北海道大学工学部研究報告
第134号（昭和62年）

Bulletin of the Faculty of Engineering,
Hokkaido University. No. 134 (1987)

会話型連続系シミュレーション・ソフトウェア・システム“ICOSS”

牧野　圭二

（昭和61年9月30日受理）

“ICOSS”: An Interactive Continuous-System Simulation Software System

Keiji MAKINO

(Received September 30, 1986)

Abstract

This paper describes a software system of a CSMP-type continuous-system simulator ICOSS, which is installed in the multi-purpose simulator HOSS in Hokkaido University, which is open to researchers. ICOSS is an interactive on-line simulation system. It is characterized by a trial-and-error process improved by on-line simulation techniques and man-machine communication in the process of the simulation. The program body is composed of only the model description with a simulation environment consisting of separate input and output descriptions. Control operations are a series of system commands issued in the exectution. ICOSS supports a parallel processing system PPA of a master-slave architecture. It emulates, through ICOSS, a conventional analog computer for high-speed and real-time simulations. The language system and software configuration are also discussed.

1．はじめに

連続系シミュレーションでは，対象システムにおける時間の経過に伴って生起する状態変化がそれぞれ継続的かつ連続的であると仮定し近似を行ってその記述，分析を行う。この目的のために長い間アナログ計算機あるいはハイブリッド計算機が用いられてきたが，1955年に最初のディジタル計算機によるアナログ・シミュレータ（ディジタル・アナログ・シミュレータ）の報告がなされ，大きな問題を高い精度で解くことができることが示されて以来，多くの研究がなされてきた。[1]

1960年代にはブロック・オリエンティドの言語とイクェーション・オリエンティドの言語が多数発表されたが，1960年代後半にこれらの長所をまとめたCSMP(Continuous System Modeling Program)に集大成された。その後CSMPはCSSL(Continuous System Simulation Language)に発展したが，CSMPはそれ以後の多くの連続系シミュレーション言語の基礎となって現在まで引き継がれている。[2]

この間，アナログ計算機は，スケーリングやパッチングなどに代表されるプログラミングの柔軟性の欠如から，ディジタル計算機の発達と共に次第にこれら連続系シミュレーション言語を用

いたディジタル・シミュレーションに置き換られてきた。しかし，その処理形態はCSMP完成当時の計算機利用形態であるバッチ型の処理のままで，アナログ計算機の長所でもあったシミュレーションに不可欠なオンライン性能の問題や実行時における操作性の問題などに対する検討はあまりなされず，会話型処理方式への移行が遅れているのが現状である。

ICOOS[4)~7)](Interactive Continuous-System Simulation Software System)は，そのモデル記述部分の表現に一部簡素化や簡略化がなされているが，従来のCSMP型の言語の系統に属する連続系シミュレーション・ソフトウェア・システムである。しかし，従来のバッチ処理型への反省から，アナログ計算機様の操作・実行方法の実現がはかられており，オンライン・シミュレータであると共に，ソース・プログラムの作成から実行・解表示にいたるまで，その間の試行錯誤を含めて，一貫して会話形式で行えるよう設計されている。

さらに，ICOSSは，実時間シミュレーションを可能とするため同時に開発された総計34台のプロセッサからなるマスタースレーブ方式の並列演算装置PPA[6),8),9)](Parallel Processor Array)を完全にサポートしている。PPAに対するコード生成はもちろんタスクの分割，割付けなどもトランスレータで自動的に行っており，ハードウェア構成をまったく意識する必要がないため，標準(非実時間)時でもPPAを使用した高速のシミュレーションを行うことができる。

ICOSSは，PAAと共に北海道大学全学共同利用施設汎用シミュレータ施設の全ディジタル式の会話型汎用シミュレータHOSS[3),5),7)](Hokkaido University High-Speed System Simulator)にその核として組み込まれており，実用システムとして広く学内の一般研究者に利用されてきている。

2．ICOSSの言語体系と特徴

ICOSSは，利用者が単にシミュレータを制御するというのではなく，利用者とシミュレータとが一体となったフィードバック・ループを形作るように考慮された[5),7)]実時間系のオンライン・シミュレータである。

ICOSSの言語体系は，シミュレーション問題の標準的問題解決過程[5),7)]を分析して得られたシミュレーションの標準処理パターンに沿って，大きく，モデルの記述，実行環境の記述，実行の3層に階層化している。その結果，CSMPの系統に属しながら，モデル記述部から入出力関係文をプログラム記述上も完全に分離すると共に，実行コマンドをこれらと独立したものとしている。このことによって，ソフトウェア・システム自体も，次節で述べるように，プログラムの作成から実行に至るまでの処理の流れに沿って処理単位毎のモジュール化が容易となり，利用者とシミュレータとのインタフェースとなる部分を極力他の部分から独立させることが可能となった。

その言語としての体系[4)]とシミュレータとしての特徴は次のようである。

- モデル記述文(狭義シミュレーション言語)[11)]
  - 定義文　(付録Ⅰ)
  - 実行文　(付録Ⅱ，Ⅲ)
  - (テキスト編集コマンド……専用エディタ　(付録Ⅳ))[11)]
- 入出力制御データ[5),7),10)]
  - パラメータ
  - 実行・解出力コントロール・データ
    - 実行方法制御データ

出力方法制御データ(プリント出力，ファイル出力，ディスプレイ出力)
(入出力制御データ登録編集コマンド)
実行時コマンド(シミュレーション・コンソール専用キー)[10)]
実行制御コマンド　(START, HALT, RESET, EXIT)
実行環境設定コマンド(PARAM, COMND, CASE)
チェック・コマンド　(CHECK)

(1) オンライン・シミュレータであること

オンライン・シミュレータであることの必要性は，実行状態を直接観察し制御できることにある。実行状態は，解波形として計算の進行と共に，非実時間実行時(標準)にはストレージ型グラフィック・ディスプレイ装置(標準出力装置)かリフレッシュ型グラフィック・ディスプレイ装置に表示され，実時間実行時にはアナログ出力ポートを通してオシロスコープあるいはペン・レコーダか XY レコーダに出力される。この関係を図1に示す。

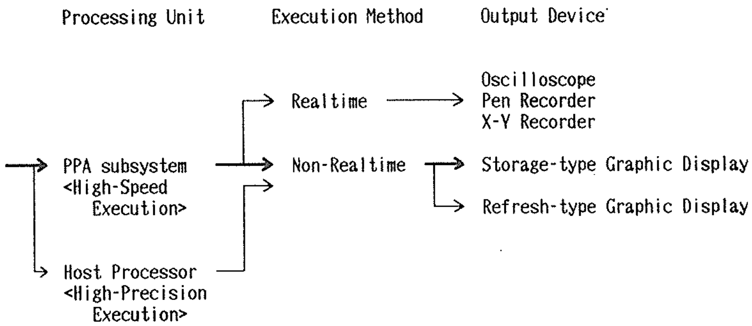

図1　実行モードと処理装置，出力装置の関係

実行段階で実行の制御に用いる機能は，実行時コマンドとしてモデルに関する部分と完全に独立させ，また実行方法や表示装置に依存せず共通のものとしており，ハードウェア構成上も ICOSS 専用のシミュレーション・コンソール上の専用コマンド・キーとして実現している。実行時コマンドには，純粋に実行過程を制御する実行制御コマンド，実行のための環境設定を行う実行環境設定コマンド，実行途中で個々の変数の値の調査や変更を行うためのチェック・コマンドの3群がある。これらのコマンド間の状態の遷移を図2に示す。

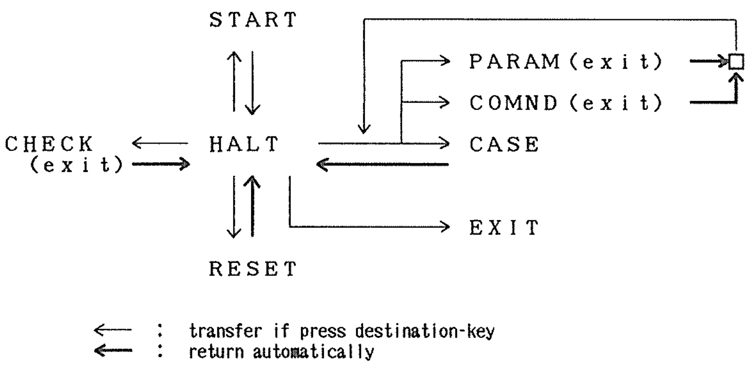

図2　実行時コマンドでの状態の遷移

(2) **会話型であること**

会話型シミュレータであることの重要性は，試行錯誤を繰り返す利用者の負担を減らすことにある。そのために，ICOSSでは，コマンドを操作の主体にすると共に，個々の場面においてメニュー選択ガイドライン表示方式とディフォールト値設定方式を全面的に採用している。また，シミュレーションの準備から実行に至る全体についても標準的実行の流れを想定し，その流れに沿ってシミュレータの状態が推移するようにしている。このことは，言語体系上モデルの記述と環境の記述とが完全に分離されており，実行環境の変更がモデルの記述と独立して行えることにも現れている。

モデル記述文を用いて記述されるモデルの記述部(プログラム)には入出力に関する文は含まれず，それ単独で翻訳される。そのとき，実行文の右辺にしか現れない変数はパラメータと仮定され，左辺に書かれた変数は後に出力変数として用いられる可能性があると判断され，それぞれその名前に関するリスト・ファイルが作られるだけである。

一方，実行の環境は入出力制御データとして設定され，パラメータ，及び，実行・解出力コントロール・データとしてそれぞれ最大10通りまで独立に登録可能となっている。登録や変更時には，ディスプレイ上にパラメータ変数名あるいは項目名が表示されるので順次必要な値を入力していけばよく，ディフォールト値も用意されている。さらに，既に登録されているデータの複写をはじめとする編集コマンドがメニュー選択方式で用意されている。[10]

モデルの記述部と実行環境との組み合せは，実行時にCASEコマンドを用いて使用する環境を指定することにより，ダイナミックに行われる。これらの関係を図3に示す。

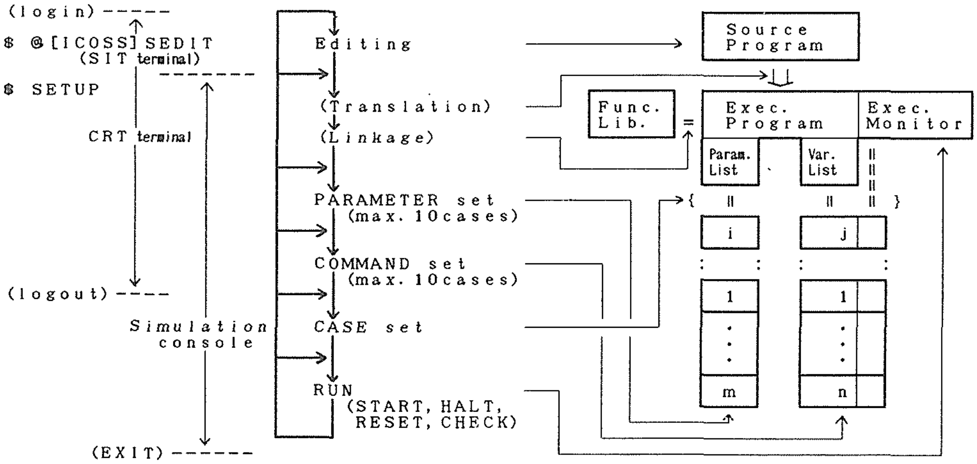

図3　実行の流れと実行の方法

会話型の処理になじみずらいモデルの記述部の入力には，文法的誤りを発生時点で検出，修正が行えるように，1ステートメントの入力毎にローカルな文法チェックを行い，入力完了時にはグローバルな文法チェックを行う専用のエディタを用意している。従って，実行時に発生するエラーは，モデル化の不適切などに起因するロジカル・エラーにしぼられる。また，編集コマンドの中に，偏微分方程式を連立常微分方程式に展開する場合などによく生じる同型方程式を自動的に展開し入力する反復展開用コマンド($REPTコマンド)を採り入れるなどの工夫がなされている。

(3) **実時間シミュレーションが可能であること**

実時間シミュレーションが可能であることは，シミュレータ外部の実システムや実験装置あるいは測定器などと連動したシミュレーションを行いたい場合に不可欠である。また，それを可能とするような計算処理の高速性は，シミレーション自体その実行を繰り返し行うことが多いことからも重要である。このためICOSSでは，連続系シミュレーションの高速実行のため並列処理方式のPPA[6),8),9)]を開発し標準の実行装置として用いている。PPAでの実行モードが高速度モードであり，さらに実時間モードと非実時間モードとに別れている。一方，実行精度に重点を置いたホスト計算機単独での実行モードが高精度モードである。(図1参照)。

高精度モードを含めて非実時間モードでは解をグラフィック・ディスプレイ装置に表示する。しかし実時間モードでは，その実行を解の表示を含めて他から独立してPPA単独で行う。すなわち，データはPPA内のプロセスI/Oポートを通して直接外部に出力し，直接外部から取り入れる。従って，実時間モードでは，本シミュレータ内部の他装置の処理速度あるいは他装置とのデータ転送速度からの影響を受けずに，PPAの最高速度で実行することができる。

ICOSSは，これらのPPAでのシミュレーションの実行を完全にサポートしており，ソース・プログラムは高精度モードとも共通であり，解の出力装置を除くとどの実行モードでも処理や操作上の基本的な違いは無く，利用者はハードウェア構成をまったく意識する必要がない。いずれの実行モードでも，演算は浮動小数点演算であり，シミュレーション時間の管理は固定時間方式である。従って，数値積分法は必然的に浮動小数点数による固定積分刻み法であるが，実行モードにより次のような次数の差がある。

実行装置(データ長(ビット))		ルンゲ=クッタ法	アダムス予測子法	矩形法
PPA	(32)	3次	2次	1次
ホスト計算機(32，64)		4次	3次	2次

3．ICOSSのソフトウェア構成と生成されるファイル

ICOSSを構成するソフトウェアは，大きく次の四つのグループからなっている。

- 入力プログラム
    - 会話型ソース・プログラム・エディタ
        - シンタックス・チェック・プログラム
    - パラメータ入力処理プログラム
    - 実行・解出力コントロール・データ入力処理プログラム
- 変換プログラム
    - トランスレータ
        - 分割・配置プログラム
    - リンケージ・プログラム
- 制御プログラム
    - シミュレーション・コントロール・プログラム
        - 実行管理プログラム
        - 解表示プログラム
    - 並列部(PPA)コントロール・プログラム
        - 実行管理プログラム
        - 制御用マイクロプログラム

その他ユーティリティなど
　並列部プロセッサ診断管理ユーティリティ
　プログラム登録管理ユーティリティ
　簡易出力ユーティリティ

ICOSS の実行の流れの概念図を図 3 に示したが，このときのモデルの作成から実行までの各段階と、それぞれの段階において関係する ICOSS のソフトウェアとその処理を以下に示す。また，このとき生成されるファイル間の関係を図 4 に示す。なお，ソース・プログラムの入力を除く他のすべての処理は，シミュレーション・コントロール・プログラムの管理下で行われる。

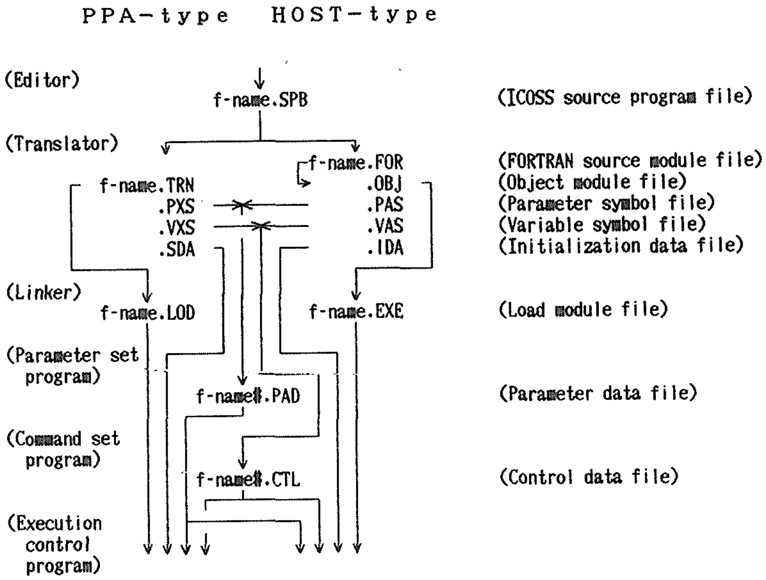

図 4　各処理段階で生成されるファイルとその相互関係

1）モデル化　　　　　（シミュレーション・モデルの作成）
　連立常微分方程式系またはブロック・ダイアグラムで対象システムを記述する。
2）モデル入力　　　　（シミュレーション・プログラムの記述と入力）
　会話型ソース・プログラム・エディタでソース・プログラム・ファイルを作成する。
3）トランスレーション（実行可能形式プログラムの生成）
　トランスレータでオブジェクト・モジュール・ファイルや各種のシンボル・ファイルを生成し，さらにリンケージ・プログラムで実行可能形式のロード・モジュール・ファイルを生成する。
4）実行環境入力　　　（パラメータ値，初期値，実行方法，出力方法の指定，登録）
　パラメータ入力処理プログラムのもとで，シンボル・ファイルを参照しながらパラメータ値や初期値を保持するパラメータ・データ・ファイルを作成する。最大 10 個まで作成できる。
　実行・解出力コントロール・データ入力処理プログラムのもとで，シンボル・ファイルでチェックしながら実行方法，出力表示方法を指示するコントロール・データ・ファイルを作成する。最大 10 個まで作成できる。

5）実行準備　　　　　　（プログラムのロード）

実行管理プログラムのもとで，指定されたロード・モジュール・ファイルをPPAへローディングすると共に，パラメータ・データ・ファイルとコントロール・データ・ファイルの指示に従って初期値や実行・解出力方式の設定を行い実行環境をととのえ，実行可能な状態にセットする。なお，その後実行環境のみを変更する場合，ロード・モジュール・ファイルを再ロードする必要はない。

6）実行　　　　　　　　（実行制御コマンドによる実行）

実行管理プログラムの制御下において，解表示プログラムや制御用マイクロプログラムの補助のもと，シミュレーション・コンソールの実行制御コマンド・キーの指示に基き実行を行う。なお必要に応じ，実行環境設定コマンド・キーを用いて前の段階にもどることができる。

以上の各段階の実行に至るまでのすべての処理はホスト計算機で行われる。会話型ソース・プログラム・エディタは，SIT端末(Source Input Terminal)においてコマンド$@[ICOSS]SEDITにより起動され実行される。[10] このSIT端末にはエディタの機能の一部が組み込まれている。また，エディット完了後の処理は，連続系シミュレーション・コンソールで，一連の作業として実行できるが，トランスレーションと実行環境入力の両段階の処理は通常端末からも実行できるように考慮されており，コマンド$SETUPで起動し実行することができる。[10]従って，実行以外の各段階の作業は，複数利用者が並行して行うことが可能である。

4．トランスレータでの処理並びにその他の補助機能

ICOSSでは連続系シミュレーションに対していくつかあらたな試みがなされており，また，高速度モードで並列演算装置PPAを使用するための特有の機能も組み込まれている。以下にこれらの処理機能について概説する。

4.1　トランスレータ(PPAに対するトランスレーション)

ICOSSは従来のCSMP型の連続系シミュレーション言語の系統に属しており，従って，これらの言語に共通の特徴であるステートメント(実行文)のソーティング(記述順序から実行順序への並べ換え)の処理は当然のこととして行っている。さらに，マスタ-スレーブ方式の並列演算装置PPAに対するタスクの分割や割り当て並びに変数のシェアド・メモリへの割り当てや実行時の同期に関する処理などもすべて自動的に行っている。ここで行っているスレーブ・プロセッサへのタスク割り当ては必ずしも最適割り当てとはいえないが，以下のような方法で行うことによって極力負荷の均等化をはかっている。なお，このとき，後に述べるユーティリティ・プログラムを使用することにより，実行に使用するプロセッサを任意に選択することができる。

1）ソース・プログラムのソーティングを行うと共に積分演算(一般的には過去の値に依存する演算)を抽出し，積分演算相互間の隣接関係を解析する。

2）積分演算を含むステートメントを単位として，積分負荷が均等になるように，隣接する積分演算を，順に，利用するスレーブ・プロセッサの台数(最大32台)に等しいグループに分ける。

3）各グループに対し，そのグループ内での積分演算に必要なすべての算術論理演算式(ステートメント)を組み入れ，プロセッサへの割り当て単位となるブロックを構成する。

4）ブロック単位での入出力の関係に着目し，隣接するブロックがPPA上でも極力隣接するように各スレーブ・プロセッサにブロックを割り当てる。

5）各ブロックの変数をスレーブ・プロセッサ間の各シェアド・メモリ・バンクに割り当てる。

6）各ブロック毎に各スレーブ・プロセッサのローカル・メモリに対してオブジェクト・コードの生成を行う。

4.2 テスト・ラン機能[9)]

実時間シミュレーション実行時にPPAがどの程度のシミュレーション速度で実行が可能かを一定時間実行させて計測する機能である。すなわち，指定された時間の間分解能10 μsecの内部クロックで自動的に計測を行い，その間の1積分刻みの実行時間の中で最も時間のかかった積分刻みの実行時間を表示する。この計測結果より大きい任意の値を，実時間シミュレーションの実行に用いるシミュレーション・クロックの値として設定することができる。

4.3 妥当性検証補助機能

連続系ディジタル・シミュレーションでの解の安定性あるいは解やモデルの妥当性を検討する手段の一つの試みとして，次の機能を組み込んでいる。

NOISE（ノイズ自動重量実行機能）

コンパイル時オプションである。コンパイル時にこれを指定することによって，指定した変数に指定したスケールの雑音(一様乱数)を重畳するルーチンが自動的に組み込まれる。従って，プログラムの変更なしに特定の変量に雑音が加わった状態のシミュレーションを自動的に行うことができる。

CHECK（解チェック機能）

実行時コマンドの一つである。実行途中での実行の一時中断(ホールト)時点で使用することができ，その後実行を再開，継続することができる。大きく二つの機能からなる。

第1は，解表示をしていない変数を含めて任意の変数の値を調べ(EXAMINE)また変更する(DEPOSIT)機能である。

第2は，時間の進行を強制的に逆向きにして実行を進める(REWIND)機能である。このサブコマンドを実行するとシミュレーション・クロックの積分刻み幅は逆向き($-\Delta t$)に設定され，引き続きスタート・コマンドを実行すると，実行中断時の各変数の値をそれぞれの初期値とみなし，その時点からシミュレーション・クロックを逆向きに実行し，すでに得られている解波形を反対方向へたどる形で出力表示をする。

4.4 ユーティリティ

並列部プロセッサ診断管理ユーティリティ

コマンド　$@[ICOSS]PPA　で起動する。

PPAの障害プロセッサの登録(トランスレータへの使用禁止情報となる)，障害プロセッサの登録状況の出力，トランスレーション完了プログラムのプロセッサ使用状況の出力を行う。

プログラム登録管理ユーティリティ

コマンド　$@[ICOSS]FILE　で起動する。

コンパイル状況並びにパラメータ・データ・ファイルとコントロール・データ・ファイルの登録状況の表示，不用な登録済みパラメータ・データ・ファイルやコントロール・データ・ファイルの消去，ソース・プログラムのみを残しそれ以外の全関連ファイルの消去を行う。

簡易出力ユーティリティ

コマンド　$@[ICOSS]LIST　で起動する。

ソース・プログラム・ファイル，パラメータ・データ・ファイル，コントロール・データ・ファイル，並びに解データ・ファイルの内容を，直接任意の端末(出力装置)にリスト出力する。各実行段階のそれぞれの処理プログラムを通さずに，各ファイルの内容を検討することができる。

5. おわりに

従来，計算機の能力の発展にもかかわらず，ディジタル・シミュレータの開発，改良が，当初のシミュレータ・モデル開発時のCPU効率を重視したバッチ処理方式のまま，操作性よりもモデル記述の面にのみ向けられる傾向にあった。ICOSSはこの点の反省からオンライン・シミュレーションの実現を第一に会話型処理化を進め，さらに，専用の並列演算装置PPAのもとシミュレーション実行の高速化をはかると共に実時間シミュレーションをも可能とした。

この成功の原因としては，文法上，シミュレーション・モデルの記述とシミュレーション環境の記述を明確に分離したことと，実行の制御は実行時コマンドとしてこれらから完全に独立させたことがあげられる。また，システム全体を通してコマンド指示方式を操作の主体に置き，その場合もメニュー選択，ガイドライン表示方式を採用し，各項目にはディフォールト値を用意したことも重要な点である。この結果，ICOSSでは，解の重ね書き機能などに加えて，モデル記述部分の再トランスレーション無しにシミュレーション環境だけを換えて再実行ができるため，必要な条件を変えての反復試行やその際の解波形の比較が非常に容易なものとなっている。

このように，ICOSSシステムでは，連続系の会話型オンライン・シミュレーションの実現という面での基本的な問題は解決されている。今後の問題としては，解の安定性あるいは解やモデルの妥当性を検討する手段の一つの試みとして今回取り入れた，妥当性検証補助機能の利用方法の確立と他の手法を含めての体系化を進めることがあげられる。また，本ICOSSシミュレーション・システムには取り入れることができなかったマルチ・タイム・スケール・シミュレーションや実行時におけるダイナミックなパラメータ値変更などの機能の実現方法の検討が残されている。

謝　辞

ICOSSはPPAと共に北海道大学「高速システム・シミュレーション装置」HOSSの中心をなす連続系シミュレーション・サブシステムであり，このHOSSは1978年度及び1979年度の2年度にわたる文部省特別設備費によって設置された。開発，設計から稼動まで長期にわたり検討，協力いただき，また実現のため多大な努力を払われた北海道大学全学共同利用施設汎用シミュレータ施設利用開発小委員会，三井造船株式会社システム本部(当時)を始めとする関係各位に深く感謝致します。

上記小委員会の委員として長期にわたり実現に熱意を傾けられた故小山昭一先生(当時北海道大学工学部精密工学科助教授)に謹んで哀悼の意を表します。

参考文献

1) Chu, Y.: Digital simulation of continuous systems, (1969), p. 423, McGraw-Hill

2) Speckhart, F. H. and Green, W. L.: A guide to using CSMP-the continuous system modeling program, (1976), p. 325, Prentice-Hall

3) Makino, K., Koyama, S., Miki, N., Iseki, Y., Kobayashi, H. and Sakai, Y.: Rroc. 9th IMACS World Congress, (1979), p. 171-177

4) Iino, Y., Makino, K., Iseki, Y., Koyama, S. and Miki, N.: Proc. 9th IMACS World Congress, (1979), p. 261-270

5) 牧野圭二，小山昭一，三木信弘：信学技報，vol. 80，(1980)，No. 75，p. 55-66

6) 牧野圭二，三木信弘：信学技報，vol. 82，(1982)，No. 62，p. 57-68

7) 牧野圭二：北海道大学工学部研究報告，第129号，(1986)，p. 45-55

8）牧野圭二：北海道大学工学部研究報告，第131号，(1986)，p. 49-59
9）牧野圭二：北海道大学工学部研究報告，第131号，(1986)，p. 61-70
10）「高速システム・シミュレーション装置」HOSS-入門編(1)：ICOSSが使えるまで，(SC-0007-04)，汎用シミュレータ施設
11）ICOSS Language Reference Manual, (SC-0006-02)，汎用シミュレータ施設
12）ICOSS Editing Guidel (SC-0005-03)，汎用シミュレータ施設

付録 I　ICOSSの定義文の種類

1	PROGRAM	プログラム名の宣言
2	DOUBLE	倍精度実行の宣言
3	INTEGER	整数型変数名の宣言
4	PARAMETER	パタメータ変数名の宣言
5-1	DIMENSION	配列型変数名の宣言
-2	TABLE	テーブル名の宣言
6-1	FUNCTION	ユーザ定義関数名の宣言
-2	DUMMY	ダミー・ブロック名の宣言
7-1	MEMORY	メモリ関数名の宣言
-2	HISTORY	ヒストリ関数名の宣言
8-1	SORT	ソート・ブロック
-2	NOSORT	ノーソート・ブロック
9-1	INITIAL	イニシャル・セグメント
-2	DYNAMIC	ダイナミック・セグメント
-3	TERMINAL	ターミナル・セグメント
10	END	プログラムの終り
11-1	PROCEDURE	プロシジャの始り
-2	ENDPROCEDURE	プロシジャの終り
12-1	ENTRY	ユーザ定義関数の始り
-2	RETURN	ユーザ定義関数の終り
13-1	SBLK	サブブロックの始り
-2	ENDSB	サブブロックの終り

付録 II　ICOSSの実行文の種類

1	関数引用文	v_1, …, $v_k = f(v_{k+1}, \cdots v_m)$
2	算述代入文	$v = expression$
3	無操作文	NOP
4	単純GOTO文	JUMP l
5	計算型GOTO文	GOTO v $l_1, \cdots l_m$
6	論理IF文	IF v_1 op v_2 l
7-1	ループ制御文	DO $v = m_1$, m_2
-2	ループ制御文	NEXT

(フォートラン標準関数が使用できる。)

付録 III　シミュレーション標準関数一覧

積分	INTGRL
モード積分	MODINT
1次遅れ	REALPL
2次遅れ	CMPXPL
リード・ラグ	LEDLAG
一般ラプラス変換	TRANSF
微分	DERIV
むだ時間	DELAY
陰関数	IMPL
ステップ関数	STEP
ランプ関数	RAMP
インパルス列	IMPULS
パルス	PULSE
正弦波	SINE
正規乱数	GAUSS
一様乱数	RANDA
直線近似任意関数	AFGEN
曲線近似任意関数	NLFGEN
リミッタ	LIMIT
不感帯	DEADSP
ヒステリシス	HSTRSS
量子化関数	QNTZR
零次ホールド	ZHOLD
サンプリング	SAMPLE
比較器	COMPAR
出力スイッチ	OUTSW
入力スイッチ	INSW
関数スイッチ	FCNSW
フリップ・フロップ	RST
論理否定	NOT
論理積	AND
否定論理積	NAND
内包的論理和	IOR
排他的論理和	EOR
否定論理和	NOR
同値	EQUIV
スカラ-アレイ変換	ARRAY
アレイ-スカラ変換	SCALAR

付録IV テキスト編集コマンド一覧([…] は省略可能を示す)

\$ADD n	行番号 n の文の後にこの後入力する文を順次挿入する。
\$ALT ～ n	行番号 n の文の修正を行う。
\$CHG n	行番号 n の文をこの後入力する文で置き換える。
\$COPY l, m, n	行番号 l から m までの文を行番号 n の文の後へ複写する。
\$DELT l, m	行番号 l から m までの文を削除する。
\$DEND	コマンド"\$ADD", "\$CHG", "\$REPT"に続く文入力モードの終了を指示する。
\$DISP [l[, m]]	行番号 l から m までの文を SIT 端末に表示する。
\$END	現在のプログラムについてテキスト編集を終了する。
\$EXIT	テキスト編集モードを抜ける。
\$GO	現在編集中のプログラムの最後の部分を表示する。
\$LIST [l[, m]]	行番号 l から m までの文を PR/PL に出力する。
\$MOVE l, m, n	行番号 l から m までの文を行番号 n の文の後へ移動する。
\$REPT v, l, m	この後入力する文に対し，指定された文字 v を l から m までの数値文字で順次置き換えて得られる文に展開する。