

連続系シミュレーション言語とその応用

松 坂 知 行

Continuous System Simulator DECS and its Application

Tomoyuki MATSUZAKA*

Abstract

Rapid development of semiconductor technology has enabled us to realize a continuous system simulation language such as CMSP or CSPL on small computers, which was implemented on large computers. This paper presents a new simulation language which can be implemented on not only large computers but also small computers or workstations. This language is able to generate the codes for simulation of the system to be described by a set of linear or nonlinear simultaneous differential equations, and enables users to save the time of tedious programming works. This paper describes the constitution and practical applications of this language.

1. ま え が き

工学の世界には制御系、電気回路の過度現象、機械系の振動、水位系の液面変動など種々の動的現象が現われる。このような系の現象は一般に連立常微分方程式で表わされるので、系のシミュレーションは積分演算、線形、非線形演算などの各種演算モジュールを相互に接続し、各種入力を加えることにより実行される。このようなシミュレータの代表的なものはアナログ計算機であるが、非線形演算や複雑な論理判断が不得意なこと、計算精度、スケーリングなどの問題があるため、現在は専らデジタル計算機が用いられている。しかしデジタル計算機では、アナログ計算機のようにパッチボード上で積分器や加算器を物理的に接続するプログラミングに比較して、プログラミングの作成が面倒である。そこで汎用デジタル計算機を用いて、アナログ計算機なみの容易さでプログラミングを行うた

めの専用シミュレーション言語が開発されており、その代表的なものには CSMP¹⁾、CSPL²⁾ などがあり、連続系シミュレーション言語と呼ばれている。しかしこれらの言語は汎用機を対象として開発されているため、それなりの利用環境の整っていないユーザでないと使用できない。一方ユーザには、このような汎用機ばかりでなく、よりパーソナルな利用環境で、シミュレーションを行いたいという要望がつよい。このような観点からパーソナルコンピュータ上で実行できる言語として MCSP³⁾、 Σ ⁴⁾、ACSL⁵⁾ などが発表されている。本稿で報告する DECS^{6),7)} もこのような考え方から開発した言語であり、基本的にはパーソナルコンピュータ上での利用環境を目指したものであるが、システムの記述言語として FORTRAN を用いているので移植性は高く、実際グラフィック部分を除き EWS、汎用機にも容易に移植可能であった⁸⁾。したがって、同一ソースプログラムで、問題のサイズにより計算機資源を選ぶことができる。

本稿では本言語の構成、応用について述べる。

平成 4 年 12 月 15 日受理

* 情報システム工学研究所教授

2. 連続系シミュレーション言語 DECS

本稿で述べる言語を DECS (Differential Equation based Continuous system Simulator) と名づけた。以下 DECS の基本的な考え方、構成について述べる。

2.1 基本的な考え方

(1) システムの表現方法

制御系、電気回路の過度現象、機械系の振動、水位系の液面変動などの動的現象を、シミュレーションするためのシステムの記述方法には種々の方法が考えられる。

システムが線形な場合には、入出力に着目し伝達関数で記述する方法、積分器、加減算器などの基本演算要素の組合わせで記述する方法、行列状態方程式の行列要素で記述する方法、ボンドグラフで記述する方法¹⁵⁾ などがある。

しかし現実のシステムには一般に非線形要素が含まれるので上記の方法は部分的にしか適用できなかったり、適用できたとしても複雑であったりする。一方動的システムは線形、非線形を問わず一般に状態方程式 (1) で記述される。すなわち

$$\frac{dx}{dt} = f(x, u, t) \quad (1)$$

ここで x は状態ベクトル、 u は入力ベクトル、 f は関数ベクトルである。

本稿で述べる DECS では状態方程式表現をそのままの形で記述する方法をとった。この方法の長所は、本来動的システムのモデルが微分方程式で表現されているので状態方程式が容易に得られること、線形、非線形システムを問わず適用可能なこと、記述順序を問わないのでシステムプログラムを作成する上でソーティングが不要なことなどである。

(2) コンピュータへの入力方法

コンピュータにシミュレーションの対象とするシステムの構成要素を入力する方法には、ア

イコンなどのグラフィックエディタを用いて入力する方法と言語形式で記述する方法とが考えられる。アイコン方式は、人間にとって分かりやすく、図形と物理的システムの要素が対応している時には便利であるが、図形だけでは表現し難い要素や論理的に細かい対応が要求される場合には不便である。一方言語形式はアイコンのようにマンマシンインターフェースは良くないが、論理的にきめ細かいシステムの表現が可能である。またシステムのソフトウェア的な記述も容易である。このため DECS では言語タイプを採用することにした。

(3) システム記述言語

記述言語としては FORTRAN, C, PASCAL などが考えられるが、数値計算のソフトウェア資産は FORTRAN が圧倒的に多いため、FORTRAN に馴染んだユーザが多い。記述言語として FORTRAN を用いた場合、ソースプログラムも FORTRAN もどきになり、FORTRAN のユーザに理解しやすいので、記述言語として FORTRAN を採用することにした。

2.2 ソフトウェアの構成

DECS のソースプログラムは、基本的に 6 つの文から成る。それらはプログラムを制御する制御文、状態変数の初期値を設定する初期値設定文、定数の値を設定する定数設定文、配列宣言文、状態方程式および後述するシステムマクロ関数、FORTRAN 命令文などを用いてシステムを記述するシステム記述文、サブルーチンを記述するサブルーチン文である。

2.2.1 制御文

以下制御文について述べる。なお記述形式の詳細については 3 の例題を通して説明する。

(1) タイトル文

>TITLE の文頭で始まり、出力画面なプリンタにプログラムのタイトルを表示する。

(2) パラメータ文

>PARAM の文頭で始まり、各シミュレーションの実行毎にパラメータを変えて実行させる文

である。

(3) 時間制御文

>TIMER の文頭で始まり、演算時間刻み、最終時間、スキップ間隔、画面スクロール間隔などを制御する。画面スクロールを可能とすることにより過度状態から定常状態までの現象を観察できる。

(4) スケール文

>SCALE の文頭で始まり、画面出力のフルスケール、目盛りの種類を設定する。

(5) 出力制御文

>PRINT の文頭で始まり、出力様式(数値, グラフィック), 出力機器, 出力すべき変数を指定する。出力機器としては画面, プリンタ, ディスクである。

(6) 入力制御文

>INPUT の文頭で始まり、ディスクファイルを読み出す制御文である。これにより予め作成しておいた実データをシステムに取り込みながらシミュレーションを行うことができる。

2.2.2 初期値設定文

初期値設定文はつぎの形式で記述される。

* INITCON

$$X_1 = C_1$$

$$X_2 = C_2$$

•

•

•

$$X_N = C_N$$

* END

ここで $X_1, X_2, X_3 \dots X_N$ は状態変数, $C_1, C_2, C_3 \dots C_N$ は初期値である。

2.2.3 定数設定文

定数設定文はつぎの形式で記述される。

* CONSTANTS

$$V_1 = T_1$$

$$V_2 = T_2$$

$$V_3 = T_3$$

•

•

•

$$V_N = T_N$$

* END

ここで $V_1, V_2, V_3 \dots V_N$ は変数で $T_1, T_2, T_3 \dots T_N$ は定数である。

2.2.4 配列宣言文

配列宣言文はシステム記述文に使用される配列を予め確保しておくための宣言文であり, 以下の形式で記述される。

* ARRAY

DIMENSION A(100), B(200)

•

•

•

* END

2.2.5 システム記述文

システムのダイナミックスを表わす状態方程式, システムへの入力信号, 表示すべき出力変数, 各種演算などを記述する。さらに後述するシステムマクロ関数, FORTRAN 命令文などを用いることができる。システム記述文は以下のように記述される。

* DYNAMICS

$$\text{DOT}(X_1) = F_1(X_1, X_2, X_3 \dots X_N)$$

$$\text{DOT}(X_2) = F_2(X_1, X_2, X_3 \dots X_N)$$

$$\text{DOT}(X_3) = F_3(X_1, X_2, X_3 \dots X_N)$$

•

•

•

$$\text{DOT}(X_N) = F_N(X_1, X_2, X_3 \dots X_N)$$

* END

ここで DOT は微分, F_n は関数, $X_1, X_2, X_3 \dots X_N$ は状態変数を示す。

2.2.6 サブルーチン文

システム記述文の中にはメインルーチンしか記述できないので、サブルーチンを用いる場合にはここに記述する。その形式は以下のようになる。

```
* SUBROUTINE
  SUBROUTINE SUB1 (A, B, C)
    .
    .
    .
  RETURN
  END
* END
```

2.2.7 システムマクロ関数

DECS のもっているシステムマクロ関数は以下のとおりである。

- (a) 信号発生関数

RAMP	ランプ関数
STEP	ステップ関数
SINE	正弦波関数
RANF	正規乱数発生関数
- (b) 論理関数

NOT	否定
AND	論理積
OR	論理和
- (c) スイッチング関数

FCNSW	関数スイッチ
INSW	入力スイッチ
- (d) 非線形関数

LIMIT	リミッタ
DEADSP	不感帯
HSTRSS	ヒステリシス
FITTER	折れ線関数
- (e) 数学関数

INTGRL	積分器
DERIV	微分器
DELAY	時間遅れ要素
ZHOLD	零次ホールド

などである。

2.3 状態方程式の数値解法

つぎに状態方程式の数値解法について述べる。(1) 式で表わされる状態方程式の解は、システムが線形の場合には、刻み幅を粗く選んで安定に収束するという長所があるため、通常行列指数関数を計算することにより求められる。しかしシステムが非線形の場合には平衡点の周りで線形近似する必要がある、システムプログラムが複雑になるので適当な方法ではない。むしろ Rung-Kutta 法のように 1 ステップあたりの計算量が少なく、線形、非線形両システムに適用可能な方法の方が適当である。そこで DECS では 4 次の Runge-Kutta 法を用いている。

つぎに 1 ステップあたりの演算時間の進み方には固定ステップ法と可変ステップ法がある。後者の可変ステップ法は次数の高い計算法と低い計算法を併行して用い、2つの計算結果の差が、ある一定以上になった時誤差の許容範囲を越えたものと見なし、演算ステップを少なくするプログラム制御によって可変ステップを実現する方法である。このアルゴリズムは、シミュレーションの対象とするシステムに、大きい時定数と極端に小さい時定数が存在する、いわゆる硬い (stiff) システムには有効な方法であるが、パーソナルな環境で利用できる連系シミュレーション言語に組み込もうとするとシステムプログラムのメモリ容量が大きくなり、またかなり複雑になる。一方固定ステップ法はプログラム作成は簡単であるが、硬いシステムに対しては収束が悪く、ステップの選び方によっては、本来安定なシステムでも発散することも起こり得る。しかし筆者の経験によると電子回路系を除けば実際の制御系、電気回路系、機械振動系、液面振動系にはシミュレーションが不可能になるような硬いシステムは少なく、固定ステップ法でもほとんど問題がない。以上の理由で DECS では固定ステップ法を用いている。

2.4 プログラムの実行手順

つぎにソースプログラムの作成から実行までの手順について述べる。

〈ステップ1〉

ソースプログラムをエディタを用いて作成する。

〈ステップ 2〉

システムプログラムを起動する。システムプログラムはソースプログラムを翻訳し、DECSの文法に一致しているかどうかを逐次チェックする。もし文法に違反している場合にはエラーメッセージを出し、その時点で翻訳を停止する。この場合はステップ1に戻り、ソースプログラムを修正する。文法違反がない場合にはFORTRANプログラムが自動生成され、ステップ3に進む。

〈ステップ3〉

生成された FORTRAN プログラムから FORTRAN コンパイラ、リンカーにより実行形式のコードが生成される。これ以後のエラーは FORTRAN プログラムのエラーであるので、コンパイラのエラーメッセージにより対処する。この場合 FORTRAN プログラムが生成されているので、FORTRAN に馴染んでいるユーザーは容易にエラーを確認できる。エラーが起きた場合は、ほとんどがモデルの論理的な誤りであるのでステップ 1 へ戻る。エラーが

ない時はステップ4に進む。

〈ステップ4〉

FORTRAN のプログラムが実行され、各種出力機器に結果が出力される。

DECS のシステムプログラムは yacc, lex などのツールを用いておらず、すべて FORTRAN 言語で作成しており、システムプログラムの大きさは約 3,000 行である。また状態変数の最大数は 50、ソースプログラムの行数は 700 行までである。

3. シミュレーションの応用例

ここでは具体的なシステムへの応用について述べる。

3. 応用例 1

応用例 1 として線形制御系への応用を取り挙げた。この例として文献 (9) で取り上げた電力システムの自動周波数制御装置へ適用してみた。図 1 はこのブロック図を示す。この図より

$$\frac{dx_1}{dt} = -K_{i1}(b_1 \cdot x_4 + x_9) \quad (2)$$

$$\frac{dx_2}{dt} = (x_1 - x_2 - x_4/R_1)/T_{sg1} \quad (3)$$

$$\frac{dx_3}{dt} = (x_2 - x_3)/T_{t1} \quad (4)$$

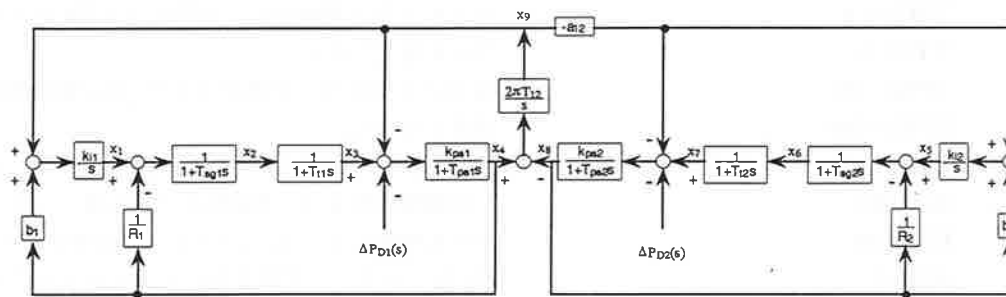

$$T_{sq1}=T_{sq2}=0.4(s), T_{11}=T_{12}=0.5(s), T_{ps1}=T_{ps2}=20(s), k_{ps1}=k_{ps2}=100, R_1=R_2=3, b_1=b_2=0.425, k_{11}=k_{12}=0.09, 2\pi T_{12}=0.05, a_{12}=1$$

図1 応用例1

Fig. 1 Application 1

$$\frac{dx_4}{dt} = (K_{ps1}(x_3 - x_9 - \Delta P_{d1}) - x_4) / T_{ps1} \quad (5)$$

$$\frac{dx_5}{dt} = -K_{i2}(b_2 \cdot x_8 - a_{12} \cdot x_9) \quad (6)$$

$$\frac{dx_6}{dt} = (x_5 - x_8 / R_2 - x_6) T_{sg2} \quad (7)$$

$$\frac{dx_7}{dt} = (x_6 - x_7) / T_{t2} \quad (8)$$

$$\frac{dt_8}{dt} = (K_{ps2}(x_7 - \Delta P_{d2} + a_{12} \cdot x_9) - x_8) / T_{ps2} \quad (9)$$

$$\frac{dx_9}{dt} = T_{12}(x_4 - x_8) \quad (10)$$

が成り立つ。上記の方程式に基づいて DECS によるソースプログラムは以下のように記述される。

```

10  >TITLE ["POWER SYSTEM"]
20  >TIMER [20, .01]
30  >PRINT [D: DTP1, X4, X9]
40  * CONSTANTS
50    TSG1=.4
60    TT1=.5
70    TPS1=20
80    KPS1=100
90    R1=3
100   B1=.425
110   KI1=.09
120   TSG2=.4
130   TT2=.5
140   TPS2=20
150   KPS2=100
160   R2=3
170   B2=.425
180   KI2=.09
190   A12=1
200   T12=.05
210   DPD2=0
220  * END
230  * DYNAMICS
```

```

240   U=STEP (1)
250   DPD1=.01 * U
260   DOT(X1)=-KI1 * (B1 * X4 +
      X9)
270   DOT(X2)=(X1 - X2 - X4/R1)/
      TSG1
280   DOT(X3)=(X2-X3)/TT1
290   DOT(X4)=(KPS1 * (X3-X9-
      DPD1)-X4)/TPS1
300   DOT(X5)=-K12 * (B2 * X8-
      A12 * X9)
310   DOT(X6)=(X5 - X8/R2 - X6)/
      TSG2
320   DOT(X7)=(X6-X7)/TT2
330   DOT(X8)=(KPS2 * (X7-DPD2+
      A12 * X9) - X8)/
      TPS2
340   DOT(X)=T12 * (X4-X8)
350   * END
```

10 行から 30 行は制御文であり、この部分の各行の説明を行うと、
 10 行：タイトル文で、グラフィック画面に出力されるタイトルを記述
 20 行：時間制御文で、シミュレーションの最終時間は 20 秒、演算刻み時間は 0.01 秒であることを意味する。
 30 行：出力制御文で、シミュレーション結果はディスク D に格納され、DPD1, X4, X9 は出力される変数である。
 40 行から 220 行：定数設定文で、各定数の値を設定している。
 230 行から 350 行はシステム記述文である。また初期値設定文は、初期値をすべて零としているため省略している。以下その中の説明を行う。
 240 行：ステップ関数を用いてシステムへの入力を記述した。
 250 行：DPD1 はブロック図中の ΔP_{d1} を表わし、電力系統の負荷変動 (P.U.) を意味する。
 260 行から 340 行：方程式 (2)~(10) に対応す

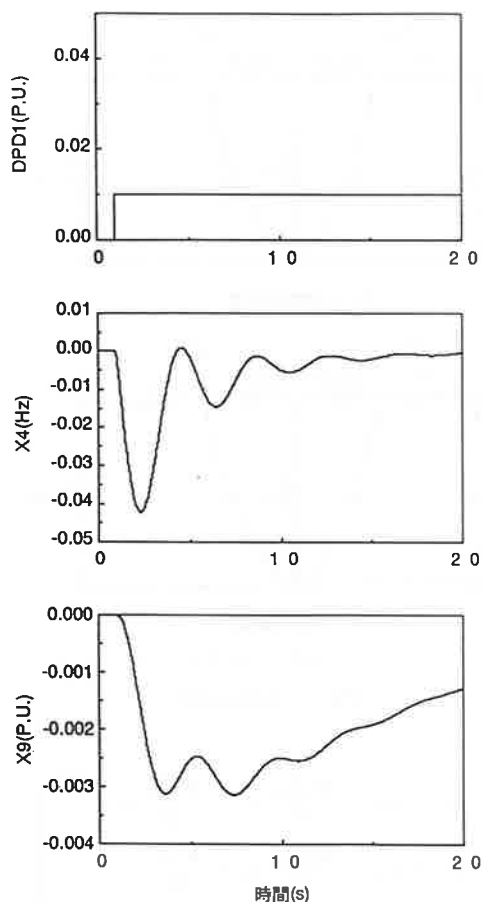


図2 応用例1のシミュレーション結果

Fig. 2. Simulation of application 1

る。なお X_4 は系統の周波数変動 (Hz), X_9 は連系線の潮流変動 (P.U.) を意味する。

ソースプログラムのコンパイルと実行は2.4で述べた手順で実行され、230行から350行の範囲が時間制御文で指定された回数だけ反復計算される。

図2はシミュレーション結果を示す。これらの出力結果は文献(9)と一致していることが確認された。

3.2 応用例2

応用例2として非線形制御系への応用を試みた。図3はこの制御系¹⁰⁾を示し、リミッタおよび

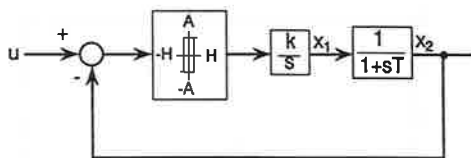


図3 応用例2

Fig. 3 Application 2

ヒステリシスの二つの非線形要素を含んでいる。このシステムをシミュレーションするソースプログラムは以下になる。

```

10  <TITLE ["NON LINEAR"]
20  >TIMER [40, .04]
30  >PRINT [S: U, X1, X2]
40  * CONSTANTS
50    K=1
60    KS=100
70    T=1
80  * END
90  * DYNAMICS
100  U=STEP (2)
110  SU=U-X2
120  Y1=HSTRSS (0, -.2, .2, SU)
130  Y2=KS * Y1
140  Y3=LIMIT (-1, 1, Y2)
150  DOT(X1)=K * Y3
160  DOT(X2)=(X1-X2)/T
170  * END

```

30行の出力制御文のSは変数U, X_1 , X_2 を画面にグラフィック表示することを意味している。120行はヒステリシス要素を示し図3の $h=0.2$ を与え、また初期値0の値を与えている。また140行はリミッタ要素を示し図3の $A=1$ の値を与えている。130行はリミッタ要素の線形部分の傾斜角度を急峻にするために利得KSを乗じている。

図4はシミュレーション結果である。定常状態における発振周期は1.57 Hz, また振幅は0.8で文献(10)で解析されている結果と一致している。

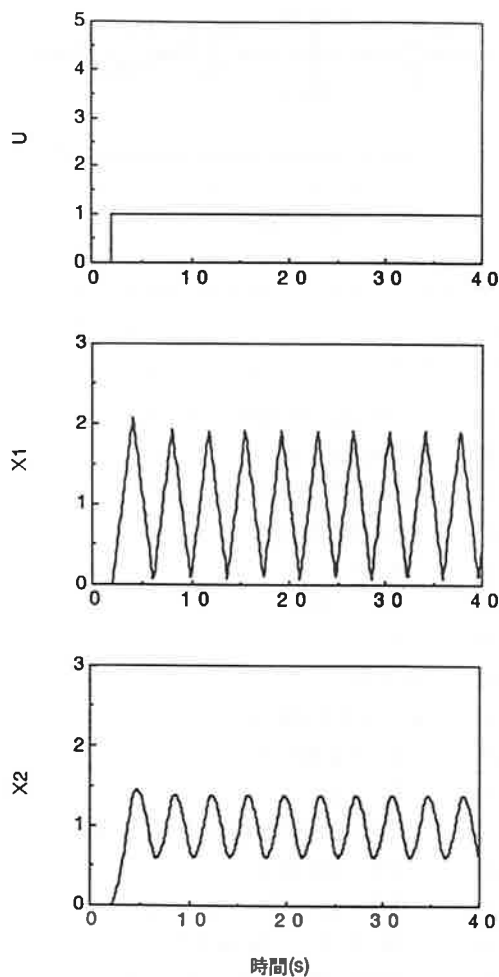


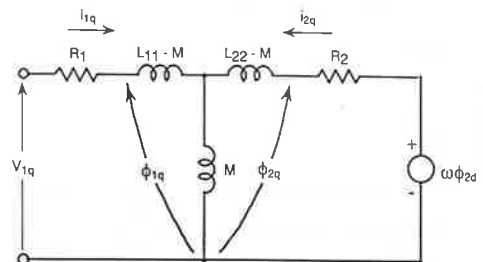
図4 応用例2のシミュレーション結果

Fig.4 Simulation of application 2

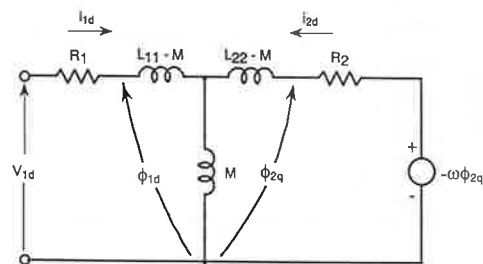
3.2 応用例3

つぎに電気機械への応用を取り挙げ、誘導機の起動特性のシミュレーションを行った。図5に誘導機のq座標軸, d座標軸等価回路を示す¹¹⁾。この等価回路から以下の式が導かれる。

状態方程式は



(a) q座標軸等価回路



(b) d座標軸等価回路

図5 応用例3

Fig.5 Application 3

$$\left. \begin{aligned} p x_1 &= v_{1d} - \frac{R_1 L_{22}}{D} x_1 + \frac{R_1 M}{D} x_3 \\ p x_2 &= v_{1q} - \frac{R_1 L_{22}}{D} x_2 + \frac{R_1 L}{D} x_4 \\ p x_3 &= v_{2d} - \frac{R_2 L_{11}}{D} x_3 + \frac{R_2 M}{D} x_1 - x_5 x_4 \\ p x_4 &= v_{2q} - \frac{R_2 L_{11}}{D} x_4 + \frac{R_2 M}{D} x_2 + x_5 x_3 \\ p x_5 &= (T - T_d)/J \end{aligned} \right\} \quad (11)$$

ここで

$$(x_1, x_2, x_3, x_4, x_5)^t = (\phi_{1d}, \phi_{1q}, \phi_{2d}, \phi_{2q}, \omega)^t$$

$$p = \frac{d}{dt}$$

また電流は

$$\left. \begin{aligned} i_{1d} &= \frac{L_{22} x_1 - M x_3}{L_{11} L_{22} - M^2} \\ i_{1q} &= \frac{L_{22} x_2 - M x_4}{L_{11} L_{22} - M^2} \end{aligned} \right\} \quad (12)$$

$$\begin{cases} \dot{i}_{2d} = \frac{L_{11}x_3 - Mx_1}{L_{11}L_{22} - M^2} \\ \dot{i}_{2q} = \frac{L_{11}x_4 - Mx_2}{L_{11}L_{22} - M^2} \end{cases} \quad (13)$$

トルクは

$$T = \phi_{2q}i_{2q} - \phi_{2d}i_{2q} \quad (14)$$

したがって (11) 式の状態方程式を求めることにより (12), (13) 式より $i_{1d}, i_{1q}, i_{2d}, i_{2q}$ が求められ, 実際の電流 i_a, i_b, i_c が以下のようにして得られる¹²⁾。

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & 0 \\ -1/2 & \sqrt{3}/2 \\ -1/2 & -\sqrt{3}/2 \end{bmatrix} \begin{bmatrix} i_{1d} \\ i_{1q} \end{bmatrix} \quad (15)$$

以上の式を基にしてソースプログラムを作成すると以下ようになる。

```

10  >TITLE ["INDUCTION
    MACHINE"]
20  >TIMER [.25, .0005]
30  >PRINT [D: V, I, T, X5]
40  * CONSTANTS
50  R1=.723
60  R2=.59
70  L1=.0023
80  L2=.0017
90  M=.0574
100 L11=L1+M
110 L22=L2+M
120 J=.01
130 W=2 * 3.14159 * 50
140 D=L11 * L22-M ** 2
150 TD=0
160 * END
170 * DYNAMICS
180 V=STEP (0.015)
190 V=V * SQRT(2.0/3.0) * 210 *
    COS(W * TIME)
200 V1D=V * COS(W * TIME)

```

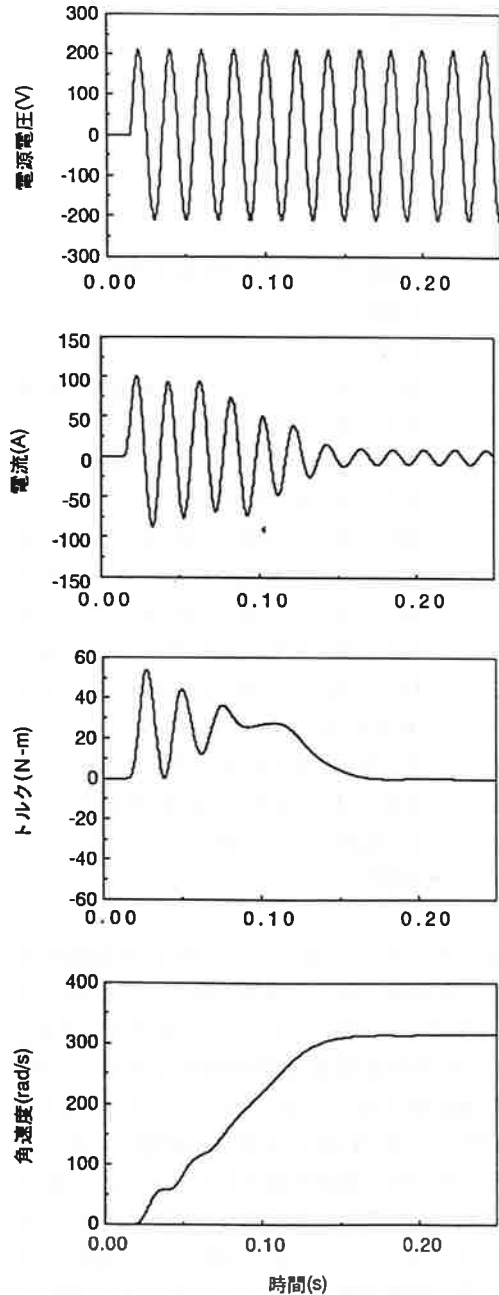


図 6 応用例 3 のシミュレーション結果

Fig.6 Simulation of application 3

表1 シミュレーション時間

例 題	ソースプログラム行数	コンパイル時間 (秒)	実行時間 (秒)
応用例 1	36	9	13
応用例 2	17	7	5
応用例 3	32	9	4

```

210  V1Q=V * SIN(W * TIME)
220  V2D=0
230  V2Q=0
240  DOT(X1)=V1D-(R1 * L22/D) *
    X1+(R1 * M/D) * X3
250  DOT(X2)=V1Q-(R1 * L22/D) *
    X2+(R1 * L22/D) * X4
260  DOT(X3)=V2D-(R2 * L11/D) *
    X3+(R2 * M/D) * X1-X5 * X4
270  DOT(X4)=V2Q-(R2 * L11/D) *
    X4+(R2 * M/D) * X2+X5 * X3
280  DOT(X5)=(M * (X3 * X2-X1
    * X4)/D-TD)/J
290  T=M * (X3 * X2-X1 * X4)/D
300  I1D=(L22 * X1-M * X3)/D
310  I=SQR(2.0/3.0) * I1D
320  * END

```

上記プログラムで40行から160行は定数設定文で、各定数は図5の等価回路の記号に対応する。170行から320行はシステム記述文である。180行のSTEP関数は電源電圧Vの投入時間を15ms遅らせるために用いている。200行、210行のV1D、V1Qはd軸、q軸電圧を表している。またかご型誘導機を用いたため、回転子のd軸、q軸電圧は零であり、220行、230行はこのことを示している。240行から280行は(11)式の状態方程式を示している。また290行のTはトルクを表し、(13)、(14)式から導かれる。また310行は電流を示し、(12)、(15)式から得られる。図6はシミュレーション結果である。誘導機に印加される電源電圧の投入位相は0°である。文献(12)に示した通り、過度電流、回

転角速度については実測値と良く一致することが確かめられた。トルクは実測していないが、他の諸量が実測値と一致していることから判断して妥当な値であると考えられる。

4. シミュレーション時間

つぎにシミュレーションの時間について述べる。シミュレーションの時間はコンパイルと実行時間から成る。コンパイル時間はソースプログラムの行数に関係し、また実行時間は時間制御文で指定される繰り返し回数、出力制御文で指定される出力変数の数、出力機器の種類に依存する。またシミュレーション速度はCPUの種類にも関係するので、ここでは32ビットパーソナル・コンピュータのCPU80386と数値演算プロセッサ-80387の構成でRAMディスクに出力した場合について述べる。

表1は3で述べた応用例についてシミュレーション時間を比較した表である。例題3.2で、コンパイル時間が例題3.3よりも短いにもかかわらず実行時間が長いのは、シミュレーションの繰り返し回数が多いためである。いずれの場合もシミュレーションに要する時間は現実的な時間内で演算が行われており、DECSによるシミュレーションは実用性が高いと思われる。なおより速度を上げる必要のある時はトランスピュータ^{16),17)}を用いたり、ワークステーション⁸⁾で実行させることも可能であり、この場合の演算速度はおおよそ3~5倍程度であることが確かめられた。

5. む す び

以上連続系シミュレーション言語 DECS の構成, 応用例, 演算速度について述べた。DECS はシステム記述言語として FORTRAN を用いているため, FORTRAN 言語の利用者にとって理解しやすく, また移植性が高くパーソナルコンピュータ, ワークステーション, 大型機など各種計算機資源を利用できる。ただしグラフィック出力機能に関しては各計算機資源に依存するので, 移植に関してはそれぞれの計算機資源の有するグラフィック機能を配慮する必要がある。しかし近時グラフィック機能に関しても国際的な標準化が進められており, GKS, PHIGS などが ISO の標準になっている。本稿でもワークステーションへの移植に際しては GKS を用いた。

本言語は各種制御系, パワーエレクトロニクス回路¹⁴⁾, 電力系統¹³⁾などの実システムのシミュレーションに適用した結果有用性が確かめられ, 現実的な利用に耐える言語であると思われる。

6. 参 考 文 献

- 1) IBM: CSMP Reference Manual
- 2) NEC: 連続系シミュレーション言語説明書
- 3) 松坂: パーソナルコンピュータによる連続系シミュレーション言語, シミュレーション, 第5巻第3号, pp. 47-55
- 4) 臼井: 会話型連続系シミュレータ: Σ , 第3回シミュレーション・テクノロジー・コンファレ

- ンス 1B-3 (1983)
- 5) ACSL, サイバネットシステム株式会社
- 6) 松坂: パソコンから大型機まで実行可能な連続系シミュレーション言語, SENAC, Vol. 21, No. 3, 76/88 (1988)
- 7) T. Matuzaka, S. Ookawa: Differential equation oriented simulation language implemented on personal computers, Proc. of system simulation and scientific computing, 319/324, Beijing, China, 1989
- 8) 松坂: ワークステーションによる連続系シミュレータ, 平成3年度電気関係学会東北支部連合大会講演論文集, 116, p. 353
- 9) I.J. Nagrath, D.P. Kothari: Modern power system analysis, p. 262, TATA Macgraw Hill Publishing Company Ltd., 1980
- 10) 松坂: パソコンによる動的システムとシミュレーション, p. 155, 工学図書出版, 昭和61年
- 11) 秦泉寺, 内藤訳: パワーエレクトロニクス & AC ドライブ, 電気書院, p. 61
- 12) 松坂, 佐々木: 誘導機の起動電流シミュレーション, 第10回シミュレーションテクノロジー・コンファレンス, p. 165
- 13) 佐々木, 松坂, 土屋: 風力駆動誘導発電機の系統並列時における電圧変動シミュレーション, 電気学会論文誌, p. 33-39, Vol. 110-B, No. 1, 1990
- 14) 松坂, 佐々木: 適応型 PWM インバータのシミュレーション, 平成2年度電気関係学会東北支部連合大会講演論文集, 1F15
- 15) Francois E. Celler: Hierarchical nonlinear bond graphs: a unified methodology for modeling complex physical system, SIMULATION, 230/247, April, 1992
- 16) 松坂: トランジュータによるシミュレーションの高速化, 第9回シミュレーションテクノロジー・コンファレンス, p. 243
- 17) 松坂: トランスポュータ・アクセラレータによる連続系シミュレーション, 平成元年度電気関係学会東北支部連合大会講演論文集, 1C16