所属学科	システム創成情報工学科	指導教員	古賀 雅伸　教授
学生番号	12236211	氏　名	林　裕之
論文題目	線形陰的な微分代数方程式で表されるシステムのための 多倍長シミュレーションパッケージの開発		

1　はじめに

ダイナミクスを表す常微分方程式と制約条件を与える代数方程式を連立した機構系や流体系，電気系などの多くのシステムは微分代数方程式で表現できる．そして，高次な微分代数方程式に対する汎用ソルバが開発され，Simulink や MapleSim，Mathematica などのモデリング・シミュレーションツールから利用されている．

本研究の目的は，モデリング・シミュレーションツール Jamox への導入を視野に入れた，線形陰的な微分代数方程式で表されるシステムのための多倍長シミュレーションパッケージの開発である．多倍長により，高度化・複雑化したシステムのシミュレーションの高精度な計算が可能となる．

2　DAE(微分代数方程式)

DAE は，代数方程式の代数制約条件の影響を受ける常微分方程式である [1]．一般的に陰的常微分方程式が

$$F(t, y, y')$$

で与えられ，y' に関して線形な場合

$$My' = f(t, y)$$

のような線形陰的な形式で表現できる．ただし，M は非正則行列である．また，特別な場合として

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

というような半陽的な形式に変換できる．しかし，陰的な形式から陽的や半陽的な形式に変換可能である場合は多いが，この変換は必ずしも数値的に容易ではなく，計算コストも高い．

DAE において重要な概念となる指数というものがある．これは，$y(t)$ を 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．ただし，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．

3　シミュレーションパッケージ

これまでに，TestSetForIVPSolvers[2] の中で紹介されている FORTRAN 言語で提供されているソルバを Java 言語で実装した DAE パッケージが開発されている [3]．このパッケージは，FORTRAN 言語から Java 言語に移植することでプラットフォームに依存しなくなり，オブジェクト指向を導入することで拡張性・保守性の向上を図ることを目的に開発された．Fig 1 に DAE パッケージの UML を示す．

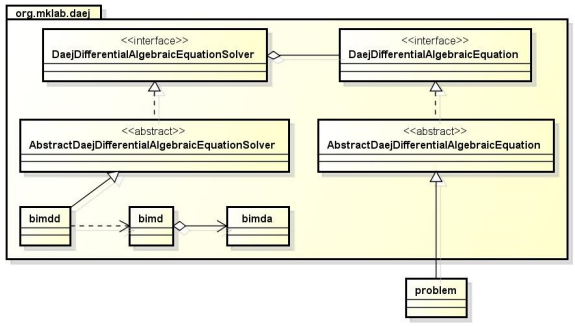

Fig 1: DAE パッケージの UML

ソルバと問題クラスのために用意されたインタフェースを実装した，抽象クラスを継承する形でソルバのドライバクラスと問題クラスの実装を行うことができる．そして，ドライバクラスは実際に問題を解くソルバクラスを呼び出し，演算に必要なルーチンを含む補助クラスを利用することで計算を行う．

3.1　BIMD ソルバの実装

これまでに，DASSL ソルバと RADAU5 ソルバの実装が完了していたが，本研究では新たに BIMD ソルバの実装を行う．BIMD とは，常微分方程式と指数 3 以下の DAE に対応したソルバである．第 3 章で説明したように，BIMD ソルバのドライバクラス，ソルバクラス，補助クラスを実装する．

4　例題による性能評価

本章では，FORTRAN と Java のソルバでそれぞれ DAE 問題を解き，計算結果と計算時間を比較することで実装した BIMD ソルバの性能評価を行う．実行環境は，CPU が Intel Core i5-3570，OS が Windows 7 Professional，メモリが 16GB となっており，FORTRAN のコンパイラとしては gfortran を使用する．また，評価の際に使用した DAE 問題を Table 1 に示す (以下，問題は指数で表現する)．

Table 1: 使用する DAE 問題

問題名	次数	指数
Transistor amplifier	8	1
Water tube system	49	2
Andrews' squeezing mechanism	27	3

4.1 計算結果

Table 2 は，FORTRAN の解を基準とした場合に Java の解でどれくらい誤差が発生しているかを示している．Table 2 に示すように，誤差は小さく収まっているため実装はほぼ問題ないと言える．

Table 2: FORTRAN に対する誤差

指数	最大絶対誤差	最大相対誤差
1	1.99×10^{-9}	1.23×10^{-9}
2	3.51×10^{-6}	5.38×10^{-6}
3	33.04	8.50×10^{-4}

4.2 計算時間

Table 3 は，FORTRAN と Java の計算時間と FORTRAN に対して Java が計算に何倍時間がかかったかを示している．Table 3 に示すように，FORTRAN に対して Java の方が遅い．原因としては，FORTRAN のデータ構造の仕様を満たすために配列操作を行っていることとガーベジ・コレクションの起動が考えられる．

Table 3: 計算時間の比較

指数	言語	計算時間 [ms]	比率
1	Fortran	37.10	9.319 倍
	Java	345.7	
2	Fortran	790.7	4.637 倍
	Java	3666	
3	Fortran	35.50	6.758 倍
	Java	239.9	

4.3 多倍長演算

Fig 2 に，性能評価に使用する硬い 2 自由度系を示す．この系を線形陰的な形式と陽的な形式で表し，線形陰的なものは BIMD ソルバで解き，陽的なものは定係数線形微分方程式なので厳密解を求めることができる．

Table 4 は，厳密解を基準とした場合に，多倍長精度浮動小数点数の仮数部ビット長を 53(倍精度)，80，100，150[bit] と設定して出力された解でどのくらい誤差が発生しているかを示している．Table 4 に示すように，精度を上げていくにつれて誤差が小さくなっていく傾向が見られる．したがって，多倍長演算を用いて精度を上げていくことにより，より良い解を得られる可能性があると言える．

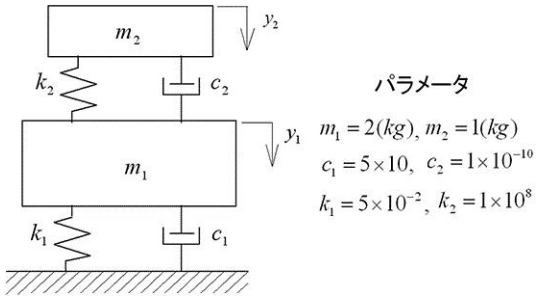

Fig 2: 2 自由度振動モデル

Table 4: FORTRAN に対する誤差

精度 [bit]	方程式	最大絶対誤差	最大相対誤差
53	y_1	2.12×10^{-9}	1.87×10^{-9}
	y_2	4.25×10^{-8}	7.78×10^{-8}
80	y_1	1.57×10^{-9}	1.38×10^{-9}
	y_2	3.14×10^{-8}	5.75×10^{-8}
100	y_1	5.50×10^{-9}	4.85×10^{-9}
	y_2	1.08×10^{-9}	1.98×10^{-9}
150	y_1	3.15×10^{-9}	2.78×10^{-9}
	y_2	6.08×10^{-10}	1.11×10^{-9}

5 まとめ

本研究では，DAE で表されるシステムのシミュレーションを行うため，FORTRAN で記述された BIMD ソルバを既存の DAE パッケージに Java 言語で実装し，例題を用いて性能評価を行った．性能評価では，実装は上手くいっているものの FORTRAN と比較して計算時間が遅いため，インスタンスを再利用するなどして高速化を図る必要がある．多倍長演算では，多倍長に対応できていることと精度を上げることでより精度の高い解を得られる可能性があることを確認できた．

今後の課題としては，新たなソルバの調査および実装を進めていくとともに Jamox との連携が挙げられる．

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] Francesca Mazzia and Cecilia Magherini. Test set for initial value problem solvers release2.4. Technical report, Department of Mathematics, University of Bari, April 2008.

[3] 景山貴宏, 古賀雅伸. 代数拘束をもつシステムの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. 自動制御連合講演会, Vol. 53, p. 111, 2010.