイミングをコントロールしながら，時間を進めて行く。そして終了条件に達すると出力ルーチンをコールして，収集された統計量を出力させる。出力が終ると，再びシミュレーションを繰返すかどうかを，ユーザーに判断させ繰返す場合には再び入力ルーチンを呼ぶ所まで戻る。繰返しをしない場合には，最初サブルーチンSPACEをコールしたメインプログラムにもどる。

(2)　テンポラリーエレメントの扱い

テンポラリーエレメントの扱いは，離散系システムのシミュレーションを行なう場合はかなり重要な問題を持つ。SPACEではファイルと呼ばれる集合を持っており，その中に各エレメントがストアされる。まずファイルの構造について述べる。

エレメントはその属性値からなり，その属性値により決定された優先度に従がって，ファイルはチェイン構造になっている。優先度を決定する為に比較の対象とする属性値番号およびその型（属性値の小さいものほど優先度を高くするか，あるいはその逆かという事）はユーザーにより

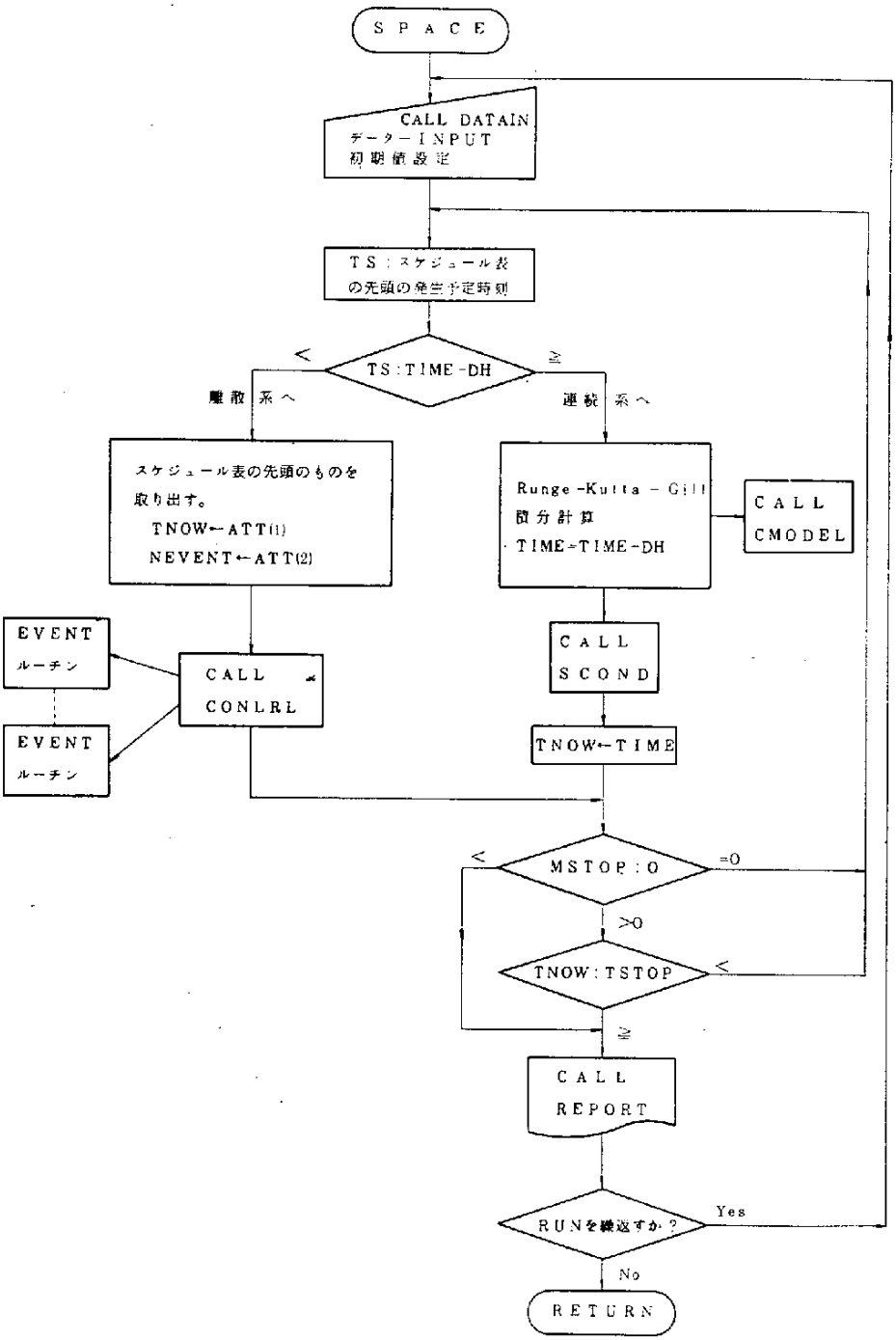


図3－5　サブルーチンSPACE概略図

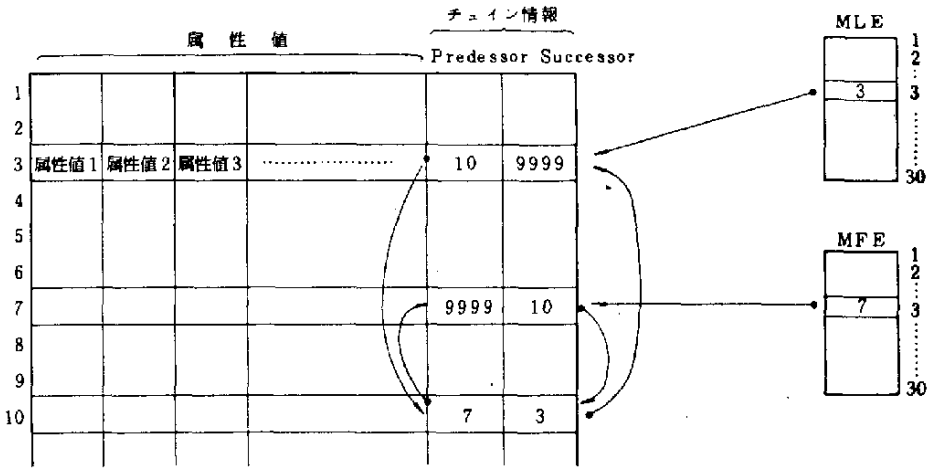


図３－６　ファイルの構造

データーとして指定される。第１の優先度の決定要素が等しかった場合に用いる為に，第２優先度まで指定する事もできる。このように優先度決定要素を与える事により，ユーザーはファイルに自由に性格を与える事ができる。

ファイルは全部で30個用意されていて，それらを順にファイル1……ファイル30と呼ぶ。ファイル１はシステムがスケジュール表として使用するが，その他のファイルはユーザーが自由に使用する事ができる。

これらのファイルは配列ＳＥＴの中に構成され，ＳＥＴの１行に対し１つのエレメントが対応する。この１行の中にエレメントの各属性値及びチェイン情報がいれられる。図３－６はファイルｎのエレメントが，第３行，７行，10行にストアされ，その優先度が７－10－３の順で高い場合のチェイン情報を示している。

この図の様に各行の最後の２列はチェイン情報をいれる為に使用され，優先度が１つ高いエレメントと，１つ低いエレメントがストアされている行の番号がいれられる。また各ファイルの先頭と最後のエレメントのストアされている行番号は，ＭＦＥ(Ｎ)およびＭＬＥ(Ｎ)を参照する事により知る事ができる。これらのチェイン情報およびＭＦＥ(Ｎ)，ＭＬＥ(Ｎ)等はシステムにより自動的に管理されているので，ユーザーが操作する必要はない。

この他，次の統計量が自動的に収集される。ただしＮＱＺ(Ｎ)はユーザーが自分でカウントしなければならない。

ＬＱ(Ｎ)　　：現在ファイルに入れられているエレメントの個数

ＭＸＬＱ(Ｎ)　：ファイルに一時にいれられたエレメントの個数

ＮＱ(Ｎ)　　：ファイルにいれられたエレメントの合計個数

QTIME(N) ：ファイルから出し入れをした最終時刻

QINTG(N) ：累積使用時間

NQZ(N) ：待ちゼロのエレメントの個数

次にファイルからエレメントを出し入れ等をする時の為のサブルーチンについて説明する。

(i) ファイルへの挿入

ファイルにエレメントを挿入する為にサブルーチンINSERTがあり，そのコーリングシークエンスは次の通りである。

```
CALL INSERT(IQ, SET, ISET, JSET)
  IQ：ファイル番号
```

これにより配列ATT(1)～ATT(ICOL)の中に入れられている属性値からなるエレメントが，セットIQに挿入される。INSERTではエレメントを挿入した後，サブルーチンSETUPをコールしチェイン情報等の修正，更新を行なう。

例えば第1属性値10，第2属性値40，第3属性値20からなるエレメントをファイル3へ挿入する場合，次の様にプログラムすればよい。

```
ATT(1) = 10
ATT(2) = 40
ATT(3) = 20
CALL INSERT(3, SET, ISET, JSET)
```

これにより指定したエレメントをファイル3のチェインの中の，情報等に従がった位置に挿入される。

(ii) ファイルからの取り出し

ファイルからエレメントを取り出す為にサブルーチンREMOVEがあり，そのコーリングシークエンスは次の通りである。

```
CALL REMOVE(KROW, IQ, SET, ISET, JSET)
  KROW：行番号
  IQ  ：ファイル番号
```

これにより，配列SETの第KROW行にストアされているファイルIQのエレメントを取り出し，その属性値をATT(1)～ATT(ICOL)の対応する番地に入れる。その後サブルーチンSETUPをコールし，取り出したKROW行を御破算にし，チェイン情報等の修正，更新等を行なわせる。

KROWの値は，MFE(N)，MLE(N)の項を参照するが，あるいは次に述べるサブルーチ

ンSFINDにより決定する等して必ず指定しなければならない。

また指定したセットが空であった場合にはエラーとなるので，セットにストアされているエレメントの個数が不明な場合には，LQ(N)の値により数を確認した上でREMOVEをコールする必要がある。

例えばファイル2の先頭のエレメントを取り出したい時，次の様にプログラムすればよい。

```
IF(LQ(2).LE.0)GO TO 100
CALL REMOVE(MFE(2),2,SET,ISET,JSET)
```

(iii) ファイルの管理

ファイルについてのチェイン情報，その他の情報を管理する為にサブルーチンSETUPがあり，そのコーリングシークエンスは次の通りである。

```
CALL SETUP(IQ,SET,ISET,JSET)
  IQ：ファイル番号
```

但しSETUPはユーザーがコールする必要はなく，ファイルからエレメントを出し入れした時，およびファイルを御破算にする時に自動的にコールされる。(すなわちサブルーチンINSERT，REMOVE，DATAINの中で自動的にコールされる。)

(iv) ファイルの検索

ファイルの中から特定の条件を満足するエレメントを検索する為にサブルーチンSFINDがあり，このコーリングシークエンスは次の通りである。

```
CALL SFIND(XVALUE,LCOL,IQ,MCODE,KROW,SET,ISET,
           JSET)
  XVALUE：比較の基準となるデーター
  LCOL：検索する列の番号
  IQ：検索するセットの番号
  MCODE：検索する型を指定するコードで次の種類がある。
    MCODE＝1  XVALUEより大きい数のうち最大値
    MCODE＝2  XVALUEより大きい数のうち最小値
    MCODE＝3  XVALUEより小さい数のうち最大値
    MCODE＝4  XVALUEより小さい数のうち最小値
    MCODE＝5  XVALUEに等しい数
  KROW：検索した結果，条件を満たしたエレメントのストアされている行の番
    号がいれられる。該当するエレメントがなかった場合はゼロがセットされる。
```

MCODE＝5の場合には最初に条件を満たしたエレメントのストアされている行番号がいれられる。

これによりファイルIQのエレメントについて，LCOL行の値をXVALUEと比較してMCODEで指示された条件を満たすエレメントがあれば，そのエレメントがストアされている行の番号をKROWにいれる。また該当するエレメントがなかった場合はゼロがセットされる。

例えば，ファイル4で第1属性値が20より大きいもののうち，最小値をサーチし，見つかればそれを取り出す場合，次の様にすればよい。

```
CALL SFIND(20., 1, 4, 2, K, SET, ISET, JSET)
IF(I.LE.0)GO TO 100
CALL REMOVE(I, 4, SET, ISET, JSET)
```

サーチは優先度の高い順に行なわれるので，同一の値のものが複数個ある場合には優先度の高いものが選ばれる。

(3) 統計量収集のためのサブルーチン

SPACEでは3種類の統計収集ルーチンが用意されており，ユーザーがプログラムの中でコールする事により後述する統計量が収集される。収集された統計量はシミュレーション終了後，出力ルーチンにより自動的に計算が行なわれて出力される。

(i) 平均値等の計算ルーチン

平均値等を求める為にサブルーチンSTATSがあり，そのコーリングシークエンスは次の通りである。

```
CALL STATIS(X, N)
  X：データー値
  N：使用する統計表の番号
```

このルーチンをコールする事により次の統計量が得られる。

① $\sum_{i=1}^{m} X_{Ni}$ （和）

② $\sum_{i=1}^{m} X_{Ni}^{2}$ （二乗和）

③ m （観測個数）

④ $MIN(X_{Ni})$ （最小値）

⑤ $MAX(X_{Ni})$ （最大値）

収集されたデーターについて次の値が計算されて出力される。

① 平 均 値

② 標準偏差

③ 最 大 値

④ 最 小 値

⑤ 観測されたデーターの個数

(ii) 時間統計ルーチン

シミュレーション実行中に種々の統計を時間の関数として求めたい場合がある。SPACEではその為にサブルーチンTMSTATがあり，そのコーリングシークエンスは次の通りである。

```
CALL TMSTAT ( X, T, N )
  X：データー値
  T：観測時間
  N：使用する時間統計表の番号
```

このルーチンをコールする事により，次の統計量が得られる。

① $\sum_{i=1}^{m} (TT_{Ni})$ （観測間隔和）

② $\sum_{i=1}^{m} (TT_{Ni} * X_{Ni})$ （和）

③ $\sum_{i=1}^{m} (TT_{Ni} * X_{Ni}^{2})$ （二乗和）

④ T （観測時刻）

⑤ $MIN (X_{Ni})$ （最小値）

⑥ $MAX (X_{Ni})$ （最大値）

⑦ 観測開始時刻

式を見ると明らかな通り②の和を求める場合は，右図における斜線の部分の面積を求めている事になる。また③の二乗和の場合も同様である。

収集されたデーターについて，シミュレーション終了後次の値が計算されて図3－7の様な表の形で出力される。

① 平 均 値

② 標準偏差

③ 最 大 値

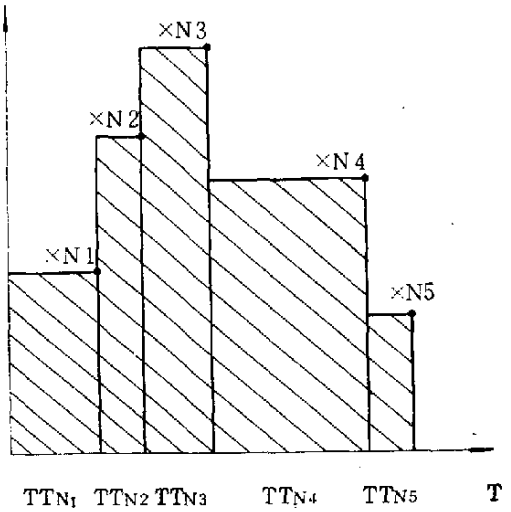


図3－7 時間統計

④ 最 小 値

⑤ 全観測時間

⑥ 観測開始時刻

⑦ 最終観測時刻

（例） ある部屋にいる人数Nについての平均値等を求めたい場合，部屋に人間がはいる時点と，部屋から出る時点でこの命令をいれる。時間統計表1を使うとすると次の様にすればよい。

```
      :
      :
CALL TMSTAT ( FLOAT (N), TNOW, 1 )
N = N + 1
      :
      :
CALL TMSTAT ( FLOAT (N), TNOW, 1 )
N = N - 1
      :
      :
```

厳密に計算を行なうなら，シミュレーション終了の直前にもう一度TMSTATをコールする必要がある。

STATISTICAL TABLE

NUMBER	MEAN	STD.DEV.	MINIMUM	MAXIMUM	COUNT
2	4.868	1.405	1.147	8.714	200
4	117.162	44.401	22.886	255.057	200
5	9.998	2.990	3.504	16.405	200
10	256.407	159.973	13.775	953.729	200

図3－8 統計表出力例

TIME STATISTICAL TABLE

NUMBER	MEAN	STD.DEV.	MINIMUM	MAXIMUM	TOTAL TIME	START TIME	LAST TIME
3	53.582	21.113	9.374	120.515	100.000	0.000	100.000
4	4.868	1.401	1.147	8.714	100.000	0.000	100.000
9	48.714	20.131	7.550	112.480	100.000	0.000	100.000

図3－9 時間統計出力例

(iii) 度数分布表作成ルーチン

データーの平均値等の統計量の他に，任意にクラス分けをした度数表を作成する為にサブルーチンHSTGRMがあり，そのコーリングシークエンスは次の通りである。

```
CALL HSTGRM(X, T, N)
  X：データー値
  N：使用する度数分布表の番号
```

このルーチンをコールする事により，次の統計量がシミュレーション終了後出力される。まず全体に関して

① 観測個数
② 平均値
③ 標準偏差
④ 最大値
⑤ 最小値
⑥ データー値の総和

および各々のクラスに関して

⑦ 度数
⑧ 全体に対するパーセント
⑨ そのクラス以下の値の全体に対するパーセント
⑩ そのクラス以上の値の全体に対するパーセント
⑪ そのクラスの上限値の平均値に対する比
⑫ そのクラスの上限値と平均値の差の標準偏差に対する比

またオーバーフローしたものに対しては

⑬ 度数
⑭ 全体に対するパーセント
⑮ オーバーフローした値の平均値

が得られる。

度数表のクラスの出発値，きざみ幅，きざみ数はユーザーによりデーターとして与えられる。

また度数分布表のグラフを作成する為にサブルーチンHSTGRPがあり，そのコーリングシークエンスは次の通りである。

```
CALL HSTGRP(I)
  I：度数表番号
```

このルーチンにより，サブルーチンHSTGRMにより作成された度数表をもとに，グラフを作成する為のもので，サブルーチンREPORTの中で度数表の出力を終えた後で，ユーザーの必要に応じてシステムがコールする。

図３－10は度数表およびそのグラフの出力例である。

HISTOGRAM TABLE

TABLE NUMBER 1

TOTAL ENTRIES	MEAN VALUE	STD.DEV.	MINIMUM	MAXIMUM	SUM OF VALUES
2502	5.959	2.021	-0.881	12.526	14907.617

UPPER LIMIT	OBSERVED FREQUENCY	PERCENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER	MULTIPLE OF MEAN	DEVIATION FROM MEAN
0.0	4	0.16	0.2	99.8	0.0	-2.949
1.00	11	0.44	0.6	99.4	0.168	-2.454
2.00	44	1.76	2.4	97.6	0.336	-1.959
3.00	116	4.64	7.0	93.0	0.504	-1.464
4.00	245	9.79	16.8	83.2	0.671	-0.969
5.00	366	14.63	31.4	68.6	0.839	-0.474
6.00	503	20.10	51.5	48.5	1.007	0.021
7.00	475	18.98	70.5	29.5	1.175	0.516
8.00	338	13.51	84.0	16.0	1.343	1.010
9.00	223	8.91	92.9	7.1	1.511	1.505
10.00	113	4.52	97.4	2.6	1.679	2.000
11.00	46	1.84	99.3	0.7	1.846	2.495
12.00	15	0.60	99.9	0.1	2.014	2.990
OVERFLOW	3	0.12	100.0	0.0		

AVERAGE VALUE OF OVERFLOW = 12.43

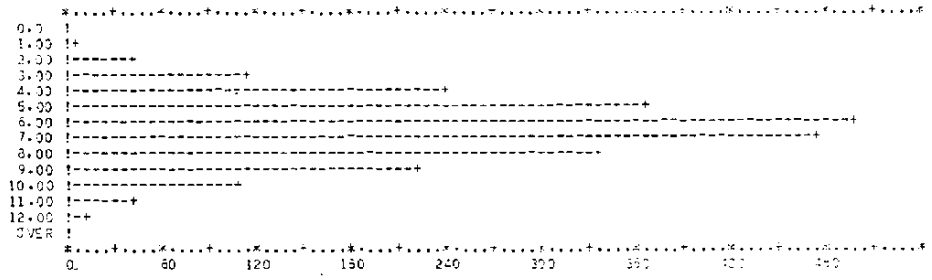


図３－10　ヒストグラム表示例

(4) 連続系要素

連続系モデルを記述する為に固有なものとしてDELAY，IEQUL，SWICH，CMPAR，RAMP，STEP，IMPULS，DESPAがある。

(1) むだ時間　DELAY

```
Y = DELAY ( P, X, YZ, I, H, K )
    P  ：遅れ時間
    X  ：入力変数
    YZ ：初期出力
    I  ：サンプリング間隔
    H  ：配列名
    K  ：配列の大きさ
```

これにより時刻 t が

t ＜ Pの時　Y＝YZ

t ≧ Pの時　入力変数Xが時間Pだけ遅れてYに出力される。

ユーザーはDELAYを使用するサブルーチン（CMODEL）の中で，DELAYの中でデーターを記憶しておく為の配列を宣言してその配列名と大きさを引数としてDELAYを使用する。必要な配列の大きさは，DELAYの中の制御情報等もストアする為に多少大きめのものが必要となり，それは次式により求められる。

$$\text{必要の配列のサイズ} = \frac{\text{おくれ時間}}{\text{積分きざみ幅} * \text{サンプリング間隔}} + 3$$

但し少数点以下は切上げる。

サンプリング間隔が2以上の場合，サンプリングされていない時の値は線形補間により求められる。

(ⅱ) スイッチ　SWICH

```
Y = SWICH ( R, X1, X2, X3 )
    R  ：スイッチ切換判定要素
    X1 ：
    X2 ：  入力変数（定数）
    X3 ：
```

これにより

R ＜ 0の時　Y＝X1

R ＝ 0の時　Y＝X2

R ＞ 0の時　Y＝X3

となる。この関数SWICHはサブルーチンCMODELの中でのみ使用可能である。

(iii) 条件一致の判定　IEQUL

```
J = IEQUL ( X, S, L )
    X：入力変数
    S：設定値
    L：一致の条件
        L = 1     Xが上り勾配でSと一致
        L = − 1   Xが下り勾配でSと一致
        L = 0     Xが勾配に無関係にSと一致
```

これにより　X = Sの時　J = 1　，

X ≒ Sの時　J ≒ 1　となる。

この関数はサブルーチンSCONDの中でのみ使用可能である。

(iv) 比較器　CMPAR

```
Y = CMPAR ( X 1, X 2 )
    X 1
          入力変数
    X 2
```

これにより

X 1 ＞ X 2の時　Y = 1.

X 1 = X 2の時　Y = 0.

X 1 ＜ X 2の時　Y = − 1.

となる。

(v) ランプ入力　RAMP

```
Y = RAMP ( P )
    P：ランプ入力のスタート時刻
```

これにより時刻 t が

t ＜ Pの時　Y = 0.

t ≧ Pの時　Y = t − P

となる。

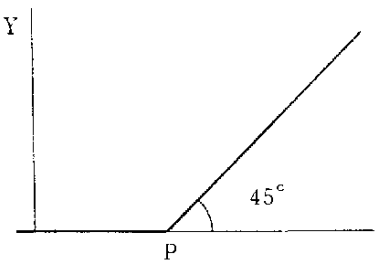


図3－11　ランプ入力

(vi) ステップ入力　STEP

```
Y = STEP ( P )
    P：ステップ入力のスタート時刻
```

これにより時刻ｔが

ｔ ＜ Ｐの時　Ｙ＝０．

ｔ ≧ Ｐの時　Ｙ＝１．

となる。

(vii)　パルス入力　IMPULS

Ｊ＝ＩＭＰＵＬＳ（Ｔ）
Ｔ：パルス入力時刻

これにより時刻ｔが，積分きざみ幅をＤＨとして

Ｔ－ＤＨ／２＜ｔ＜Ｔ＋ＤＨ／２の時

Ｊ＝１

Ｔ－ＤＨ／２＞ｔ　あるいは

Ｔ＋ＤＨ／２＜ｔの時　　Ｊ＝０

となる。

(viii)　ＤＥＡＤ　ＳＰＡＣＥ　ＤＥＤＳＰＡ

Ｙ＝ＤＥＤＳＰＡ（ＴＡ，ＴＢ）
ＴＡ
ＴＢ

これにより時刻ｔが

ｔ＜ＴＡの時　Ｙ＝Ｔ－ＴＡ

ＴＡ＜ｔ＜ＴＢの時　Ｙ＝０．

ＴＢ＜ｔの時　Ｙ＝Ｔ－ＴＢ

となる。

(5)　乱数発生ルーチン

ＳＰＡＣＥシステムでは次の乱数発生関数を用意している。

1)　一様乱数　　ＲＡＮＤＯＭ（Ｎ）

2)　ポアソン乱数　　ＮＰＯＩＳＮ（Ｎ）

3)　アーラン乱数　　ＥＲＬＡＮＧ（Ｎ）

4)　正規乱数　　ＲＮＯＲＭＬ（Ｎ）

5)　対数正規乱数　　ＲＬＯＧＮＭ（Ｎ）

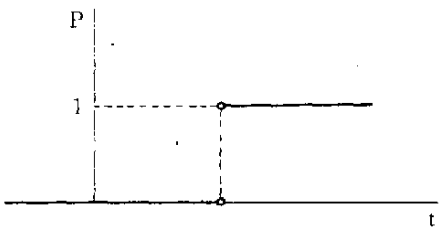


図３－12　ステップ入力

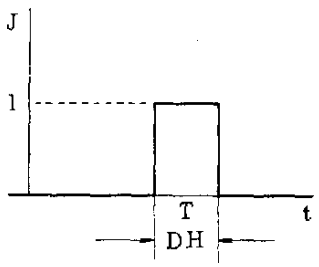


図３－13　パルス入力

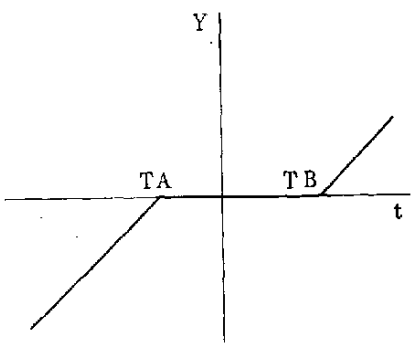


図３－14　デッドスペース

乱数のパラメーターをストアする為に，テーブルPARAMSが用意されていて，その中にデーターとしてパラメーターを与える。そしてプログラムの中で乱数FUNCTIONを使用する時に，引数としてパラメーターのストアされている行の番号を指定する事により，FUNCTIONの中でテーブルPARAMSの中の指定された行を参照し，そのパラメーターに従がった乱数を発生する。

乱数の種類により，必要なパラメーターの種類は異なり，それは次ページの図の通りである。また使用できる乱数発生ルーチンの合計個数は20個以内である。

（例） 平均値0，標準偏差1，最小値－5，最大値5の正規乱数Xを発生させる。PARAMSの第2行を使用するとする。

プログラムの中ではX＝RNORML(2)と書き，PARAMSの第2行には，第1列＝0，第2列＝－5，第3列＝5，第4列＝1とセットすればよい。

表3－1 乱数の一覧表

乱数名	FUNCTION名	パラメーター			
		列 1	列 2	列 3	列 4
一様乱数	RANDOM	最小値	最大値		
ポアソン乱数	NPOISN	平均値	最小値	最大値	
アーラン乱数	ERLANG	平均値	最小値	最大値	型
正規乱数	RNORML	平均値	最小値	最大値	標準偏差
対数正規乱数	RLOGNM	平均値	最小値	最大値	標準偏差

(6) 任意の分布に従がう関数

SPACEシステムでは任意な分布に従がう関数として，離散数値関数および連続数値関数の2つを用意している。どちらも　　値は点の座標としてデーターにより与えられる。

(i) 連続数値関数

Y＝CSELCT（X，N）
X：データー値
N：関数表番号

(X_i, Y_i) $(i = 1, 2 \cdots\cdots n)$ を結んだ右図の様な関数である。X1より小さいデーターXに対しては関数値は常にY1をとり，Xnより大きいデーターに対しては，関数

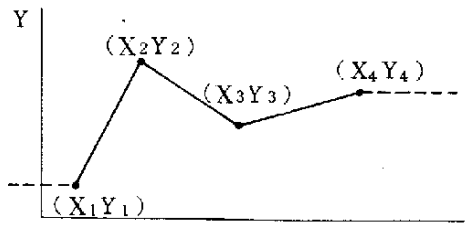


図3－15

値は常にＸｎをとる。

(ii) 離散数値関数

Y＝DSELCT（X，N）
X：データー値
N：関数表番号

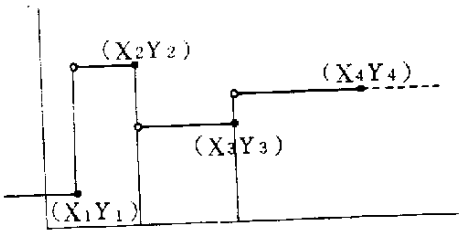


図３－16　ステップ関数

（－∞，Ｘ１）ではＹ１，（Ｘ１，Ｘ２）（左に開き右に閉じる区間）ではＹ２，……（X_j，X_{j+1}）ではY_{j+1}，最終のX_nの値より大きいデーターに対しては，常に最終のY_nの値をとる階段関数である。

(7) 出力機能

SPACEの出力機能としては次の機能がある。

(i) 離散系のトレース

イベントが発生するたびに，現時刻と発生イベント番号をそのイベントの処理が終了した時点でプリントする。それと供に次に発生するイベントについて，発生時刻，イベント番号および他の属性値がプリントさせる。ユーザーはこのプリントアウトを参照する事により，イベントが正しい順序で発生しているか，情報が正しくストアされているか等のチェックをする事ができる。下にその出力例を示す。

```
**  S I M U L A T I O N   T R A C E  **

CURRENT EVENT....TIME=          83.5404 EVENT=   1
NEXT EVENT.......       86.62       1.00        0.0         0.0

CURRENT EVENT....TIME=          86.6177 EVENT=   1
NEXT EVENT.......       87.29       3.00       19.61       37.29

CURRENT EVENT....TIME=          87.2925 EVENT=   3
NEXT EVENT.......       89.11       1.00        0.0         0.0

CURRENT EVENT....TIME=          89.1092 EVENT=   1
NEXT EVENT.......       92.10       3.00       38.74       38.74

CURRENT EVENT....TIME=          92.1047 EVENT=   3
NEXT EVENT.......       93.33       4.00       86.62        0.0
```

図３－17　トレース出力例

(ii) 連続系状態変数のプリント

連続系モデルの中でS(I)により定義した状態変数の値を現在時刻と供にシミュレーション実行中にプリントする。プリントする変数の番号，プリント間隔および開始，終了時刻はデーターとして指定される。図3－18に出力例を示す。

** S I M U L A T I O N T R A C E **

TIME	S(1)	S(2)	S(3)	S(5)	S(7)
	S(9)	S(10)	S(12)		
0.0	0.1000000E 04	0.6000000E 03	0.3000000E 03	0.3000000E 03	0.4000000E 03
	0.5000000E 03	0.7000000E 03	0.9000000E 03		
10.00	0.3676795E 03	0.2207277E 03	0.1103638E 03	0.1103638E 03	0.1471518E 03
	0.1839397E 03	0.2575156E 03	0.3310915E 03		
20.00	0.1353353E 03	0.8120117E 02	0.4060059E 02	0.4060059E 02	0.5413412E 02
	0.6765765E 02	0.9473470E 02	0.1218018E 03		
30.00	0.4978708E 02	0.2987224E 02	0.1493612E 02	0.1493612E 02	0.1991483E 02
	0.2489354E 02	0.3485095E 02	0.4480836E 02		
40.00	0.1831565E 02	0.1098939E 02	0.5494693E 01	0.5494693E 01	0.7326258E 01
	0.9157823E 01	0.1282095E 02	0.1648408E 02		
50.00	0.6737950E 01	0.4042770E 01	0.2021385E 01	0.2021385E 01	0.2695180E 01
	0.3368975E 01	0.4716563E 01	0.6064154E 01		
60.00	0.2478753E 01	0.1487252E 01	0.7436259E 00	0.7436259E 00	0.9915015E 00
	0.1239377E 01	0.1735127E 01	0.2230878E 01		
70.00	0.9118825E 00	0.5471294E 00	0.2735647E 00	0.2735647E 00	0.3647531E 00
	0.4559412E 00	0.6383175E 00	0.8206941E 00		
80.00	0.3354628E 00	0.2012777E 00	0.1006388E 00	0.1006388E 00	0.1341852E 00
	0.1677314E 00	0.2348239E 00	0.3019165E 00		
90.00	0.1234099E 00	0.7404592E-01	0.3702296E-01	0.3702296E-01	0.4936397E-01
	0.6170494E-01	0.8638690E-01	0.1110689E 00		

図3－18　連続系状態変数のプリント例

(iii) 連続系状態変数のプロット

連続系モデルの中でS(I)により定義した状態変数の値をシミュレーション実行中にプロットし，時系列のグラフを作成する。プロットする変数の番号，開始，終了時刻等は②と同様にデータにより与える。ただしプロットできる変数の個数は5個以内に限られる。図3－19に出力例を示す。

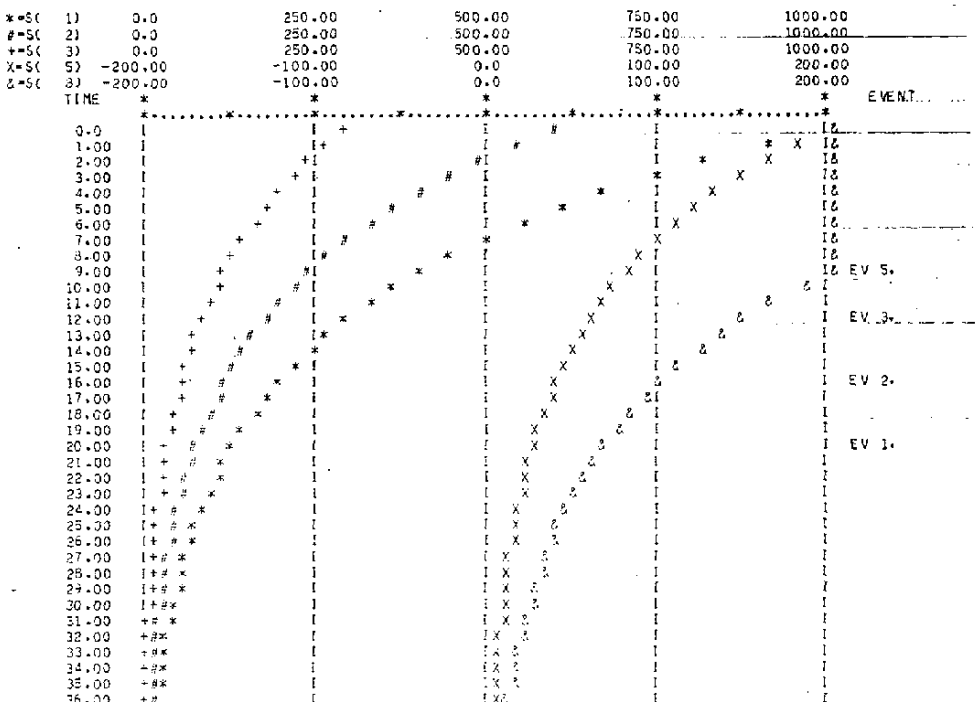


図3－19　連続系状態変数のプロット例

(v) 連続系の任意の変数のプロット

連続系の変数の値をプロットする。この為にサブルーチンLPPLOTがある。ユーザーはサブルーチンSCONDの中で，プロットしたい変数およびサンプリング間隔を引数としてサブルーチンLPPLOTをコールする事により，シミュレーション終了後に自動的にスケーリングされた時系列グラフが作成される。ただしプロットできる変数は3個，各々の変数についてプロットできる点数は90点以内に限られる。

図3－20に出力例を示す。

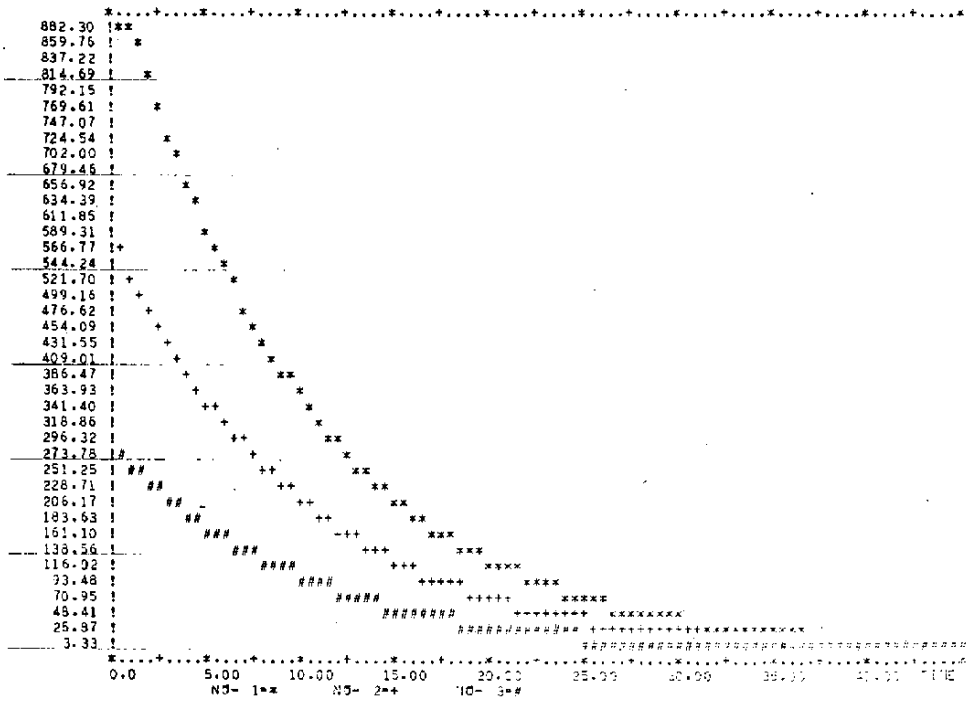


図3－20　時系列グラフの出力例

(v) 待行列統計表の作成機能

使用したファイルについての待行列統計表を作成する為にサブルーチンQPRINTがある。QPRINTはシミュレーション終了後に自動的にコールされるが，ユーザーが必要に応じて途中でコールする事もできる。待行列統計表の項目は次の通りである。図3－21に出力例を示す。

① 最大待行列長

② 平均待行列長

③ 待行列へ入れられたエレメントの個数

④ 待ちゼロのエレメントの個数

⑤ 待ちゼロであったエレメントの割合

⑥ 平均待時間（待ちゼロであったエレメントを含む）

⑦ 平均待時間（待ち行列へ入れられたエレメントのみ）

⑧ 現在の待行列表

QUEUE STATISTICS AT . 200.00 TIME UNITS

QUEUE NUMBER	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/TRANS	*AVERAGE TIME/TRANS	CURRENT CONTENTS
1	5	1.37	29	0	0.0	33.65	33.65	5

図3－21　待行列統計表出力例

* THE CONTENTS OF QUEUE 1 AT 0.0 TIME UNITS

0.0	200.00	5.00
4.00	1.00	0.0
10.00	2.00	0.0
16.00	3.00	0.0
22.00	4.00	0.0
28.00	5.00	0.0
100.00	200.00	2.00

** S I M U L A T I O N T R A C E **

図3－22　待行列ダンプ出力例

(vi) 待行列のダンプ

ファイルにストアされているエレメントの属性値をプリントさせる為にサブルーチンQDUMPがある。QDUMPはシミュレーション開始時と終了時にシステムによりコールされるが，ユーザーが必要に応じて途中でコールする事もできる。

図3－22に出力例を示す。

3.1.4 コンバインド・シミュレーション言語SPACEの問題点

今まで，SPACEの概要やその使い方について述べてきたが，もう一度SPACEの特徴について大まかに挙げて見ると，

(i) 離散・連続結合系のシミュレーション言語であること。

(ii) FORTRANをベースとしたサブルーチン群で構成されていること。

(iii) TSS-FORTRANを母体とした会話型である。

以上の三点になるが，これらの特徴はそれぞれに様々な問題点をもたらしている。そこで，上の3点の特徴によってもたらされる問題点の検討をして行く。

(i) 離散・連続両系を結合したコンバインド・タイプシミュレーション言語であるということ。

SPACEは結合系シミュレーターであり，離散系シミュレーション言語のスケジューリング機能と連続系シミュレーション言語の積分機能とを合せもったタイミング・ルーチンにより時刻をコントロールしている。

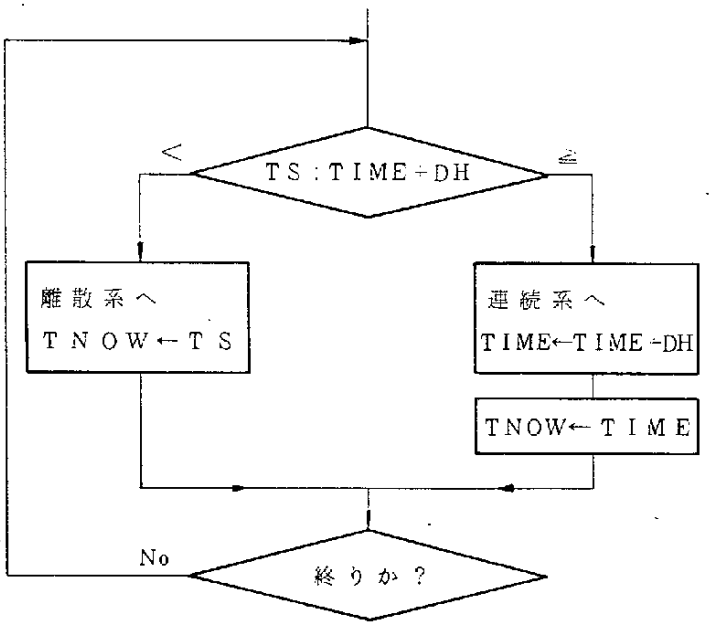


図3－23　連続系と離散系の時間の進め方

図3－23はSPACEのタイミングルーチンの概略を示すものであるが，離散系の時刻をＴＮ－ＯＷ，連続系の時刻をＴＩＭＥとしてこの２つのクロックを平行して走らせている。ここでＴＳはスケジュール表の先頭にあるイベントの発生予定時刻，ＤＨは積分のきざみ幅であり，図からも分る様に連続系の時刻ＴＩＭＥから積分のきざみ幅を加えた時刻（ＴＩＭＥ＋ＤＨ）の間にイベントがスケジュールされているかどうか調べ，スケジュールされていなければ連続のモデルを実行し，連続系の時刻ＴＩＭＥをＤＨだけ進め，その後離散系の時刻ＴＮＯＷもＤＨだけ進める。一方（ＴＩＭＥ）から（ＴＩＭＥ＋ＤＨ）の間に離散系のイベントがスケジュールされている場合，つまり（ＴＩＭＥ＋ＤＨ）がイベントの発生予定時刻ＴＳより大きい場合は，離散系の時刻ＴＮＯＷをＴＳまで進めイベントを発生させ，連続系の時刻ＴＩＭＥはそのままにしておく。

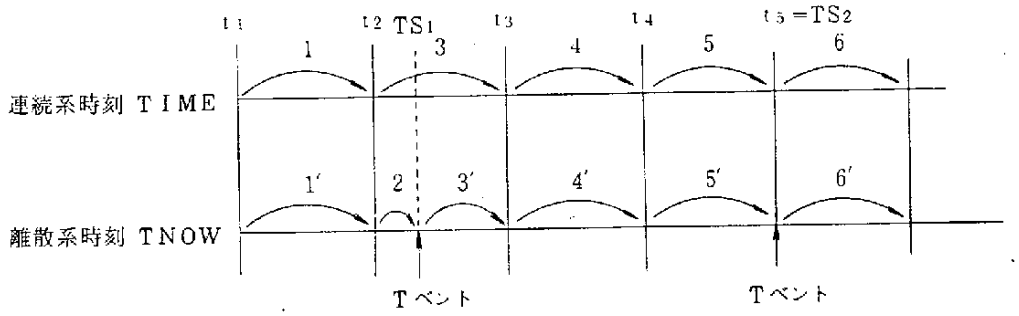


図３－24

図３－24はSPACEの時刻の進め方の例であるが，時刻ＴＳ１，ＴＳ２にイベントがスケジュールされている。連続系の時刻ＴＩＭＥ及び離散系の時刻ＴＮＯＷがＴ１にあったとする。（ＴＩＭＥ＋ＤＨ）がＴＳ１より小さいので連続系のモデルを実行しクロックをＴ２に進める。従ってＴＩＭＥ，ＴＮＯＷともにＴ２となる。次に（ＴＩＭＥ＋ＤＨ）とＴＳ１を比べると（ＴＩＭＥ＋ＤＨ）＝Ｔ３であり，Ｔ２＜ＴＳ１＜Ｔ３であるので，離散系の時刻ＴＮＯＷをＴＳ１としイベントを発生させる。するとＴＳ１はスケジュール表からはずされ，次のイベント発生時刻ＴＳはＴＳ２に更新される。イベントに関する処理が終った後，再びタイミングルーチンにもどり，（ＴＩＭＥ＋ＤＨ）とＴＳ（＝ＴＳ２）を比べると（ＴＩＭＥ＋ＤＨ）の方が小さいので連続系のモデルをＴ２からＴ３に関して実行する。又ＴＳ２にスケジュールされているイベントはＴＳ２＝Ｔ５であるので，Ｔ４からＴ５について連続系の実行を終了した後イベントが発生することとなる。

以上，SPACEにおけるタイミングの取り方について説明したが，ここで次の様な問題点がある。即ち，積分のきざみ幅の間でイベントが発生し，しかもそのイベントが連続系のモデルの状態を参照する様な場合，イベントが発生した時刻の状態ではなく少し前の状態を参照することになるということである。前の先で，ＴＳ１にスケジュールされているイベントが発生した時連続系モデルの状態はＴ２のままであり，離散系のモデルはこれを参照することになる（離散系のモ

デルが連続系の状態に無関係の場合には問題はない。）SPACEでは，積分のきざみ幅を微少なものとしてこの問題を無視しているが，これは次の様な解決方法を考えている。

① イベントの発生時刻の積分のきざみ幅の整数倍にする。

② 積分法を可変増分のものとしイベント発生時刻まで連続系を実行してしまう。

上の2つの解決策によって，離散系のモデルのスケジューリング時刻，連続系モデルの積分によって進められる時刻との関連づけが可能となり，コンバインド・シミュレーション言語におけるタイミングの問題は解決される。しかしながらこれらの解決策によってタイミングの問題を解決すべきなのかどうかが又一つの問題となる。例えば①では離散系の事象は積分のきざみ幅の整数倍の時刻でしか起らないということになるし，②ではシミュレーションの実行時間が非常に長いものとなる可能性がある。

(ii) FORTRANのサブルーチン群によって構成されていることにより起ってくる問題点

SPACEはFORTRANのサブルーチン群によって構成されているシステムであるので，手軽にインプリメントできFORTRANのライブラリーが自由に使え，しかもサブルーチンとしてシステムに追加することにより機能の拡張が容易に行なえる。

しかしながら，FORTRANサブルーチン群であり，プロセッサーを持っていないために次の様な問題点が起ってくる。それはSORTING機能がないということである。ソーティングがないために大規模な連続系モデルを組むのが容易に行なえない。これはシミュレーション・プロセッサーを作ることによってしか解決できず，初めに書いたFORTRANサブルーチン群によって構成されているための長所とは相反する方向となるが，やはりシミュレーション言語としてはプロセッサーを持っていた方が有利であると言えよう。

(iii) 会話型であることに関する問題点

SPACEはTSS-FORTRANによって会話型としてインプリメントされており，シミュレーションをインターアクティブに行なうことができ，この点に関しては何ら問題がない。ただTSS-FORTRANの機能が弱く，QUITをかけてもシミュレーションの途中の様子が全く分らないという欠点がある。

また，インプットすべきデータが多すぎるという欠点がある。これはSPACEの母体となった離散系のシミュレーション言語をそのまま受け継いでいるためであるが，このデータをプログラムの中に組み込んだ場合，パラメータ変更などによって何回かランをさせる時，その都度プログラムをコンパイルしなければならないという無駄を生じてしまうことになる。従ってどのデータをデータとしてインプットさせ，どのデータをプログラムの中に組み込むかという問題は今後検討しなければならない問題であると思う。

インプットデータに関する問題点としてもう一つ，入力するデータを途中で間違えた時再び最初からインプットしなおさなければならないという問題がある。これはTSS-FORTRANの機

能がこの様になっているためであり，この点を改良するにはSPACE専用の会話型プロセッサーの開発を待つ以外にない。

3.2 SIMBOL－Ⅱ

3.2.1 はじめに

3.1でコンバインド・シミュレーション言語の実験システムとしてSPACEを作成したが，その結果コンバインド・シミュレーション言語をFORTRANのサブルーチン群として構成するには限界があり，専用のプロセッサーを作成すべきであることを強く感じた。特にシミュレーションのランに入ってから，インタラクションを取ることが不自由である点，あるいは連続系要素を表現する手段が充分でない点などについては，FORTRANサブルーチン群のシミュレータでは解決できない問題である。

SIMBOL－Ⅱは，プロセスタイプの離散系シミュレーション言語に連続系のプロセスを導入したコンバインド・シミュレーション言語であり，しかも従来SIMBOLが持っている会話システムの機能を受け継いだコミュニケータブルなコンバインド・シミュレーション言語である。

3.2.2 SIMBOL－Ⅱ 設計方針

コンバインドタイプのシミュレーション言語を設計するに当ってベースとなる離散系・連続系のそれぞれの言語をどの様に取り扱うかが一つの大きな問題点となるが，SIMBOL－Ⅱはプロセスタイプ離散系シミュレーション言語SIMBOLをベースとし，これに連続系シミュレーション用の機能を追加する形で設計した。

SIMBOLはSIMULA同様プロセスタイプの言語であり，離散系のシミュレーション・モデルをプロセスという概念を用いて表現する。

このプロセスという概念については，先のSIMBOLの報告書に詳しく紹介してあるので，ここでの説明は省くが，一般にシミュレーション・モデルを作る際プロセスという概念が有効であることが認められつつある。そこでコンバインド・シミュレーション言語SIMBOL－Ⅱでは，SIMBOLの離散系プロセスに連続系のプロセスという概念 ── 次節にて説明 ── を導入し，プロセスタイプのシミュレーション言語という基本的思想をそのまま生かす形で設計することにした。

この意味では，Case Werter Reserve Universityで開発したGSLと似かよったシステムである。しかし，SIMBOL－ⅡはTSSシステムのもとで会話型利用が出来る点で特徴がある。近年，特にシミュレーションシステムに於ける会話型利用の有効性が認められ，システム開発もさかんに成って来た。ことに連続系シミュレーション言語に於てはグラフィックディスプレイを使ったCSSL－Ⅲの会話型システム，CSMP－Ⅲの会話型システム等のかなり実用的なシステムが開発

され，ある意味では実用段階に入っているとも言える。一方離散系シミュレーションの会話型システムは，ごく最近になって開発が進んで来てはいるが，連続系シミュレーションシステムに較べると，一歩立ち遅れている。これは離散系言語の会話型利用にメリットが無いと思われている為ではなく，むしろインプリメントに関連した技術的な問題の方が大きな理由ではないかと思われる。連続系言語に較べて離散系言語の方がより大きなメモリースペースも必要とするし，一般にＣＰＵタイムもかかりがちである。この様な点が会話型にした場合不都合が生じやすい為である。

しかし，どちらの言語にしろ会話型でシミュレーションの出来る事自体のメリットに対してはかなり昔しから有用性が認められている。この意味でコンパインド言語に於ても会話型である事のメリットは大きいと考えている。

3.2.3　ＳＩＭＢＯＬ－Ⅱの新しい機能

(1)　プロセスについて

プロセスの具体的な取扱に関しては，今までＳＩＭＢＯＬで用いていたプロセス，即ちＳＩＭＢＯＬ－Ⅱの離散系プロセスとほとんど同じである。例えば，アクティビティと言うＦＯＲＴＲＡＮのサブルーチンやＡＬＧＯＬのプロセデュアーと似た形のプログラム単位でプロセスを定義し，アクティビティ名の前にＮＥＷと言うオペレータを付けた項を算定するとプロセスが発生し，プロセスには個有のローカルデータを持って，プロセスで構成されるセットやキューと言ったグループの取扱いが出来るなど連続系プロセスでも同じ様に通用する。従って，ここでアクティビティと呼んでいるものとプロセスの関係は従来のＳＩＭＢＯＬの場合と同じである。唯，新しく連続系のプロセスを導入した為に離散系プロセスと連続系プロセスを定義するアクティビティをそれぞれにして区別したいので，前者にはデスクリートアクティビティ，後者にはコンティニュアスアクティビティを対応付けている。

デスクリートアクティビティは，結局今まで単にアクティビティと呼んでいたものとまったく同じなのでこの内容については45年度の報告書に譲る事にして，次にコンティニュアスアクティビティについて説明する。

従来，ＳＩＭＢＯＬで扱っていたプロセスと言う概念は離散的に変化する様子を記述するには適していたが，連続的な変化を記述するには不向きであった。一般に連続系シミュレーションに於ては，この様な連続的な変化を微分方程式や積分方程式などで記述して，シミュレーションを行なっている。ＳＩＭＢＯＬ－Ⅱで新たに導入した連続系プロセスも，通常の連続系シミュレーション言語の様に連続的な変化の様子を微分方程式を主体に記述する。この点では，連続系プロセスと通常の連続系シミュレーション言語，例えばＣＳＭＰなどと類似している。

唯，記述の仕方は似ていても，その振いはＣＳＭＰ－Ⅲのダイナミックレジョンとは，かなり違っている。

(2) コンティニュアスアクティビティの構造

コンティニュアスアクティビティは図3－25に示す通り，4つのセクションから構成される。

1. デフィニションセクション（definition section）
   これはローカルデータの定義をする部分
2. イニシャルセクション（initial section）
   このセクションは，プロセスが発生した時に1回だけ実行される部分で，このプロセスの初期設定をする為に用いる事が出来る。
3. ダイナミックセクション（dynamic section）
   このセクションは，プロセスの定義でも一番中核になる部分で，系の変化の様子を表わす。
4. コミニュケーションセクション（communication section）
   このセクションでは自分自身のプロセスの状態を調べたり，他のプロセスに対してアクションを取ったりする事が出来る。この部分はダイナミックセクションと異なりプロセデュラルな記述が出来る。

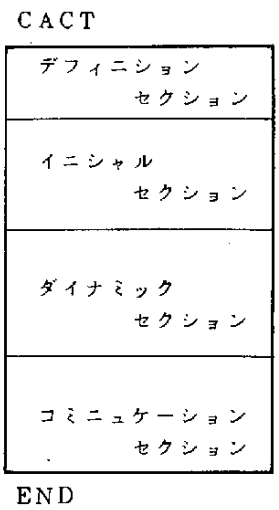


図3－25　コンティニュアス
アクティビティの構造

以上のセクションについて，もう少し詳しく説明していく事にする。

1. デフィニションセクション

このセクションは特別にこれを区別する為のステートメントがあるわけでなく宣言用のステートメントがあれば，これをまとめて1つのセクションと呼んでいると言うだけである。ここで宣言された変数はこのアクティビティから発生したプロセスのローカルデータとなる変数で，シミュレーションの実行が開始されてからダイナミックにエリアが割付けられる。言わば，その時のパターンを定義している様なものである。

```
CACT  INGOT
REAL  OTEMP, W
REAL  SIZE
  .
  .
  .
  .
  .
END
```

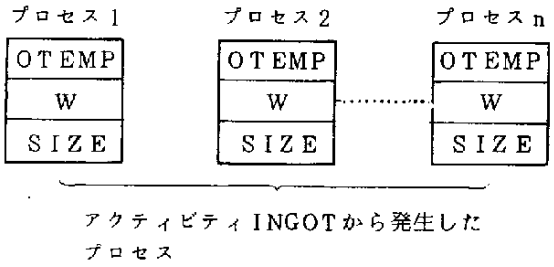


図3－26　プロセスのローカルデータとアクティビティの関係

図3－26は，ローカルデータの定義とプロセスが発生した時に割当てられるエリアとの関係を示したものである。この例でも解る様に同じ名前を持った変数が複数個出来てしまう。従って，それぞれを区別して参照する時にはプロセス名でクォリファイする。この点については，プロセス間コミニュケーションの所で説明する事にする。

2. イニシャルセクション

イニシャルセクションは文字通りプロセス実行開始時に，ローカルデータなどの初期設定を行なう為に設けたセクションである。このセクションを設けるに当っては，プロセスの初期設定を発生する側で発生前に値を決め，発生時に設定してやる方法も考えたが，かえって繁雑になりそうなので，発生された側でまず初期設定をし，それから実行に入る方法にした。このセクションは次に説明する，ダイナミックセクション，コミニュケーションセクションと異って，プロセスが発生した時に，ただ一度だけしか実行されない。

3. ダイナミックセクション

このセクションはCSMP-Ⅲで言えば，ダイナミックレジョンに相当する。ダイナミックセクションはプロセスを定義する一番重要な部分であり，SIMBOL-Ⅱでは連続系プロセス用のファンクションを用いてシステムの連続的要素をモデル化する。これは通常の連続系シミュレーション言語に於ても同じであるが，この部分では，微分方程式などの関係式を定義する。又，ダイナミックセクションの特徴の一つは，SIMBOL-Ⅱのモデルの他の部分がプロセデュラルに記述されるのに対して，この部分だけノンプロセデュラル（non procedural）に記述されている事である。

今までSIMBOLで使っていたステートメントは全てプロセデュラルな処理をする為のものであったので，上記の様な関係式を定義する為，新しくEQUステートメントを導入した。このステートメントを使って，

$y = \int x\,dt$　などの関係を定義するには，

```
EQU  Y = INTG ( X, IC )
```

の様に書く。

EQUステートメントには単に積分だけでなく，一般に連続系シミュレーションで用意している函数，例えば1次遅れ，2次遅れ，時間遅れ，ファンクションスイッチなども書ける様になっている。

ダイナミックセクションでは前述した様に，ノン・プロセデュラルな方法で記述をしているが，この意味は，ステートメントの出現順がユーザ側から見て意味を持たない事でもある。ところが実際にシステム側がダイナミックセクションを実行する時にはステートメントの順番が大事な意味を持っている。これはコンパインドシミュレータの問題と言うわけではなく一般の連続系シミュレーション言語でステートメントのソーティング（SORTING）問題として解決

されているものである。そもそもダイナミックセクションで定義している式に使われている変数の変化は全てがパラレルにしかも同時に起るべき所を現実には1ステートメントずつ処理しなくてはならない為に，その順番をうまく考えて，あたかも全体の変化がパラレルに起っている様にしようと言うものである。連続系シミュレーションに於けるステートメントのソーティング機能については，実験システムSPACEが，この機能を持っておらず不便である点を指摘しておいたが，もちろんSIMBOL－Ⅱではこの点を考慮してソーティングが出来る様に設計している。ただSIMBOL－Ⅱは会話型システムである為にバッチ処理言語，例えばCSMP－Ⅲで行なっている様に，間にSORTとかNOSORTと言ったステートメントを入れてソーティングの指示を与えたりはせず，コマンドで行なう様にしている。

4. コミニュケーションセクション

このセクションを設けた事の目的は，他プロセスとのコミニュケーションを行なう事にある。これはダイナミックセクションと異り，プロセデュラルな記述を行なう部分である。この部分では今まで離散系プロセスで使えた機能のほとんどが同じ様に使える。

コミニュケーションセクションをダイナミックセクションと分離して独立のセクションとしたのは，インプリメント上の問題と密接に関係がある。連続系シミュレーションでは積分器を中心にシミュレーションが進行するが，積分方法に依って異なるが通常，1ステップ進めるのに複数回ダイナミックセクションをスキャンしなくてはならない。その為にもしこのセクション中にプロセデュラルな記述を許すと，複数回これが実行され，不都合が起ってしまう。そこでこの様な部分を分離し，別のセクションにして，1ステップの終了時に1度だけ実行する部分を設けたわけである。

(3) プロセス間コミニュケーション

SIMBOL－Ⅱに於てはシミュレーションモデルを組み立る場合，対象とするシステムをプロセスと言う単位に分割して記述した上で，あらためてプロセス間でお互に関連付けを行なう方法を取っている。

プロセス間コミニュケーションと言うのはプロセスとプロセスの間でお互に関連付ける手段の事を言っている。即ち，1つはプロセスのローカルデータをアクセスする事，もう1つはプロセスを操作する事の2つに大別出来る。第2の表現はやや抽象的であるが，具体的に言うと①プロセスを発生したり消去したりする，②セットやキューに入れたり出したりする，③プロセスの実行をコントロールする，が主な事がらである。

① ローカルデータのアクセス

プロセスのローカルデータは，それが連続系プロセスであるか離散系プロセスであるかにかかわらずどちらのプロセスからもアクセスする事が出来る。前にも一寸触れたがローカルデータの変数は同じアクティビティから発生したプロセスでは全て同じなので，これらを区別する

為にプロセスを指示する名前を前に付けて参照する形態をとる。これをリモートリファレンス（remoto referrence）と呼んでいる。ただ，プロセスが自分にローカルなデータを参照する時に限って，変数の名前だけで参照する事が出来る。

プロセスの名前についてはSIMBOLの時と扱いが変っていないので，詳しくは45年度報告書を参照されたい。一つの方法としてはエレメント変数にプロセスを示すポインターを入れておき，エレメント変数名でプロセスを示す方法がある。

そこでエレメント変数Eの示すプロセスがAと言うアクティビティから発生したもので，そのローカルデータXを参照するには，リモートリファレンス

E：A.X　の形で行なう事になる。

② プロセスの発生と消去

プロセスは他のプロセスに依って発生させられる。プロセスを発生したプロセスは，と逆にたどっていくと一番始めに発生したプロセスは，誰か別のものに発生されたのでなくてはつじつまが合わない。SIMBOL-IIにもSIMBOLと同じくメインプログラムが必要で，これがプロセスを始めに発生する。

プロセスの実行がENDステートメントまでたどりつくと自然に消滅するが他のプロセスを消去する事も出来る。プロセスを消去するにはDESTROY ステートメントを用いるが，一寸注意が必要で，次に説明するセットやキューのメンバーになっているプロセスを消去する事は出来ない。

③ セットやキューに関連した扱い

セットやキューと言うのはプロセスから成る順序の付いた集合を扱う手段で，セットとキューの相異は統計データのカウントをしてくれるか否かだけである。今までの連続系シミュレーション言語では，この様な扱いはまるっきり出来なかった。しかし，プロセスのグループを扱える事には色々とメリットがあると思われる。

④ プロセス実行のコントロール

これは離散系シミュレーションでスケジューリングと呼んでいる問題である。他のプロセスの実行時間を決めたり，実行を取り消させたりなどプロセス間コミニュケーションの中でも一番直接的に働きかける手段である。

3.2.4 連続系プロセス導入による問題点

連続系プロセスに関連した問題点はいくつかあるが，一番大きな問題はスケジューリングの問題である。この問題は，中に2つの問題があって離散系プロセスと連続系プロセスが混在する事，即ちコンバインドシミュレータとしての基本的な問題と連続系プロセスが同時に複数個存在し得る事によって引き起される問題である。

第1の問題点については，実験システムSPACEの問題点としてすでに指摘してある。この問題は離散系シミュレーションと連続系シミュレーションではシミュレーション時間進行の管理が本質的に異っている点に起因している。つまり離散系シミュレーションでは生起するイベントの生起予定時刻に合せてシミュレーションクロックを動して行くのであるが，イベントを実行しても時間の経過が無い。一方の連続系シミュレーションでは，時間の進行が積分器を中心に行なわれ，1ステップの実行は，即ち積分ステップサイズだけ時間の進行をともなっている。つまりプロセスを実行すると時間が進んでしまう。そこで積分ステップサイズの間で起るイベントを問題にしなくてはならない。

第2の問題についても，連続系プロセスが積分器を中心に時間が進められる為に問題が起る。

これらの問題はともにプロセスの実行時における取り扱いにより発生するものであり，ここで問題点について説明する前に両プロセス実行の様子を述べておく必要がある。

まず個々にプロセスの実行を考える。離散系プロセスでは実行の論理的な流れを考えると図3－27に示す通りで，ある一時刻での変化を記述する一群のステートメントと，間に時間経過を示すDELAYだとかWAIT などが入った形とみなせる。時間経過が無い部分をイベントと呼んでいる。従ってプロセスはイベントとイベント間の時間経過を示す表現とから成っている。同時に幾つものプロセスが存在しているとそれぞれのプロセスに含まれているイベントの生起順序に従ってプロセス間を順次移行しながら全体の実行が進められる。その様子を概念的に示すと次の図3－28のごとくなる。

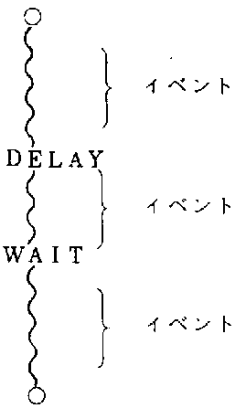


図3－27　離散系プロセスの構造

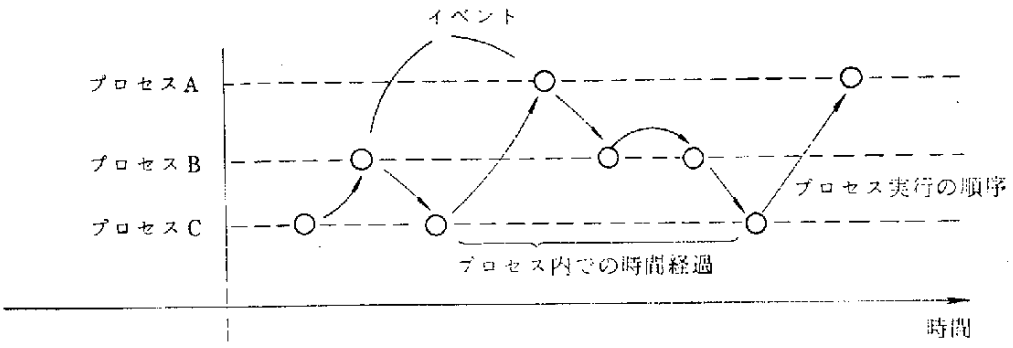


図3－28　離散系プロセス実行の様子

次に連続系プロセスの実行を考えてみる。図3－29は連続系プロセス実行の様子を説明する為のもので，この図からもわかる様にダイナミックセクションがシステム側から複数回呼び出された後

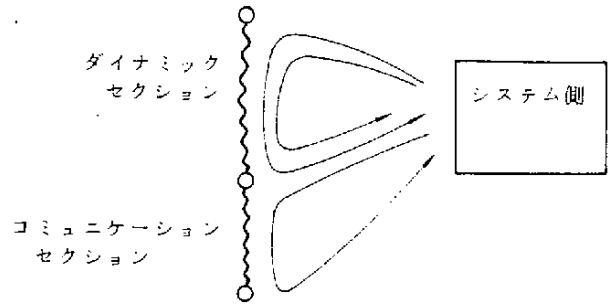


図3－29　連続系プロセス実行の様子

コミニュケーションセクションが実行され，1回の実行を終了する。

ダイナミックセクションの呼び出し回数は積分法によって決り，ＲＫＧ法では4回呼び出される。

こうしてダイナミックセクションとコミニュケーションセクションの実行が終ると積分間隔時間 ΔT だけ先のステータスが得られる。

離散系プロセスの場合と同じ様に，複数個の連続系プロセスの実行の様子を示したのが図3－30である。

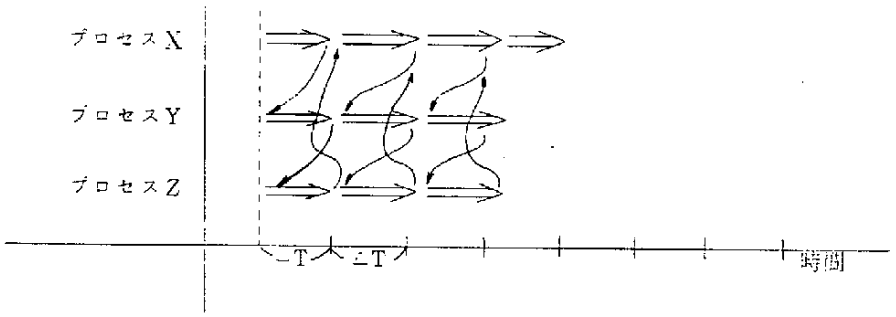


図3－30　複数の連続系プロセスの実行

この図ではどのプロセスも同じ積分間隔を持っているものとしてある。

これらプロセス実行上の相異は離散系プロセスの場合プロセスを実行してもそれ自身で時間の経過がないのに対して，連続系プロセスの場合には積分間隔 ΔT だけ時間が進んでしまう点にある。これはコンバインドシミュレータとして本質的な意味を持っている。

(1)　離散系プロセスと連続系プロセスが混在することによって起る問題点

実際には両タイプのプロセスが複数個づつ存在するはずであるが，この問題を扱うについては，連続系プロセスと離散系プロセスが各々1つの場合を考えれば充分である。今，2つのプロセスが実行される様子を図に表わしてみる。（図3－31参照）

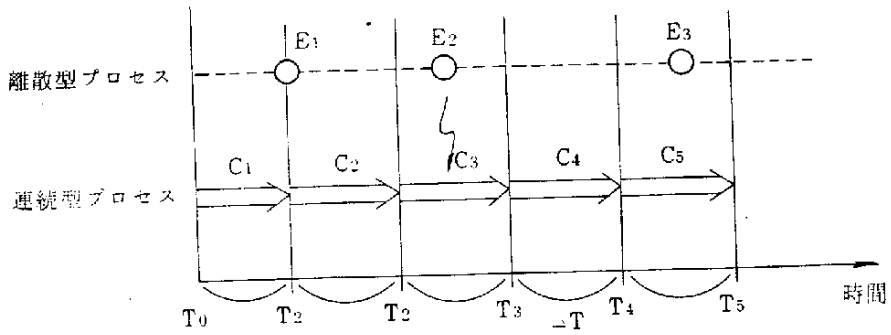


図3−31　両型プロセスを実行する時の様子

実は，2つのプロセスを単に時間の順に従ってうまく実行するだけならさして問題は起らず，現に実験システムと同じ方法で行なえば済む事である。即ち図の場合には，離散系プロセスのイベントE_1，E_2，E_3などと連続系プロセスの実行サイクルC_1，C_2，C_3，……等について，連続系プロセスの方では実行サイクルの終了時刻とイベントの生起時刻とを比較しながら早く起る方を実行して行けば良い。

問題は，離散系プロセスと連続系プロセスとがお互にコミニュケートする場合にある。例えば図中でイベントE_2が連続系プロセスのステータスを参照しようとする時，C_3の途中であるから本来なら，丁度この時点での連続系プロセスのステータスを知りたいにもかかわらず連続系プロセスの実行がΔT間隔でしか行なわれておらず，結局C_2終了時でのステータスしか見る事が出来ない。この問題は実験システムでも指摘しておいたが，SIMBOL−Ⅱの場合単にステータスを参照するだけではなく，相手プロセスの実行を中断したり再開したりなど色々とプロセス間の関係を入りくんで扱う事が出来る為に事情はSPACEの場合より複雑である。

(2)　連続系プロセスの同時現象の問題点

もう一つの問題はコンバインドシミュレータだから起ると言った問題ではなく，むしろ複数個の連続系プロセスが同時に実行される為にこれらプロセスの間で起る問題である。前の図3−31を参照するとこの図からほぼ明らかであるが，連続系プロセスを順次実行して行く間で時間のズレが生じている。つまり実行を終了してしまったプロセスは基準時刻TからΔTだけ進んだ時刻の状態に成っているにもかかわらず，まだ実行を終了していないプロセスは依然，時刻Tの状態のままである。この様に一時的な時間のズレは，プロセス間でお互に他のものと無関係に進行している場合，特に問題とならないが各プロセスのコミニュケーションセクションで他のプロセスのローカルデータをアクセスしたり，実行をコントロールしたりしている場合には，ΔTだけ時間を進める間でプロセスの実行順序によってはまるっきり異った結果を生んでしまいかねない。離散系シミュレーションに於ては同時現象の問題が指摘されている。この問題は，まったく同時刻に起るイベントの実行順序にかかわるものであり，ごく消極的な解決策としてイベントにプラ

イオリティを与える方法が一般に行なわれている。連続系プロセス間での実行順序の問題は，離散系シミュレーションの同時現象の問題と良く似ていて，連続系シミュレーションに於ける同時現象の問題と言う事が出来る。この問題はCSMP－Ⅲの様にダイナミックセグメントを１つしか許していない言語では発生しなかった。ＳＩＭＢＯＬ－Ⅱ流に言えばダイナミックセグメントはプロセスであり唯一つしかプロセスが存在しない為にプロセス間の同時現象が問題とされなかったのである。しかし，ＣＳＳＬ－Ⅲでは１つのダイナミックレジョンに複数個のデリバティブを書く事が出来る。デリバティブはＳＩＭＢＯＬ－Ⅱの連続系プロセスに相当するもので，ＣＳＳＬ－Ⅲでは結局複数のプロセスが同時に存在する事が出来るわけでＳＩＭＢＯＬ－Ⅱの場合と事情は同じである。従ってＣＳＳＬ－ⅢにはＳＩＭＢＯＬ－Ⅱと同じく同時現象の問題があると思われる。唯，この言語ではＳＩＭＢＯＬ－Ⅱほどプロセス間（ＣＳＳＬ－Ⅲではデリバティブ間）の相互関係が複雑に持てない為，この問題の影響がいくぶん緩和されている様である。

しかし，いずれにしても連続系シミュレーションの同時現象の問題は，言語のプログラム構造が増々複雑なものまで扱える方向に向っている事からも，より大きくなって行く事はまちがいなさそうである。

4．道路交通シミュレーション

第4章　道路交通シミュレーション

自動車の交通手段としての急激な進歩と，自動車交通の需要の増加は，近年多くの都市に交通渋滞，騒音，排気ガス公害など多くの都市問題を引き起こしている。死亡事故の増加は，自動車の過密状態とそれに伴うドライバーのストレス或は注意力の欠如などがもたらした結果であるとさえ言われている。

自動車に取って代わるべき，新しい交通手段としてcity carなど様々なものが研究されているが，輸送効率，安全性という面で，自動車に匹敵する程，決定的なものと成り得ない現状では，道路の建設・道路網の整備が唯一の解決手段ではないだろうかと考えられている。しかしながら，新らしい道路の建設はその建設費用の面や，建設計画立案から工事完了に至るまでに長期間を必要とすること，或は又道路建設が自然破壊の原因になる等の理由により問題解決の最善策とはなり得ないのである。

それでは，交通問題に対処するにはどの様な方策を取れば良いのであろうか？それは道路の建設拡充を行なう前に，現在ある道路網の through put を最大にする様な信号制御方式をさぐり出すことである。ところが大都市の信号制御方式は新しいパラメータをつけ加えることにより，柔軟性に富んだものとなり，それとともに制御方式も複雑化し単純な思考では，道路の through put を最大にすることができなくなってきている。

この様な複雑化した交通制御システムの解析，評価，最適化に対して，シミュレーションは大いにその力を発揮する手法であると言える。

本章では道路交通問題に関するシミュレーションを取り扱う訳であるが，これはあくまでも，Combined Type Simulation Languageの応用例であり，交通問題を解決するには，とうてい及ばない程度のモデルしか，用意できなかったが，交通現象の解析という点で，多少なりとも役立つことを期待している。

4.1　道路交通問題の解析方法とモデル

道路交通システムを解析するに当っては，解析の対象となるシステムをできるだけ忠実に表現（近似）しているモデルを作らなければならない。しかもそのモデルは，解析方法に合致したものでなければならない。

現在広く用いられている交通問題の解析としては，純粋に数学的な解析法或はAnalog及びDigital Computerによるシミュレーションとがある。

(1) 数学的解析法

道路における交通流を，個々の車輌についてその特性を捉えた微視的モデル，車群として捉える巨視モデルに対し，前者には自動制御理論，後者には流体力学或は気体運動力学を応用し，解析して行く。

数学的解析法には，問題の対象となるシステムの構造により control parameter を計算しなければならないという厄介な面があり，システムが飽和状態に到した時，理論的な関係が成り立たなくなってしまうという欠点がある。従って交通流の理論的解析にはある種の限界があると言わざるを得ない。

(2) アナログ・コンピュータによるシミュレーション

アナログ・コンピュータによるシミュレーションでは，交通流を流体としてとらえ，システムを微分方程式により表現し解析する方法もあるが，これらの多くはアナログ・コンピュータではなくCSSLによって解かれることが多い。

道路交通問題をアナログ・コンピュータを使って解いた例は少ないが，パルス信号により車輌を表現した例は興味深いものであるので，アナログ・コンピュータによる道路交通シミュレーションの代表例として挙げておく。

図4－1は，シミュレーションの対象となるシステムである。これは単一交差点であるが，各道路は2車線あるので8つの input section と8つの output section を持っている。図4－2はシミュレーション・ブロック・チャートであるが，RANDOM GENERATOR 'R' により発生したパルス ── 一つのパルスは一つの車輌を表わし，ポアソン分布に従って発生される ── 信号待行列に到達するまで，時間的な遅れ 'S' を経た後，待行列の最後部に並ぶ。信号は待行列に並んだ車輌と対しサービスする窓口であり，QUEUE EXTERNAL CONTROL によって制御されている。

この様なアナログ・コンピュータによるシミュレーションでは，アナログ・コンピュータの parallel computation により 1／30 程度の実時間で実行でき，又man－machine的性格を生かし，シミュレーションを行なうことができるという特徴をもっているが，アナログ・コンピュータに，AND，OR，NOT他の論理演算機能をつけた専用の hardware を開発しなければならず，非常にコストがかかるという点がネックとなっている。

(3) 汎用シミュレーターによるシミュレーション

交通問題をディジタル・コンピュータによって解析する場合，GPSS或はSIMSCRIPT などの汎用シミュレータが用いられる。汎用シミュレータを用いてシミュレーションを行なう時，そのシミュレータのworld－viewに従ってシステムをモデル化しなければならない。GPSSによるトラフィック・シミュレーションでは車輌をXact，信号をFacility或はLogic Switch，又道路をいくつかのStorage或はFacilityに対応させることによりモデル化する。SIMS－

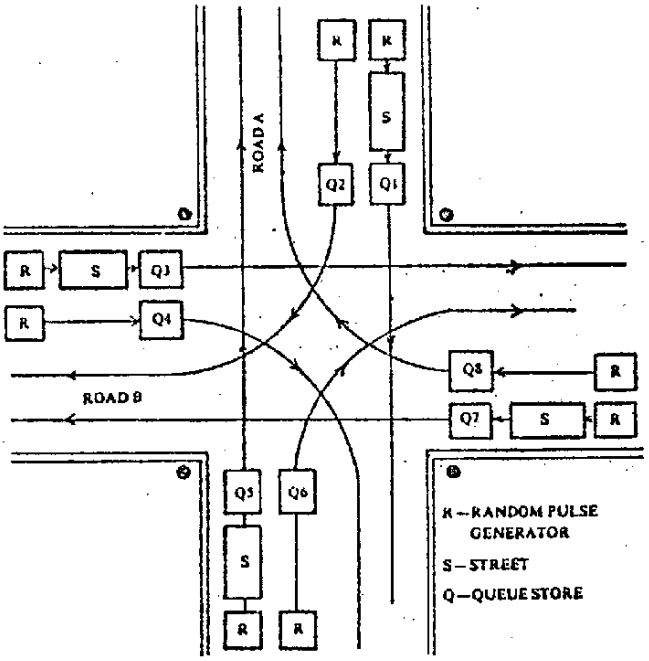


図4－1　Traffic interaction at an isolated intersection

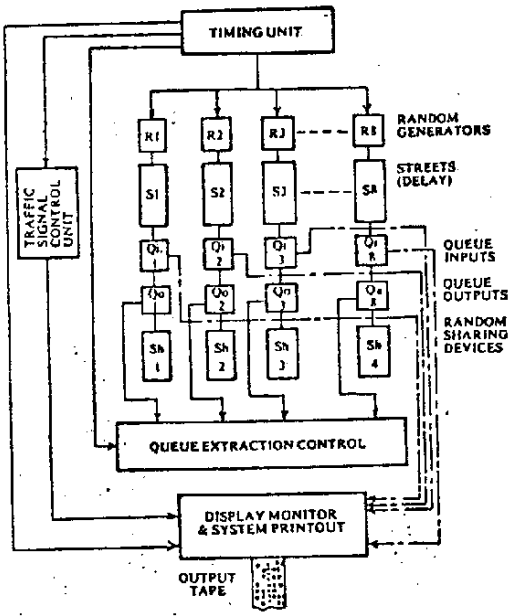


図4－2　Schematic of basic simulator for a single intersection

図4－1，4－2はE. T. POWNER他による"Road traffic simulation employing a hardware approach philosophy and design considerations"より一部掲載。

CRIPTでは信号をPermanent Entity，車輌をTemporary Entityと見なし，到着イベント，待行列に並ぶイベント，信号によりサービスを受けるというイベントを発生させることによりシミュレーションを行なっていくが，SIMSCRIPTに比べGPSSの方が，トラフィックという概念に適しているため，SIMSCRIPTによるトラフィック・シミュレーションはあまり行なわれていない。

図4－2はGPSSによる一交差点のシミュレーション例であるが，各方向の道路は各々2レーンを有し，1つのレーンはいくつかのFacilityより出きている。各道路でGENERATEされたXact（車輌）は，Parameterの値によって ── 右折・左折或は直進かによって ── 2つのレーンのうちのどちらかのQUEUEにつく。QUEUEの最前線はFacilityになっており，このFacility内にあるXactは信号が赤であるか青であるかによって ── 信号FacilityのSNjの値によって ── そのまま待っていたり或は交差点内のFacilityをその進行方向によって次々に使用して行くのである。これらのシミュレーション結果はQUEUEに関する統計，あるいはFacilityに関する統計という形で出力される。

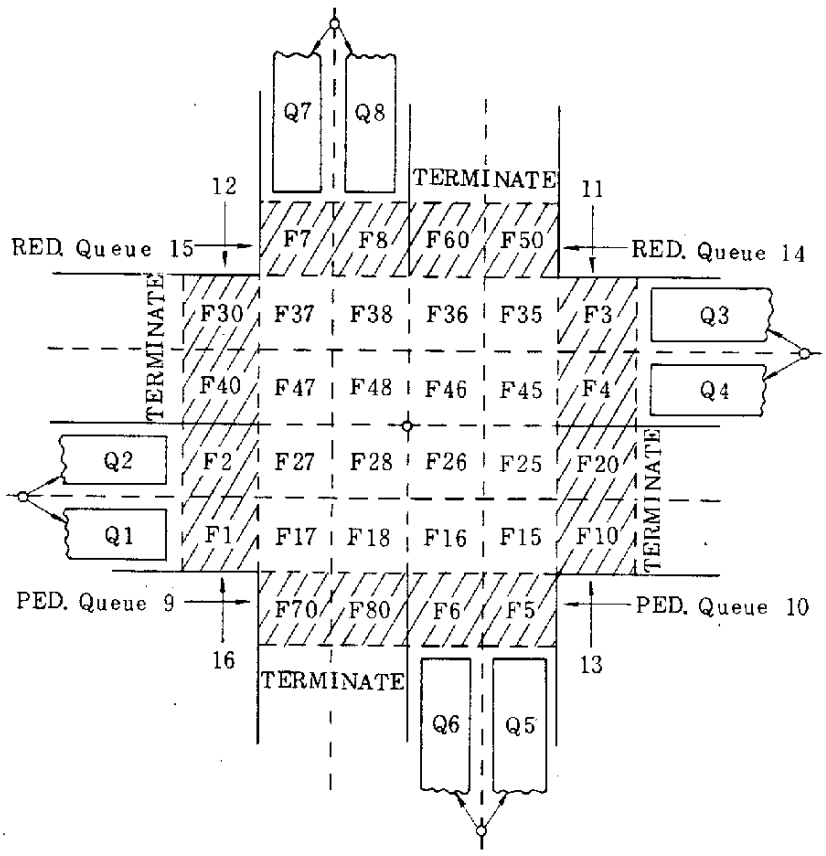


図4－3

4.2　トラフィック・シミュレーションの実行

シミュレーションは，システム・エンジニアリングでのシステム評価の一手段である。従って，シミュレーションの実行は，対象となるシステムの詳細な調査，理論的裏付け，評価基準に沿ったシステムのモデル化を必要としている。図４－４は一般的なシミュレーションの手順を示したものであり，この図に沿ってトラフィック・シミュレーションの概略を示す。

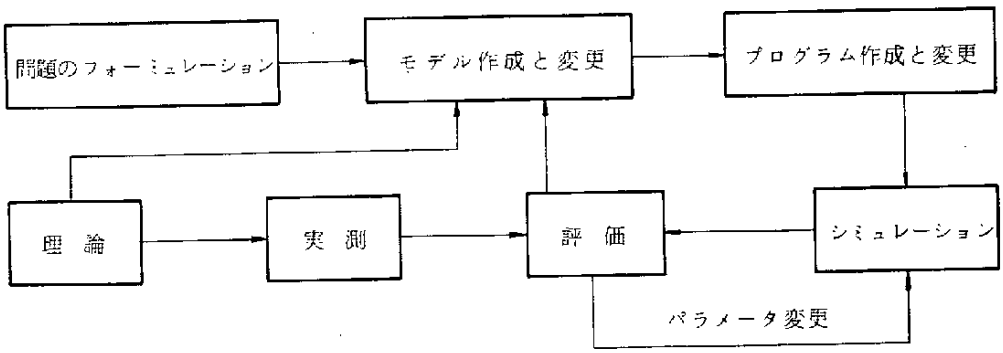


図４－４　一般的なシミュレーションの手順

4.2.1　問題のフォーミュレーション

シミュレーションにより一般的な問題を解くことは困難である。シミュレーションを行なう前にその目的を明確に限定し，評価基準を設定しこれに沿ってシステムの最適化を計ることがシミュレーションの目指すところである。

トラフィック・シミュレーションの評価基準としては次の様なものを代表例として挙げることができる。

(1)　交差点での待行列　（最小化）

(2)　旅行時間　（最小化）

(3)　道路専有率　（最大化）

4.2.2　モデル作成

シミュレーションは本来，試行錯誤的な手法である。従って，シミュレーション・モデルは計算結果からのフィードバックとしてのパラメータ変更，モデルの手直しなどを容易に受け入れしファインできる柔軟性に富んだものでなければならない。

トラフィック・シミュレーションにおけるモデル化は具体的には，交通流をどの様に捉えるかによって次の二つに大別できる。

(1)　微視的モデル

交通流を構成している粒子，即ち個々の車輌に注目して，モデルを作成する。微視的モデルは，

ハイウェイや一般郊外道路などの様に車輌の相互干渉が，問題となる場合のシミュレーションに用いられる。

交通流を微視的にとらえモデル化する時，次の様な要素をモデルに組み込むと良い。

(i) 個々の車に付随する特性 ： スピード，車長，最大加速度，信号に対するドライバーの反応時間。

(ii) 各車輌に共通の特性 ： 走行方程式，ドライバーの意志決定の方法，追越し，車線変更論理。

(iii) システムの特性 ： 信号パラメータ，歩行者の取扱い，システム内での車の出入。

微視的なトラフィック・シミュレーションは，GPSS などを用い広く行なわれているが，個々の車輌を追跡しているためどうしても広いエリアをシミュレーションの対象にできないという欠点がある。

(2) 巨視的モデル

規模の大きい都市交通網を，マクロ的に捉え信号の制御方式の最適化をねらうシミュレーションでは，交通流を近似的にしか捉えることができないが，巨視的なモデルを用いなければならない。

巨視的なモデルの例としては，交通流を車群として取り扱うタイプ，或は流体として取り扱うタイプのものがある。

(i) 車群を取り扱うトラフィック・シミュレーション

都市における交通では交通流の密度が比較的高く，信号などの影響により車が一団となって動いていく。この様な状況のシステムをシミュレートするには個々の車輌を取り扱うより，車群として交通流を捉えるのが有利である。

車群を取り扱うシミュレーションの例として道路を適当な長さにブロック化したシミュレーションについて説明する。

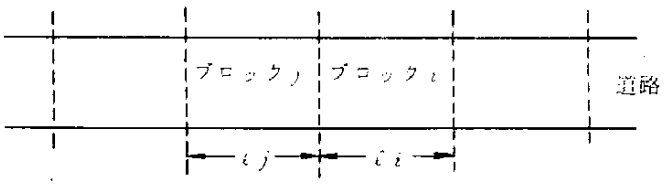


図 4−5　道路のブロック化

図 4−5 において，ブロック i の長さを ℓ_i，ブロック i に入ることのできる車輌の最大数を Q_i，ある時刻においてブロック i に存在する車の数を q_i，許容最大速度を V_i，さらにブロック i における車の平均速度を v_i とする。v_i は次の式により求まる。

$$v_i = V_i \; v\,(\,q_i / Q_i\,) \quad \cdots\cdots (1)$$

ただし，$v(q/Q)$は図4−6の様な関数である。

又，ブロックiからブロックjに移動する車の数Δq_{ij}は次式より求まる。

$$\Delta q_{ij} = \frac{1}{2}\left(V_i\, v(q_i/Q_i) + V_j\, v(q_j/Q_j)\right)\Delta t \cdot q_i/Q_i \quad \cdots\cdots\cdots\cdots \quad (2)$$

(2)式は次の様に考えることにより導かれる。つまり，移動する車の速度はブロックiとブロックjにある車の速度であり，q_iはブロックiの長さl_iに等しく分配されているのである。（ただし，$q_{ij} > q_i$のときは$q_{ij} = q_i$とする）

以上の移動公式をシミュレーション・インターバルΔtの間に各ブロックに適応し，車の移動によるシステムの各ブロック内の車の数の変化を，イテレーティブに計算して行く。

車群を取り扱うタイプのトラフィック・シミュレーションでは，近似的ではあるが，中規模程度の交通システムを取り扱え，しかも車群を連続的に追跡できることが利点である。

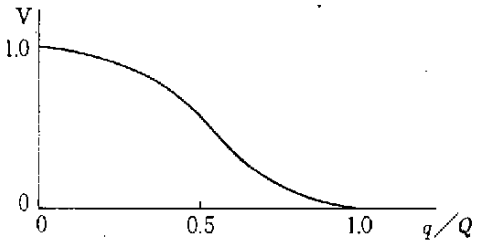


図4−6　密度 ── 速度曲線

(ii) 交通流を流体として捉えるトラフィック・シミュレーション

大規模な道路網の信号制御方式の評価を目的とするシミュレーションでは，非圧縮性流体の理論を交通流に適応したモデルが有利であるとされている。本論文のメイン・テーマである，Combined Type Simulatorの応用例としてのトラフィック・シミュレーションが流体モデルによるものであるので，流体モデルについては次節で詳しく紹介することにし，交通現象のモデル化ということについて総体的な意見を述べることにする。

道路交通システムに現われる諸々の現象は，個々の車によってひき起こされる現象と個々の車が集まり，その集団によってもたらされる現象とを含んだものである。この様なシステムをmacroなモデル，或はmicroなモデルにより捉える時，前者では個々の車の動きが曖昧なものとなり，後者では全体像が不明瞭なものとなるきらいがある。では，トラフィック・シミュレーションはどの様にあるべきものなのであろうか？トラフィック・シミュレーションの最終目的は，信号の最適制御方式を探り出すことにあり，その意味では，どうしてもマクロなシミュレーションが必要であろう。しかしこのマクロ・モデルにおける不明確な点 ── 例えば右折車と対向車の関係，左折車と歩行者の関係或は発進遅れの表現等 ── をモデル化したミクロなモデルを用意し，そのシミュレーション結果をマクロなモデルにフィードバックするというシミュレーション方式がトラフィック・シミュレーションにおける最も真摯な方法ではないだろうか。

もう一つ，交通現象のモデル化についての意見を書いておく。それは交通現象は，Discrete な現象と continuous な現象を内含しているということである。例えば信号の切り換わり，或は車の到着や車の台数などはDiscrete な数であり，車の速度，位置或はシステム内の車の密度といったものは continuous な量である。continuous なものと discrete なものとを包含したシステムをモデル化するには，従来の連続系シミュレーター或は離散系のシミュレーターでは cover しきれない現象を無理やりモデル化している傾向がある。従ってトラフィック問題を取り扱うには離散，連続両用のシミュレーターが必要なのではないだろうか？

4.2.3 シミュレーション・プログラムの作成及び実行

モデル化されたトラフィック・プログラムは，所定のシミュレーション・プログラミング言語によりプログラムされインプリメンテーションされる訳であるが，ここでは多くを語る必要性はないと思う。

ただ，シミュレーションは，最適化を計る手法であり，その意味で，各パラメータの変更やモデルの変更が容易である様なプログラムをつくることが望ましい。ＴＳＳなどを用いシミュレーションをOn-Line 化すること，或はGraphic Display を用い，パラメータの影響が容易に把握できる様なシミュレーションは，シミュレーションの best feature であろう。

4.3 結合系シミュレーション言語SPACEによるトラフィック・シミュレーション

前節でも多少触れたが，従来のトラフィック・シミュレーションはＧＰＳＳなどの汎用のシミュレーション言語を用いて行なわれることが多かった。しかしながらＧＰＳＳなどの離散系のシミュレーション言語では個々の車輌を一つのX'act として取り扱うため，コンピュータのメモリーの制限を受け広いエリアを，カバーできるモデルを作ることが困難であり，又X'act が動いて行く過程を離散的にしか表現できず，交通流の解析を行なうという点では不便なところがある。

本報文における，トラフィック・シミュレーションは，先にわれわれが開発した結合型実験シミュレータ 'SPACE' の応用例として行なわれたものであり，交通流の定量的解析 ── 実際には計数的な車両を連続的な量として取り扱い，信号のDiscrete な変化によってこれがどの様な影響を受けるかを解析 ── することを目的としている。

シミュレーションの目的が交通流の解析という点にあり，実際のシステムの信号制御方式最適化という様な具体的なものでないため，対象となるシステムは架空のものであり事前の調査も行なわなかった。しかしながら，今回開発したモデルは後述する様な問題点が解決され，それによって，修正を加えればきわめて高速にしかもかなりの精度を期待できるシミュレーションモデルとなるで

あろう。

4.3.1　モデル及びそのインプリメンテーション

今回のトラフィック・シミュレーションを行なうに当って２つのモデルを開発した。一つは交差点の停止線後方に停止している一連の車輌が信号が青になった時点でどの様な動きをするかを解析するためのモデル（ミクロモデル）そしてもう一つは交通流を流体と見なし，５つの交差点の信号によりこの流体がどの様な影響を受けるかを解析するためのモデル（街路モデル）の２つである。

(1)　ミクロモデル

図４－７のように交差点の停止線前後200ｍ程度の１つのレーンを想定し，このレーン内で赤信号のために停止線後方に並んでいる20台の車両の動きを考える。（ただし車両の追越しやレーン変更は考えないものとする。）

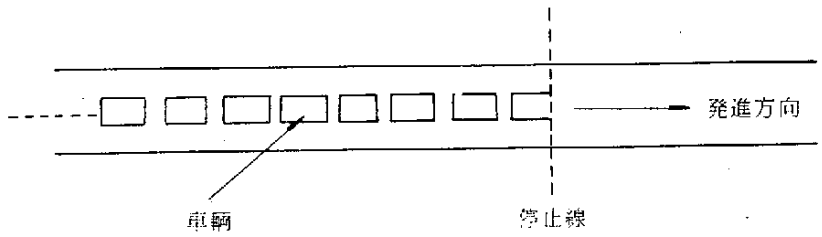


図４－７　ミクロモデル

信号が青になると車は序々に動き出す訳であるが，その動きは次の２つのモードに分けられる。

(i)　自由走行モード（Cruising Mode）

このモードではドライバーは自分の車に自由な加速度を与え，スピードがある一定値に到達したら，これを維持して行こうとする。先頭の車輌はこのCruising Modeに従って動きをする。図４－８は自由走行モードのブロック・ダイヤグラムである。

(ii)　追跡モード（Following Mode）

追跡モードにある車のドライバーは前の車との車間距離或は相対速度を意識しながら加速したり，減速したりする。停止線の後方に停止している20台の車輌のうち２台目以降の車輌は全てこの追跡モードに従って動くものとする。

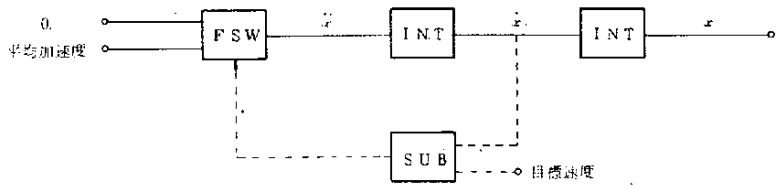


図４－８　自由走行モードのブロック・ダイヤグラム

リスト4－1　SPACEによるシミュレーションモデルのソースリスト（その1）

```
* SOURCE STATEMENT *


      1           DIMENSION SET(30,5)
      2           ISET=30
      3           JSET=5
      4           CALL SMTRN(SET,ISET,JSET)
      5           STOP
      6           END




      1           SUBROUTINE CONTRL(SET,ISET,JSET)
      2           COMMON /SYS1/ INDCTR,TNOW,NEVENT,ATT(30),MFE(30),NQZ(30),LQ(30)
      3           RETURN
      4           END


      1           SUBROUTINE START(SET,ISET,JSET)
      2           COMMON /SYS1/ INDCTR,TNOW,NEVENT,ATT(30),MFE(30),NQZ(30),LQ(30)
      3           COMMON /SYS4/ ICLL,DH,TIME,ISAI,KZSB,IEQ,IDEL,ISW,NLL,VIC(100),
                 1              S(100),D(100),Q(100),LA(100)
      4           COMMON /US1/XP(20,100),LC,KKA
      5           KKA=10
      6           LC=0
      7           COMMON SP(40)
      8           VIC(1)=0.0
      9           VIC(2)=0.0
     10           DO 10 I=3,40,2
     11           VIC(I)=0.0
     12           SP(I)=RANDOM(1)
     13           VIC(I+1)=VIC(I-1)-SP(I)
     14        10 CONTINUE
     15           RETURN
     16           END


      1           SUBROUTINE CMODL
      2           COMMON /SYS4/ ICLL,DH,TIME,ISAI,KZSB,IEQ,IDEL,ISW,NLL,VIC(100),
                 1              S(100),D(100),Q(100),LA(100)
      3           COMMON SP(40)
      4           DIMENSION SHAKAN(40),STAND(40),DEFZ(40),DEF(40),ACR(40)
      5           DIMENSION DPP(5,40)
      6           DEF(1)=S(1)-11.1111
      7           D(1)=SWICH(DEF(1),1.75,0.0,0.0)
      8           D(2)=S(1)
      9           DO 100 I=3,40,2
     10           SHAKAN(I)=S(I-1)-S(I+1)
     11           STAND(I)=SWICH(S(I-2)-1.750,SP(I),SP(I),S(I)*1.2)
     12           DEFZ(I)=SHAKAN(I)-STAND(I)
     13           DEF(I)=SWICH(S(I)-5.00,DEFZ(I),0.0,(S(I-2)-S(I))*3.0)
     14           ACR(I)=CSELCT(DEF(I),1)
     15           D(I)=DELAY(0.2,ACR(I),0.,1,DPP(1,I),5)
     16           D(I+1)=S(I)
     17       100 CONTINUE
     18           RETURN
     19           END
```

リスト４－１　SPACEによるシミュレーションモデルのソースリスト（その２）

```
      SUBROUTINE ENDRUN(SET,ISET,JSET)
      COMMON /SYS4/ ICLL,DH,TIME,ISAI,KZSB,IEQ,IDEL,ISW,NLL,VIC(100),
     1              S(100),D(100),Q(100),LA(100)
      COMMON /US1/XP(20,100),LC,KKA
      DIMENSION BUF(1024)
      CALL PLOTS(BUF(1),1024)
      CALL PLOT(10.,60.,3)
      CALL PLOT(250.,60.,2)
      CALL PLOT(20.,5.,3)
      CALL PLOT(20.,180.,2)
      CALL PLOT(20.,60.,-3)
      CALL PLOT(0.,100.,3)
      CALL PLOT(250.,100.,2)
      CALL PLOT(0.,0.,3)
      I=0
  100 I=I+1
      I2=I*2
      CS=VIC(I2)
      CALL PLOT(0.,CS,3)
      T=0.0
      DO 200 J=1,80
      T=T+2.5
      IF(XP(I,J).GT.150.) GO TO 150
  200 CALL PLOT(T,XP(I,J),2)
  150 IF(I.LT.KKA) GO TO 100
      CALL PLOT(300.,0.,999)
      RETURN
      END
```

```
      SUBROUTINE SCOND(SET,ISET,JSET)
      COMMON /SYS4/ ICLL,DH,TIME,ISAI,KZSB,IEQ,IDEL,ISW,NLL,VIC(100),
     1              S(100),D(100),Q(100),LA(100)
      COMMON /US1/XP(20,100),LC,KKA
      LC=LC+1
      DO 100 I=1,KKA
      I2=I*2
      XP(I,LC)=S(I2)
  100 CONTINUE
      RETURN
      END
```

図４－９は追跡モードのブロック・ダイヤグラムである。

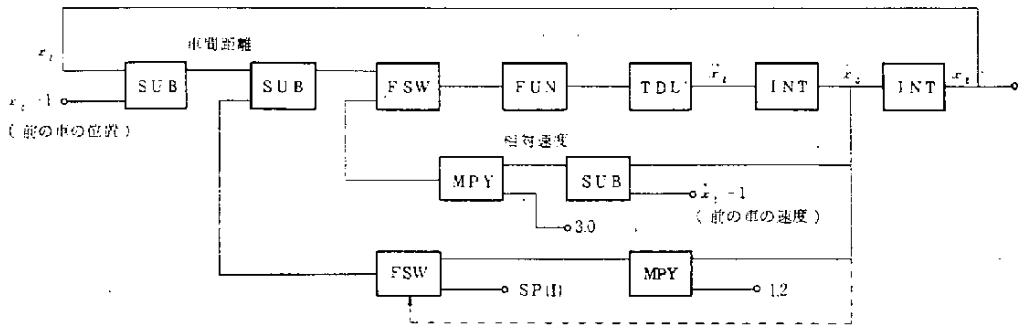


図４－９　追跡モードのブロック・ダイヤグラム

以上のブロック・ダイヤグラムをSPACEプログラムのCMODELとして，プログラミングし連続系のシミュレーションを行なう。ＬＩＳＴ－１はミクロモデルのソースリストである。

(2)　街路モデル

街路モデルは５つの交差点がシリアルに連なったものであり，非圧縮性流体と見なした交通流が信号によってどの様な影響を受けるかを，解析するためのモデルである。道路は片側だけを考え，道路幅も均一，しかも粘性は考えない。この様に５つの交差点にまたがっているシステムを，交差点から交差点を一つのサブシステムとして考えてみる。

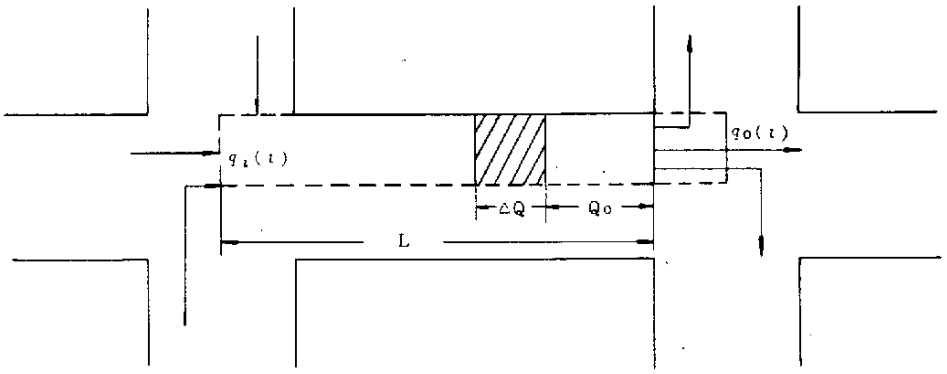


図４－10　街路モデル

図４－10において，時刻 t にサブシステムに入って来た交通流を $q_i(t)$，サブシステムから出て行く交通流を $q_0(t)$ とする。道路の長さを L とすると，次式で求まる時間 t だけ遅れて待ちの後ろにつく。

$$t = (L-Q)/v \quad 〔\text{sed}〕 \qquad \cdots\cdots (1)$$

〔ここで，v は交通流の平均速度，Q は待ちの長さとする〕

$q_0(t)$は，ミクロモデル解析より得られる関数$f(t)$或は待ちがないときは$q_i(t-\tau)$として与えられる。従ってΔt秒間の待ちの長さの増減は次式によって決まる。

$$S\,dQ = q_i(t-\tau)\,\Delta t - q_0(t)\,\Delta t \quad \cdots\cdots (2)$$

〔ここでSは道路単位長に入ることのできる車の量である〕

我々はこのサブシステムの評価基準として道路の混雑度を定量的にとらえることのできる待ちの長さを採用している。つまりサブシステムの待ちの長さを表わしている微分方程式が欲しいのであるが，それは次式より得られる。

$$Q(t) = Q_0(t) + \int_t^{t+\Delta t} \{ q_i(t-\tau) - q_0(t) \} / S\,dt \quad \cdots\cdots (3)$$

(3)式をブロックダイヤグラムとしてあらわしたのが図4－11である。

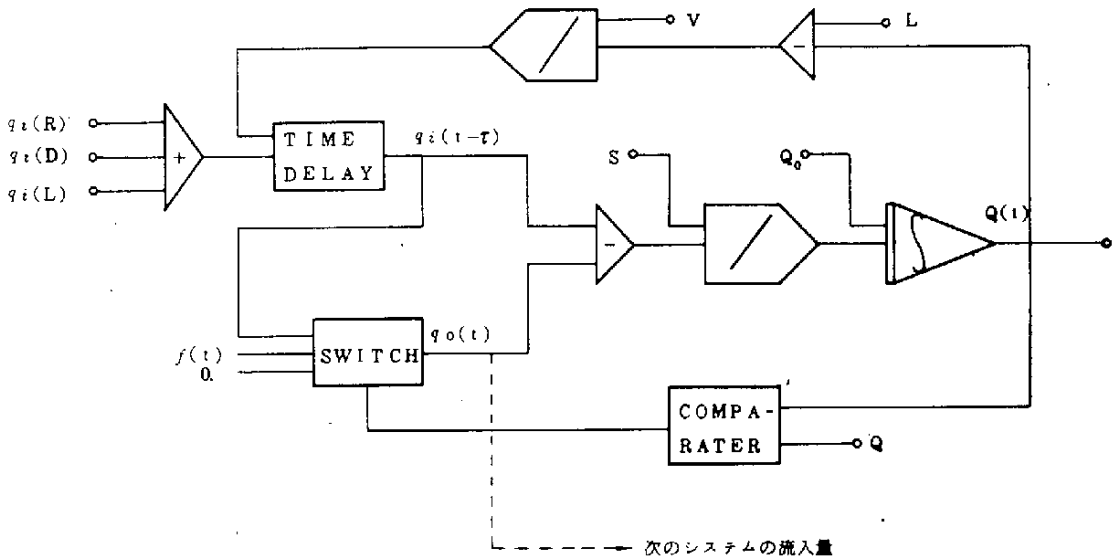


図4－11　街路モデルのブロックダイヤグラム

(3)であらわされる一つのサブシステムを5つシリアルに接続 ── サブシステムの流出量を次のサブシステムの流出量とする ── して街路モデルを構成する。街路モデルは微分方程式(3)による連続系モデルと，信号の切り換えをイベントにより制御し，個々の信号イベントを別々に起すことにより，オフセットを与えた離散系モデルからなる結合型モデルであり，On-Line SPACEによりインタアクティブなシミュレーションとして行なわれた。LIST－2は街路モデルのソースリストである。

リスト４－２　街路モデル

```
?????? 09:27 .I TSS SERVICE ? ?AZIMETA, DATE 1974.02.27
?READ
? USER-ID...MATSUMOTO
? ?USR-PSWD...0174
/?SUR JOB-NO....00IC001
?ACRO DOM ??YOURYOKU.
*RC SL
JOB ??TSCI.
¥LOAD SCOND,CMODEL
    SCOND    A LOAD ｼﾚﾀ
    CMODEL   A LOAD ｼﾚﾀ
¥LIST SCOND
                SUBROUTINE SCOND (SET,ISET,JSET)
                COMMON /SYS4/ICLL,DH,TIME,A(106),S(100),F(300)
                COMMON /US/PT(10),TF(10),TDE(10)
                CALL LPPLOT(S(1),2,1)
                CALL LPPLOT(S(3),2,2)
                CALL LPPLOT(S(5),2,3)
                DO 100 I=1,5
                CALL STATIS(S(I),I)
                TDE(I)=(200.-S(I))/12.
        100     IF(TDE(I).LT.0.) TDE(I)=0.
                RETURN
                END
¥LOAD START,ENDRUN,CONTRL,E1,E2,E3,E4,E5,STEP,MAINK
    START    A LOAD ｼﾚﾀ
    ENDRUN   A LOAD ｼﾚﾀ
    CONTRL   A LOAD ｼﾚﾀ
    E1       A LOAD ｼﾚﾀ
    E2       A LOAD ｼﾚﾀ
    E3       A LOAD ｼﾚﾀ
    E4       A LOAD ｼﾚﾀ
    E5       A LOAD ｼﾚﾀ
    STEP     A LOAD ｼﾚﾀ
    MAINK    A LOAD ｼﾚﾀ
¥LIST START
                SUBROUTINE START (SET,ISET,JSET)
                COMMON /US/PT(10),TF(10),TDE(10)
                COMMON /SYS4/ F(9),VIC(100),S(100),Z(300)
                DO 100 I=1,5
                PT(I)=0.
        100     TDE(I)=(200.-S(I))/12.
                RETURN
                END
¥LIST CMODEL
                SUBROUTINE CMODEL
                COMMON /SYS4/ ICLL,DH,TIME,AQQ(106),S(100),D(100),AWW(300)
                DIMENSION AA(100,5),QOUT(10)
                COMMON /US/ PT(10),TF(10),TDE(10)
                XCOS=0.75
                DO 100 I=1,5
                RX=RANDOM(2)+RANDOM(3)
                IF(I.EQ.1)QIN=RANDOM(1)*XCOS+RX
                IF(I.GT.1)QIN=QOUT(I)*XCOS+RX
                SG=STEP(PT(I))*2.
                ATF=TIME-TF(I)
                FTX=CSELCT(ATF,1)
                FXX=AMIN1(FTX,S(I))
                SGN=SG+CMPAR(0.,S(I))
                QIND=DELAY(TDE(I),QIN,0.1,1,AA(1,I),100)
                QOUT(I)=SWICH(SGN,FXX,QIND,0.)
                S(I)=(QIND-QOUT(I))/0.645
        100     CONTINUE
                RETURN
                END
¥RUN
```

4.3.2　シミュレーション結果

(1)　ミクロモデルのシミュレーション結果

図4－12は，ミクロモデルの微分方程式の解をX－Yプロッターに出力させたものである。スタートしてから徐々に加速され速度が大きくなるとともに車間距離が大きくなって行く様子がうかがえる。

点線は車が信号によって停らず一定の速度で通過したと仮定しての直線である。交差点から適当な距離 ── スタートした車が一定の速度に到しうる距離以上 ── はなれた地点（ここでは100m前方）で車の通過時間を測定してみる。点線に沿った車が通過する時刻とS(1)との差，S(1)とS(2)の差，S(2)とS(3) …… と車の通過する時間隔を取ったものが，図4－13の車頭間隔時間である。又，図4－15はS(1)，S(2) …… が，X軸と交わる時刻つまり交差点を抜ける時刻のグラフであり，図4－16はV(1)，V(2) …… などが30cm／sec以上になる時刻，即ち発進遅れのグ

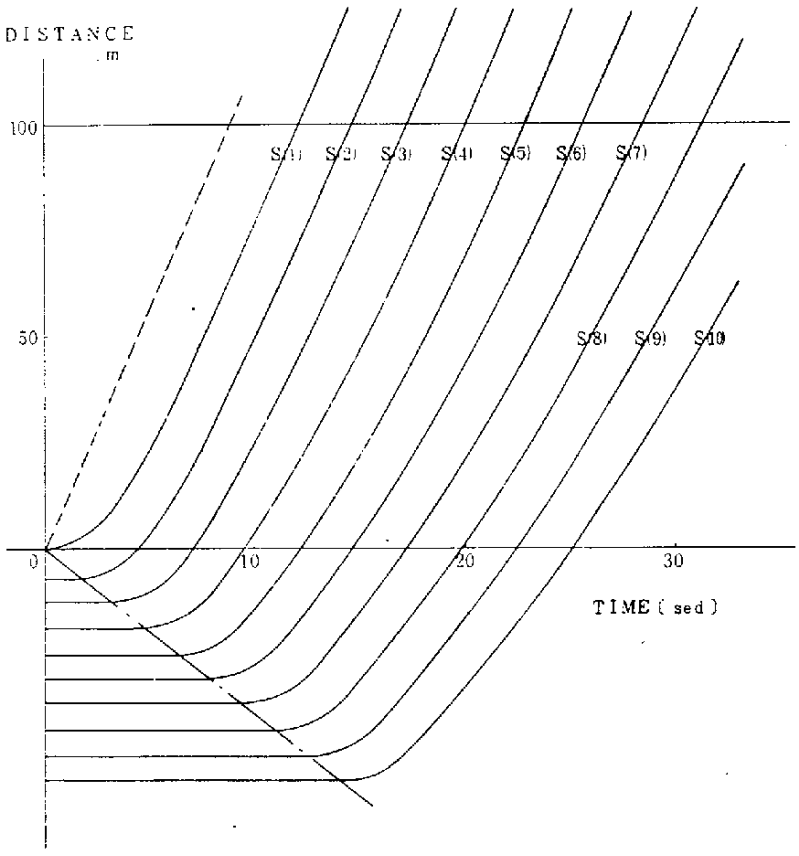


図4－12　ミクロモデル，X－Yプロッターへの出力グラフ

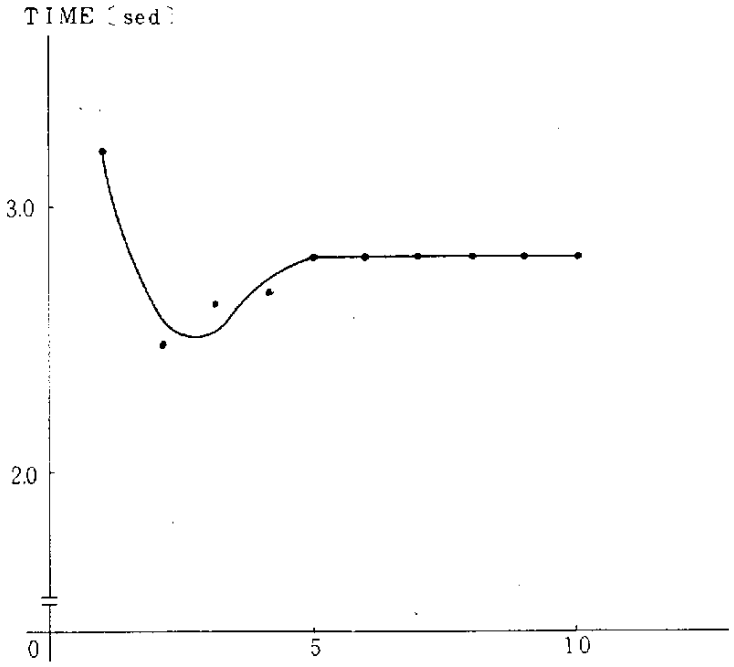


図４－13　車頭間隔時間

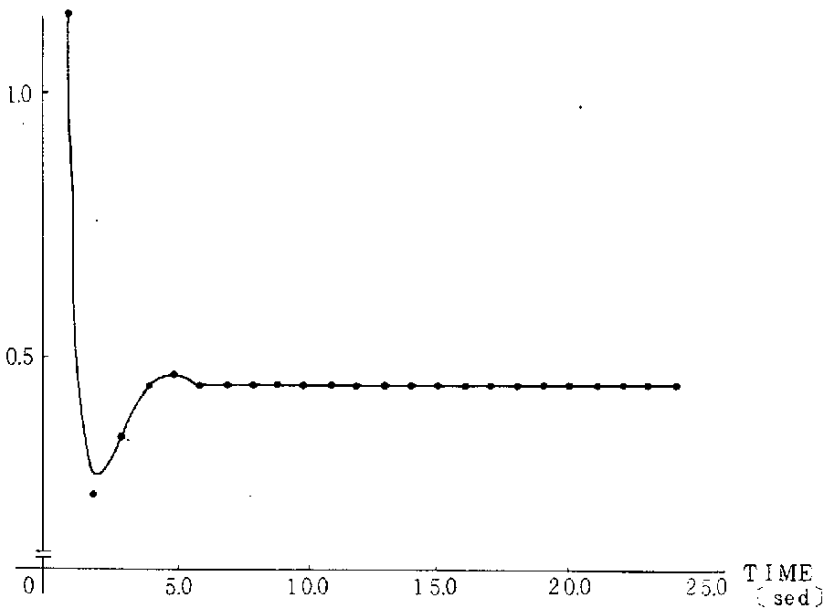


図４－14　単位時間当りの流出量

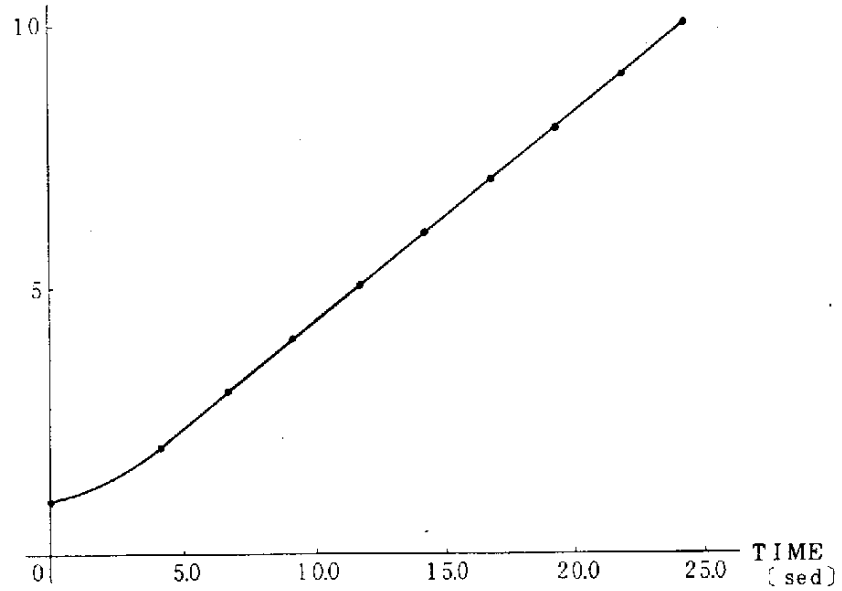


図 4－15　交差点を抜けるまでに要する時間

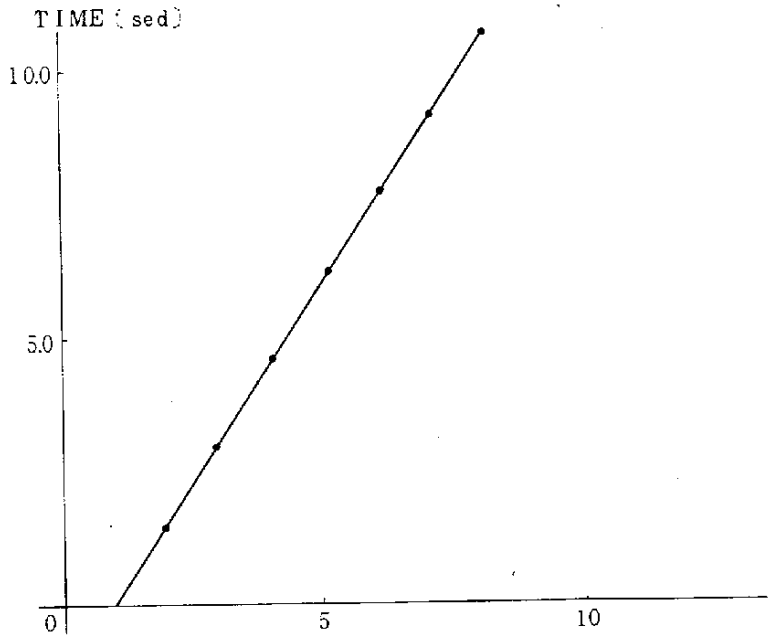


図 4－16　発　進　遅　れ

ラフである。

10台の車を連続量と見なし図4－15で単位時間当りの増分を計算すると図4－14の単位時間当りの流出量が得られる。この図4－14を街路モデルの流出量関数 ── 信号が青で，待ちがあり，しかも流出量値が待ちより小さいときの流出量 ── として用いる。

(2) 街路モデルのシミュレーション結果

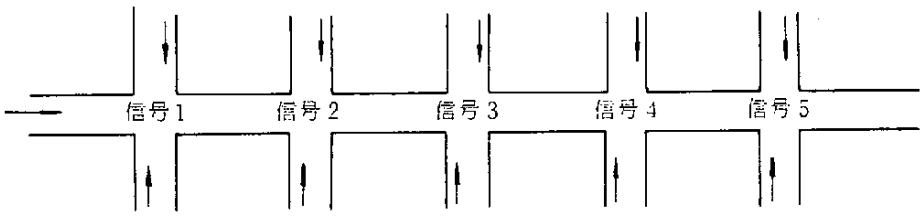


図4－17　街路モデル全体の構成

表4－1は，信号イベントを外生事象として発生させた時刻であり，信号1はシミュレーションがスタートしてから5 sec後，信号2は10 sec後 …… に信号を青にしている。又イベントの起る間隔は30 secである。

つまり，周期が60 sec，オフセット5 sec，スピリット50％の信号系を表現している。

街路モデルは5交差点をもつ一方向のみのモデルであり，その交通流の流入部は矢印の部分である。図4－18はOn-Line SPACEがランタイムに打ち出した各交差点にできる待行列長であり，図4－19はこのグラフをラン終了時にスケーリングしなおして打たせたグラフである。

表4－1　シミュレーションを開始させる外生事象

** I N I T I A L Q U E U E P R I N T **

* THE CONTENTS OF QUEUE 1 AT	0.0	TIME UNITS
0.0	200.00	4.00
5.00	1.00	0.0
10.00	2.00	0.0
15.00	3.00	0.0
20.00	4.00	0.0
25.00	5.00	0.0

** S I M U L A T I O N T R A C E **

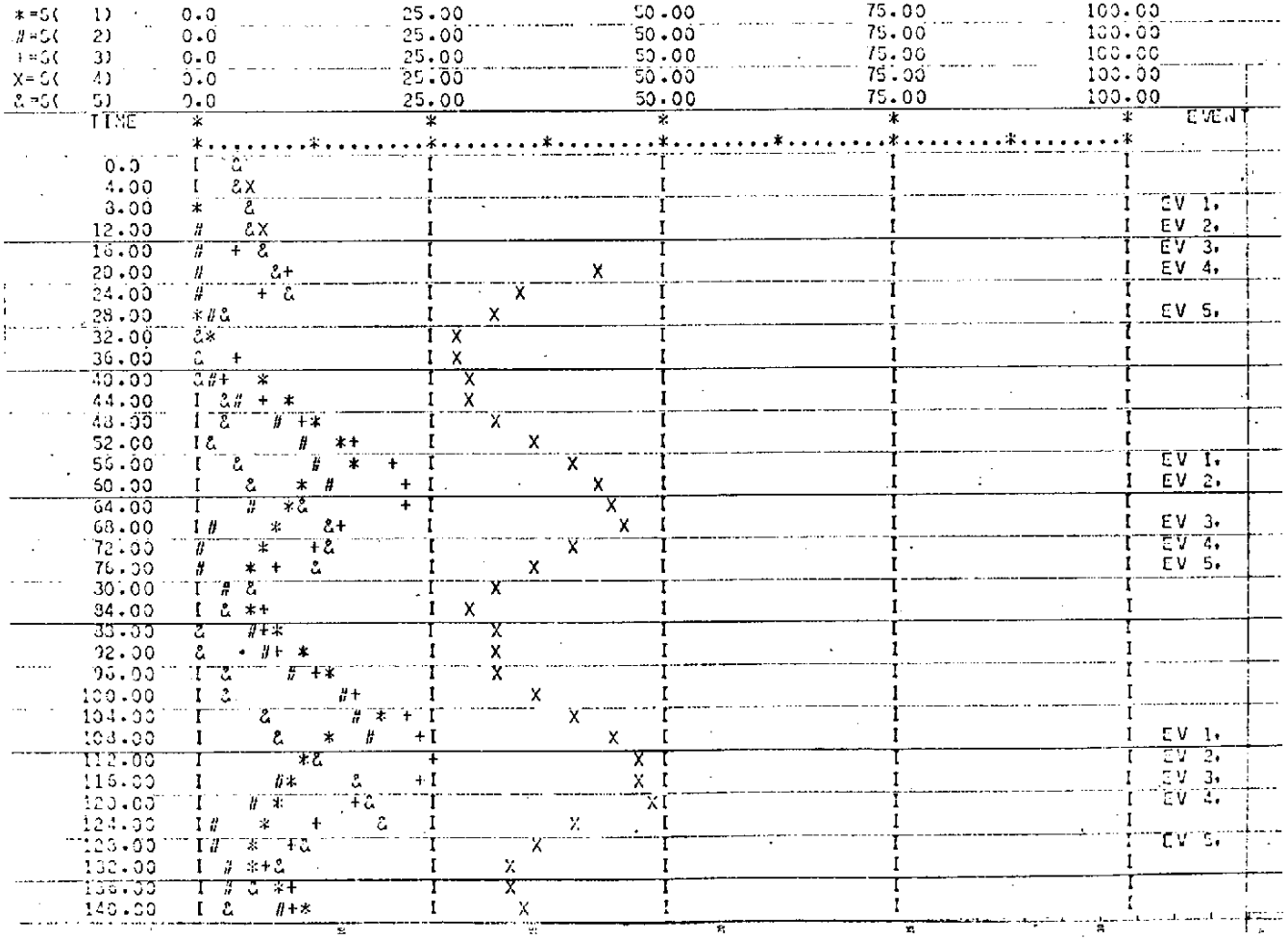


図 4－18　各交差点に於ける待行列長の変化

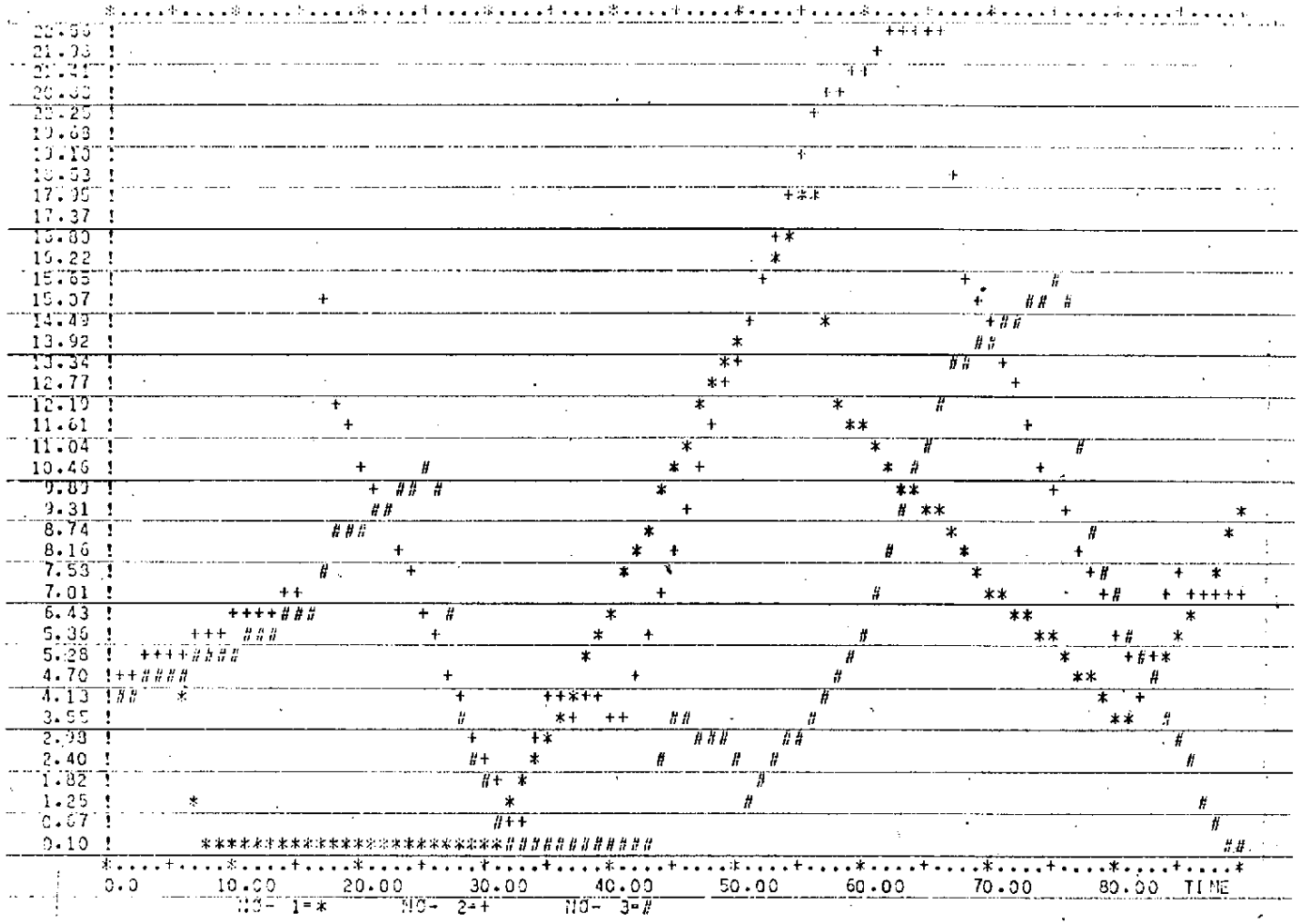


図 4 － 19　各交差点に於ける待行列長の変化

表４－２　各交差点に於ける待行列の統計

STATISTICAL TABLE

NUMBER	MEAN	STD.DEV.	MINIMUM	MAXIMUM
1	10.248	5.929	0.100	22.039
2	1.150	0.247	0.101	22.383
3	14.135	7.107	0.720	28.378
4	33.003	10.593	5.078	51.003
5	3.819	3.069	0.069	23.206

4.3.3　ミクロモデル及びストリートモデルの問題点とトラフィック・シミュレーションに対する反省

今回のトラフィック・シミュレーションは，コンバインド・タイプのシミュレーション言語の応用例として行なったものであり，その意味ではある程度目的は達せられたと思うが，トラフィックシミュレーションそのものは交通流を解析するには力が及ばなかった。実測のデータがなく，架空のシステムを対象としていること，及び理論的な裏付けが不足していることが原因であろうが，今回トラフィック・シミュレーションを行なうに当って開発したミクロモデル及びストリートモデルで用いたサブモデルは，次の様な点を考慮に入れさらに改良を加えれば，多少なりともトラフィック・シミュレーションに役立つであろうと思われる。

(1)　ミクロモデルに関する問題点

ミクロモデルでは，１つのレーンを想定し前の車両との関係にのみ注目していたが，２つ以上のレーンを考え，左右の車両との関係も考慮に入れなければならないと思う。又２つ以上のレーンを考えれば追越しやレーン変更も考えに入れることができる。今回のミクロモデルは片側一車線であったが，対流の車両もモデルに組み込み右折車が直進或は左折車にどの様な影響を及ぼすかを解析したい。

今回行なったミクロシミュレーションでは先頭の車輌は一定速度を維持して行くものとしていたが，これをある地点から減速させることによって待ちの最後尾につける動作を解析できると思う。

(2)　流体モデルに関する問題点

街路モデルは５つのサブモデル（流体モデル）をシリアルに構成されているが，このサブモデルを４方向パラレルに配置することによって交差点モデルをつくることができる。交差点モデルは実際にプログラムしたのであるが，右折車の影響及び対流の交通流が右折車に及ぼす影響を全く考慮に入れていないため，非常に曖昧なものとなってしまったので，本報文には載せなかった。右折車の影響が曖昧であるという点は街路モデルにも言えることで，今後この点をミクロモデルによる解析でサポートしたいと思う。

又流体モデルで流入した交通量が待ちの最後尾につくまでの遅れをタイム・ディレイによって表現しているがこの点も改良を加えたい。流入した交通流は流入時の密度が，サブシステム全体の密度に一致する様にちらばる傾向がある。これはSPACEに特殊なディレイをもうけるか，あるいは時間軸の外に密度軸の様なものを考え，偏微分方程式の様な形にサブモデルを作り変えなければならないと思う。

以上，トラフィック・シミュレーションモデルについての問題点を簡単に書いてきたが，今後上に書いた様な点の改良がなされ，トラフィック・シミュレーションモデルらしくなることを望む。

5．連続系シミュレーション言語の使用比較

第5章　連続系シミュレーション言語の使用比較

5.1　はじめに

5.1.1　目　　的

連続系シミュレーション言語はMIMICの時代を経て今日の連続系言語の代表であるCSMP-3，CSSL-3に至っているが，その間には非常に急激な進歩があった。そこで第4章で開発した交通流のミクロモデルをこれらの言語にコンバートし，MIMICからCSSL-3，CSMP-3に至る発展の過程を求明し，合わせてCSMP-3，CSSL-3の言語比較を行ない，我々の開発する結合系シミュレータの連続系の機能の設計参考にした。

5.1.2　言語比較の際の着目点

連続系シミュレーション言語は微分方程式によって表現されている連続系モデルのシミュレーションを行なうわけであるが，言語のもつ機能は大きく分けるとモデルの記述に関する機能，シミュレーションの実行に関する機能及びアウトプットレポート作成に関する機能とに分けられる。さらにこれらの機能の中には次の様な要素が含まれる。

(1)　モデルの記述
- function blockライブラリ
- 同一のパターンをまとめて記述する方法（マクロ機能）
- 手続きを取る機能

(2)　実行に関する機能
- プロセッサーの構造
- シミュレーション・スタディに関すること
- ランのコントロール
- パラメータの変更方法
- プログラムの構造

(3)　レポート作成に関する機能
- ランタイムにおける出力或は一括出力
- グラフの出力
- プロッターへの出力
- グラフィック出力

これらの機能について（MIMIC）vs（CSMP-3，CSSL-3）と（CSMP-3）vs（CSSL-3）との比較を行ない連続系言語の機能を検討して行く。

5.2 MIMIC. vs. CSSL－3，CSMP－3

5.2.1 プログラム構造についての比較

CSMP-3，CSSL-3のソース・プログラムはINITIAL，DYNAMIC，TERMINAL の3つのセクションに分けられているが，MIMICではこれらの概念はなくプログラムをシンプルな形でかける。しかしながらMIMICでは時間更新毎にイニシャライズを行ないしかもその結果はどこにも反映されないという無駄がある。〔リスト5－1参照〕又，このために有効な初期設定を行なえないという面もある。

5.2.2 プロセッサーの構造上の相違点

CSMP-3，CSSL-3はともにプリ・コンパイラでありFORTRANステートメントを中間言語としているのに対し，MIMICインタープリターであり，ステートメントが直接マシーン・コードに対応している。従ってMIMICではFORTRAN，ライブラリーとのインターフェイスができないことやproceduralな機能が組み込めない，などの欠点を有しプログラムの柔軟性にかける。

5.2.3 個々のステートメントとfunction block

MIMICでの個々のステートメントは1つのfunction blockに対応しており，そのfunction blockのinputには数式を書くことも許されている。表5－1はMIMICのfunction blockライブラリーであるが，CSMP-3，CSSL-3に比べ，MIMICはラプラス変換を取り扱う機能が小さく一次遅れのみであること，ステップ関数，ランプ関数，或はパルスジェネレーター，インパルスジェネレーターがないなどの点があるが，MIMICでもかなりの機能をもっていると言えよう。

又MIMICではFUNCTION TABLEと呼んでいる選択関数（arbitrary function）は独立変数は2つ（三次元関数）までであり，データで与えられた座標を線型補間するというシンプルなものである。

5.2.4 積 分 法

CSMP-3，CSSL-3ではいくつかの積分法から，任意に積分アルゴリズムを選択することができるが，MIMICでは可変増分のルンゲ・クッタ法でのみ行なわれる。きざみ幅の最大値，最小値及び時間増分は，DTMAX，DTMIN，DTというコンスタントにデータとして与える。又DT＝DTMAX＝DTMINとすると，きざみ幅は固定され，固定増分，ルンゲ・クッタ法となる。

リスト5－1　MIMICによるミクロモデル

```
***MIMIC SOURCE-LANGUAGE PROGRAM***

          F         CFN(7.0)
                    CON(K1,K2)
                    CON(K3)
                    CON(RCT)
                    CON(DTMAX)
                    CON(DT)
                    CON(DTMIN)
                    CON(K4)
FIRST CARS EQUATION OF ITS MOVEMENT
          1DX01     INT(2DX01,0.)
          X01       INT(1DX01,0.)
          DEF01     SUB(1DX01,K2)
          2DX01     FSW(DEF01,K1,0.,0.)
SECOND CARS EQUATION OF ITS MOVEMENT
          X02Z      ADD(0.,K3)
          1DX02     INT(2DX02,0.)
          X02       INT(1DX02,X02Z)
          SHKN02    SUB(X01,X02)
          CRT02     SUB(1DX02,2.7777)
          STDL02    FSW(CRT02,5.,5.,1DX02*0.5)
          DEFZ02    SUB(SHKN02,STDL02)
          DEF02     FSW(1DX02-11.1,DEFZ02,0.,(1DX01-1DX02)*K4)
          2DX02D    FUN(F,DEF02)
          2DX02     TDL(2DX02D,RCT)
THIRD CARS EQUATION OF ITA MOVEMENT
          X03Z      ADD(X02Z,K3)
          1DX03     INT(2DX03,0.)
          X03       INT(1DX03,X03Z)
          SHKN03    SUB(X02,X03)
          CRT03     SUB(1DX03,2.7777)
          STDL03    FSW(CRT03,5.,5.,1DX03*0.5)
          DEFZ03    SUB(SHKN03,STDL03)
          DEF03     FSW(1DX03-11.1,DEFZ03,0.,(1DX02-1DX03)*K4)
          2DX03D    FUN(F,DEF03)
          2DX03     TDL(2DX03D,RCT)
FOURTH CARS EQUATION OF ITS MOVEMENY
          X04Z      ADD(X03Z,K3)
          1DX04     INT(2DX04,0.)
          X04       INT(1DX04,X04Z)
          SHKN04    SUB(X03,X04)
          CRT04     SUB(1DX04,2.7777)
          STDL04    FSW(CRT04,5.,5.,1DX04*0.5)
          DEFZ04    SUB(SHKN04,STDL04)
          DEF04     FSW(1DX04-11.1,DEFZ04,0.,(1DX03-1DX04)*K4)
          2DX04D    FUN(F,DEF04)
          2DX04     TDL(2DX04D,RCT)
FIFTH CARS EQUATION OF ITS MOVEMENT
          X05Z      ADD(X04Z,K3)
          1DX05     INT(2DX05,0.)
          X05       INT(1DX05,X05Z)
          SHKN05    SUB(X04,X05)
          CRT05     SUB(1DX05,2.7777)
          STDL05    FSW(CRT05,5.,5.,1DX05*0.5)
          DEFZ05    SUB(SHKN05,STDL05)
          DEF05     FSW(1DX05-11.1,DEFZ05,0.,(1DX04-1DX05)*K4)
          2DX05D    FUN(F,DEF05)
          2DX05     TDL(2DX05D,RCT)
```

```
C 6TH CARS EQUATION OF ITS MOVEMENT
          X06Z      ADD(X05Z,K3)
          1DX06     INT(2DX06,0.)
          X06       INT(1DX06,X06Z)
          SHKN06    SUB(X05,X06)
          CRT06     SUB(1DX06,2.7777)
          STDL06    FSW(CRT06,5.,5.,1DX06*0.5)
          DEFZ06    SUB(SHKN06,STDL06)
          DEF06     FSW(1DX06-11.1,DEFZ06,0.,(1DX05-1DX06)*K4)
          2DX06D    FUN(F,DEF06)
          2DX06     TDL(2DX06D,RCT)
C 7TH CARS EQUATION OF ITS MOVEMENT
          X07Z      ADD(X06Z,K3)
          1DX07     INT(2DX07,0.)
          X07       INT(1DX07,X07Z)
          SHKN07    SUB(X06,X07)
          CRT07     SUB(1DX07,2.7777)
          STDL07    FSW(CRT07,5.,5.,1DX07*0.5)
          DEFZ07    SUB(SHKN07,STDL07)
          DEF07     FSW(1DX07-11.1,DEFZ07,0.,(1DX06-1DX07)*K4)
          2DX07D    FUN(F,DEF07)
          2DX07     TDL(2DX07D,RCT)
C 8TH CARS EQUATION OF ITS MOVEMENT
          X08Z      ADD(X07Z,K3)
          1DX08     INT(2DX08,0.)
          X08       INT(1DX08,X08Z)
          SHKN08    SUB(X07,X08)
          CRT08     SUB(1DX08,2.7777)
          STDL08    FSW(CRT08,5.,5.,1DX08*0.5)
          DEFZ08    SUB(SHKN09,STDL08)
          DEF08     FSW(1DX08-11.1,DEFZ08,0.,(1DX07-1DX08)*K4)
          2DX08D    FUN(F,DEF08)
          2DX08     TDL(2DX08D,RCT)
C 9TH CARS EQUATION OF ITS MOVEMENT
          X09Z      ADD(X08Z,K3)
          1DX09     INT(2DX09,0.)
          X09       INT(1DX09,X09Z)
          SHKN09    SUB(X08,X09)
          CRT09     SUB(1DX09,2.7777)
          STDL09    FSW(CRT09,5.,5.,1DX09*0.5)
          DEFZ09    SUB(SHKN09,STDL09)
          DEF09     FSW(1DX09-11.1,DEFZ09,0.,(1DX08-1DX09)*K4)
          2DX09D    FUN(F,DEF09)
          2DX09     TDL(2DX09D,RCT)
C10TH CARS EQUATION OF ITS MOVEMENT
          X10Z      ADD(X09Z,K3)
          1DX10     INT(2DX10,0.)
          X10       INT(1DX10,X10Z)
          SHKN10    SUB(X09,X10)
          CRT10     SUB(1DX10,2.7777)
          STDL10    FSW(CRT10,5.,5.,1DX10*0.5)
          DEFZ10    SUB(SHKN10,STDL10)
          DEF10     FSW(1DX10-11.1,DEFZ10,0.,(1DX09-1DX10)*K4)
          2DX10D    FUN(F,DEF10)
          2DX10     TDL(2DX10D,RCT)
C11TH CARS EQUATION OF ITS MOVEMENT
          X11Z      ADD(X10Z,K3)
          1DX11     INT(2DX11,0.)
          X11       INT(1DX11,X11Z)
          SHKN11    SUB(X10,X11)
```

```
          CRT11     SUB(1DX11,2.7777)
          STDL11    FSW(CRT11,5.,5.,1DX11*0.5)
          DEFZ11    SUB(SHKN11,STDL11)
          DEF11     FSW(1DX11-11.1,DEFZ11,0.,(1DX10-1DX11)*K4)
          2DX11D    FUN(F,DEF11)
          2DX11     TDL(2DX11D,RCT)
C12TH CARS EQUATION OF ITS MOVEMENT
          X12Z      ADD(X11Z,K3)
          1DX12     INT(2DX12,0.)
          X12       INT(1DX12,X12Z)
          SHKN12    SUB(X11,X12)
          CRT12     SUB(1DX12,2.7777)
          STDL12    FSW(CRT12,5.,5.,1DX12*0.5)
          DEFZ12    SUB(SHKN12,STDL12)
          DEF12     FSW(1DX12-11.1,DEFZ12,0.,(1DX11-1DX12)*K4)
          2DX12D    FUN(F,DEF12)
          2DX12     TDL(2DX12D,RCT)
C13TH CARS EQUATION OF ITS MOVEMENT
          X13Z      ADD(X12Z,K3)
          1DX13     INT(2DX13,0.)
          X13       INT(1DX13,X13Z)
          SHKN13    SUB(X12,X13)
          CRT13     SUB(1DX13,2.7777)
          STDL13    FSW(CRT13,5.,5.,1DX13*0.5)
          DEFZ13    SUB(SHKN13,STDL13)
          DEF13     FSW(1DX13-11.1,DEFZ13,0.,(1DX12-1DX13)*K4)
          2DX13D    FUN(F,DEF13)
          2DX13     TDL(2DX13D,RCT)
C14TH CARS EQUATION OF ITS MOVEMENT
          X14Z      ADD(X13Z,K3)
          1DX14     INT(2DX14,0.)
          X14       INT(1DX14,X14Z)
          SHKN14    SUB(X13,X14)
          CRT14     SUB(1DX14,2.7777)
          STDL14    FSW(CRT14,5.,5.,1DX14*0.5)
          DEFZ14    SUB(SHKN14,STDL14)
          DEF14     FSW(1DX14-11.1,DEFZ14,0.,(1DX13-1DX14)*K4)
          2DX14D    FUN(F,DEF14)
          2DX14     TDL(2DX14D,RCT)
C15TH CARS EQUATION OF ITS MOVEMENT
          X15Z      ADD(X14Z,K3)
          1DX15     INT(2DX15,0.)
          X15       INT(1DX15,X15Z)
          SHKN15    SUB(X14,X15)
          CRT15     SUB(1DX15,2.7777)
          STDL15    FSW(CRT15,5.,5.,1DX15*0.5)
          DEFZ15    SUB(SHKN15,STDL15)
          DEF15     FSW(1DX15-11.1,DEFZ15,0.,(1DX14-1DX15)*K4)
          2DX15D    FUN(F,DEF15)
          2DX15     TDL(2DX15D,RCT)
C16TH CARS EQUATION OF ITS MOVEMENT
          X16Z      ADD(X15Z,K3)
          1DX16     INT(2DX16,0.)
          X16       INT(1DX16,X16Z)
          SHKN16    SUB(X15,X16)
          CRT16     SUB(1DX16,2.7777)
          STDL16    FSW(CRT16,5.,5.,1DX16*0.5)
          DEFZ16    SUB(SHKN16,STDL16)
          DEF16     FSW(1DX16-11.1,DEFZ16,0.,(1DX15-1DX16)*K4)
          2DX16D    FUN(F,DEF16)
```

```
          2DX16     TDL(2DX16D,RCT)
C17TH CARS EQUATION OF ITS MOVEMENT
          X17Z      ADD(X16Z,K3)
          1DX17     INT(2DX17,0.)
          X17       INT(1DX17,X17Z)
          SHKN17    SUB(X16,X17)
          CRT17     SUB(1DX17,2.7777)
          STDL17    FSW(CRT17,5.,5.,1DX17*0.5)
          DEFZ17    SUB(SHKN17,STDL17)
          DEF17     FSW(1DX17-11.1,DEFZ17,0.,(1DX16-1DX17)*K4)
          2DX17D    FUN(F,DEF17)
          2DX17     TDL(2DX17D,RCT)
C18TH CARS EQUATION OF ITS MOVEMENT
          X18Z      ADD(X17Z,K3)
          1DX18     INT(2DX18,0.)
          X18       INT(1DX18,X18Z)
          SHKN18    SUB(X17,X18)
          CRT18     SUB(1DX18,2.7777)
          STDL18    FSW(CRT18,5.,5.,1DX18*0.5)
          DEFZ18    SUB(SHKN18,STDL18)
          DEF18     FSW(1DX18-11.1,DEFZ18,0.,(1DX17-1DX18)*K4)
          2DX18D    FUN(F,DEF18)
          2DX18     TDL(2DX18D,RCT)
C19TH CARS EQUATION OF ITS MOVEMENT
          X19Z      ADD(X18Z,K3)
          1DX19     INT(2DX19,0.)
          X19       INT(1DX19,X19Z)
          SHKN19    SUB(X18,X19)
          CRT19     SUB(1DX19,2.7777)
          STDL19    FSW(CRT19,5.,5.,1DX19*0.5)
          DEFZ19    SUB(SHKN19,STDL19)
          DEF19     FSW(1DX19-11.1,DEFZ19,0.,(1DX18-1DX19)*K4)
          2DX19D    FUN(F,DEF19)
          2DX19     TDL(2DX19D,RCT)
C20TH CARS EQUATION OF ITS MOVEMENT
          X20Z      ADD(X19Z,K3)
          1DX20     INT(2DX20,0.)
          X20       INT(1DX20,X20)   X20Z.
          SHKN20    SUB(X19,X20)
          CRT20     SUB(1DX20,2.7777
          STDL20    FSW(CRT20,5.,5.,1DX20*0.5)
          DEFZ20    SUB(SHKN20,STDL20)
          DEF20     FSW(1DX20-11.1,DEFZ20,0.,(1DX19-1DX20)*K4)
          2DX20D    FUN(F,DEF20)
          2DX20     TDL(2DX20D,RCT)
          RD3       X01-X03
          RD4       X01-X04
          RD5       X01-X05
          RD6       X01-X06
                    FIN(T,30.)
                    OUT(T,X01,X02,X03,X04,X05)
                    OUT(1DX02,X06,X07,X08,X09,X10)
                    OUT(1DX07,X11,X12,X13,X14,X15)
                    OUT(1DX13,X16,X17,X18,X19,X20)
                    UT
                    PLO(T,SHKN02,RD3,RD4,RD5,RD6)
                    END

                    ***SORT DIAGNOSTICS FOLLOW***
```

リスト５－２　ミクロモデル（ＭＩＭＩＣ）の出力結果

-1.50000+01	-4.00000+00	0.00000
-1.00000+01	-1.00000+00	0.00000
-5.00000+00	-5.00000-01	0.00000
0.00000	0.00000	0.00000
1.00000+00	5.00000-01	0.00000
1.00000+01	1.00000+00	0.00000
1.50000+01	4.00000+00	0.00000

K1	K2
3.00000+00	1.11000+01

K3
-5.00000+00

RCT
1.00000-01

DTMAX
5.00000-01

DT
5.00000-01

DTMIN
5.00000-02

K4
3.00000+00

T	0.00000	X01	0.00000	X02	-5.00000+00	X03	-1.00000+01	X04	-1.50000+01	X05	-2.00000+01
IDX02	0.00000	X06	-2.50000+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	0.00000	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	0.00000
T	5.00000-01	X01	3.79284-01	X02	-4.99836+00	X03	-1.00000+01	X04	-1.50000+01	X05	-2.00000+01
IDX02	1.63185-02	X06	-2.50000+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	0.00000	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-5.05711-01
T	1.00000+00	X01	1.50866+00	X02	-4.95884+00	X03	-9.99972+00	X04	-1.50000+01	X05	-2.00000+01
IDX02	1.78494-01	X06	-2.50000+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	0.00000	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-2.01154+00
T	1.50000+00	X01	3.38803+00	X02	-4.80238+00	X03	-9.99511+00	X04	-1.50000+01	X05	-2.00000+01
IDX02	4.54765-01	X06	-2.50000+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	0.00000	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-4.51738+00
T	2.00000+00	X01	6.01741+00	X02	-4.49526+00	X03	-9.97013+00	X04	-1.49994+01	X05	-2.00000+01
IDX02	7.83937-01	X06	-2.50000+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	0.00000	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-8.02321+00
T	2.50000+00	X01	9.39678+00	X02	-4.00689+00	X03	-9.89280+00	X04	-1.49957+01	X05	-1.99999+01
IDX02	1.18242+00	X06	-2.50000+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	0.00000	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-1.25290+01
T	3.00000+00	X01	1.35262+01	X02	-3.29626+00	X03	-9.71670+00	X04	-1.49808+01	X05	-1.99994+01
IDX02	1.71250+00	X06	-2.50000+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	1.39740-07	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-1.80349+01
T	3.50000+00	X01	1.84055+01	X02	-2.14114+00	X03	-9.41191+00	X04	-1.49370+01	X05	-1.99968+01
IDX02	3.09350+00	X06	-2.49999+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	5.10990-06	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-2.45407+01
T	4.00000+00	X01	2.39066+01	X02	-9.51864-02	X03	-8.96669+00	X04	-1.48540+01	X05	-1.99878+01
IDX02	5.09192+00	X06	-2.49995+01	X07	-3.00000+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	4.55617-05	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-3.20465+01
T	4.50000+00	X01	2.94727+01	X02	2.95079+00	X03	-8.36100+00	X04	-1.46319+01	X05	-1.99617+01
IDX02	7.09192+00	X06	-2.49978+01	X07	-2.99999+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	2.54533-04	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-4.05524+01
T	5.00000+00	X01	3.50387+01	X02	6.99675+00	X03	-7.56328+00	X04	-1.43033+01	X05	-1.98988+01
IDX02	9.09192+00	X06	-2.49922+01	X07	-2.99996+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	1.08914-03	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-5.00582+01
T	5.50000+00	X01	4.06047+01	X02	1.20427+01	X03	-6.50310+00	X04	-1.38407+01	X05	-1.97682+01
IDX02	1.10919+01	X06	-2.49762+01	X07	-2.99985+01	X08	-3.50000+01	X09	-4.00000+01	X10	-4.50000+01
IDX07	3.84424-03	X11	-5.00000+01	X12	-5.50000+01	X13	-6.00000+01	X14	-6.50000+01	X15	-7.00000+01
IDX13	0.00000	X16	-7.50000+01	X17	-8.00000+01	X18	-8.50000+01	X19	-9.00000+01	X20	-6.05640+01

表5－1　MIMICのfunction blockライブラリー

関　数	コード	インプット	結果および注意
1. 算術関数			
ADDITION	ADD SUM	A,B(,C,D,E,F)*	R=A+B(+C+D+E+F)
SUBTRACTION	SUB	A,B	R=A−B
MULTIPLICATION	MPY	A,B(,C,D,E,F)	R=A*B(*C*D*E*F)
DIVIDE	DIV	A,B	R=A/B
MULTIPLY AND ADD	MAD	A,B,C(,D,E,F)	R=A*B+C(*D+E*F)
NEGATION	NEG	A	R=−A
ABSOLUTE VALUE	ABS	A	R=|A|
EQUALITY	EQL	A	R=A
SQUARE ROOT	SQR	A(>O)	$R=\sqrt{A}$
SINE	SIN	A(rads.)	R=sin A
COSINE	COS	A(rads.)	R=cos A
ARCTANGENT	ATN	A(,B)	$R=\tan^{-1}(A/B)$. Bが指定されなければ＋1とみなされる。
EXPONENTIAL	EXP	A(,B)	$R=B^{A}$ Bが指定されなければeとみなされる。
LOGARITHM	LOG	A(,B)	$R=\log_{B}A$ Bが指定されなければeとみなされる。

＊　カッコで囲まれた引数は，必ずしも指定しなくてよい。

関　数	コード	インプット	結果および注意
2. 論理関数およびコントロール関数			
FUNCTION SWITCH	FSW	A,B,C,D	R=B　A<0 =C　A=0 =D　A>0
LOGICAL SWITCH	LSW	A,B,C	R=B　AがTRUEのとき =C　AがFALSEのとき
AND	AND	A,B(,C,D,E,F)	R=A∩B(∩C∩D∩E∩F)
EXCLUSIVE OR	EOR	A,B	R=(A∩B′)∪(A′∩B)
INCLUSIVE OR	IOR	A,B(,C,D,E,F)	R=A∪B(∪C∪D∪E∪F)
COMPLEMENT	COM NOT	A	R=A′
GO TO NEXT CASE	FIN	A,B	A≧Bのとき，次のランに進む。
END OF PROGRAM	END		MIMICプログラムの終りおよびMIMICデータの始まりを指定する。
3. インプット／アウトプット関数			
NAME CONSTANTS	CON	A.B.C.D.E.F	定数の名前を入れる
NAME PARAMETERS	PAR	A.B.C.D.E.F	パラメータの名前を入れる
NAME FUNCTION (CONSTANT)	CFN	A	テーブルの名前を結果のフィールドに指定する。Aは点の組の数である。
NAME FUNCTION (PARAMETER)	PFN	A	テーブルの名前を結果のフィールドに指定する。Aは点の組の数である。
PRINT OUTPUT	OUT PRI	A,B,C,D,E,F	TがDT進む度にA,B,C,D,E,Fがプリントされる。
PRINT HEADERS	HDR HEA	A,B,C,D,E,F	ヘッダーとして，A,B,C,D,E,Fがプリントされる。
PLOT	PLO	A,B,C,D,E,F	TがDT進む度に，A,B,C,D,E,Fの値がPLOサブルーチンにわたされる。

関　数	コード	インプット	結果および注意
4. 積分関数			
INTEGRATION	INT	A, B(, C, D)	R=B+$\int$adt TRUE FALSE TRUE OPERATE HOLD FALSE RESET OPERATE
LIMIT INTEGRATORS	LIN	A, B, C, D	R=0 B<C =A C≦B≦D =0 B>D
FIRST ORDER TRANSFER FUNCTION	FTR	A, B	R= [A(S)/(BS+1)] ここでAはインプットで，Bは時間に関して一定である。
5. ハイブリッド関数			
MONOSTABLE MULTIVIBRATOR	M	A, B	AがTRUEのときRはTRUEにセットされ，AがFALSEになっても，B単位時間はRはTRUEのまま，t＝0のとき R＝A
TRACK AND STORK	TAS	A, B, C	R=A BがTRUEのとき =R BがFALSEのとき R=C at t=0
FLIP-FLOP	FLF	A, B, C	R= A∪(B′∩R_p) ここでR_pは1単位時間前のRの値である。 t＝0のときR＝C
ZERO ORDER HOLD	ZOH	A, B	Aが値が単位時間B毎にとり出され，Rに入れられる。 t＝0のときR＝A

関　数	コード	インプット	結果および注意
6. 特別の関数			
DEAD SPACE	DSP	A.B.C	R=A−B　A<B =0　B≤A≤C =A−C　A>C
TIME DELAY	TDL	A.B(.C)	R=A(t−B) Cは貯えられるべきAのポイントの数。Cが指定されなければ，１００とみなされる。 Bは変数でもよい。t≦Bのとき　R＝A(0)
FUNCTION	FUN	A.B A.B.C	R=A(B) R=A(B.C)
IMPLICIT FUNCTION	IMP	A.B	R=A where|A−B(A)|≤5×10^{-6}|A|
MAXIMUM	MAX	A.B(.C.D.E.F)	R=max(A.B(.C.D.E.F))
MINIMUM	MIN	A.B(.C.D.E.F)	R=min(A.B(.C.D.E.F))
RANDOM NO.GENERATCR	RNG	A.B.C	平均A．標準偏差Bのガラス分布の乱数がRに入れられる。Cは乱数生成の際の出発値である。
RANDOM NO.GENERATOR	RNU	A.B.C	下限A．上限がBの一様乱数がRに入れられる。Cは乱数生成の際の出発値。
DERIVATIVE	DER	A.B.C	R=dA/dB．t＝0のとき R＝C
LIMITER	LIM	A.B.C	R=B　A<B =A　B≤A≤C =C　A>C

関　数	コード	インプット	結果および注意
7. サブプログラム			
BEGIN SUBPROGRAM	BSP	A(,B,C,D,E,F)	BSPおよびESPカードの結果のフィールドにサブプログラム名を書く。インプット用およびアウトプット用引数はそれぞれBSP，ESPカードで指定する。
END SUBPROGRAM	ESP	A(,B,C,D,E,F)	
CALL SUBPROGRAM	CSP	A(,B,C,D,E,F)	CSP カードの結果のフィールドに，コールするサブプログラムの名前を書く。RSPカードは，CSPの直後に続けなければならない。
RETURN SUBPROGRAM	RSP	A(,B,C,D,E,F)	

MIMIC マニュアルより一部掲載

5.2.5　ランのコントロール，シミュレーション・スタディに関する相違点

CSMP－3，CSSL－3ではランの終了時にTERMINALセクションを通るが，ここでシミュレーションを最適解に近づける様パラメータを変更させ再びランを始めることができる様なプログラムを組むことができ，これをシミュレーション・スタディと呼んでいる。MIMICではパラメータを変更し何回ものランを一回のJOBで行なうことはできるが，シミュレーション・スタディを行なうことはできない。

MIMICではラン毎に変わる値をパラメータ，ジョブ全体を通じて変わらない値をコンスタントと呼び，これらはデータとして与えられる。

値をコンスタントと呼び，これらはデータとして与えられる。

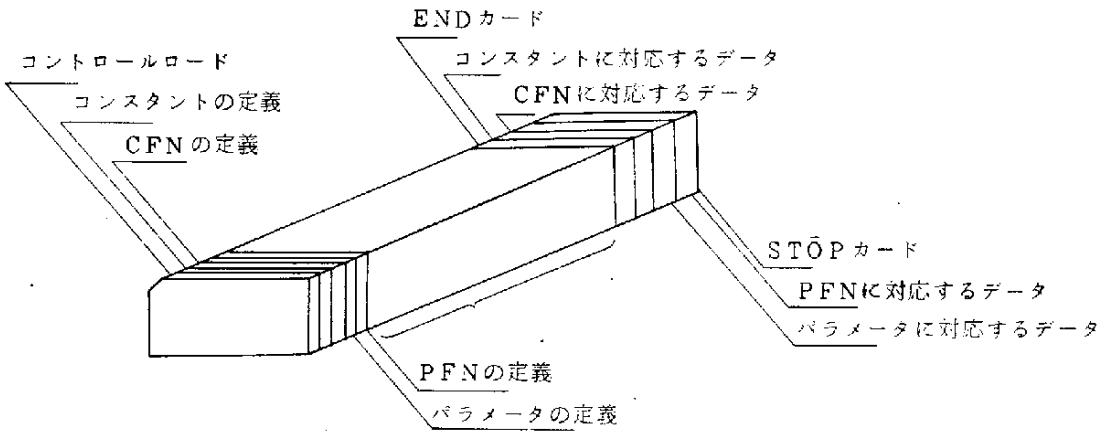


図5－1　MIMICプログラムカード

5.2.6　レポート作成に関する機能の相違点

ラインプリンターへの出力に関しては，CSMP－3，CSSL－3はランタイムに出力させる方式と，計算結果をドラムにセーブしておきランの終了時に一括してプリントさせる方式の二つがあるが，MIMICではランタイムに出力させる方式のみである。

又MIMICではラインプリンターにグラフを出力させるには，座標軸，レーベル，タイトルなどの指定が必要であるが，CSMP－3で5変数以下の出力では自動的にグラフを出力してくれる。X－Yプロッターは MIMIC，CSMP，CSSLともに可能である。

LIST－2はMIMICのミクロモデルのプリントアウト，図5－2はそのグラフである。

5.2.7　マクロ機能

CSMP－3，CSSL－3にはマクロ機能があり同一のパターンをまとめて記述することができるが，MIMICにはそれがなく，くり返しの数だけプログラムを作らなければならず非常にわずらわ

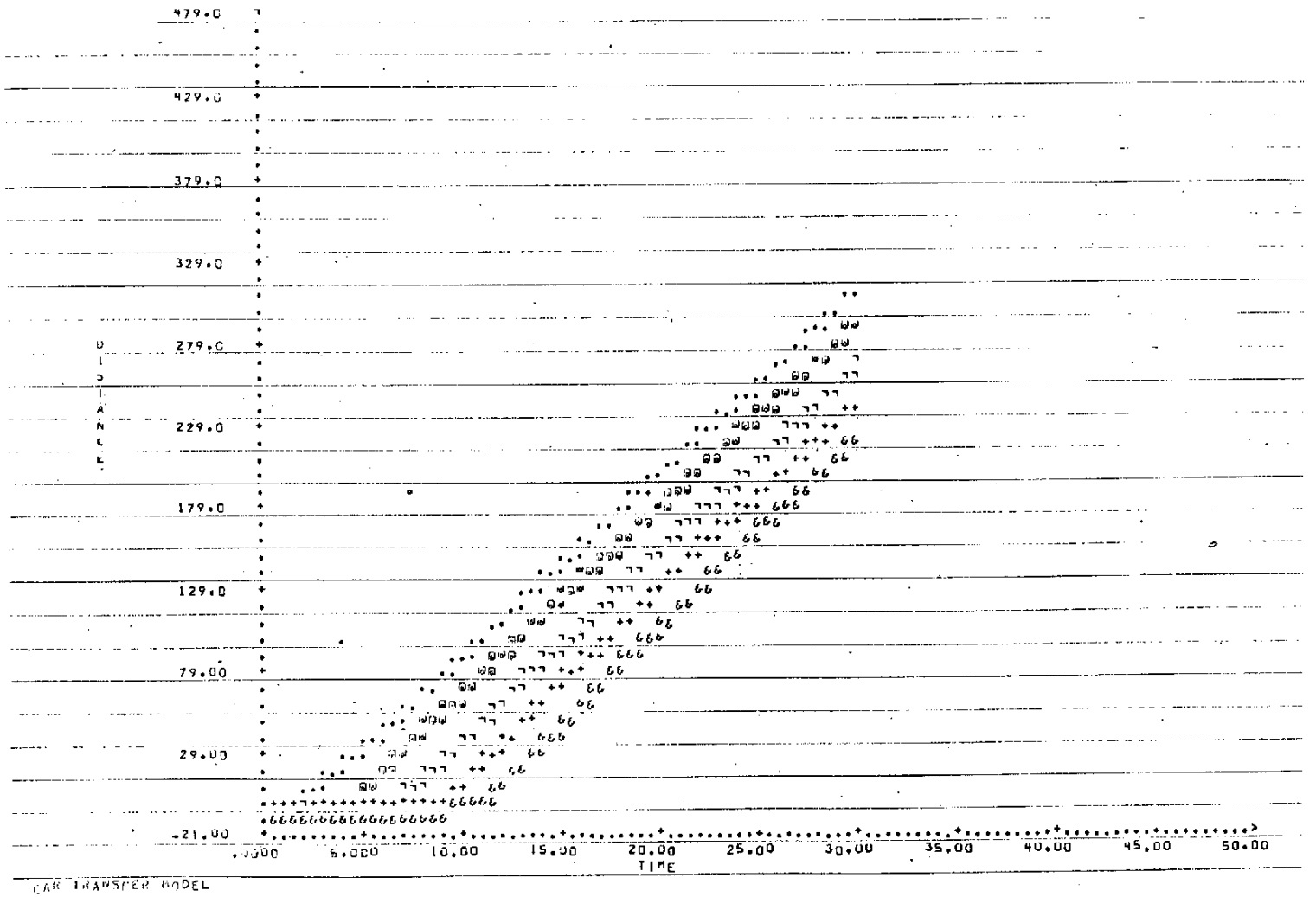


図5－2　ミクロモデルのMIMICグラフ出力

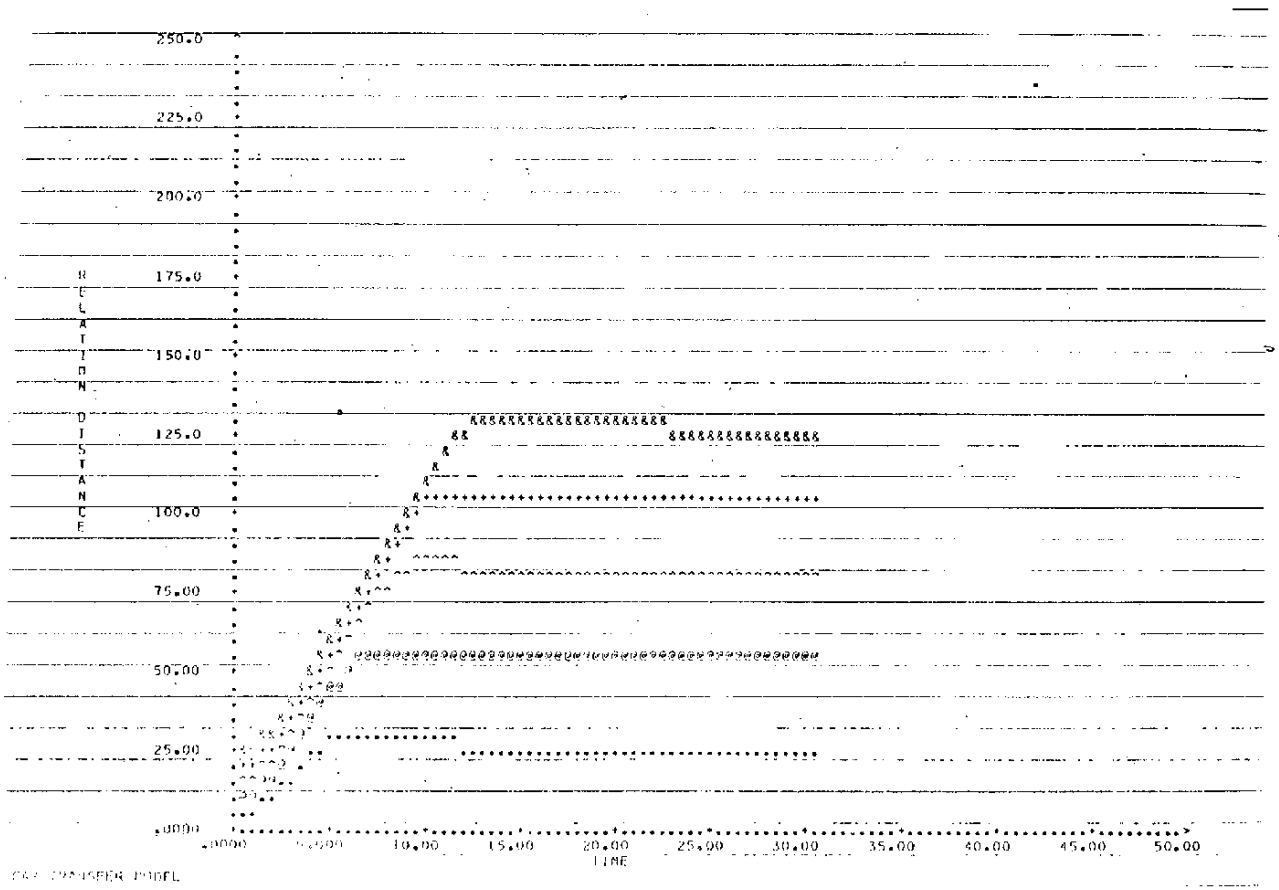


図５－２　ミクロモデルのMIMICグラフ出力（車間距離）

しいものとなっている。この点を補うためにMIMICはサブプログラム（FORTRANのサブルーチンに対応するもの）を組むことを許しているが，これとて同一のパターンを記述することはできない。

以上，MIMICとCSSL-3，CSMP-3の相違点について述べてきたが，これらを表5－2に簡単にまとめておく。

表5－2　MIMICとCSMP-3，CSSL-3の相違点

着　目　点	MIMIC	CSMP-3，CSSL-3
プログラムの構造	単一のセクションでありシンプルな構造	INITIAL，DYNAMIC，TERMINAL の3つのセクションに分けられる。
プロセッサー	インタープリター	FORTRANプリコンパイラー
function block	ラプラス変換を扱う機能が小さい。 ランプ，ステップ関数，パルス，インパルスジェネレーターなし	
選択関数 arbitrary function	独立変数2つまで線型補間データとして与える	CSSL-3では3つ，CSMP-3では線型，2次，3次補間と選択できる。 プログラム中に任意に与えられる。
積　分　法	可変のルンゲ・クッタ法	いくつかのアルゴリズムの中から選択できる。
シミュレーション ラン・コントロール	パラメータをデータとして与えデータの個数によりコントロール	ランコントロールセクションで変数の値を変えることができる
シミュレーション スタディ	不　可	可　能
レポート作成	ランタイムに出力 グラフは座標軸の指定要 X－Yプロッタ出力可	ランタイム，一括出力両方可 CSMP-3では自動的にグラフを出力する。 X－Yプロッタ出力可
マクロ機能	無	有

5.3 CSMP－3．vs．CSSL－3

5.3.1 プログラム構造上の相違点

CSMP－3のプログラムは図5－3のようにイニシャルセクション，ダイナミックセクション，ターミナルセクションの3つから構成される。一回のランはイニシャルセクションにおいて初期設定を行ない，ダイナミックセクションにコントロールを渡す。ダイナミックセクションにおいては，FINTIM或はFINISHステートメントの条件が満足されるまで，シミュレーションを行ない，これらの条件が満足されると，コントロールをターミナルセクションに渡しシミュレーションスタディを行なったり，一括出力を行なったりする。

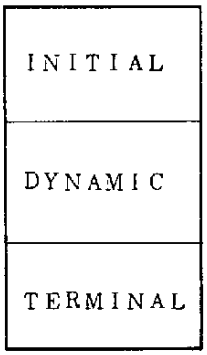


図5－3　CSMP－3のプログラム構造

一方，CSSL－3のプログラム構造にはインプリシット（implicit）構造とエクスプリシット（explicit）構造の二つの形式がある。エクスプリシット構造のプログラムは，複数個のセグメントが書け，一つのセグメントはCSMP－3の一つのプログラムに対応しており，このセグメントを単位として独立変数の指定ができる。又CSSL－3のダイナミックセクションには，複数個のデリバティブ（Derivative）が書けるがこのデリバティブという概念は，CSMP－3のダイナミックに相当するものであり，CSSL－3ではこのデリバティブを単位として積分アルゴリズム，積分ステップ・サイズを与えることができる。従ってCSSL－3ではいくつかのプロセスをコンカレントにシミュレートすることができる。

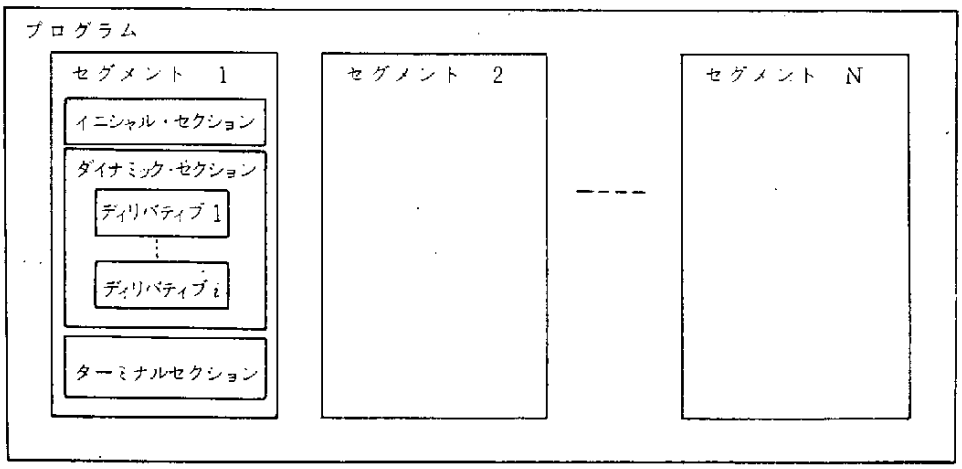


図5－4　CSSL－3のイクスプリシット構造

一方，CSSL-3のインプリシット構造は1つのセグメントから成り，イニシャル，ダイナミック，ターミナルセクションの枠を取り除いたものである。従ってMIMICの様にシンプルな構造であり，簡単にプログラムが作れる。

リスト5−3はCSMP-3，リスト5−4はCSSL-3によるミクロモデルのプログラムである。

プログラムの構造には直接関係ないが，ソーティング機能の違いについてふれておきたい。MIMICではプログラム全体をソートするが，（CSSL-3のインプリシット構造では，プログラム全体がダイナミックセクションと見なされソーティングの対象となる）CSSL-3，CSMP-3では，ダイナミックセクションのみがソーティングの対象となる。ソートしてほしくないステートメントは図5−5，図5−6の様にNO SORT，又はPROCEDURAL～ENDとすれば良い。

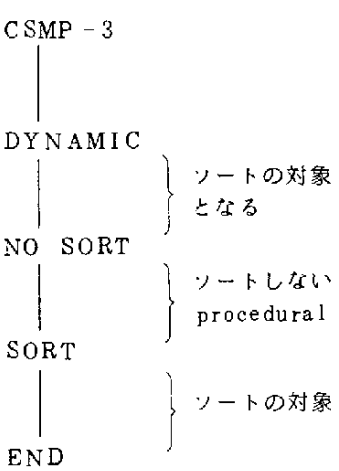


図5−5

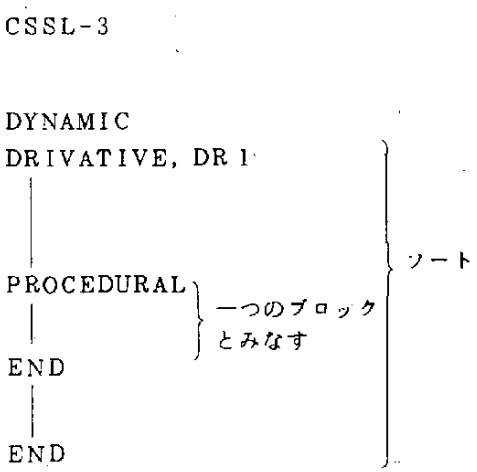


図5−6

5.3.2 プロセッサーについて

CSSL-3，CSMP-3ともにFORTRANプリコンパイラーであり，オブジェクトとしてFORTRANステートメントを作り出す。従ってFORTRANライブラリーとのリンクが可能であり，プログラム中にFORTRANステートメントを書くことができる。又FORTRANサブルーチンを用意すれば，これとコミュニケーションを取ることもできる。

CSMP-3では，ソースプログラムをFORTRANステートメントにコンパイルしてUPDATEというファイルを作り，システム変数はこのUPDATEとは別にファイルされる。従ってラン毎に積分アルゴリズムやコミュニケーションインターバルの変更が許される。一方CSSL-3ではこの様な変更はできるが，CSMP-3の様にファイルの変更という方法とは異なり，積分アルゴリズムやコミュニケーション・インターバルを任意の変数に値を与え，これをパラメータとして行なわれるのである。

5.3.3 function blockについての多少の相違点

function blockはCSMP-3，CSSL-3ともに同じ様な機能をもっているが，〔表5－4，表5－5参照〕多少違いがある。それらを表5－3にまとめてみた。

表5－3　CSMP-3，CSSL-3のFUNCTION BLOKの相違点

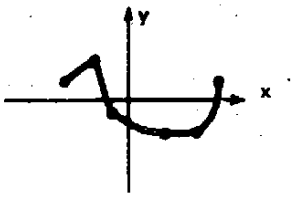
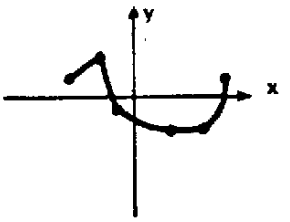
着目点	CSMP-3	CSSL-3
任意選択関数	Arbitrary function 独立変数は2つまで取れる 線型 補間法 二次 三次 ポイント数の指定不要	Function Table 独立変数は3つまで取れる 線型補間法 ポイント数の指定必要
LOADING FUNCTION	有	無
論理演算子	NOT, AND, NOTAND, INCLUSIVE OR, EXCLUSIVE OR, NOT OR, EQUIVALENT	NOT AND, AND, INCLUSIVE OR,

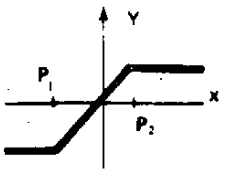
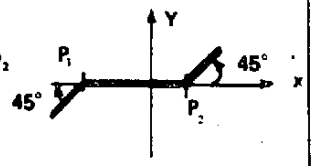
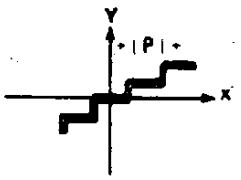
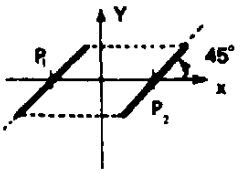
LOADING FUNCTIONとはArbitrary functionの一種であるが，あらかじめデータとして座標を与えるのではなく，計算結果をポイントとの指定として用いるfunction Generatorである。

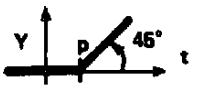
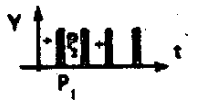
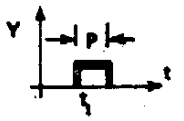
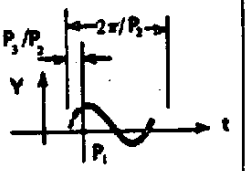
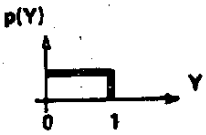
表5－4　CSMP－3 function block ライブラリ

CSMP III Statement	Equivalent Mathematical Expression
INTEGRATOR Y = INTGRL (IC, X) where: IC = $y\|_{t=t_s}$ Alternative 'Specification' Form: Y = INTGRL (IC, X, N) where: Y = output array IC = initial condition array X = integrand array N = number of elements in the integrator array	$y(t) = \int_{t_s}^{t} x dt + y(t_s)$ where: t_s = start time t = time $\overset{*}{y} = \int_{t_s}^{t} \overset{*}{x} dt + \overset{*}{y}(t_s)$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = \frac{1}{s}$
DERIVATIVE Y = DERIV (IC, X) where: IC = $\frac{dx}{dt}\Big\|_{t=t_s}$	$y = \frac{dx}{dt}$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = s$
* **1ST ORDER LAG (REAL POLE)** Y = REALPL (IC, P, X) where: IC = $y\|_{t=t_s}$	$p\frac{dy}{dt} + y = x$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = \frac{1}{ps+1}$
* **LEAD–LAG** Y = LEDLAG (P1, P2, X)	$p_2\frac{dy}{dt} + y = p_1\frac{dx}{dt} + x$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = \frac{p_1 s+1}{p_2 s+1}$
* **2ND ORDER LAG (COMPLEX POLE)** Y = CMPXPL (IC1, IC2, P1, P2, X) where: IC1 = $y_{t=t_s}$ IC2 = $\frac{dy}{dt}\Big\|_{t=t_s}$	$\frac{d^2y}{dt^2} + 2p_1p_2\frac{dy}{dt} + p_2^2 y = x$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = \frac{1}{s^2 + 2p_1p_2s + p_2^2}$
* **GENERAL LAPLACE TRANSFORM** Y = TRANSF (N, B, M, A, X) STORAGE TABLE B(N + 1), A(M +1) B(1 – [N + 1]) = B(1), B(2) . . . . B(N + 1), A(1 – [M + 1]) = A(1), A(2), ... A(M+1)	$\frac{Y(s)}{X(s)} = \frac{\sum_{i=1}^{m} a_i s^i + a_{m+1}}{\sum_{j=1}^{n} b_j s^j + b_{n+1}}$ where: $m \leq n$

CSMP III Statement	Equivalent Mathematical Expression
DEAD TIME (DELAY) Y = DELAY (N, P, X) where P = delay time N = number of points sampled in interval p (integer constant) and must be ≥ 3, and $\leq 16,378$	$y = x(t-p)$; $t \geq p$ $y = 0$; $t < p$ Equivalent Laplace Transfer Function: $\frac{Y(s)}{X(s)} = e^{-ps}$
ZERO-ORDER HOLD Y = ZHOLD (X1, X2)	$y = x_2$; $x_1 > 0$ y = last output ; $x_1 \leq 0$ $y_{\mid t=t_0} = 0$ Equivalent Laplace Transfer Function $\frac{Y(s)}{X(s)} = \frac{1}{s}(1 - e^{st})$
* **MODE-CONTROLLED INTEGRATOR** Y = MODINT (IC, X1, X2, X3)	$y = \int_{t_0}^{t_f} x_2 dt + ic$; $x_1 > 0$, any x_2 $y = ic$; $x_1 \leq 0$, $x_2 > 0$ y = last output ; $x_1 \leq 0$, $x_2 \leq 0$
* **VARIABLE FLOW TRANSPORT DELAY** Y = PIPE (N, IC, P, X1, X2, ND) where: N = number of intervals necessary to define holdup quantity IC = initial condition of entire pipeline P = holdup quantity X1 = flow rate X2 = delayed characteristic ND = degree of interpolation for retrieving delayed characteristic. May be 1 or 2.	$fv = \int_{t_0}^{t} x_1 dt$ $q = \int_{t_0}^{t} x_2 x_1 dt$ $y = ic$; $fv < p$ $y = \frac{q(fv-p) - q(fv(t-\Delta t)-p)}{fv - fv(t-\Delta t)}$; $fv \geq p$ where: fv = flow volume q = weighted volume
IMPLICIT FUNCTION Y = IMPL (IC, P, FOFY) where IC = first guess P = error bound FOFY = output name from final statement in algebraic loop definition	$y = f(y)$ $\lvert y - f(y) \rvert \leq p \lvert y \rvert$

CSMP III Statement	Equivalent Mathematical Expression
ARBITRARY FUNCTION GENERATOR (LINEAR INTERPOLATION) Y = AFGEN (FUNCT, X)	 y = f(x)
ARBITRARY FUNCTION GENERATOR (QUADRATIC INTERPOLATION) Y = NFLGEN (FUNCT, X)	 y = f(x)
FUNCTION GENERATOR WITH DEGREE OF INTERPOLATION CHOSEN BY USER Y = FUNGEN (FUNCT, N, X) where: FUNCT = function name N = degree of interpolation to be used. May be 1, 2, 3, 4, or 5 X = value of abscissa	 y = f(x)
ARBITRARY FUNCTION OF 2 VARIABLES Y = TWOVAR (FUNCT, X, Z)	y = f(x,z)
SLOPE OF A CURVE Y = SLOPE (FUNCT, N, X) where: FUNCT = name of curve N = degree of interpolation to be used X = value of abscissa	$y = \frac{df}{dx}$ at x

CSMP III Statement	Equivalent Mathematical Expression
LIMITER Y = LIMIT (P1, P2, X)	$y = p_1 : x < p_1$ $y = p_2 : x > p_2$ $y = x : p_1 \leq x \leq p_2$ 
DEAD SPACE Y = DEADSP (P1, P2, X)	$y = 0 : p_1 \leq x \leq p_2$ $y = x - p_2 ; x > p_2$ $y = x - p_1 ; x < p_1$ 
QUANTIZER Y = QNTZR (P, X)	 $y = kp : (k - \frac{1}{2})\,p < x \leq (k + \frac{1}{2})\,p$ $k = 0, \pm 1, \pm 2, \pm 3, \ldots$
HYSTERESIS LOOP Y = HSTRSS (IC, P1, P2, X) where: IC = $y|_{t_1}$	 $y = x - p_2 : (x - x(t - \Delta t)) > 0$ and $y(t - \Delta t) \leq (x - p_2)$ $y = x - p_1 : (x - x(t - \Delta t)) < 0$ and $y(t - \Delta t) \geq (x - p_1)$ otherwise: $y = y(t - \Delta t)$

CSMP III Statement	Equivalent Mathematical Expression
STEP FUNCTION Y = STEP (P)	$y = 0 \;;\; t < p$ $y = 1 \;;\; t \geq p$ 
RAMP FUNCTION Y = RAMP (P)	$y = 0 \;;\; t < p$ $y = t - p \;;\; t \geq p$ 
IMPULSE GENERATOR Y = IMPULS (P1, P2)	$y = 0 \;;\; t < p_1$ $y = 1 \;;\; (t - p_1) = kp_2$ $y = 0 \;;\; (t - p_1) \neq kp_2$ $k = 0, 1, 2, 3, \ldots$ 
PULSE GENERATOR (WITH X > 0 AS TRIGGER) Y = PULSE (P, X) where: P = minimum pulse width	$y = 1 \;;\; t_1 \leq t < (t_1 + p)$ or $x > 0$ $y = 0$; otherwise (t_1 = time of trigger) 
TRIGONOMETRIC SINE WAVE WITH DELAY, FREQUENCY AND PHASE PARAMETERS Y = SINE (P1, P2, P3) where: P1 = delay P2 = frequency (in radians per unit time) P3 = phase shift in radians	 $y = 0 \;;\; t < p_1$ $y = \sin(p_2 (t - p_1) + p_3) \;;\; t \geq p_1$
NOISE (RANDOM NUMBER) GENERATOR WITH NORMAL DISTRIBUTION Y = GAUSS (N, P1, P2) where: N = any odd integer P1 = mean P2 = standard deviation	Normal Distribution of Variable y p(y) = probability density function 
NOISE (RANDOM NUMBER) GENERATOR WITH UNIFORM DISTRIBUTION Y = RNDGEN (N) where: N = any integer	Uniform Distribution of Variable y p(y) = probability density function 

CSMP III Statement	Equivalent Mathematical Expression
SAMPLING INTERVAL SWITCH Y = SAMPLE (P1, P2, {P3 / N}) where: P_1 = start time for sampling to occur P_2 = last time for sampling to occur P_3 = time interval between samples if entered as a floating-point number N = number of sampling intervals if entered as a fixed-point number	y = 1.0 ; TIME = $p_1 + k\ p_3 \leq p_2$ k = 0, 1, 2, . . . or y = 1.0 ; TIME = $p_1 + k(p_2 - p_1) / n \leq p_2$ k = 0, 1, 2. y = 0.0 otherwise

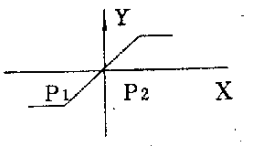
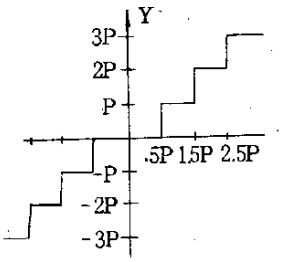
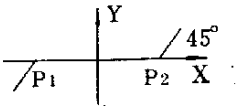
CSMP III Statement	Equivalent Mathematical Expression
SCALAR-TO-ARRAY CONVERTOR CALL ARRAY (V1, V2, . . . , VN, X)	x(1) = v1 x(2) = v2 x(3) = v3 x(n) = vn
ARRAY-TO-SCALAR CONVERTOR Y1, Y2, . . . , YM = SCALAR (X(2))	y1 = x(2) y2 = x(3) ym = x(m + 1)
DOUBLE PRECISION FLOATING-POINT TO SINGLE PRECISION where: DNUMBR is a double precision floating-point number. Y = rounded single precision value of DNUMBR	To convert the double precision floating-point number, DNUMBR, to single precision, rounding to hexadecimal digit.

CSMP III Statement	Equivalent Mathematical Expression
COMPARATOR Y = COMPAR (X1, X2)	$y = 0;$ $x_1 < x_2$ $y = 1;$ $x_1 \geq x_2$
OUTPUT SWITCH Y1, Y2 = OUTSW (X1, X2)	$y_1 = x_2, y_2 = 0;$ $x_1 < 0$ $y_1 = 0, y_2 = x_2;$ $x_1 \geq 0$
INPUT SWITCH RELAY Y = INSW (X1, X2, X3)	$y = x_2;$ $x_1 < 0$ $y = x_3;$ $x_1 \geq 0$
RESETTABLE FLIP-FLOP Y = RST (X1, X2, X3)	$y = 0;$ $x_1 > 0$ $y = 1;$ $x_2 > 0, x_1 \leq 0$ $y = 0;$ $x_1 < 0,$ $x_3 > 0, y(t - \Delta$ $y = 1;$ $x_1 < 0,$ $x_3 > 0, y(t - \Delta t) = 0$ $y = 0;$ $x_2 < 0,$ $x_3 \leq 0, y(t - \Delta t) = 0$ $y = 1;$ $x_3 \leq 0, y(t - \Delta t) = 1$
FUNCTION SWITCH Y = FCNSW (X1, X2, X3, X4)	$y = x_2;$ $x_1 < 0$ $y = x_3;$ $x_1 = 0$ $y = x_4;$ $x_1 > 0$

CSMP III Statement	Equivalent Mathematical Expression	
NOT Y = NOT (X)	y = 1 ; y = 0 ;	$x \leq 0$ $x > 0$
AND Y = AND (X1, X2)	y = 1 ; y = 0 ;	$x_1 > 0, x_2 > 0$ otherwise
NOT AND Y = NAND (X1, X2)	y = 0 ; y = 1 ;	$x_1 > 0, x_2 > 0$ otherwise
INCLUSIVE OR Y = IOR (X1, X2)	y = 0; y = 1 ;	$x_1 \leq 0, x_2 \leq 0$ otherwise
EXCLUSIVE OR Y = EOR (X1, X2)	y = 1 ; y = 1 ; y = 0 ;	$x_1 \leq 0, x_2 > 0$ $x_1 > 0, x_2 \leq 0$ otherwise
NOT OR Y = NOR (X1, X2)	y = 1 ; y = 0;	$x_1 \leq 0, x_2 \leq 0$ otherwise
EQUIVALENT Y = EQUIV (X1, X2)	y = 1 ; y = 1 ; y = 0;	$x_1 \leq 0, x_2 \leq 0$ $x_1 > 0, x_2 > 0$ otherwise

CSMP－3 マニュアルより一部掲載

表5－5　CSSL－3 function blockライブラリ

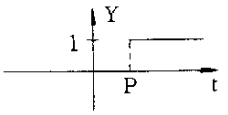
COMPARATOR Y = COMPAR(X1,X2)	Y = 0. for X1＜X2 = 1. for X1≧X2
FUNCTION SWITCH Y = FCNSW(P,X1,X2,X3)	Y = X1 for P＜0. = X2 for P＝0. = X3 for P＞0.
INPUT SWITCH Y = SWIN(P,X1,X2)	Y = X1 for P＜0. = X2 for P≧0.
LIMITER Y = BOUND(P1,P2,X)	Y = P1 for X＜P1 = P2 for X＞P2 = X for P1≦X≦P2 
QUANTIZER Y = QNTZR(P,X) P = Value of the quantization interval	Quantizer function is defined by the input output characteristics as shown. 
DEAD SPACE Y = DEAD(P1,P2,X)	Y = 0. for P1≦X≦P2 = X-P2 for X＞P2 = X-P1 for X＜P1 

STEP FUNCTION

Y = STEP(P,T)

Y = 0. for t<P

= 1. for t≥P

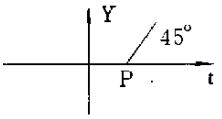


RAMP FUNCTION

Y = RAMP(P,T)

Y = 0. for t<P

= t-P for t≥P



PULSE GENERATOR

Y = PULSE(TZ,P,W,T)

TZ = start time

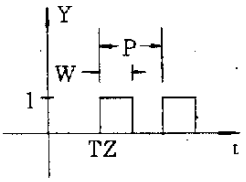
P = period

W = pulse width

T = independent variable

Y = 0. for T<TZ

= pulse train as illustrated for T≥TZ



HARMONIC FUNCTION

Y = HARM(P1,P2,P3,T)

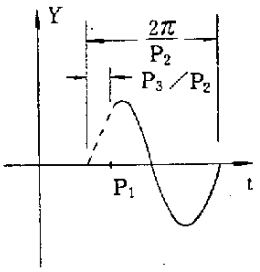
P1 = delay in seconds

P2 = frequency in radians/sec.

P3 = phase shift in radians

Y = 0. for t<P1

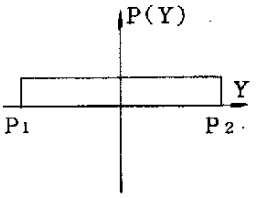
SIN(P2*(t-P1)+P3) for t≥P1



UINFORM DISTRIBUTION

Y = UNIF(P1,P2)

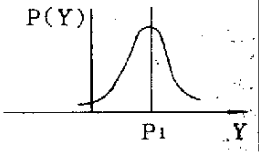
Y = a random number sequence uniformly distributed between P1 and P2.



NORMAL DISTRIBUTION

Y = GAUSS(P1,P2)

Y = a random number sequence with Gaussian distribution having mean P1 and standard deviation P2.



FIRST ORDER LAG

Y = REALPL(P,X,IC)

Y(0) = IC

$P\dot{Y} + Y = X$

EQUIVALENT LAPLACE TRANSFORM:

$$\frac{1}{PS + 1}$$

SECOND ORDER LAG

Y = CMPXPL(P,Q,X,IC1,IC2)

$\dot{Y}(0)$ = IC1

Y(0) = IC2

$P\ddot{Y} + Q\dot{Y} + Y = X$

EQUIVALENT LAPLACE TRANSFORM:

$$\frac{1}{PS^2 + QS + 1}$$

LEAD-LAG Y = LEDLAG(P,Q,X,IC) Y(O) = IC	$Q\dot{Y} + Y = P\dot{X} + X$ EQUIVALENT LAPLACE TRANSFORM: $\frac{PS + 1}{QS + 1}$
RESOLVER Y1,Y2 = PTR(X1,X2) Y1,Y2 = RTP(X1,X2)	Y1 = X1* COS(X2) Polar to rectangular Y2 = X1* SIN(X2) Y1 = SQRT ($X1^2 + X2^2$) Rectangular to polar Y2 = TAN^{-1}(X2/X1)
LIMITED INTEGRATOR Y = LIMINT(D,IC,LB,UB) D = derivative of Y IC = initial value of Y LB = lower bound on Y UB = upper bound on Y	The LIMINT operator should be used in place of the BOUND function when the output of an integrator is to be hard limited.
NOT AND Q = NANDL(X1,X2,...) X_i = any number of variables	Q = 1.0 if any input is zero = 0.0 otherwise (Inputs greater than 0.5 are considered to be 1.).

AND Q = ANDL(X1,X2,...) X_i = any number of variables	Q = 1.0 if all inputs are one = 0.0 otherwise (Inputs greater than 0.5 are considered to be 1.).
INCLUSIVE OR Q = ORL(X1,X2,...) X_i = any number of variables	Q = 1.0 if one or more inputs are one = 0.0 otherwise (Inputs greater than 0.5 are considered to be 1.).
SINGLE STEP DELAY Y = SDELAY(IC,X,TS) X = input TS = storage area which must have a unique name and be dimensioned for 8	Y = IC initially; otherwise = last valid value of X

MULTIPLE STEP DELAY	
Y = DELAY(IC,N,X,TS) X = input N = number of steps of delay TS = storage area which must have a unique name and be dimensioned for 8*N	Y = IC initially and for the first N-1 steps; otherwise = last valid value of X.
ZERO ORDER HOLD	
Y = ZHOLD(IC,P,X) X = input P = sampling switch	Y = IC initially if the initial P = 0. = last value of Y when P was one if P = 0. = X if P = 1.

DERIVATIVE FUNCTION

Y = DERIVT(IC,X,T,TS)

T = independent variable

X = input

TS = storage area which must have a unique name and be dimensioned for 15.

Y = IC initially

$= \frac{dX}{dt}$ otherwise

(The derivative is approximated by $\Delta X/\Delta T$, where the Δ's extend over the past step).

BAND LIMITED WHITE NOISE

Y = OU(TAU,T,P1,P2)

TAU = time constant

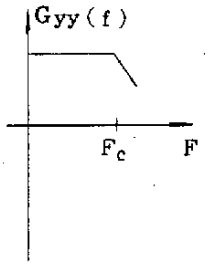
T = independent variable

P1 = RMS value of Y

P2 = average value of Y

fc is the cutoff frequency (break frequency) of the signal which is set by TAU where:

$$fc = \frac{1}{2\pi TAU}$$



EXPONENTIAL FUNCTION

Y = EXPF(IC,TAU,T,ON)

IC = initial value (0. to 1.)

TAU = time constant

T = independent variable

ON = exponential switch:

ON > .5, exponential increases:

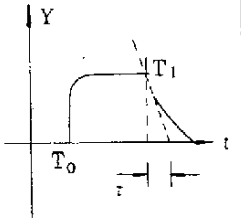
ON < .5, it decays

$Y = 1.-e^{(T_o/\tau)}$

where T_o = time when ON switched from 0. to

$1.-e^{(-T_1/\tau)}$

where T_1 = time when ON switched from 1. to 0.



OPERATOR

COMMENTS

HYSTERESIS LOOP - TYPE 1

Y = HSTRSS(IC,P1,P2,X)

IC = initial condition on Y

P1 = leftmost X intercept

P2 = rightmost X intercept

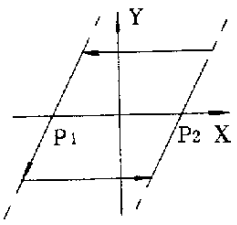
X = input

Y_n = IC initially

= X-P2 if $Y_{n-1} \leq$ X-P2

= X-P1 if $Y_{n-1} \leq$ X-P1

= Y_{n-1}, last output



HYSTERESIS LOOP - TYPE 2

Y = HSTRSS(IC,P1,P2,S,X)

IC = initial condition on X

P1 = leftmost X intercept

P2 = rightmost X intercept

S = slope of transition region

If $X_n > X_{n-1}$:

$Y_n = Y_{n-1} + S(X_n - Y_{n-1})$ or

$= X_n - P1$, whichever is smaller.

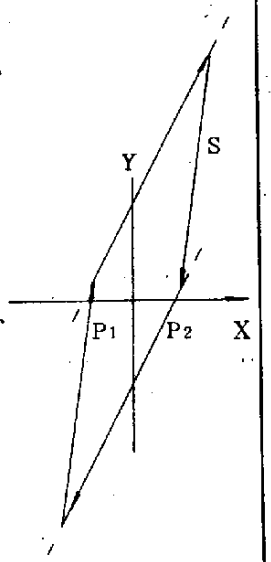
If $X_n < X_{n-1}$:

$Y_n = Y_{n-1} + S(X_n - X_{n-1})$ or

$= X_n - P2$, whichever is larger.

If $X_n = X_{n-1}$:

$Y_n = Y_{n-1}$



CSSL-3 マニュアルより一部掲載

5.3.4　実行時のコントロールに関する方法とパラメータの比較

(1)　積分法に関するコントロール

CSMP－3，CSSL－3は各々表5－6，表5－7に挙げてある積分法から，任意にそのアルゴリズムを選択でき，可変ステップの積分法に関してはきざみ幅の最大値，最小値，最大許容誤差を指定することができる。又CSSL－3，LSMP－3ともにダミーのルーチンがあり，ユーザの用意した積分法にも対処できる。

表5－6　CSMPの積分法

NAME	METHOD
ADAMS	固定増分，2次のADAMS積分法
CENTRAL	ユーザが用意する積分法に対するダミールーチン
MILNE	可変増分，5次予測修正子MILNE法
RECT	長方形積分法
RKS	可変ステップ，4次のRunge－Kutta法
RKSDP	倍長精度の可変4次Runge－Kutta法
RKSFX	固定増分4次のRunge－Kutta法
SIMP	シンプソン法
STIFF	STIFF方程式による方法
TRAPZ	Trapezoidal積分法

CSMP－3ではMETHODステートメントにより，積分アルゴリズムを指定し，ランによって積分法を変えたい時はデータとして与える。

図5－7の例では最初のランは4次のルンゲ・クッタ法によって行なわれ，二回目のランはMILNE法によって行なわれる。

CSSL－3では，任意の変数名＝Algorithm Numberとすることにより，積分アルゴリズムを与え，ランによって変更したい時は，データとして与えれば良い。又CSSL－3ではデリバティブ単位に積分法を与えることができる。

```
METHOD  RKS

END
METHOD  MILNE
END
```

図5－7

プログラムの構造のところでもふれたが，CSSL－3では，セグメント単位に独立変数を指定することができ，これがCSSL－3の最大の特徴の一つになっている。独立変数の指定は図5－7の様にVARIABLEステートメントによって行なわれる。

表5－7　CSSL－3の積分法

Algorithm	Algorithm Number	Step Size	No. of Previous Values Needed
Euler	1	Fixed	0
Trapezoidal	2	Fixed	1
Four-Point Predictor	3	Fixed	3
Adams-Moulton	4	Fixed	3
Runge-Kutta-Gill (4th order)	5	Fixed	0
User Supplied	6	--	--
User Supplied	7	--	--
Runge-Kutta/Moulton	8	Variable	Self-Starting
Runge-Kutta/Milne	9	Variable	Self-Starting

〔CSSL－3マニュアル〕

図5－8の例では；SEGMENT 1に対しては独立変数をXとしてその初期値は1.0に取り，SEGMENT 2に対しては，Yを独立変数として初期値を0.0に取り，積分を行なうのである。

なお，積分のコントロールということについては関係がないが，積分のアルゴリズムについてその方法を紹介しておく。

```
PROGRAM
    SEGMENT  SEG  1
      |
    DYNAMIC
      VARIABLE  X=1.0
      |
    END
    SEGMENT  SEG  2
      |
      DYNAMIC
      VARIABLE  Y=0.0
      :
    END
END
```

図5－8　セグメント単位に対する独立変数の指定

1. **Milne Fifth-Order Predictor-Corrector Predictor:**

$$Y^{p}_{t+\Delta t} = Y_{t-\Delta t} + \frac{\Delta t}{3}\left(8.X_t - 5X_{t-\Delta t} + 4.X_{t-2\Delta t} - X_{t-3\Delta t}\right)$$

Corrector:

$$Y^{c}_{t+\Delta t} = \frac{1}{8}\left(Y_t + 7.Y_{t-\Delta t}\right) + \frac{\Delta t}{192}\left(65.X_{t+\Delta t} + 243.X_t + 51.X_{t-\Delta t} + X_{t-2\Delta t}\right)$$

Estimate:

$$Y_{t+\Delta t} = .96116.Y^{c}_{t+\Delta t} + .03884.Y^{p}_{t+\Delta t}$$

Integration interval control is based on the following criteria:

$$\frac{|Y^c - Y^p|}{R \cdot |Y^c| + A} \leq 1$$

2. **Runge-Kutta (Fourth Order)**

$$Y_{t+\Delta t} = Y_t + \frac{1}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)$$

$$K_1 = \Delta t \cdot f\left(t, Y_t\right) \qquad K_3 = \Delta t \cdot f\left(t + \frac{\Delta t}{2}, Y_t + \frac{K_2}{2}\right)$$

$$K_2 = \Delta t \cdot f\left(t + \frac{\Delta t}{2}, Y_t + \frac{K_1}{2}\right) \qquad K_4 = \Delta t \cdot f\left(t + \Delta t, Y_t + K_3\right)$$

ただし，可変ステップの場合は次の条件を満足する様に
インターバルを小さくして行く。

$$\frac{|Y_{t+\Delta t} - Y^s|}{A + R \cdot |Y_{t+\Delta t}|} \cong \frac{\text{Error}}{A + R \cdot |Y_{t+\Delta t}|} \leq 1$$

3. Method for handling stiff equations

Given the differential equation
$y = f(t,y)$, $y(t_0)=y_0$

the solution is computed by the following explicit step-by-step procedure:

1) $\dot{y}_A(t) = [y(t) - y(t-h_0)]/h_0$

2) $d_1 = \dot{y}(t) - \dot{y}_A(t)$

3) $y_p(t+\delta) = y(t) + \delta\dot{y}(t)$ where $\delta \leq h/4$

4) $\dot{y}_p(t+\delta) = f[t+\delta, y_p(t+\delta)]$

5) $d_2 = [\dot{y}_p(t+\delta) - \dot{y}(t)]/\delta$

$$\lambda = \begin{cases} d_2/d_1, d_1 \neq 0, \\ 0 \quad d_1 = 0, \end{cases} \qquad c_1 = \begin{cases} (e^{\lambda h} - 1)/\lambda h & \lambda < 0, \\ 1 + \lambda h/2 & \lambda \geq 0 \end{cases}$$

6)
$$c_0 = \begin{cases} e^{\lambda h} & \lambda < 0 \\ 1 + \lambda h & \lambda \geq 0 \end{cases}$$

7) $y_c(t+h) = y(t) + h\dot{y}_A(t) + hc_1d_1$

8) $\dot{y}_c(t+h) = f[t+h, y_c(t+h)]$

9) $E = h[\dot{y}_c(t+h) - (\dot{y}_A(t) + c_0d_1)]$

10) $E_e = 2\delta\,[\dot{y}_p(t+\delta) - \dot{y}(t)]$

Note that y, y_C, y_p, y_A, y_{pe}, d_1, d_2, c_0, c_1, λ are vector quantities.

Step 1	To start up $h_0 = 0$ $\dot{y}_A(0) = \dot{y}(0)$
Step 2	y is assumed to be the sum of an asymptotic part and a perturbation from the asymptote.
Steps 3 and 4	These steps constitute Euler integration with step size δ.
Step 5	$d_2 \approx \ddot{y}(t)$. It is assumed that the form of f makes calculation of $\ddot{y}$ difficult.
Step 7	y_c is computed solution at end of current step; h is stepsize.
Step 9	E is used to monitor step-size h.
Step 10	E_e is used to monitor Euler step-size.

4. Adams-Second Order

$$Y_{t+\Delta t} = Y_t + \frac{\Delta t}{2}\left(3 \cdot \dot{Y}_t - \dot{Y}_{t-\Delta t}\right)$$

5. Simpson's Rule

Predictor:

$$Y^P_{t+\frac{\Delta t}{2}} = Y_t + \frac{\Delta t}{2} X_t$$

$$Y^P_{t+\Delta t} = Y^P_{t+\frac{\Delta t}{2}} + \frac{\Delta t}{2} X_{t+\frac{\Delta t}{2}}$$

Corrector:

$$Y^C_{t+\Delta t} = Y_t + \frac{\Delta t}{6}\left(X_t + 4X_{t+\frac{\Delta t}{2}} + X_{t+\Delta t}\right)$$

6. Trapezoidal

Predictor:

$$Y^P_{t+\Delta t} = Y_t + \Delta t \cdot X_t$$

Estimate:

$$Y_{t+\Delta t} = Y_t + \frac{\Delta t}{2} \cdot \left(X_t + X_{t+\Delta t}\right)$$

7. Rectangular

$$Y_{t+\Delta t} = Y_t + \Delta t \cdot X_t$$

Note: In these equations, the common terminology is $X_t \equiv \dot{Y}_t \equiv f(t)$. A value $X_{t+\Delta t}$ used in the estimation is based on the prediction $Y^P_{t+\Delta t}$.

（以上CSMP－3マニュアルより一部掲載）

(2) コミュニケーション・インターバルに関するパラメータ

CSMP-3ではTIMERステートメント，CSSL-3ではCINTERVALステートメントにより，コミュニケーション・インターバルを指定することができる。

CSSL-3	CSMP-3
CINTERVAL CI=1.0	TIMER PDEL=1.0, OUTDEL=2.0

図5－9　コミュニケーション・インターバルに関する指定

CSSL-3では，ランタイムの出力，一括出力ともに任意の変数名CIの値により行なわれるのに対し，CSMP-3ではシステム変数PDELの値によりランタイムの出力がOUTDELよりの値によりcase終了時の一括出力が行なわれる。

（caseについては，シミュレーション・スタディに関する項で説明する。）

(3) ランの終了に関するコントロール

CSSL-3では，TERMTステートメント及びSTPCLKステートメントにより，シミュレーションランの打ち切りが行なわれる。

```
TERMT ( T.GE. 20 )
STPCLK  10
```

図5－10　CSSL-3のラン終了に関するコントロール

図5－10の例では，独立変数Tが20 secに到した場合或はCPUタイムが10 sec に到した時コントロールがターミナルセクションに渡されランが終了となる。

一方CSMP-3ではTIMERステートメント内にシステム変数FINTIMに終了時刻をアサインする方法と，FINISHステートメントによる指定方法がある。FINISHステートメントによるランの終了指定では，指定された変数がある値に到した場合にコントロールがターミナルセクションに渡される。

5.3.5　ランのコントロール方法とシミュレーション・スタディに関すること

CSMP-3，CSSL-3では，パラメータの変更を行ない，複数回のランを一回のジョブとして行なえるが，これは変更すべきパラメータとその値をデータとして与えることによりコントロールされる。

図5－11の例で，ソースプログラムの実行が終りENDと出会うと1回のランが終る。次にP＝(1.0, 2.0, 3.0)というファイルされているデータを実行するのであるが，P＝1.0として1回

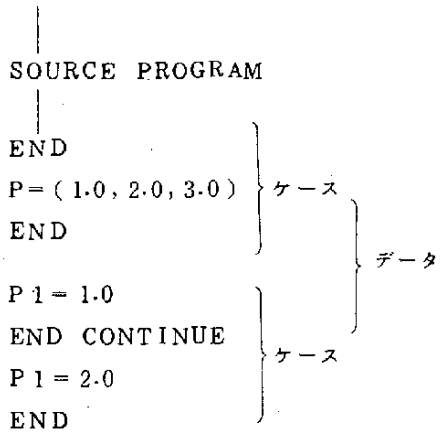


図5－11　CSMP-3ランのコントロール方法

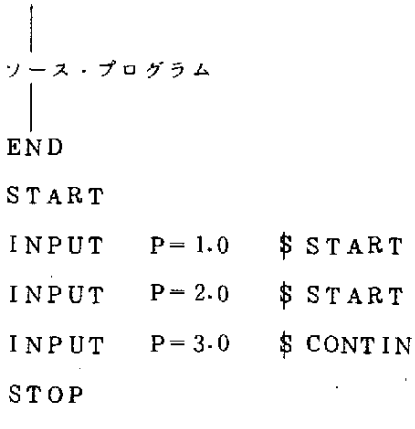


図5－12　CSSL-3ランのコントロール

のラン，P＝2.0，P＝3.0として各々1回のランと見なされる。これらのランが終るとENDステートメントと出会うが，CSMP-3ではENDステートメントとENDステートメントの間の複数回のランをケースと呼んでいる。次にP1＝1.0としてプログラムを実行しEND CONTINUEに出会い，P1＝2.0に変更し，END CONTINUEに出会った時刻から再びプログラムを実行し，これを1回のランと見なす。又CSMP-3ではシステム変数はソースプログラムとは別のファイルに登録されるため，これらを変更したい場合でもデータとして与えればよい。

一方CSSL-3ではINPUTステートメントにより，パラメータの値を変更しランをコントロールする。

図5－12の例で，最初のSTARTによりプログラムが実行され1回目のランを行なう。次にINPUT P＝1.0 $ STARTでP＝1.0としてコントロールをイニシャルセクションに渡し，2回目のランを行なう。さらにP＝2.0としてイニシャルセクションを実行し，終了条件が満たされると再びINPUT P＝3.0 $ CONTINに出会い，Pの値を3.0として前の終了時刻から再び実行する。即ち，コントロールをダイナミックセクションにもどすのである。CSSL-3では実行のコントロール――積分アルゴリズムやコミュニケーションインターバルに関するコントロール――は（任意変数名＝constant）という形式で与えるため，これらを変更してランをくり返すことも容易に行なえる。

シミュレーション・スタディとは，ターミナル・セクションにシミュレーションを最適解に近づける様なパラメータの変更をプログラムしておき，コントロールをイニシャル・セクションにもどすのであるが，CSSL-3ではGO TO文，CSMP-3ではCALL RERUNステートメントによって行なうことができる。

5.3.6　レポート作成に関する機能

レポート作成に関しては，CSMP-3 の方がはるかに大きな機能を備えている。表5−8はCS−SL-3，CSMP-3のレポート作成に関する機能をまとめたものである。

表5−8

	CSMP−3	CSSL−3
ランタイムにおけるプリントアウト	PRINTステートメント コミュニケーション間隔指定可 TITLEをつけられる	OUTステートメント コミュニケーション間隔は OUT PUTと同じ
一括出力	CASE終了時に出力 OUT PUT ステートメントアレイの出力も容易 LABELをつけられる PAGEにより，フォーマットの指定ができる	ラン終了時に出力 OUT PUTステートメント VOUTステートメント
グラフ出力	5変数以下のOUT PUT ステートメントでは自動的にグラフを出力する。〔図5−13参照〕 PAGEステートメントにより濃淡図〔図5−15〕，等高線図〔図5−16〕がかける。	プログラム中にPREPAREステートメントによりグラフ出力を指定，STARTステートメントの後にPLOTステートメントを置く。 〔図5−14〕
レンジレポート	指定された変数の最大値，最小値及びその時の時刻を出力する。	
X−Y・プロッター	使　用　可	

図 5－13　"SHADE" Output from Diffusion Model

（CSMP－Ⅲ マニュアルより一部掲載）

図 5－14　"CONTOR" Output from Diffusion Model

（CSMP－Ⅲ マニュアルより一部掲載）

5.3.7　マクロ機能

CSMP－3，CSSL－3ともにマクロ機能を持ち，同一のパターンを容易に記述することができる。図5－15はマクロの定義のしかたと呼び出し方である。

CSSL－3では，マクロ機能を充分に発揮させるためマクロディレクティブステートメントが用意されている。マクロディレクティブ・ステートメントはインプット・アーギュメント・リストによりFORTRANステートメントもしくはCSSL－3 ステートメントを作り出すのをコントロールするものであり次の様なものがある。

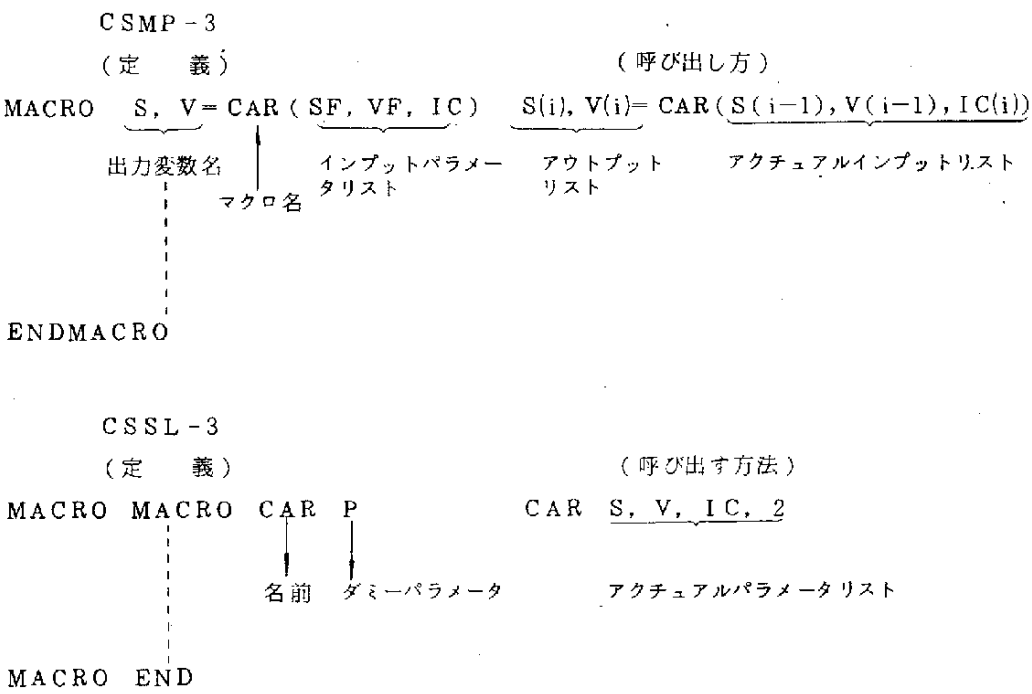


図5－15　マクロの定義と呼び出し方

MACRO RELABEL label〔, label〕……

マクロの内部のレーベルを，ローカルに定義する。

MACRO REDEFINE variable〔, variable〕……

マクロ内部でローカルに定義したい変数を指定する。

MACRO GOTO S

レーベルSのマクロディレクティブステートメントへの無条件ブランチ。

MACRO CONTINUE

レーベルをつけるためのダミー用のディレクティブステートメント。

MACRO EXIT
マクロのローカル・ターミネーションに用いる。

MACRO ASSIGN N
変数Nにマクロを呼んだ時のパラメータの数が入る。

MACRO INCREMENT i
Nにiをプラスする。

MACRO DECREMENT i
Nからiを引く。

MACRO MULTIPLY i
NをN＊iとする。

MACRO DIVIDE i
N／iとする

MACRO IF $e_1 = e_1$ S
$e_1 = e_2$となった時レーベルSのディレクティブステートメントへジャンプ

MACRO STANDVAL V(i)= e_i 〔, V(i)= e_i〕 ……
MACROを呼ぶ時パラメータが省略されている場合数値を直接アサインするために用いる。

MACRO PRINT charactor string
アルファニュメリックなキャラクターの出力に用いる。

このマクロ・ディレクティブ・ステートメントにより，CSSL-3のマクロは非常に柔軟性に富んだものとなり，マクロのNestingを可能になっている。

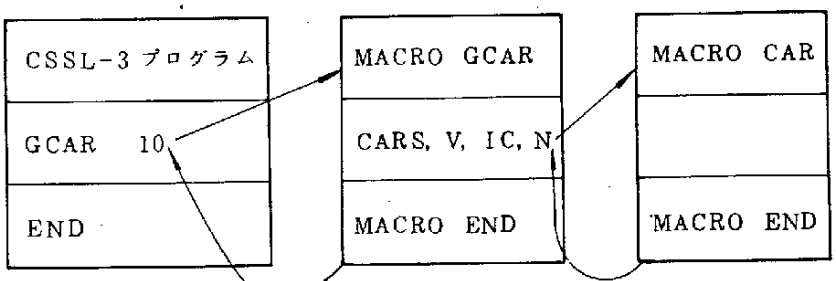


図5－16　CSSL-3マクロのNesting

上の図5－16の例で，プログラム中でGCARというマクロを呼ぶと，マクロGCARがそのパラメータの値だけ，CARを呼び結局プログラム中でCARというマクロを10回呼んだと同じになる。

CSMP-3では，マクロのネスティングはできないけれど，これに対応する機能として'INTEGRATOR ARRAYS'がある。これはインプット変数，アウトプット変数，及び初期値としてアレイを用いて行なうことを許したものであり，

Z = INTGRL (IC, ZDOT, 50)

とすると，STORAGE Z(50)，IC(50),ZDOT(50) というストレージが自動的に取られZD-OTに対する積分器が50個取られる。

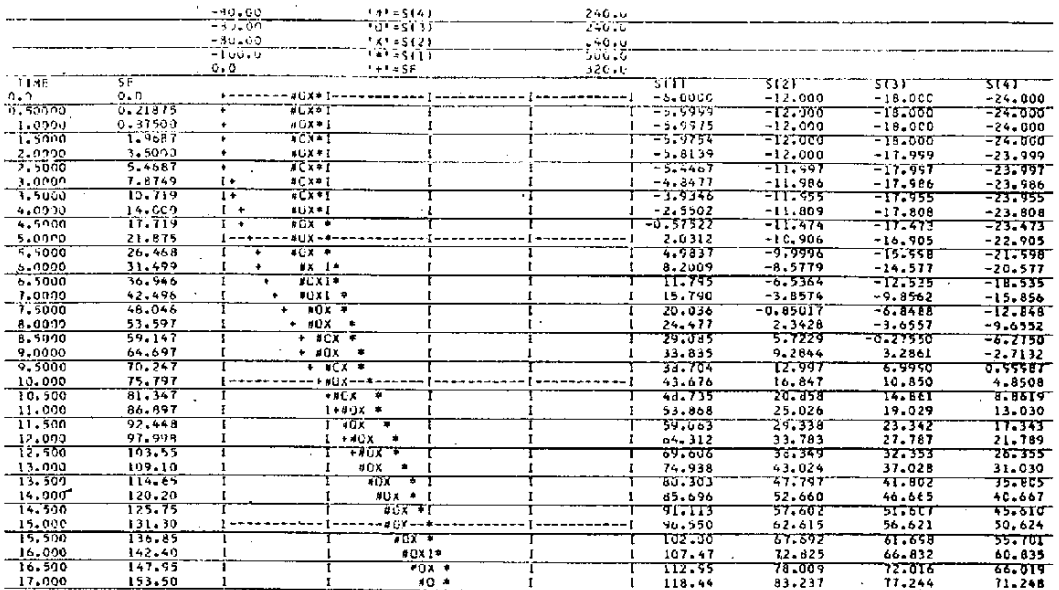
5.3.8 その他の機能

CSMP－3，CSSL－3についてのほとんどの機能は即に書きつくしたと思うが，5.3.1～5.3.7の項目に入らなかったものとして次のことを上げておく。

CSMP－3には，プログラムを部分的にファイルする機能があり，このファイルされたプログラムにシンボリックなキーを与えることにより，別のプログラムで呼び出すことができる。大規模なシステムをいくつかのサブシステムに分けそれらのプログラムをファイルし，後にこのファイルを用いてシステム全体のシミュレーションを行なうことが可能である。又CSMP－3にはグラフィック・ディスプレイ2250を用いて会話型CSMP－3がある。これについては次節で詳しくふれたいと思う。

リスト５－３　ミクロモデル（CSMP－３）のソースリストと結果

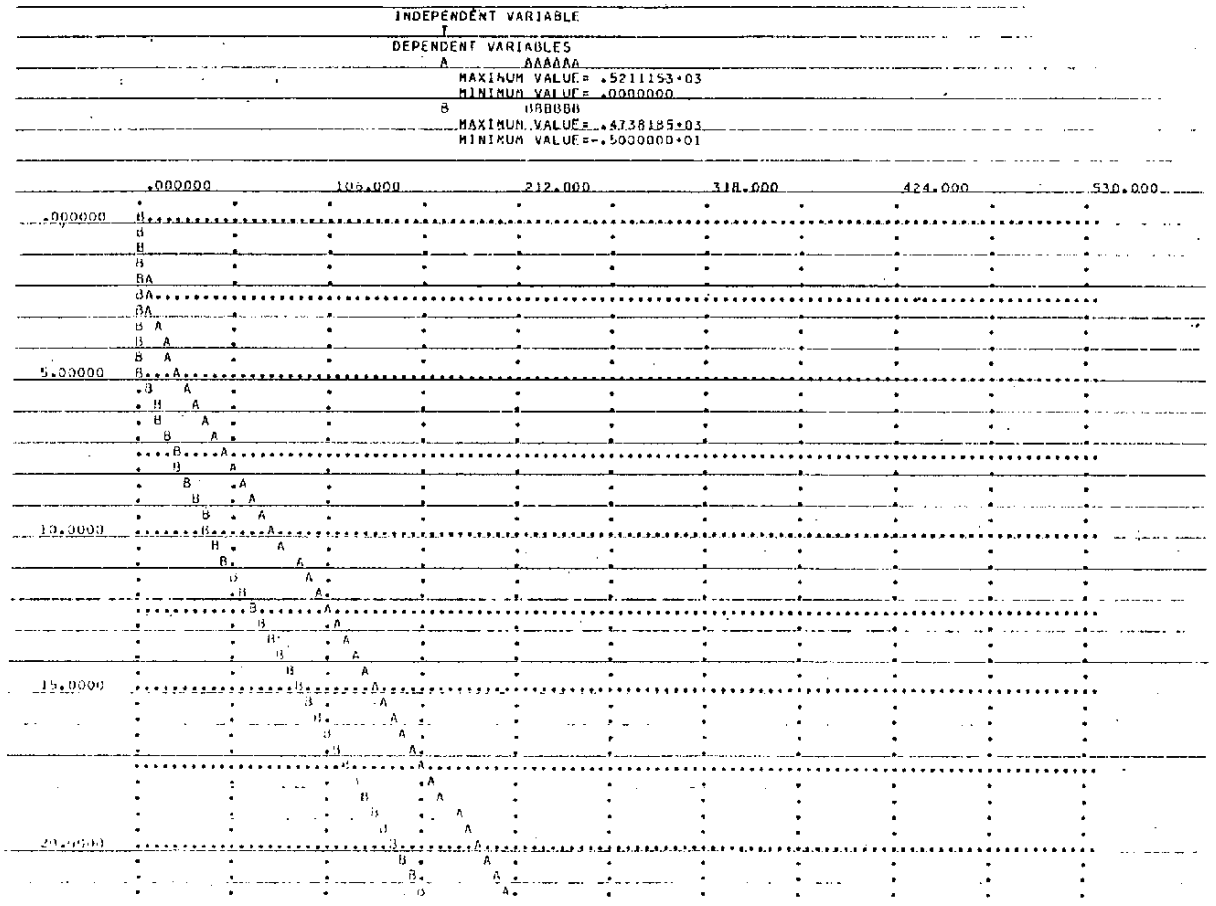
```
$$$CONTINUOUS SYSTEM MODELING PROGRAM  III   V1MC   TRANSLATOR OUTPUT$$$

FIXED I
FUNCTION  FACR  =  -15.0,    -4.0,    ...
                   -10.0,    -1.0,    ...
                   -5.0,     -0.5,    ...
                   5.0,      0.5,     ...
                   10.0,     1.0,     ...
                   15.0,     4.0
PARAMETER   K1=1.75,K2=11.1,K3=3.0,K4=1.2,V2=5.0
INITIAL
          NOSORT
          DO 1 I=1,9
          V0(I)=0.0
          S0(I)=-6.0*FLOAT(I)
        1 CONTINUE
DYNAMIC
          VF=INTGRL(0.,DX2)
          SF=INTGRL(0.,VF)
          DX2=INSW(VF-K2,K1,0.)
          V=INTGRL(V0,ACR,9)
          S=INTGRL(S0,V,9)
PROCEDURE ACR=CARMV(VF,SF)
          STD=INSW(VF-K1,6.0,V(1)*K4)
          DEFZ=SF-S(1)-STD
          DEF=INSW(V(1)-V2,DEFZ,(VF-V(1))*K3)
          ACR(1)=DELAY(3,0.25,AFGEN(FACR,DEF))
          DO 2 I=2,9
          STD=INSW(V(I-1)-K1,6.0,V(I)*K4)
          DEFZ=S(I-1)-S(I)-STD
          DEF=INSW(V(I)-V2,DEFZ,(V(I-1)-V(I))*K3)
          ACR(I)=DELAY(3,0.25,AFGEN(FACR,DEF))
        2 CONTINUE
ENDPROCEDURE
PRINT     SF,S(1-9),DX2,ACR(1-9)
OUTPUT     SF,S(1-4)
OUTPUT     S(5-9)
TIMER     FINTIM=30.0,DELT=0.5,PRDEL=0.5,OUTDEL=0.5
END
STOP

OUTPUT VARIABLE SEQUENCE
ZZ1000 DX2     VF      SF      ACR     V       S

$$$ TRANSLATION TABLE CONTENTS $$$      CURRENT          MAXIMUM

   MACRO AND STATEMENT OUTPUTS          13                600
   STATEMENT INPUT WORK AREA            45               1900
   INTEGRATORS+MEMORY BLOCK OUTPUTS      4 +  0           300
   PARAMETERS+FUNCTION GENERATORS        9 +  1           400
   STORAGE VARIABLES+INTEGRATOR ARRAYS   0 +  2/2          50
   HISTORY AND MEMORY BLOCK NAMES       21                50
   MACRO DEFINITIONS AND NESTED MACROS   6                50
   MACRO STATEMENT STORAGE              13               125
   LITERAL CONSTANT STORAGE              0               100
   SORT SECTIONS                         1                20
   MAXIMUM STATEMENTS IN SECTION        15               600
```

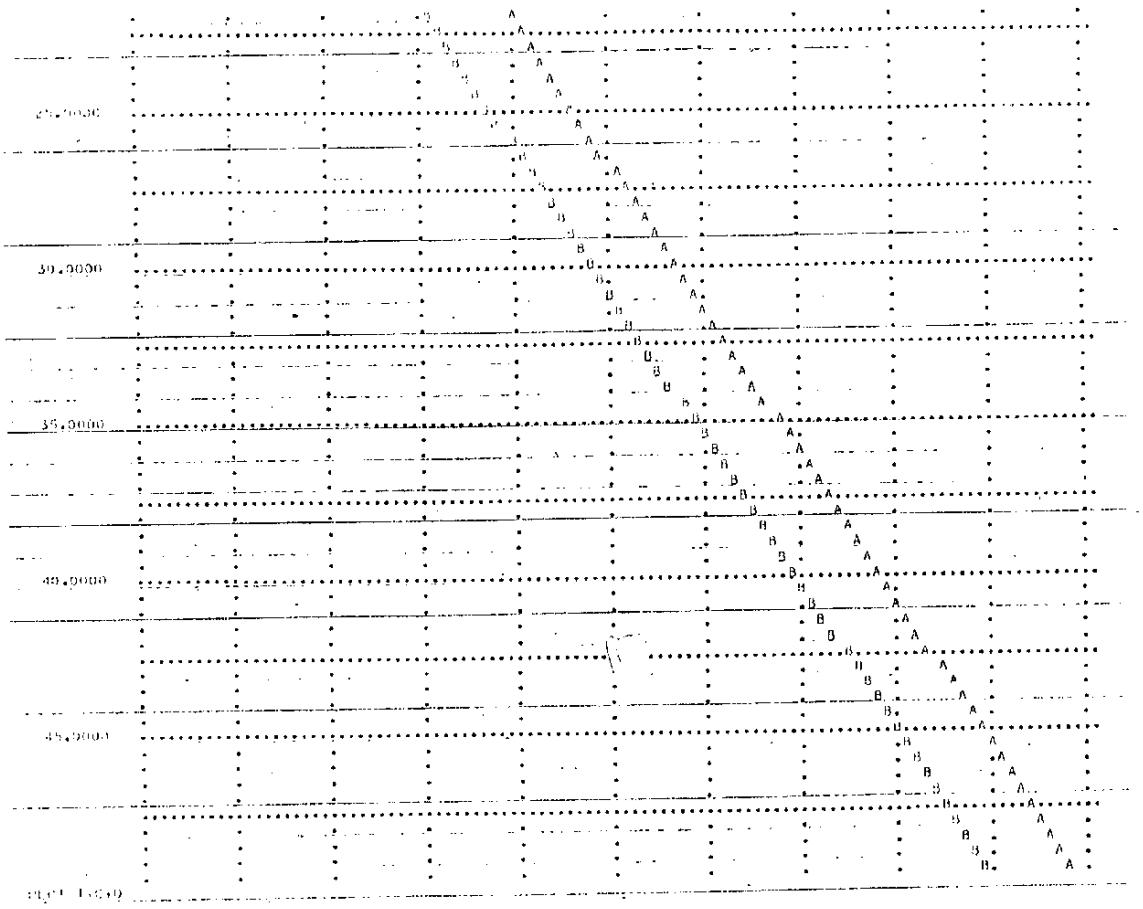


TIME	SF	S(1)	S(2)	S(3)	S(4)
0.0	0.0	-6.0000	-12.000	-18.000	-24.000
0.50000	0.21875	-5.9909	-12.000	-18.000	-24.000
1.0000	0.87500	-5.9375	-12.000	-18.000	-24.000
1.5000	1.9687	-5.8754	-12.000	-18.000	-24.000
2.0000	3.5000	-5.8139	-12.000	-17.999	-23.999
2.5000	5.4687	-5.4467	-11.997	-17.997	-23.997
3.0000	7.8749	-4.8477	-11.986	-17.986	-23.986
3.5000	10.719	-3.9346	-11.955	-17.955	-23.955
4.0000	14.009	-2.5902	-11.809	-17.808	-23.808
4.5000	17.719	-0.57522	-11.474	-17.473	-23.473
5.0000	21.875	2.0312	-10.906	-16.905	-22.905
5.5000	26.468	4.9837	-9.9996	-15.998	-21.998
6.0000	31.499	8.2009	-8.5779	-14.577	-20.577
6.5000	36.946	11.745	-6.5364	-12.535	-18.535
7.0000	42.796	15.790	-3.8574	-9.8562	-15.856
7.5000	48.046	20.036	-0.85017	-6.8488	-12.848
8.0000	53.597	24.477	2.3428	-3.6557	-9.6552
8.5000	59.147	29.085	5.7229	-0.27550	-6.2750
9.0000	64.697	33.835	9.2844	3.2861	-2.7132
9.5000	70.247	38.704	12.997	6.9950	0.95587
10.000	75.797	43.676	16.847	10.850	4.8508
10.500	81.347	48.735	20.858	14.861	8.8619
11.000	86.897	53.868	25.026	19.029	13.030
11.500	92.448	59.063	29.338	23.342	17.343
12.000	97.998	64.312	33.783	27.787	21.789
12.500	103.55	69.606	38.349	32.353	26.355
13.000	109.10	74.938	43.024	37.028	31.030
13.500	114.65	80.303	47.797	41.802	35.805
14.000	120.20	85.696	52.660	46.665	40.667
14.500	125.75	91.113	57.602	51.617	45.610
15.000	131.30	96.550	62.615	56.621	50.624
15.500	136.85	102.00	67.692	61.698	55.701
16.000	142.40	107.47	72.825	66.832	60.835
16.500	147.95	112.95	78.009	72.016	66.019
17.000	153.50	118.44	83.237	77.244	71.248
17.500	159.05	123.94	88.504	82.512	76.516
18.000	164.60	129.45	93.807	87.815	81.819
18.500	170.15	134.96	99.140	93.149	87.153
19.000	175.70	140.48	104.50	98.510	92.514
19.500	181.25	146.01	109.88	103.89	97.899
20.000	186.80	151.54	115.29	109.30	103.30
20.500	192.35	157.07	120.71	114.72	108.73
21.000	197.90	162.60	126.15	120.16	114.17
21.500	203.45	168.14	131.61	125.62	119.62
22.000	209.00	173.68	137.07	131.09	125.09
22.500	214.55	179.22	142.55	136.56	130.57
23.000	220.10	184.76	148.04	142.05	136.06
23.500	225.65	190.30	153.53	147.55	141.55
24.000	231.20	195.85	159.04	153.05	147.05
24.500	236.75	201.39	164.54	158.56	152.56
25.000	242.30	206.94	170.06	164.07	158.08
25.500	247.85	212.48	175.58	169.59	163.60
26.000	253.40	218.03	181.10	175.11	169.12
26.500	258.95	223.58	186.62	180.64	174.65
27.000	264.50	229.13	192.16	186.17	180.18
27.500	270.05	234.67	197.69	191.70	185.71
28.000	275.60	240.22	203.22	197.23	191.24
28.500	281.15	245.77	208.76	202.77	196.78
29.000	286.70	251.32	214.30	208.31	202.31
29.500	292.25	256.87	219.84	213.84	207.85
30.000	297.80	262.42	225.38	219.38	213.39

リスト５－４　ミクロモデル（ CSSL－3 ）のソースリストと結果

```
PROGRAM DELAY OF STARTING
      INITIAL
      ARRAY S(20),V(20),TS(16,20)
      ARRAY IC(20)
      DO L1 I=1,19
      IC(I+1)=-5.0*I
L1..CONTINUE
      CONSTANT V1=2.7777,V2=5.5,P1=3.0
      CONSTANT K1=1.75,K2=11.111
      CONSTANT TX=0.5
      CINTERVAL CI=0.5
      NSTEPS NST=2
      ALGORITHM IA=5,JA=5
      TABLE ACR,1,6,-15.,-10.,-5.,5.,10.,15.,-4.,-1.,-0.5,0.5,1.,4.
      END
         DYNAMIC
         OUTPUT K1,K2
         VOUT S,V
       A=S(1) $ B=S(2) $ C=S(3) $ D=S(4) $ F=S(5)
       PREPAR A,B,C,D,F
         TERMT(T.GE.TX)
                  DERIVATIVE DRI
                  V(1)=INTEG(DX2,0.)
                  S(1)=INTLG(V(1),0.)
                 DX2=FCNSW(V(1)-K2,K1,0.,0.)
                  CAR S,V,IC,2
                  CAR S,V,IC,3
                  CAR S,V,IC,4
                  CAR S,V,IC,5
                  CAR S,V,IC,6
                  CAR S,V,IC,7
                  CAR S,V,IC,8
                  CAR S,V,IC,9
                  CAR S,V,IC,10
                  CAR S,V,IC,11
                  CAR S,V,IC,12
                  CAR S,V,IC,13
                  CAR S,V,IC,14
                  CAR S,V,IC,15
                  CAR S,V,IC,16
                  CAR S,V,IC,17
                  CAR S,V,IC,18
                  CAR S,V,IC,19
                  CAR S,V,IC,20
                  END
         END
      TERMINAL
      END
      MACRO MACRO CAR P
      MACRO REDEFINE SHK,STD,DEFZ,DEF,KSD,IC,RV
          SHK=P(1)(P(4)-1)-P(1)(P(4))
           STD=FCNSW(P(2)(P(4)-1)-K1,5.0,5.0,P(2)(P(4))*1.2)
          DEFZ=SHK-STD
           RV=P(2)(P(4)-1)-P(2)(P(4))
           DEF=FCNSW(P(2)(P(4))-V2,DEFZ,0.,RV*P1)
          KSD=DELAY(0.,1,ACR(DEF),TS(1,P(4)))
          IC=P(3)(P(4))
          P(2)(P(4))=INTEG(KSD,0.)
          P(1)(P(4))=INTLG(P(2)(P(4)),IC)
      MACRO END
END
```


INDEPENDENT VARIABLE
DEPENDENT VARIABLES
A AAAAAA
MAXIMUM VALUE= .5211153+03
MINIMUM VALUE= .0000000
B BBBBBB
MAXIMUM VALUE= .4738185+03
MINIMUM VALUE=-.5000000+01
.000000
106.000
212.000
318.000
424.000
530.000
.000000
5.00000
10.0000
15.0000
20.0000


30,0000
35,0000
40,0000
45,0000

6．CSMP－IIIグラフィックヴァージョン

第6章　CSMP—Ⅲグラフィックヴァージョン

6.1　はじめに

IBMのCSMP-Ⅲは同社のグラフィックディスプレイ装置2250を使った会話型システムとしてデータセンターで利用出来る様に成っている。MIMIC，CSMP-Ⅲ，CSSL-Ⅲの比較の一環としてこのシステムを利用してみた。主な目的はSIMBOL-Ⅱの設計に当り，連続系シミュレーション言語サイドから見て会話型システムの備えるべき機能や問題点などを，検討する事であった。ただ現実にはこのシステムを使用する回数や時間にも制限があり隅々まで充分に検討したとは言い切れない。

6.2　会話型CSMP—Ⅲの概略

このシステムはバッチ用のCSMP-Ⅲとシミュレータ本体はほとんど同じで，これにグラフィックの機能を追加したものである。

従って会話型システムとは言ってもステートメントを一行一行インタープリティブに実行する様なシステムではない。グラフィックデスプレイからインプットされたステートメントは，ステートメントの構文チェックだけ行なわれ，正しいものはソースファイルとしてセーブされる。ソースプログラムが完成した段階でオブジェクトプログラムの作成を指示すると，CSMP-Ⅲのトランスレーター，FORTRANコンパイラー，リンクエディターが順に働きオブジェクトプログラムが出来上る。実行の指定があると，このプログラムが動きだす。唯，実行中にはクイット（quit）を掛ける事が出来，実行の中断やそのまま続行の指示が出来る。概略，こう言った感じのシステムで，第3章で述べたSPACEとプロセッサー自体の構造は似たシステムである。言はば後からグラフィックを使った会話型機能を加えたシステムである。

6.3　実際に使用して見て

このシステムを2回使用した。1回目は同センターで持っているデモンストレーション用のプログラムを見せてもらい，その後，こちらで用意した簡単なモデルを使って操作して見た。2回目の時は，CSMP-Ⅲ，CSSL-Ⅲ，MIMICの比較で使用した自動車速行モデルを使った。

それぞれ約15分間程度使用した。又，その時ソースプログラムはグラフィックディスプレイ端末からインプットはせず，あらかじめ，バッチのランでソースファイルを作成した後にこのファイル

を使用した。

以下にこの時の状況を会話型として特徴があると思われる項目にまとめて感想をまとめてみた。

(1) ソースプログラムのエディット

今回使用したモデルは前述した通り，前もってソースプログラムのファイルを作っておいてから会話型システムを使用した。

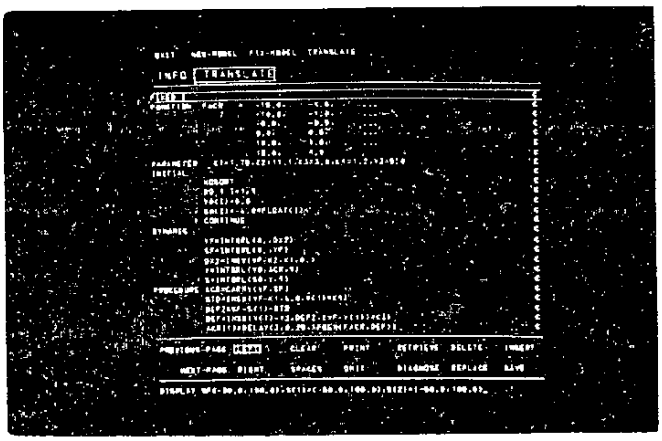


図６－１

図６－１は，ソースプログラムの一部分がディスプレイ画面に表示されている所で，右端にCと言う文字が表示されているラインがソースステートメントで，それ以外の部分はシステム側で現在のステータスを表示したり，指示を受け付ける為のコマンドのメニューである。又一番下のラインは新しくステートメントを追加する場合などキーボードからタイプインした情報を表示する部分で，この写真ではDISPLAYステートメントをキーインした時で表示されている。

ソースプログラムのエディットを行なうには，まず挿入しようと思う位置とか，修正しようと思うステートメントなどがある画面を表示する。CSMP-Ⅲのソースステートメントがディスプレイ画面に入らない場合，いくつかのページ単位に分割されていて，NEXT-PAGEとかPREVIOUS-PAGEの指示をすると見たい画面が表示出来る様になっている。これらの指示は，ソースプログラムのエディットについても同じであるが，画面の下側にこの為に必要なコマンドがリストアップされている。

その中には画面操作に関係するものも含めて，CLEAR，PRINT，RETRIEVE，DIELETE，INSERT，REPLACE，SAVEなどがある。ステートメントの修正には対象とするステートメント例えば，新しくステートメントを追加しようとする場合には挿入する位置の直前にあるステートメントをライトペンでピックする。すると，そのステートメントを知らせる為に画上下を直線ではさんで表示する。次の図６－２にこの様子が写されている。

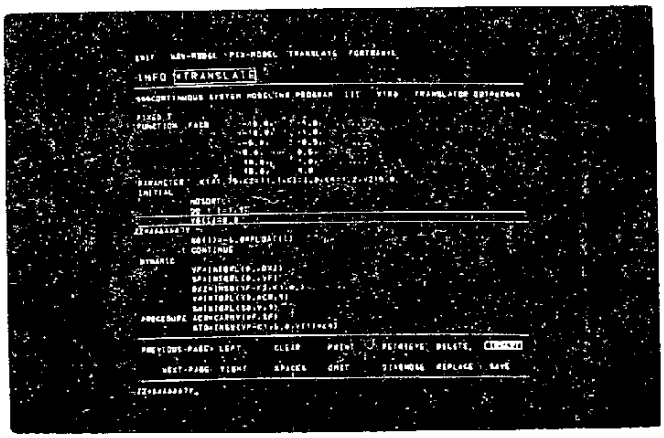


図６－２

その後，挿入すべきステートメントをキーボードからタイプインしてコマンドのメニューの中からＩＮＳＥＲＴをライトペンでピックすると指定の位置に挿入され画面上に表示される。図６－２の画面上には，わざとコンパイルエラーを起させる為に意味のないステートメントを入れた時の様子が表示されている。コマンドのメニューのどれかをピックするとそのむねを示す為にコマンドが四角い枠で囲まれる。

以上はステートメントを新しく追加する場合についてであるが，エディットをする場合，だいたいこの方法に準じている。即ち，エディットの対象は画面に表示されているものをライトペンでピックして指定する。どう言うエディットをするのかはコマンドのリストからやはりライトペンでピックして指示する。こう言った方法は，実際に操作してみると思った以上に使い安い事を感じた。

(2)　ファンクションプロセッサー（Function processor）

ＣＳＭＰ－Ⅲにはユーザーがデータ点を与えて函数を定義する機能がある。この様な函数を定義する場合，データを数値だけで扱っていると案外とまちがい安い。しかもその函数の様子が簡単につかみにくい。ファンクションプロセッサーと言うのは，こうした函数をディスプレイ画面にグラフとして図示したり函数の定義を変更したりする機能を持ったプロセッサーである。

ファンクションプロセッサーを呼び出して函数の名前をピックすると，その函数のグラフが表示される。次の図６－３は自動車速行モデルで使った，加速度を決める函数のグラフ表示である。このグラフに新しい点を追加してみた。

新しい点を追加する時にはグラフの一点をピックすると次の図６－４の様な８方向を持ったマークが表示される。次にマークの線分を再びピックするとその方向にマークが移動し，動いた位置まで線分が追加される。そこで下に表示されているコマンドリストのうちＡＤＤを指示すると

今引いた線分の端点が新しくこの函数に追加される。その様子が図６－５と図６－６に示してある。

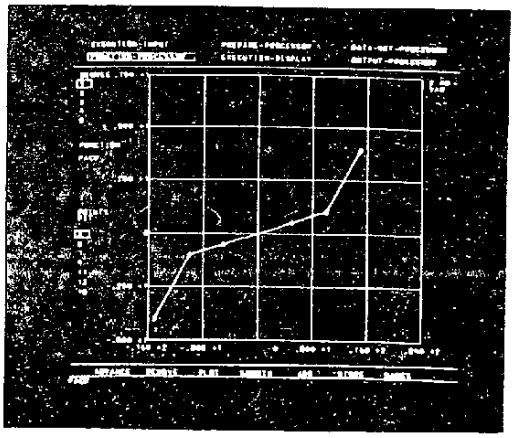


図６－３

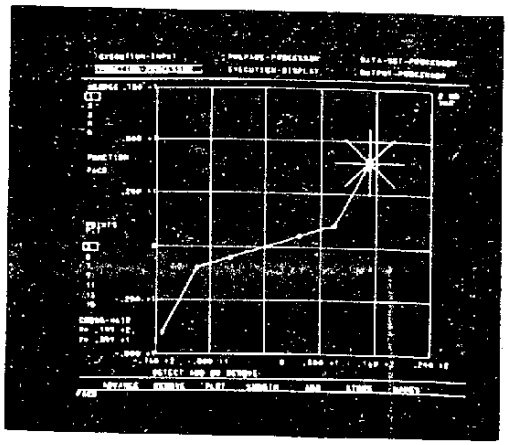


図６－４

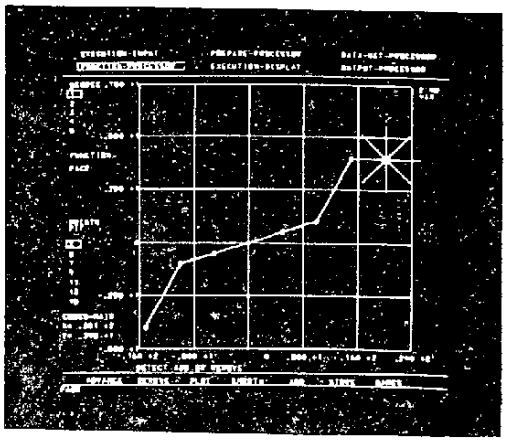


図６－５

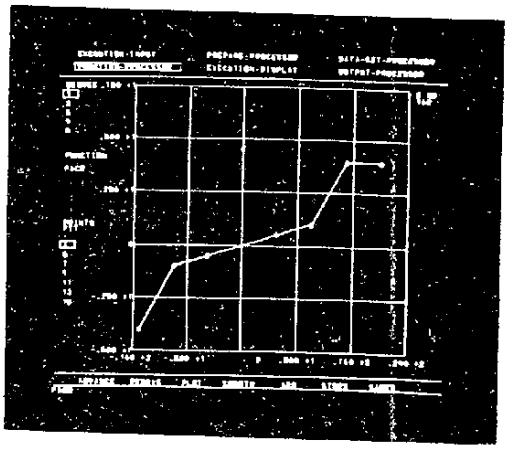


図６－６

函数を定義する時にはデータ点は飛び飛びにしか与えられないので，この函数をプログラムの中で参照する時には，途中の点を補間して値を求めなくてはならない。この為CSMP-Ⅲには直線補間と，２次から５次までの多項式で近似する機能がある。従来のバッチ用システムではこれらの補間が行なわれた時に実際どんな形の函数になったのかを視覚的に見る事が出来なかった。このグラフィックシステムの場合にはファンクションプロセッサーが補間したグラフを表示して見せてくれる。この機能などもグラフィックシステムの特徴を良く生かしていると思われる。次の図６－７と図６－８は同じデータ点を与えた函数を直線補間したものと２次曲線補間したものをグラフ表示しており，一見するとグラフの値が違う様だが，実は目盛のスケールが違っているだけで同じ函数である。

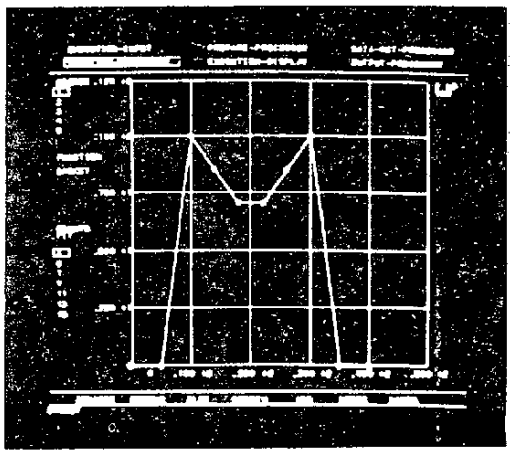


図６－７

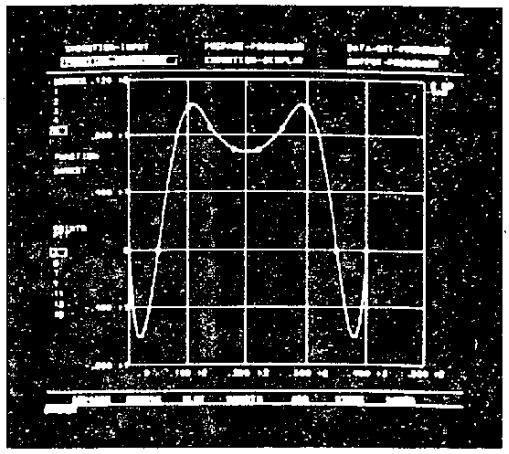


図６－８

ファンクションプロセッサーの機能は函数の表示とその修正をグラフィカルに行なう事で，こうした機能は連続系シミュレーション言語だけでなく，離散系シミュレーション言語の場合にも非常に有効な事である。いかにもグラフィックディスプレイの性格が生かされている。

(3)　シミュレーション実行中の表示

シミュレーション実行中に変数の変化の様子をダイナミックに表示する機能を持っている。対象とする変数はシミュレーション実行に入る前にＤＩＳＰＬＡＹ ステートメントであらかじめ指定しておく。指定出来る変数の数には４つまでと，制限はあるが，シミュレーションの進行状況を見るにはなかなか有効な機能である。図６－９はシミュレーションを開始した直後の様子を写したもので，次の図６－10は，もう少しシミュレーションが進んだ時に，左下隅のＨＡＬＴ をライトペンでピックして実行を中断して写した。

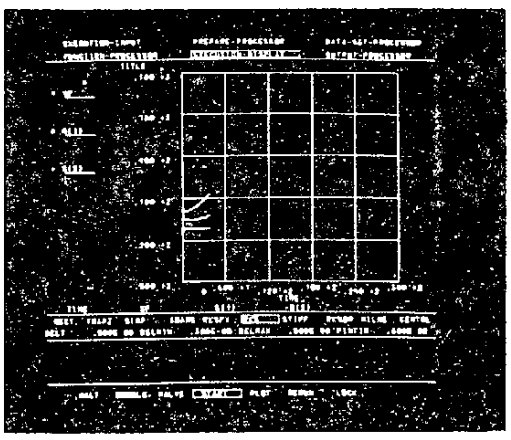


図６－９

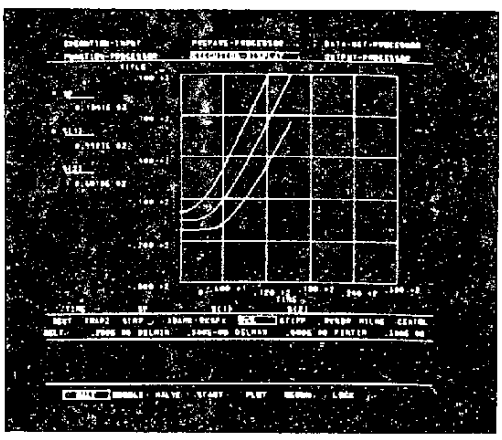


図６－10

この様にシミュレーション実行中に様子を見ながら実行を中断する機能を持っているが，この時に取れる処置が，シミュレーションを続行するか中止するかしかないのは残念で，出来る事ならこの時画面に表示出来なかった他の変数の様子を調べる手段を持っていると良いと思う。実際考えてみると実行を中断して見てもそこで調べられる事が画面に出ているグラフをじっと眺めるだけであるなら，要するに，シミュレーションの進行がおかしかったら中止する事以外に，この機能が役に立っていない事になる。それなら，なにもチャネルを画面の変化して行く間中頻繁に使用しているよりは，シミュレーションが終了してからグラフを見て，おかしかったらもう一度ランをした方が安上りの場合もあろう。そうは言うものの，実際にこのシステムを使って，じかに見ていると，なかなか迫力があり会話型のシステムを使っている実感があった。

(4) 実行が終了すると

シミュレーション実行が終了した時点で，本当はOUTPUTプロセッサーと言うのがあって色々な角度から結果を図示したりしながらシミュレーション結果を検討する事になっているが，今回はそれを使って見る事が出来なかった。

この機能とは別に，バッチ用CSMP-Ⅲで実行が終了した時に，ラインプリンターに出力する情報をディスプレイ画面に表示する事が出来る。唯，この場合には結果が数値そのものであるせいもあってあまり役に立つ機能ではないと思われる。一応，その様子を図6－11，図6－12，図6－13に示してある。この3枚の写真は，ラインプリンター用紙をめくって見るのと同じ様にページを順に送って画面に表示したものである。

この様なものは，後でゆっくりとラインプリンターに出る結果を見て検討すべきものであり，一応結果は出してあるから必要なら見て下さい，と言った感じであろう。

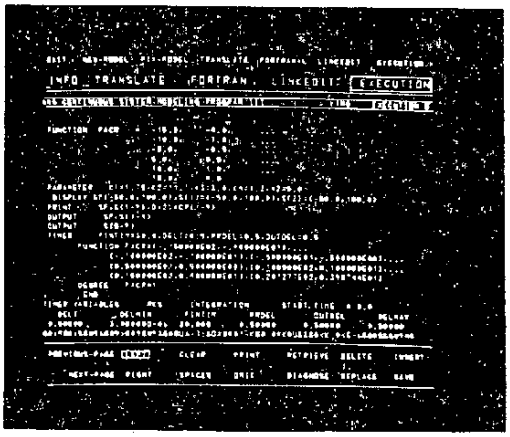


図6－11

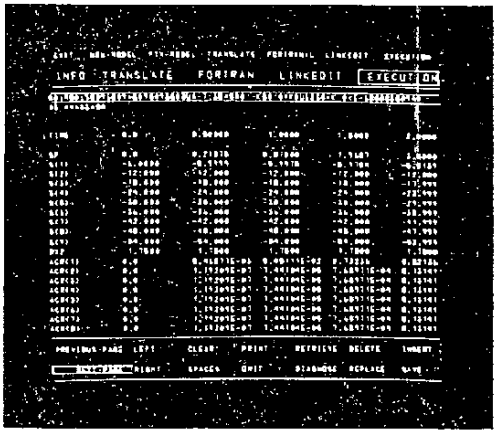


図6－12

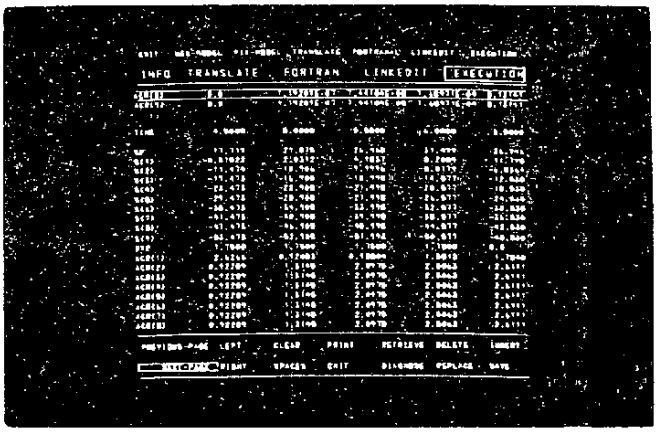


図６－13

(5) 実行の繰り返し

ＣＳＭＰ－Ⅲではプログラム実行に関連した情報だけをソースプログラム中から抜き出してプログラム自身とは別のファイルを作り出し，シミュレーションの実行に際しては，この情報をもとにシミュレーションの進行を管理している。この様な情報はＣＳＭＰ－Ⅲのステートメントのうち，Data ステートメント，Execution control ステートメント，Output control ステートメントが全てその対象となっている。この様に実行にかかわる情報をプログラムとは分離している為，バッチシステムの時と同じ様なオブジェクトプログラムが作成されていてもRecompile なしでかなり融通がきいた実行の繰り返しが出来る。今回もランタイムに表示する情報を追加したり，シミュレーション打切り時間を変えてシミュレーションを繰り返してみた。図６－14にはこのモデルで使った函数やパラメータ，ＤＩＳＰＬＡＹステートメント，ＴＩＭＥＲステートメントなどが表示されている。

図６－14にはＴＩＭＥＲステートメントを変更してシミュレーション打切り時間を変えた時の様子が表示されている。これらのステートメントを変更する時の要領はソースステートメントを修正する場合と同じである。

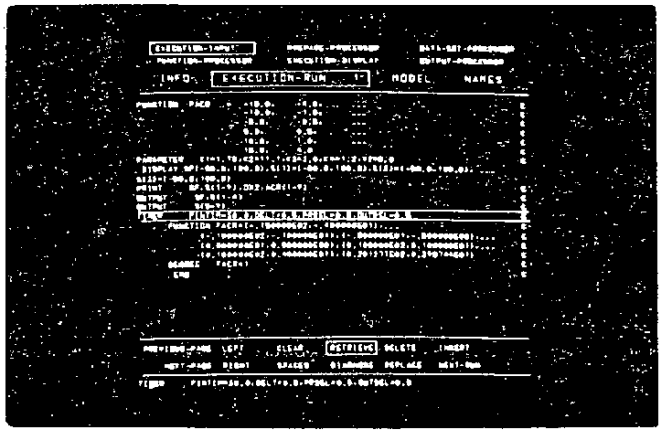


図 6 － 14

6.4　全体的に見て

このシステムを，わずかの時間ではあるが使用して見て感じた事は，あまりキーボードを使わなくて済んだと言う事である。ほとんどの指示がディスプレイ画面の必要な項目をライトペンでピックする事に依って出来るからである。即ち１つのフェーズから他のフェーズに移項する時や，１つのフェーズの中で細い指示をする場合にも全部必要なメニューが揃っていて，これを選ばせる方法をとっている。実際に使ってみると，この方式は，キーボードからインプットするより早くアクションが取れる事や，キーインする際のまちがいが防止出来る点でなかなか使い良かった。

以上総合的にみて，このシステムはなかなか良く出来ており，連続系シミュレーション言語がグラフィックディスプレイを介して会話型で使える事が非常に有効である事を感じた。しかし，経済的な面を考えるとあまり実用的とは言えない様である。これは，グラフィックディスプレイの使用料がまだ高すぎる点が大きな原因である。従って今後ディスプレイの使用料金が安くなれば本当の意味で実用的なシステムとなるのではないかと思う。

参 考 文 献

1. 日本情報処理開発センター
「オンラインシミュレーション言語ＳＩＭＢＯＬ」46－Ｓ004
2. 日本情報処理開発センター
「オンラインシミュレーション言語ＳＩＭＢＯＬの評価」47－Ｓ004

3. ユニバック
「MIMICマニュアル」
4. CONTROL DATA
6000 Computer systems
Continuous Systems Simulation Language Ⅲ (CSSL-3)
User's guide Version 1.
5. IBM
Continuous System Modeling Program Ⅲ (CSMP-Ⅲ)
Program Reference Manual SH 19-7001-2.
6. IBM
Continuous System Modeling Program Ⅲ (CSMP-Ⅲ)
Graphic Feature Program Reference Manual SH 19-7003-1.
7. Control Data Corporation
Control Data 6000 Series Computer System MIMIC Digital Simulation Language Reference Manual
Publication no 44610400 revision D 1970.
8. E. T. POWNER
Road traffic simulation employing a hardware approach philosophy and design considerations
SIMULATION vol 15 no 3 September 1970 p 113-118.
9. ROBERT D. BRENNAN
A Survey of Digital Simulation :
Digital Analog Simulator Programs
SIMULATION, MCGRAW・HILL p 244-258.
10. FAHRLAND D A
Combined Discrete-Event/Continuous-Systems Simulation
SIMULATION vol 14 no 2 February 1970 pp 61-72.
11. GOLDEN D G SCHOEFFLER J D
GSL - A Combined Continuous and Discrete Simulation Language
SIMULATION vol 20 no 1 January 1973 pp 1-8.
12. GOLDEN D G SCHOEFFLER J D
Problems in the Implementation of a Combined Continuous-Discrete Simulation Language

SIMULATION vol 20 no 2 February 1973 pp 49－52.

13. HIXSON H G

CLASS - Composite Language Approach for System Simulation

Proceedings Fourth Conference on Applications of Simulation

New York December 9－11 1970 pp 317－325.

14. HURST N R

GASP IV : A Combined Continuous Discrete FORTRAN

Based Simulation Language

Unpublished PhD thesis Purdue University 1973.

15. CHU Y

Digital Simulation of Continuous Systems

McGraw-Hill New York 1969.

昭和49年3月発行

発行所　財団法人　日本情報処理開発センター
東京都港区芝公園3丁目5番8号
機械振興会館内
TEL（434）8211（代表）

印刷所　三協印刷株式会社
東京都渋谷区渋谷3－11－11
TEL（407）7316

48－S002