Vol. 37 No. 3　　情　報　処　理　　Mar. 1996

解　説

シミュレーション技術の最近の動向

# 2.3 連続型シミュレーション言語 ACSL の概要†

土　田　行　貞††

## 1. はじめに

工学，科学のさまざまな分野では，現実のシステムのシミュレーションが盛んに行われ，システムの挙動の予測に大きな成果をあげている．

シミュレーションモデルは状態の変化の仕方によって，主として待ち行列型モデルの離散系と，連立微分方程式モデルで表現される連続系の2つに分類される．連続系の解析対象は，制御系の設計，化学プロセスの表現，ミサイルや航空機のシミュレーション，電力プラントのダイナミクス，生態系の予測，薬の投与などの生物学的・医学的ダイナミクス，車両の操縦性，マイクロプロセッサコントローラ，流体，熱伝導解析など多岐にわたる．

ACSL (Advanced Continuous Simulation Language) は，これらの連続系の現象，特に時間依存の非線形微分方程式や伝達関数などによって表現されるシステムのモデル化を容易に行えるように開発された言語で，連続型シミュレーションツールとして広く利用されている．ACSL は米国 MGA (Mitchell & Gauthier Associates, Inc.) 社が FORTRAN をベースとして設計・開発した連続型シミュレーション言語で，1975 年に製品化され，この仕様は 1967 年に制定された CSSL (Continuous System Simulation Language) に準拠している．現在のバージョンでは，不連続な現象を簡単にモデル化したり，代数-微分方程式で表されるモデル，たとえば，多体系の機構モデルなどを解析できる機能が追加されている．ここでは，連続型シミュレーション言語 ACSL の概要と最新の機能について紹介する．

## 2. シミュレーションの手順

連続系モデルのシミュレーションを行うためには，問題の定義，モデルの定義およびモデルの実行の3つの段階に分けることができる．第1段階では，解析対象の問題の定義を，代数方程式や微分方程式で記述したりブロック線図でモデルとして表す．

第2段階では，このモデルを連続型シミュレーション言語 ACSL を使って記述し，モデル定義ファイルを作成する．モデル定義ファイルは FORTRAN に翻訳されて FORTRAN オブジェクトファイルとしてコンパイルされる．さらに積分や線形解析あるいはプロット出力するための実行時ライブラリなどをリンクしてシミュレーションの実行モジュールを作成する (**図-1**)．

最後に，実行時コマンドを使ってモデルを動かしシミュレーションを行う．コマンドはバッチモードで解析できるようにファイルに用意したり，あるいは対話形式で直接入力したりできる．作成されたモデルは，シミュレーションをして動かしたり，その結果を見たり，さらに定数を変更したりして何回でも対話形式で実験を行うことが可能になる．たとえば，さまざまなばね定数やアクチュエータの値でシステムを動かしてみることができる．

このように ACSL への入力としては，解析対象を表現するモデル定義プログラムと，シミュレーションを行う実行時コマンドの2つが必要となる．

## 3. ACSL のプログラム構造

**図-2** に ACSL のモデル定義プログラムの明示的な構造を示す．構造文としては PROGRAM,

† The Advanced Continuous Simulation Language (ACSL) by Yukisada TSUCHIDA (Application Software Division, Cybernet Systems Co., Ltd.).

†† サイバネットシステム(株)応用ソフトウェア事業部

INITIAL, DYNAMIC, DERIVATIVE および TERMINAL といったものが用意されている．簡単なモデルの場合は PROGRAM と END 文，あるいは DERIVATIVE と END 文だけの構造文を利用してモデル化することもできる．

INITIAL セクションはダイナミックモデルの計算を始める前に 1 度だけ実行される部分で，状態変数の初期条件の設定や，シミュレーション中に変化しない変数などを計算することができる．

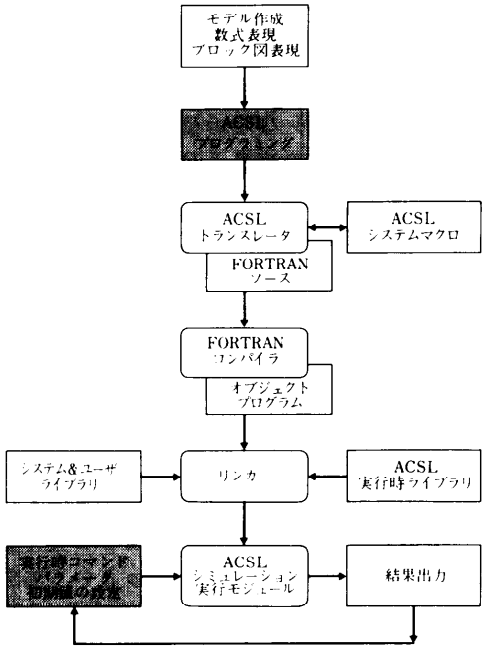


図-1 シミュレーションの流れ

```
PROGRAM
                セッションのはじめに 1 回だけ実行される
    INITIAL
                シミュレーションが始まる前に実行される
    END
    DYNAMIC
        DERIVATIVE
                連続的に積分される
        END
        DISCRETE
                離散的な時間点で実行される
        END
                出力間隔ごとに実行される
    END
    TERMINAL
                シミュレーションが終了した後に実行される
    END
END
```

図-2 ACSL プログラム構造

DYNAMIC セクションは出力刻み幅ごとに実行される部分で次に述べる DERIVATIVE セクションや DISCRETE セクションを含む．

DERIVATIVE セクションは微分方程式を書き連ね，指定した積分アルゴリズムでモデルをシミュレートする部分である．このセクションは複数設定でき，各々別々の積分アルゴリズムや積分刻み幅などを設定することができる．この部分がシミュレーションの核となる部分である．離散的な現象や不連続な現象をモデル化する場合は DISCRETE セクションを利用する．

TERMINAL セクションはシミュレーションが終了したときに実行される部分である．計算結果の判定を行い，実行の分岐を INITIAL セクションに戻し，パラメータの最適化などに利用できる．

FORTRAN サブプログラムは，ACSL の PROGRAM 文に対応した END 文の後に直接置いたり，あるいはそのオブジェクトファイルをリンクしたりすることによって利用することができる．

ACSL は DERIVATIVE セクションにあるコード，すなわち連続型モデルの方程式をソート（並べ換え）する．この点が，プログラムの実行が文の順序に依存している FORTRAN のような汎用プログラム言語とは対照的である．

## 4. ACSL のモデル例

図-3 に示す単振り子の例で，ACSL のモデル例を見てみよう．

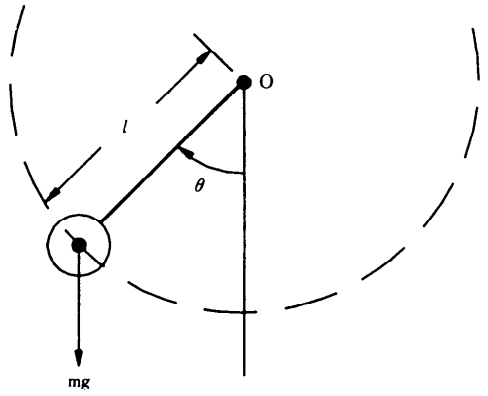


図-3 単振り子モデル

長さ $l$(m)の質量のない棒の一端をO点に固定，他の端に質量 $m$ (kg) のおもりを吊るし，これをO点を含む鉛直平面内で運動させるときの運動方程式を求める．ただし，O点では振り子の速度に比例する減衰力が働くものとすると，式は次のように表すことができる．

$$ml\ddot{\theta} + kl\dot{\theta} + W\sin\theta = 0$$

### 4.1 ACSLモデルの作成

この振り子のACSLモデル定義例を図-4に示す．ここでは，ACSLの構造を見ていただくためにINITIAL，DYNAMIC，DERIVATIVE，TERMINALなどの構造文を使ってモデルを表現した．

ACSL言語は，算術演算子，ACSL固有の演算子，FORTRAN関数（SQRT，MOD，IF-THEN-ELSEなど），シミュレーションを制御するACSL固有の文およびマクロ機能で構成される．シミュレーションを制御する文とその変数（IALG，MAXT，NSTP，CINTなど）には既定値が用意されている．したがって，このモデルは一般には約10行以内で表現できるものである．

```
PROGRAM damped nonlinear pendulum

INITIAL
     !-------Conversion factor, deg/rad
     DPR    = 45.0/(ATAN(1.0))
END   ! of Initial

DYNAMIC
DERIVATIVE
     !-------Integration algorithm and step size
     ALGORITHM IALG = 4
     MAXTERVAL MAXT = 0.0125
     NSTEPS    NSTP = 1

     !-------Constants for model (units in kg-m-sec)
     CONSTANT mass = 1.0    , g = 9.81
     CONSTANT length = 0.5  , kdamp = 0.3
     CONSTANT thdic = 0.0   , thic = 1.0

     !-------Angular acceleration of mass
     thdd   = -(mass*g*SIN(th) + kdamp*length*thd)/(length*mass)

     !-------Integrate for angular velocity and position
     thd    = INTEG(thdd, thdic)
     th     = INTEG(thd, thic)

END   ! of Derivative

     !-------Communication interval
     CINTERVAL CINT = 0.025

     !-------Termination condition
     CONSTANT  tstop = 4.99
     TERMT(T. GE. tstop)

     !-------Angle in degrees for output
     xth    = DPR*th

END   ! of Dynamic

TERMINAL

     !-------Call for debug dump
     LOGICAL  dump
     CONSTANT dump = .FALSE.
     IF(dump) CALL DEBUG

END   ! of Terminal
END   ! of Program
```

図-4　ACSLのモデル定義例

このモデルのINITIALセクションでは，シミュレーション中不変である角度変換の係数を計算している．

DYNAMICセクションはTERMT文内の論理関係が満足されるまで，出力刻み（CINT）ごとに実行される部分である．ここでは，角度の単位変換も行っている．

この中のDERIVATIVEセクションは，導関数変数を使って微係数を計算し，実際の積分を行う部分となる．積分アルゴリズム（IALG）は実行時に変更することが可能である．積分の刻み幅は，最大積分刻み幅（MAXT），出力刻み，および刻み数（NSTP）などの値で決定される．積分は，ACSL固有の演算子INTEGで実行する．この積分演算子がシミュレーションシステムの中心をなすものである．モデル作成では，微分演算子を積分演算子に変える必要がある．すなわち，階数の最も高い導関数変数を低い階数の導関数変数などで表すことが必要で，モデルの表現も分かりやすくなる．この例では，角加速度 thdd について解いて，それを順次積分して角速度 thd と角度 th を求めている．

TERMINALセクションは，終了条件すなわちTERMT内の論理関係が真になると実行されるブロックで，この例では，dumpという論理変数を実行時に真に設定したときに，デバック出力ができるように設定している．

定数はCONSTANT文を使って設定する．たとえば，減衰定数はkdampという定数で0.3を与えている．定数の記述箇所はどの位置でもかまわないが，一般にその定数が使用される直前に設定しておくと見やすくなる．なお，感嘆符“！”以下はコメントを表している．

### 4.2 シミュレーションの実行

実行時のコマンドはファイルから入力したり，直接キーボードから入力したりできる．図-5に実行時コマンド例を示す．

“prepar”コマンドは，変数のデータを記憶し，それをプロット表示できるように，変数リストを設定するものである．“start”コマンドでシミュレーションが終了条件まで実行され，“prepar”で指定した変数のデータが保存される．最初の“plot”コマンドで，振り子を初期角度1ラジアン

(rad) からはなしたときの，時間に対する角度と角速度の図が表示される．次の“plot”コマンドでは，横軸の変数を角度に変えて，角度に対する角速度の変化を表示してある．

“set”コマンドで定数の値を変更することができる．ここでは積分の初期値の初期角度を 0 rad，初期角速度を 10 rad/sec に設定し，終了時間を 9.99 秒まで延ばしている．“start”コマンドで再実行し，プロットの表示形式を“set”コマンドで変更し，“plot”コマンドで表示させたものが図-6 である．

```
prepar t, th, thd, thdd
start
plot th, thd
plot/xaxis=th, thd
set thic=0.0, thdic=10, tstop=9.99
start
set strplt=. t., calplt=. f.
plot/xaxis=t, th, thd, thdd
stop
```

図-5　実行時のコマンドの入力例

## 5. ACSL のモデル作成機能

ACSL 言語は，融通性がありだれにでも簡単にモデル化できるように設計されている．モデルの記述も 80 カラム内に自由に表現できるので見やすくできる．また，充実した演算子やステートメントにより，複雑なシステムや大規模なシステムもシンプルにモデル化できる．作成したモデルは，自動的に式の実行順序が考慮されて FORTRAN に翻訳され，既存のライブラリとのリンク，ユーザによる独自な機能の追加などができる．さらに，パラメータの設定だけでモデルを単精度モデルと倍精度モデルに切り換えることもできる．

ACSL のモデル作成上の特長は以下のようにまとめられる．

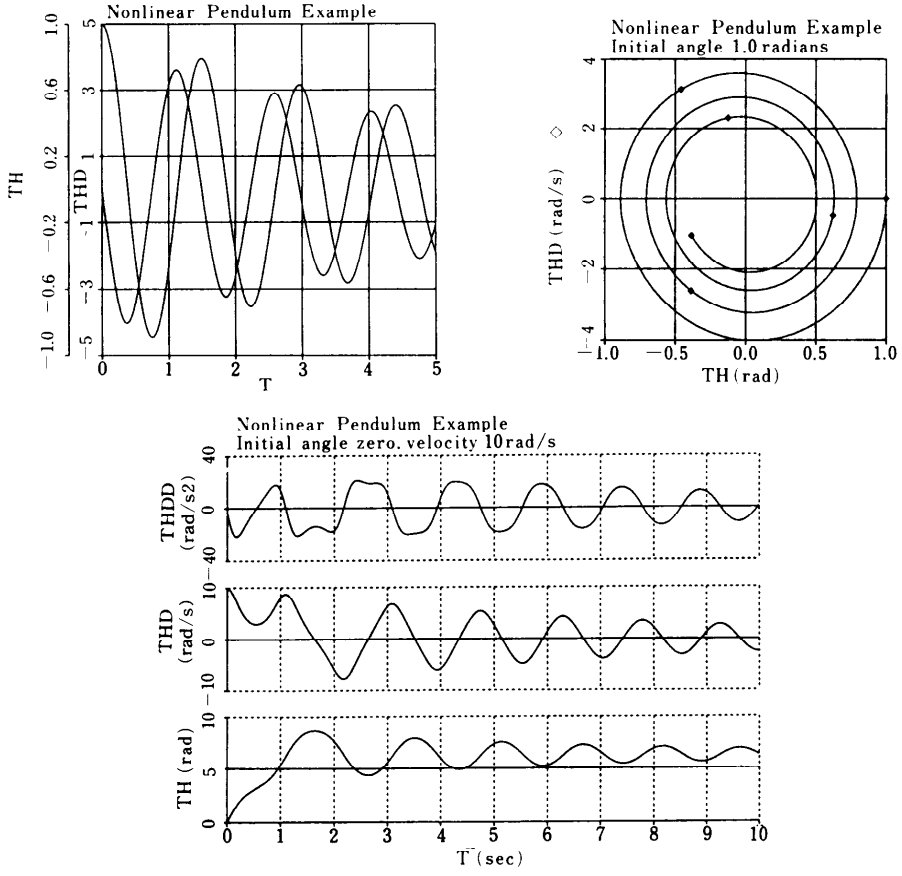


図-6　プロット出力例

●自動ソート機能により，プログラムの実行順序を考慮せずにモデル化できる．
●積分器，1次遅れ，2次遅れ，一般伝達関数などの線形演算子が利用できる．
●遅延器，制限器，スイッチ，バックラッシュ，不感帯，テーブル関数などの非線形演算子が用意されている．
●ステップ関数，パルス発生器，調和関数，ランプ関数，ノイズ発生器などの信号発生器がある．
●配列要素を使って，ベクトル積分や行列積分が表現できる．
●インプリシット演算子で，代数的に解けない代数ループシステムや代数-微分方程式で表されるシステムをモデル化できる．
●イベントファインダ機能により，衝突や摩擦などの不連続現象や，デジタルコントローラなどの周期的な離散現象を簡単かつ正確にモデル化できる．
●特殊で複雑な演算子を容易に作成できるマクロ定義機能が利用でき，新しい演算子をいくらでも定義でき，モデル化の効率を高めることができる．
●ユーザ定義サブルーチンを利用できる．

## 6. ACSLのシミュレーション実行機能

ACSLには，シミュレーション実行時の対話型処理を円滑に行うコマンドが豊富に用意されている．モデルのすべての変数や積分器の制御パラメータなどは実行時に参照・変更でき，モデルを作り直さなくてもさまざまな条件下で繰り返しシミュレーションできる．以下に，ACSLの実行時機能の特長を示す．
●コマンド列を1つの名前で定義でき，コマンド入力を省力化できる．
●次の積分アルゴリズムが利用できる．
Euler法，固定刻み
Runge-Kuttaの2次法，固定刻み
Runge Kuttaの4次法，固定刻み
Runge-Kutta-Fehlbergの2次法，可変刻み
Runge-Kutta-Fehlbergの5次法，可変刻み
Adams-Moulton法，可変次数，可変刻み
Gear's stiff法，可変次数，可変刻み
DASSL法，可変次数，可変刻み
●非線形モデルの線形化，固有値，固有ベクトルの計算，周波数応答，根軌跡などを表示できる．
●シミュレーション結果を即座にグラフィック表示でき，多彩なグラフ出力が簡単に行える．計算しながらプロットさせることもできる．
● ACSL固有のデバッグ機能により，モデル化のエラーなどを迅速に判定できる．

## 7. ACSLの最新の機能

ACSLは，改良を重ねてCSSLにはなかった機能が数多く利用できるようになっている．たとえば，scheduleという不連続現象の発生する時間を検出する文で，衝突現象やデジタル回路などを容易にかつ正確にモデル化できるようになった．また，拘束された運動方程式などの代数－微分方程式も解析できるようになり，これを効率よく解くDASSLという積分アルゴリズムも用意されている．さらに，ブロック図などのグラフィックス入力でACSLモデルが作成できるものも製品化されている．

### 7.1 グラフィックスモデラ（ACSL/GM）

ACSL/Graphic Modeller (GM) はACSLのサブセットで，ブロック線図でACSLモデルを作成し，パラメータの変更やプロット変数の指定など実行時環境をグラフィックス上で行えるようにしたものである．このACSL/GMにはACSLの基本演算子がブロックとして用意され，新しいブロックも作成することができる．図-7にロケットエンジンや人体の血液の流れをACSL/GMでモデル化した例を示す．たとえば伝達関数などのブロックを割り付け，それらを結線するだけでACSLのモデルができあがる．また，ブロックの機能を表すように描画データを各ブロックに割り当てると，モデルの内容を視覚的にとらえやすくなる．このACSL/GMの特長をまとめると次のようになる．
●作成したブロックは階層構造にでき，ビジュアルな形のアイコンで表示できる．
● ACSLコードを直接ブロックにしたり，FORTRANやCのサブルーチンをブロック内で呼び出すことができる．
● UNIXとWindowsの環境下で利用できる．

### 7.2 代数-微分方程式の解析機能

最新のACSLでは，ある残差をゼロにするように定義した，代数的に拘束された微分方程式も

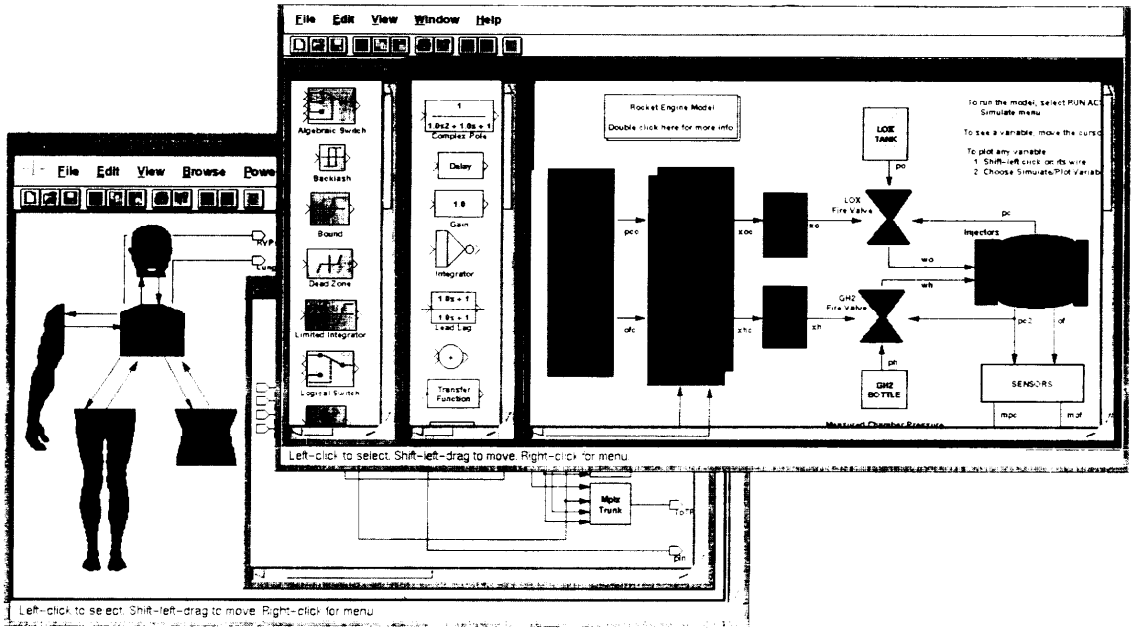


図-7 ACSL/GM でのグラフィックス入力の例

積分できるようになった.

たとえば，状態変数と代数変数が次のように表現されるときは，

$$\dot{Y}=F(Y,Z)$$
$$0=G(Y,Z)$$

ACSL では変数がスカラの場合，

$Y$=INTEG（$F$（$Y$, $Z$), YIC)
$Z$=IMPLC（$G$（$Y$, $Z$), ZIC)

あるいは $f$ と $r$ などがベクトルであれば次のように表現できる.

$y$=INTVC（$f$, yic)
$z$=IMPVC（$r$, zic)

ACSL は，Broyden 改良を行った Newton-Raphson の反復法を使って，残差（$G$ あるいは $r$）が常にゼロになるように代数ループを解くことができる.

この機能は，ジョイントなどで結合された多体系のモデル，いわゆる機構モデルを解析する場合や，あるいは電気回路の接点での電圧をその接点での電流の総和がゼロの条件下で求めたりする場合（Kirchhoff の法則)，さらに化学プラントのシミュレーションなどに対して有効である.

## 8. ACSL 関連ソフトウェア

ACSL の関連ソフトウェアとして，ブロック・ダイアグラムで ACSL コードを生成する ACSL フロントエンドプロセッサの ACSL/GM や，ACSL の解析結果を可視化しアニメーションする ACSL/Vision，さらに ACSL 言語を使ったリアルタイムシミュレーションシステム ACSL/Real Time などが MGA 社によって開発されている.

さらにサードパーティからは，ACSL モデルの最適化ツールとして Optdes や SimuSolv，電力プラント用のモデリングシステムとして MMS，ボンドグラフによるフロントエンド・プロセッサの CAMP，機構系モデルから ACSL コードを生成する Autosim などが製品化されている.

ACSL は，さらに制御系 CAD の MATLAB とインタフェースがとれている．ACSL の実行時コマンドで “matlab” と入力するだけで，ACSL の解析結果をすべて MATLAB に渡すことができ，その中の行列演算機能などを使って信号処理や制御系の設計が可能となる.

## 9. ま と め

ACSL に代表され CSSL タイプの連続型シミュレーション言語は，かつては大型コンピュータで利用されてきた．しかし，近年のコンピュータ

のダウンサイジングにより，ほとんどのユーザはワークステーション，パソコンで利用する傾向にある．ACSLも用途に応じてスーパコンピュータからパソコンまでさまざな機種で利用できるようになっており，モデル化や実行の仕方は機種に依存しない．

最近は，連続系に対応したさまざまなシミュレーション言語や具体的な用途に応じたCEAソフトができている．しかし，ACSLの場合は解析対象をユーザが目的とする最少の変数だけでモデル化できるので，無駄なく，短時間に解析できる．また，モデル化や実行方法にも融通性があり，初心者でもすぐにシミュレーションを行うことができる．CSSLに準拠したACSLは，積分アルゴリズムの信頼性，正確さから今後も連続型シミュレーション言語のスタンダードとして利用されていくものと思われる．

## 参 考 文 献


1) Simulation Councils, Inc. (SCi) : The SCi Continuous System Simulation Language, SIMULATION 9, pp. 281-303 (1967).
2) Mitchell and Gauthier Assoc. Inc.: ACSL Reference Manual, Edition 10. 1. (1993).
3) Mitchell and Gauthier Assoc. Inc.: ACSL Beginner's Guide, Edition 10. 1. (1992).
4) MGA Software: ACSL/Graphic Modeller for Windows, Version 3. 1 User's Guide (1995).


(平成7年3月6日受付)

**土田 行貞**

1956年生．1979年秋田大学鉱山学部鉱山地質学科卒業．1981年同大学院鉱山学研究科修士課程修了．1985年日本シーディーシー(株)に入社．同年サイバネットシステム(株)設立と同時に移籍．現在，同社応用ソフトウェア事業部に所属．