所 属 講 座	制御情報システム 講座	指 導 教 員	古賀 雅伸　准教授
学 生 番 号	06236025	氏　　名	景山　貴宏
論 文 題 目	Java による微分代数方程式パッケージの開発		

1　はじめに

自動車、ロケット、トランジスタ増幅器、化学プラント、閉リンク機構や外界との接触を持つロボットマニピュレータなど、陽的常微分方程式、代数方程式では表現することのできない制御システムが存在する。これらのシステムを表現するためには微分代数方程式を用いる必要がある。

本研究の目的は、微分代数方程式で表現されるシステムのモデリング・シミュレーションのための微分代数方程式パッケージの開発である。

2　DAE(微分代数方程式)

DAE（微分代数方程式）とは、代数方程式の代数拘束条件の影響を受ける常微分方程式である。[1] 一般的な形は、陰的常微分方程式

$$F(t, y, y') = 0 \tag{1}$$

の形で与えられる。また、制約関係を視覚的に確認しやすい形として、

$$\begin{aligned} x' &= f(t, x, z) \\ 0 &= g(t, x, z) \end{aligned}$$

のように半陽的な形式に変換できる。

微分代数方程式を理解する上で重要となる知識として指数があげられる。解 $y(t)$ の指数は、y' を y と t に関して解くときに（すなわち、y に対する常微分方程式を定義するために）必要な、連立方程式に対する最小の微分回数のことである。指数は解や初期条件に依存し、微分代数方程式の形だけで決まらないことに注意する必要がある。

2.1　数値解法

微分代数方程式に対する数値解法は大まかに 2 種類に分類できる。与えられた方程式の直接離散化法、また再定式化（例えば、指数下げ）と離散化を組み合わせた方法である。直接離散化法は、多段解法や Runge-Kutta 法のような離散化公式によって y と y' を近似する方法である。本質的に、指数 1 か半陽的指数 2 の微分代数方程式に限って使えるという制限がある。実際の方法としては、後退 Euler 法、後退微分公式（BDF）法、Radau 配列法などがある。

後退微分公式 (BDF) 法

例えば、一般形の微分代数方程式 (1) に等ステップ幅後退微分公式 (BDF) 法を適用すると、

$$F\left(t_n, y_n, \frac{1}{\beta_0 h}\sum_{j=0}^{k}\alpha_j y_{n-j}\right) = 0$$

となる。ただし、β_0 と $\alpha_j, j = 0, 1, ..., k$ は後退微分公式法の係数である。再定式化と離散化を組み合わせた方法としては、指数が 2 より大きい場合に適用される。ただし、この方法は、手間がかかることがあり、ユーザから多くの入力を要する場合がある。

3　org.mklab.nfc.dae パッケージ

L.Petzold 氏によって FORTRAN 77 言語で書かれた DASSL[2] を Java 言語で移植することで、パッケージを開発した。dae パッケージのアーキテクチャを図 1 に示す。 Fortran コードを Java で実装することで、

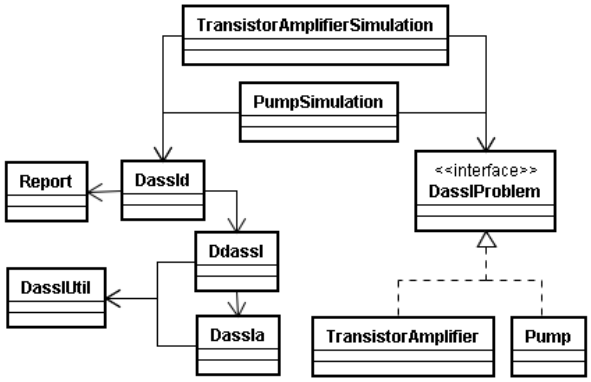

図 1: dae パッケージのアーキテクチャ

プラットフォームに依存しなくなり、オブジェクト指向でプログラムを扱えるようになるというメリットがある。DASSL は、ODE(常微分方程式)、IDE（陰的常微分方程式)、DAE(微分代数方程式) のソルバーを提供する。IDE、DAE に関しては、指数 1 以下の問題に対応している。数値解法として BDF（後退微分公式）法を利用している。

3.1　FORTRAN 77 言語から Java への実装方法

サブルーチン呼び出時、引数に配列を用いる

Fortran では、サブルーチンの呼び出し元で 1 次元配列を引数として与え、サブルーチンの定義部分で 2

次元配列として受けとるように記述されることがある。Java で実装する際は、1 次元配列と 2 次元配列の相互変換が必要となる。

GO TO 文

GO TO 文を移植する際、上方に実行が戻る場合は、戻る番地にラベル付き while(true) 文を配置し、GO TO 文をラベル付き continue 文で while 文に置換する方法を用いた。

4 例題による検証

本章では、CWI[3] の提供するテストプログラム、トランジスタ増幅器（指数 1 の硬い微分代数方程式）問題を用いて性能評価する。実験環境として、CPU に Athlon 64 X2 5200+、メモリは 2GB、OS として Microsoft Windows XP Professional を利用する。

トランジスタ増幅器

回路図を図 2 に示す。トランジスタ増幅器の回路のシミュレーションについて考える。 キルヒホッフの電

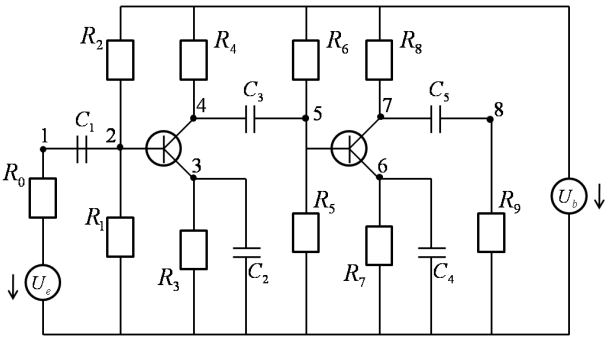

図 2: トランジスタ増幅器の回路図

流法則より、

$$M\frac{dy}{dt} = f(t, y), y \in \Re^8$$

指数 1 微分代数方程式が得られる。M はランク 5 の行列である。また、関数 $g(x)$、U_e を、

$$\begin{aligned} g(x) &= \beta(e^{\frac{x}{U_F}} - 1) \\ U_e(t) &= 0.1 \sin 200\pi t \end{aligned}$$

と与える。入力電圧 U_e を与え、増幅された出力電圧 U_8 を得る。y_0,y_0' の初期値を与え、この問題を Fortran と Java で実装された DASSL を用いて、$0 \leq t \leq 0.2$ の範囲を刻み幅 0.001sec で解く。

図 3 に、Fortran と Java の出力結果（$y(1)$ から $y(8)$）をそれぞれ重ねて描画したグラフを示す。ほとんどグラフが重なっており、計算結果が妥当であると言える。

5 おわりに

本研究では、DAE で表現されるシステムのモデリング・シミュレーションを行うため、FORTRAN 77 言語で実装された DASSL を Java に移植し、我々の研究室

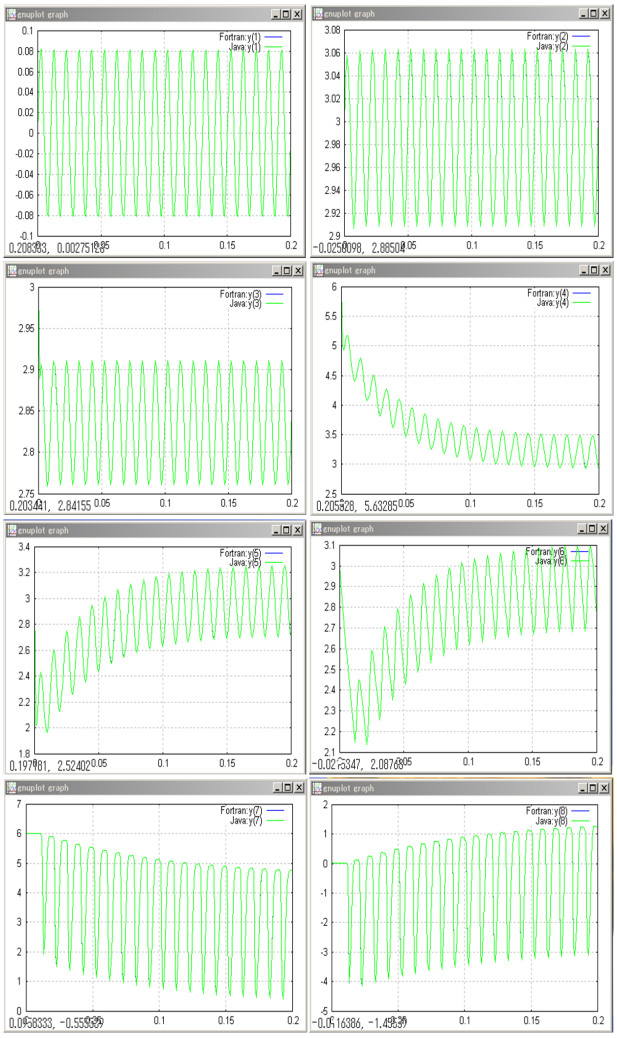

図 3: Fortran と Java の出力結果 ($y(1)$〜$y(8)$)

で開発している NFC に微分代数方程式パッケージを実装した。Java で実装したことにより、プラットフォームに依存しなくなり、オブジェクト指向の考えを用いて、理解しやすいコードにした。今後は、Radau 配列法などを用いて、高指数への拡張を行う必要がある。また、我々の研究室の多倍長計算パッケージ MPFloat との連携を図り、演算精度の向上を目指す。

参考文献

[1] U.M. アッシャー, L.R. ペツォルド. 常微分方程式と微分代数方程式の数値解析. 培風館, 2006.

[2] L.Petzold. Dassl. http://pitagora.dm.uniba.it/ testset/solvers/dassl.php/.

[3] Cwi. http://www.cwin.nl/en/.