1997年度日本オペレーションズ・リサーチ学会
秋季研究発表会

数理計画のためのモデリング言語 SIMPLE　I　概要

01701240　数理システム　　*山下浩　YAMASHITA Hiroshi
01307380　数理システム　　田辺隆人　TANABE Takahito
　　数理システム　　逸見宣博　HENMI Nobuhiro

1. はじめに

SIMPLEは，1．システムの最適化，2．偏微分方程式の数値的解析，3．離散系・連続系のシミュレーション，等における利用を想定して開発されたシステムのモデリングと解析のための言語である．

形式的にはC++上のクラスライブラリとして実現されていて，ユーザが作成したモデル及びデータを入力として1．モデル全体の情報(変数/関数の総数)，2．式同士の関係(等式/不等式)，3．式の種別(線形/2次/非線形)，4．特定の変数値(の組)に対応する関数値及び微係数，をモデルの解析を行うプログラム(ソルバ)に提供する．ソルバとは独立していて，情報の受け渡しは汎用のインタフェースを通じて行う．ソルバは必要に応じてモデルの内容をSIMPLEから得てシステムの解析を進める．インタフェースが汎用であることから，ユーザは自由に各種ソルバをSIMPLEと連結することが可能である．

上に挙げた応用分野のモデル記述を行う際に，言語に要求される機能としてたとえば次の様なものが考えられる．A．数学的関係の記述が自然な形で行える，B．ソルバとのリンクがユーザ自身により容易にできる，C．大規模モデルの記述が簡明にできる(モジュール，階層構造，反復構造の記述が可能)，D．自動微分の機能．

本稿ではSIMPLEにおけるモデル記述方法を説明し，主に上記の観点からその機能に関して論ずる．

2. SIMPLEによるモデル記述と解釈

簡単な非線形最適化問題(Hock & Schittkowski No.7)：

$$\begin{array}{rl} \text{変数} & x_1,\ x_2 \\ \text{最小化} & \log(1+x_1^2)-x_2 \\ \text{条件} & (1+x_1^2)^2+x_2^2=4 \\ \text{初期値} & x_1=2,\ x_2=2 \end{array}$$

を記述する場合を例に取る．以下はこれをSIMPLEによって記述したものである．

```
//  Hock & Schittkowski No.7
Variable x1,x2;
Objective f(type=minimize);
f = log(1+pow(x1,2)) - x2;
pow(1+pow(x1,2),2) + pow(x2,2) == 4;
x1 =2;  x2 = 2; // 変数の初期値
```

ここで Variable は SIMPLE が提供するクラスであり，変数を表す．変数 x1,x2 を用いて記述した式を目的関数を表すクラス Objective のオブジェクトの定義や，等式制約を表す演算子 == の両辺に配置することによって最適化問題(より一般的には「モデル」)を定義している．変数への代入はソルバの利用する初期値を与える．

C++ではユーザ定義のクラスオブジェクトに対する演算子や関数の動作を個別に定義する(overloading)ことができる．SIMPLEはその機能を応用してモデル記述をC++のコードとして一度だけ実行することによって解釈し，モデルの情報を計算グラフの形で蓄える．

上記の例では演算子 +,* や関数 log が Variable クラスのオブジェクトに対して実行されるたびに，計算グラフノードの作成と連結が行われる．モデル記述を実行することで内部に構成された計算グラフを利用して高速自動微分法[2]を適用することにより関数値や微係数の計算を行っている．

3. 式と方程式

SIMPLEでは通常のモデル記述と同様，段階的にモデルを記述することが可能である．たとえばこの例の場合，目的関数と制約式の共通部分をクラス Expression のオブジェクト t で表して，

```
//  Hock & Schittkowski No.7 (2)
Variable x1,x2;
Expression t = 1+pow(x1,2);
Objective f(type=minimize);
f = log(t) - x2;
```

```
pow(t,2) + pow(x2,2) == 4;
x1 =2;  x2 = 2;
```

と記述することができる．この様な式の定義方法は計算グラフの簡約化に寄与し，関数値・微係数計算の高速化のためにも有効である．また，関係演算子==, =<, => によって関係付けられた制約式は Constraint または，その別名の Equation というオブジェクトに代入することができる．

```
Constraint g;
g = pow(1+pow(x1,2),2) + pow(x2,2) == 4;
```

SIMPLE は方程式の種別 (線形/ 2 次/非線形) を識別する機能を備えていて，問題が，線形計画，2 次計画，一般の非線形計画問題のいずれであるか判定することができる．

定数を表現するのはクラス Parameter である．Parameter は式との演算や後述する添字付けが可能で，他のパラメータの演算結果を代入したりデータファイルから値を入力したりすることができる．

```
Parameter r(name="rhs");
        // 右辺値 (値はデータファイルから)
pow(1+pow(x1,2),2) + pow(x2,2) == r;
```

は制約式の右辺を Parameter としてデータファイルから与えられるものとした例である．

この機能を利用すればモデル記述とデータを分離し，モデル記述を固定して複数の具体的なシステムを生成したり，データとは別にモデルを精密化する等の応用が可能である．

4. 大規模問題のための集合の概念

SIMPLE では，集合とその要素という概念を導入することによって，我々が通常用いている数式に近い自然な形で大規模モデルを簡明に記述することを可能にしている．

既存の言語系でこの様なモデルを記述する場合には，配列変数や繰り返しを表現するイテレタ (DO, for) を用いるのが一般的だが，式の記述自体と配列やイテレタの書式とが混在し記述全体が繁雑となる．

SIMPLE では数学的記述に現れる集合とその要素にそれぞれ直接対応するクラス Set. Element を用いてモデル記述を行う．

```
Set S;    //  集合  S
Element i(set=S);  //  S の要素
```

以下はパラメータと変数の宣言である．

```
Parameter a(index=i),b(index=i);
Variable x(index=i),y(index=i),z(index=i);
```

宣言時に与えられる index=i の書式で，オブジェクトが i で添字付けられること，すなわち i の動く範囲で定義されていることを示す．

以下の例ではパラメータ a を使用して制約式を記述する．

```
a[i]/x[i] >= y[i] + z[i];// 非線形制約式
```

i は数式における添字と同じく，集合の要素の総称であり特定の要素を表すものではない．要素を含んだ式の記述は要素の取り得る値すべてにわたって展開されて解釈される．

以下に条件式を伴なった線形な制約式を記述する．

```
y[i+1] - y[i] <= z[i], i<10 ;
y[i] <= z[i], i==10 ;
```

,(コンマ) 演算子以降の条件式は先行する要素の展開される範囲を限定する働きをする．この記述が実際の数式における

$$y_{i+1} - y_i \leq z_i, \quad (i < 10)$$

等の表現に類似していることに注意されたい．

以下の記述はひとつの目的関数の宣言と定義をあらわしている．

```
Objective cost(type=minimize);
cost = sum(b[i]*log(y[i]),i);
```

上記で sum() は数式における$\sum$に対応する関数であり，引数として与えられた要素の取り得る範囲すべてについての和を示している．

また，SIMPLE には集合演算

& (積集合) | (和集合) - (差集合) * (直積)

や集合に対する操作を行う関数

add() (要素の追加) remove() (要素の削除)

が用意されており，集合の内容をモデル記述内において定義することが可能である．

参考文献

[1] 山下浩，田辺隆人，逸見信博，富永純. システムのモデリングと解析のための言語 SIMPLE 入門. Technical Report. Mathematical Systems Inc., Tokyo. 1994.

[2] 伊理正夫，久保田光一. 高速自動微分法 *(I)*. 応用数理.Vol.1.17-35.1991.

[3] 久保田光一. *C++* による高速自動微分法の処理系. OR 学会春季研究発表会予稿集.175-176.1989.