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

A Simulation Package for Systems of Differential Algebraic Equation

○ 景山 貴宏 (九州工業大学)　古賀 雅伸 (九州工業大学)

Takahiro Kageyama, Kyushu Institute of Technology

Masanobu Koga, Kyushu Institute of Technology

This papaer proposes package for modeling and simulation on the complex system such as being expressed as Differential algebraic equation. Our proposed package promote understanding of simulation method for users. And we use an example to assess the validity of our package.

Key Words: differential algebraic equation, simulation, algebraic constraint, software package

1　はじめに

マルチボディダイナミクス, 油圧機器, 電気回路などの分野においては, 微分代数方程式 (DAE:Differential Algebraic Equation, 以下 DAE) で表現される複雑な制御システムが頻繁に見られる. 本論文では,DAE で表現される複雑なシステムのシミュレーションを効率化するために開発した DAE 数値計算パッケージの紹介を行う. 本パッケージを用いることで, 方程式の形式や指数に応じて適切なソルバーの切り替えを行うことを可能にした.

パッケージ内のソルバーに関しては,TestSetForIVPSolvers[1] を参考に実装を行った. FORTRAN 言語で記述してあるコードを, プラットフォームへの依存性を抑え, 開発効率・保守性を向上させるため Java 言語での実装を行っている. また例題による性能評価として, パッケージ利用の有用性の検証,FORTRAN コードとのシミュレーション結果の比較を行った.

2　DAE

DAE とは, 微分方程式と代数方程式が連立した式のことを言う. 一般的な形式は, 陰的な常微分方程式

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

の形で与えられる. また, 微分方程式に対する代数方程式の代数拘束の関係を視覚的に確認しやすい形式として,

$$x' = f(t, x, z) \tag{2}$$

$$0 = g(t, x, z) \tag{3}$$

のように半陽的な形式でも与えられる. これら 2 つの形式は互いに単純な変換によりどちらの形式でも表現できるので同値の関係であると言える.

Fig. 1 に示す質量のない長さが l の一様な棒の先端に質量 m のおもりが取り付けられている単振り子を考える [2].

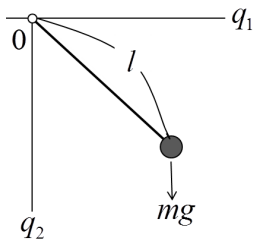

Fig. 1: single pendulum

振り子の運動を表す方程式は重力加速度を g とする 2 次元のデカルト座標で

$$mq_1'' + \lambda q_1 = 0 \tag{4}$$

$$mq_2'' + \lambda q_2 = mg \tag{5}$$

と表される. ただし λ はラグランジュの未定定数である. 棒の長さが l と一定であるから拘束条件は

$$g(q_1, q_2) \equiv q_1^2 + q_2^2 - l^2 = 0 \tag{6}$$

である. 式 (4),(5) は位置 (q_1, q_2) と速度 (v_1, v_2) を用いると次の 4 つの 1 階の微分方程式となる.

$$q_1' = v_1 \tag{7}$$

$$q_2' = v_2 \tag{8}$$

$$v_1' = -\frac{\lambda}{m} q_1 \tag{9}$$

$$v_2' = -\frac{\lambda}{m} q_2 + g \tag{10}$$

以上より, 式 (7)〜(10) は式 (2) の形の微分方程式, 式 (6) は式 (3) の形の拘束方程式となっていることが分かる.

2.1 指数

DAE 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 $y(t)$ の指数とは, 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 [3]. 連立方程式

$$y_1 = q(t)$$

$$y_2 = y_1' \tag{11}$$

に対して,y_2 の導関数を得るために連立方程式を微分すると,

$$y_1' = q'(t)$$

$$y_2' = y_1''$$

となる.y_1'' を得るために, 連立方程式をもう一度微分することで

$$y_2' = y_1'' = q''(t)$$

となり,$q(t)$ の 2 階微分が必要だったので指数は 2 となる.

また, 式 (11) は, いかなる解も自明な制約 $y_1 = q(t)$ を満たさなければならないだけではなく, 隠れた制約 $y_2 = q'(t)$ もあり, 解はすべての点 t においてこれを満たさなければならない. 従って適合する初期条件は $y_1(0) = q(0)$,$y_2(0) = q'(0)$ となる. 以上より, 指数は解や初期条件に依存し,DAE の形だけで決まらないことに注意する必要があることが分かる.

2.2 数値解法

DAE 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.

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.

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. 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.

3 DAE 数値計算パッケージ

本章では,DAE のシミュレーションを行うために作成した DAE 数値計算パッケージの説明を行う. 実際に数値計算を行うソルバーに関しては,FORTRAN 言語で提供されている TestSetForIVPSolvers[1] を参考に実装を行った.

3.1 TestSetForIVPSolvers

TestSetForIVPSolvers は, ソルバーを提供するだけではなく, 主に機械学, 電気回路解析, 天体力学などのテスト問題を提供する TestSet を構成している.

常微分方程式 (ODE:Ordinary Differential Equation, 以下 ODE), 陰的多次元常微分方程式 (IDE:Implicit Differential Equation, 以下 IDE), そして,DAE の初期値問題に対応するソルバーを持つ. それぞれのソルバーの特徴としては, 対応指数, 可解な方程式の種類の違いが存在することが挙げられる.TestSet に掲載されている数値計算ソルバーの一覧を Table 1 に示す.

Table 1: Solvers list in TestSet

solver name	index	problem type
VODE	-	ODE
DASSL	1 以下	IDE,DAE,ODE
MEBDFDAE	3 以下	DAE,ODE
MEBDFI	3 以下	IDE,DAE,ODE
PSIDE	3 以下	IDE,DAE,ODE
RADAU	3 以下	DAE,ODE
RADAU5	3 以下	DAE,ODE
BIMD	3 以下	DAE,ODE
GAMD	3 以下	DAE,ODE

また,FORTRAN 言語で記述してあり,TestSet のための構成となっているという点で, ファイル間, サブルーチン間の関係が強いものとなっており, 新たな問題を対応させたり, アルゴリズムの修正を行う時など大きな苦労を要する設計となっている.

3.2 DAE パッケージのアーキテクチャ

DAE パッケージのアーキテクチャをクラス図を用いて Fig. 2 に示す.

DifferentialAlgebraicEquatioProblem インターフェースを実装した抽象クラスを継承することで問題クラスを生成し, ソルバーの切り替えを行うために DifferentialAlgebraicEquationSolver インターフェースが存在するオブジェクト指向を用いた構成となっている. 新しくソルバーの開発, 問題の作成を行いたい場合にもインターフェースを実装した抽象クラスを継承すればよい.

問題クラス, 駆動クラスの共通部分をインターフェースとして抽出することでファイル間の役割を明確にし, 開発効率・保守性の向上を行った. このように実装する

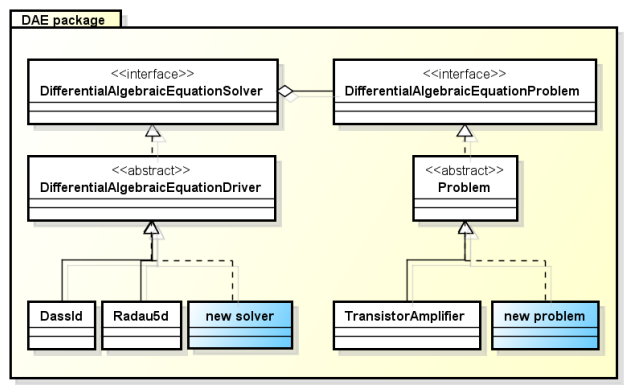

Fig. 2: DAE package

ことで,3.1 節で説明したような設計における問題は発生しない.

現在実装しているソルバーは,DASSL[4][5],RADAU5 である. 以下にそれぞれのソルバーの紹介を行う.

3.2.1 DASSL

DASSL は,L.Petzold 氏によって提供される, 指数 1 以下の IDE,DAE に対応するソルバーである. 数値解法としては, 次数 1〜5 の後退微分公式法を採用している. 以下に後退微分公式法の説明を行う.

後退微分公式 (BDF) 法

一般形の DAE(1) に等ステップ幅後退微分公式法を適用すると,

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

となる.y の導関数をこのように近似することで代数方程式に帰着することができる. 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.

3.2.2 RADAU5

RADAU5 は,E.Hairer 氏と G.Wanner 氏によって提供される, 指数 3 以下の ODE,DAE に対応するソルバーである. 数値解法としては, 陰的 Runge-Kutta 法の一種である 5 次の Radau IIA を採用しており, ステップ幅制御, 連続出力に対応しており, 問題を $My' = f(t, y)$ の形式として扱う. また,M は非正則行列である可能性をもつ. 以下に 5 次の RADAU IIA の説明を示す.

5 次の RADAU IIA

$My' = f(t, y)$ の形式に対して,s 段の陰的 Runge-Kutta 法を適用すると,

$$MK_i = f(t_n + a_i h, y_n + h\sum_{j=1}^{s} b_{ij}k_j), i = 1, 2, \cdots, s$$

となり,$k_i (i = 1, 2, \cdots, s)$ に関する方程式となり, これを解き,$k_i (i = 1, 2, \cdots, s)$ を求めることで,

$$y_{n+1} = y_n + h\sum_{i=1}^{s} w_i k_i$$

により解を求めることができる.

3.3 DAE パッケージ利用の手順

この節では, 開発したパッケージを用いてシミュレーションを行う手順の説明を行う.

1. 数学モデルの導出
   - $My' = f(t, y)$ の形式で数学モデルを導出
2. DifferentialAlgebraicEquationProblem インターフェースを実装した問題クラスに関数の実装を記述.
   - コンストラクタ
     - type,numeq の設定
     - numericaljac,lbandjac,ubandjac の設定
     - lbandmas,ubandmas の設定
     - 各方程式の指数を設定
   - init y と y' 初期化
   - feval 関数 f の定義
   - jeval ヤコビアン (df/dy) の定義
   - meval 係数行列 M の定義

type	問題の種類
numeq	方程式数
numericaljac	$\frac{\partial f}{\partial y}$ を内部的に計算させるなら true, 手動で入力するなら false
lbandjac	$\frac{\partial f}{\partial y}$ の対角下バンド幅
ubandjac	$\frac{\partial f}{\partial y}$ の対角上バンド幅
lbandma	M の対角下バンド幅
ubandmas	M の対角上バンド幅
index	指数を表す 1 次元配列

3. メインクラスの作成
   - 問題クラスのインスタンス生成

- 解を求める時刻 (初期時刻, 終端時刻) の設定
- numsamp の設定
- 利用するソルバーのインスタンスを生成
- 絶対誤差, 相対誤差, 初期ステップ幅の設定
  - DASSL の場合 → 絶対誤差, 相対誤差
  - RADAU5 の場合 → 絶対誤差, 相対誤差, 初期ステップ幅
- solve(t) による計算の実行
- printIS() による計算結果の出力

numsamp	範囲時刻におけるサンプリング回数
solve(t)	時刻 t における解を計算
printIS()	演算回数、ヤコビアン, 関数 f の評価回数などの出力

以上の流れでシミュレーションを実行する. また, 関数 jeval,meval では, それぞれヤコビアン, 係数行列 M を 2 次元配列として定義する際, バンド構造を意識する必要がある. 以下にバンド構造を持つ行列 A を示す.

$$A = \begin{pmatrix} a_{11} & a_{12} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & 0 & a_{n,n-1} & a_{nn} \end{pmatrix}$$

行列のバンド構造では, 対角成分の周りに零ではない成分が集中している. 対角成分を中心として, 右上にある成分を上バンド, 左下にある成分を下バンドとすると, 行列 A は, 上バンド幅 (1) ＋下バンド幅 (1) ＋対角成分の幅 (1), つまりバンドの幅が 3 となる 3 重対角行列であると分かる.

ヤコビアン, 係数行列 M の定義を行う際には, 以下に示すようにバンド部分を配列として考え, 各成分を代入しなければならない.

$$\begin{bmatrix} a_{12} & a_{23} & \cdots \\ a_{11} & a_{22} & \cdots \\ a_{21} & a_{32} & \cdots \end{bmatrix}$$

4 性能評価

本章では, 作成したパッケージを用いた DAE で表現されるシステムのシミュレーション容易性の検証と,DASSL について FORTRAN 言語と Java 言語のシミュレーション結果の比較を行う. また対象問題としては,TestSetForIVP-Solvers 内の, トランジスタ増幅器 (指数 1 の硬い DAE) 問題 [6] を扱う. 以下にトランジスタ増幅器問題の解析を行う.

トランジスタ増幅器

回路図を Fig. 3 に示す.

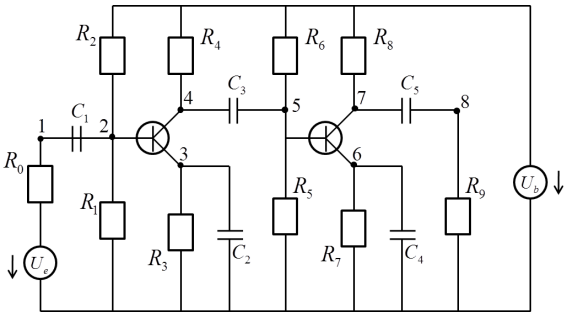

Fig. 3: Circuit diagram of Transistor Amplifier

トランジスタ増幅器回路とは, 入力電圧を与え, 内部のトランジスタを介して出力電圧を増幅させる回路である.

キルヒホッフの電流法則より各ノードにおける方程式を導出する. 得られた式を

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

の形式に変換すると, 指数 1 の DAE が得られる. 係数行列 M は,

$$M = \begin{pmatrix} -C_1 & C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_1 & -C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -C_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -C_3 & C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_3 & -C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -C_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -C_5 & C_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & C_5 & -C_5 \end{pmatrix},$$

となり, ランク 5 の 3 重対角行列を表し, 上バンド幅 (ubandmas)=1, 下バンド幅 (lbandmas)=1 となる.

関数 f は,

$$f(t,y) = \begin{pmatrix} -\frac{U_e(t)}{R_0} + \frac{y_1}{R_0} \\ -\frac{U_b}{R_2} + y_2(\frac{1}{R_1} + \frac{1}{R_2}) - (\alpha - 1)g(y_2 - y_3) \\ -g(y_2 - y_3) + \frac{y_3}{R_3} \\ -\frac{U_b}{R_4} + \frac{y_4}{R_4} + \alpha g(y_2 - y_3) \\ -\frac{U_b}{R_6} + y_5(\frac{1}{R_5} + \frac{1}{R_6}) - (\alpha - 1)g(y_5 - y_6) \\ -g(y_5 - y_6) + \frac{y_6}{R_7} \\ -\frac{U_b}{R_8} + \frac{y_7}{R_8} + \alpha g(y_5 - y_6) \\ \frac{y_8}{R_9} \end{pmatrix}$$

となる. ただし, 関数 $g(x)$,U_e を,

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

とする.

また, ヤコビアン (df/dy) のバンド構造を以下に示す. ただし, バンド構造内における, 零以外の要素が存在する成分を ∘, 零成分を • で表示する.

$$\frac{\partial f}{\partial y} = \begin{pmatrix} \circ & \bullet & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ \bullet & \circ & \circ & \ddots & & & & \vdots \\ \bullet & \circ & \circ & \bullet & \ddots & & & \vdots \\ 0 & \circ & \circ & \circ & \bullet & \ddots & & \vdots \\ \vdots & \ddots & \bullet & \bullet & \circ & \circ & \ddots & \vdots \\ \vdots & & \ddots & \bullet & \circ & \circ & \bullet & 0 \\ \vdots & & & \ddots & \circ & \circ & \circ & \bullet \\ 0 & \cdots & \cdots & \cdots & 0 & \bullet & \bullet & \circ \end{pmatrix}$$

$\frac{\partial f}{\partial y}$ は 4 重対角行列となり, 上バンド幅 (ubandjac)=1, 下バンド幅 (lbandjac)=2 となる.

$t = 0$ での y と y' の初期値を示す. また各パラメータの値を Table. 2 に示す.

$$y_0 = \begin{pmatrix} 0 \\ \frac{U_b}{(\frac{R_2}{R_1}+1)} \\ \frac{U_b}{(\frac{R_2}{R_1}+1)} \\ U_b \\ \frac{U_b}{(\frac{R_6}{R_5}+1)} \\ \frac{U_b}{(\frac{R_6}{R_5}+1)} \\ U_b \\ 0 \end{pmatrix}, y_0' = \begin{pmatrix} 51.338775 \\ 51.338775 \\ -\frac{U_b}{(\frac{R_2}{R_1}+1)(C_2 \cdot R_3)} \\ -24.9757667 \\ -24.9757667 \\ -\frac{U_b}{(\frac{R_6}{R_5}+1)(C_4 \cdot R_7)} \\ -10.00564453 \\ -10.00564453 \end{pmatrix}$$

Table 2: Parameter list

U_b=6	U_F=0.026
α=0.99	$\beta = 10^{-6}$
R_0=1000	
R_k=9000, for $k = 1, \ldots, 9$	
C_k=$k \cdot 10^{-6}$, for $k = 1, \ldots, 5$	

4.1 パッケージを用いたシミュレーション容易性の検証

本節では,3.3 節で説明した利用の手順に沿ってシミュレーションを行い, その有用性について検証する.

- DifferentialAlgebraicEquationProblem インターフェースを実装した, TransistorAmplifier クラス.
  - コンストラクタ
    - type = ”DAE ”, numeq = 8;
    - numericaljac = false; ($\frac{\partial f}{\partial y}$ を手動で計算)
    - lbandjac = 2, ubandjac = 1;
    - lbandmas = 1, ubandmas = 1;
    - index[i] = 0; for i=0,...,numeq-1
  - init y と y' を初期化
  - feval 関数 f の定義
  - jeval ヤコビアン (df/dy) の定義
  - meval 係数行列 M の定義
- メインクラスの作成
  以下にメインクラスのプログラムを示す。

```
public class TransistorAmplifierSimulation {
  public static void main(String[] args) {

    // トランジスタ増幅器クラスのインスタンス生成
    DifferentialAlgebraicEquationProblem
           prob = new TransistorAmplifier();
    double t0 = 0.0;   // 初期時刻
    double tf = 0.2;   // 終端時刻
    double dt = 0.001; // 刻み幅
    int numsamp = (int)((tf - t0) / dt);
    double[] t = new double[numsamp + 1];

    for (int i = 0; i < t.length; i++) {
      t[i] = t0 + dt * i;
    }

    // 利用するソルバーのインスタンス生成
    DifferentialAlgebraicEquationSolver
           solver = new Dassld(prob);
    // 絶対誤差の設定
    solver.setAbsoluteTolerance(1e-4);
    // 相対誤差の設定
    solver.setRelativeTolerance(1e-4);
    solver.solve(t);  // 時刻 t での計算の実行
    solver.printIS(); // 計算結果の出力
  }
}
```

以上がシミュレーションの手順である. ユーザが行う作業としては, 数学モデルの導出, ヤコビアン行列の計算,DifferentialAlgebraicEquationProblem インターフェースを実装した問題クラスの定義, 利用するソルバーのインスタンスを生成するメインクラスの定義となっている.

現在一般に ODE で表現されるシステムのシミュレーションに関する方法はよく見られるが, DAE に関して

の情報, 書籍などは稀であると考えられる. その点では, ユーザはインターフェースを 2 つ利用するだけでシミュレーションが行えるため, 開発した DAE パッケージの有用性は確認できる.

4.2 FORTRAN77 と Java の結果比較

実験環境を以下に示す。CPU に Intel Core2 Duo E8400, メモリは 4GB,OS として Mirosoft Windows 7 Professional を利用し,FORTRAN77,Java ともに 32 ビットコンパイラを用いる.

回路に入力電圧 U_e を与え, 出力電圧 U_8 が増幅されているか確認する. FORTRAN77,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 でシミュレーションを行う.

Fig. 4 に, 入力電圧 U_e,FORTRAN77,Java それぞれの電圧のグラフを示す.

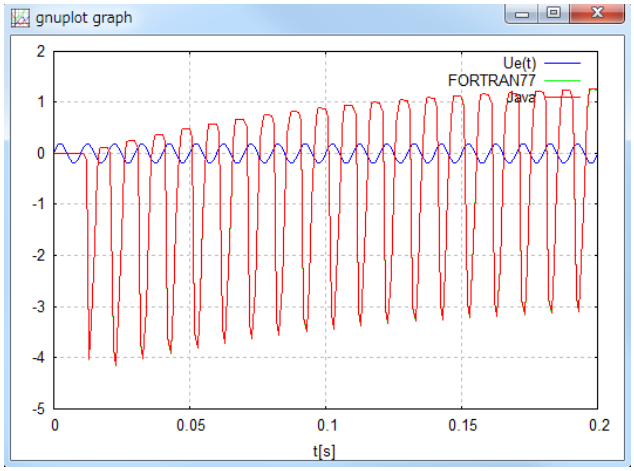

Fig. 4: input and output of FORTRAN77 and Java

Fig. 4 より, 入力電圧 U_e に対して,FORTRAN77,Java ともに電圧が増幅されているのが確認できる. また,FORTRAN77 で描画したグラフと Java のグラフが重なっており, 計算結果が妥当であると言える.

また演算速度の比較結果を Table. 3 に示す.

Table 3: operation speed of FORTRAN77 and Java

FORTRAN77	Java
0.1280[s]	0.2905[s]

結果から,FORTRAN77 の方が約 2.27 倍速いことが分かる. データ格納領域などの観点からアルゴリズム, コードが FORTRAN 仕様になっているため, 改善次第で Java においても演算速度向上が期待できる.

5 おわりに

本論文では,DAE で表現される複雑なシステムのシミュレーションを効率的に行うために, DAE 数値計算パッケージの開発を行った.

インターフェースを導入したことで, シミュレーションおけるユーザの負担を軽減し, また, それぞれの関数の役割が明確になったことで, 新たな問題やソルバーの追加において作業効率の向上に期待がもてる設計となっている.

参考文献

[1] Test set for ivp solvers. http://pitagora.dm.uniba.it/〜testset/.

[2] 藤川猛, 清水信行. 数値積分法の基礎と応用. コロナ社, 2003.

[3] 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.

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

[5] K. E. Brenan, L. R. Petzold S. l. Campbell. *Numerical Solution of Initial-Value Problems in differential-Algebraic Equations*. SIAM, 1996.

[6] parallel-IVP-algorithm group of CWI. Transistor amplifier problem. http://pitagora.dm.uniba.it/〜testset/problems/transamp.php/.