

# Solving DAEs by Taylor Series

NED NEDIALKOV

Department of Computing and Software

McMaster University

`nedialk@mcmaster.ca`

JOHN PRYCE

Computer Information Systems Engineering Dept.

Cranfield University, RMCS Shrivenham, UK

`pryce@rmcs.cranfield.ac.uk`

CMS/CAIMS Meeting

13-15 June 2004, Halifax

# The Problem

Solve numerically the **initial-value problem** for a **differential-algebraic equation** (DAE) system with

$n$  equations  $f_i$  in

$n$  dependent variables  $x_j = x_j(t)$

of the form

$$f_i(t, \text{the } x_j \text{ and derivatives of them}) = 0, \quad 1 \leq i \leq n$$

- Fully implicit
- Derivatives of order  $> 1$  are allowed

## Outline

- Theory background
  - DAEs versus ODEs
  - Pryce's structural analysis
  - Computing Taylor coefficients
- The integration process
- Theory: “nonsingularity implies validity”
- Numerical experience
- To be done

## References

J. Pryce, *A Simple Structural Analysis Method for DAEs*, BIT, 41(2), pp. 364–394, 2001

N. S. Nedialkov and J. D. Pryce, *Solving Differential-Algebraic Equations by Taylor Series (I): Computing Coefficients*, October 2003  
URL: [www.cas.mcmaster.ca/~nedialk/PAPERS](http://www.cas.mcmaster.ca/~nedialk/PAPERS)

# Theory Background

## DAEs versus ODEs

Example:

$$\begin{array}{l}
 0 = x - g(t) \\
 x' = y - h(t) \\
 y' = z - k(t)
 \end{array}
 \quad \text{versus} \quad
 \begin{array}{l}
 \epsilon z' = x - g(t) \\
 x' = y - h(t) \\
 y' = z - k(t)
 \end{array}$$

where  $g, h, k$  are given functions and  $\epsilon \neq 0$

On right: an ODE with the expected 3 degrees of freedom (DOF)

On left: a DAE with zero DOF and the unique solution

$$x = g(t)$$

$$y = g'(t) + h(t)$$

$$z = g''(t) + h'(t) + k(t)$$

## Some Observations

- An ODE solution is always **smoother** than driving functions such as  $g, h, k$ .

A DAE solution can be **less smooth** than driving functions

- **Cause and effect** can be reversed.

In DAE above, eqn 2,

$$x' = y - h(t),$$

$x'$  determines  $y$ , whereas in ODE,  $y$  determines  $x'$

- In addition to obvious algebraic constraints, DAEs typically have **“hidden” constraints**

obvious constraint

$$x = g(t)$$

hidden constraints

$$y = g'(t) + h(t) \quad \text{and} \quad z = g''(t) + h'(t) + k(t)$$

## Index

- There are various definitions of **Index** of a DAE, which all broadly measure “how different it is from an ODE”
- Informally, the index of a DAE is the minimum number of differentiations needed to reduce the DAE to an ODE
- ODEs have index 0. The DAE above has index 3

$$\boxed{\begin{array}{l} 0 = x - g(t) \\ x' = y - h(t) \\ y' = z - k(t) \end{array}}$$

↓

$$x = g(t)$$

$$y = g'(t) + h(t)$$

$$z = g''(t) + h'(t) + k(t)$$

$$x' = g'(t)$$

$$y' = g''(t) + h'(t)$$

$$z' = g'''(t) + h''(t) + k'(t)$$

A **consistent initial condition** must satisfy the constraints

- The higher the index of a DAE, the more difficult it is to solve it
- Current methods can deal with at most **index 3** DAEs, usually in the form

$$f(t, x, x') = 0$$

**We do not restrict the index**

Approach: Pryce's structural analysis + Taylor series

## Pryce's Structural Analysis (SA)

1. Form the  $n \times n$  signature matrix  $\Sigma = (\sigma_{ij})$  where

$$\sigma_{ij} = \begin{cases} \text{order of the derivative to which} \\ x_j \text{ occurs in } f_i \\ -\infty \text{ if it does not occur} \end{cases}$$

2. Find a Highest Value Transversal (HVT):  
 $n$  positions  $(i, j)$  in  $\Sigma$  with one entry in each row & column  
 such that  $\sum \sigma_{ij}$  is maximized
3. Find the smallest "offsets"  $c_i, d_j \geq 0$  satisfying

$$d_j - c_i \geq \sigma_{ij} \quad \text{for all } i, j = 1, \dots, n$$

with

$$d_j - c_i = \sigma_{ij} \quad \text{on the HVT}$$

4. Form the **System Jacobian  $\mathbf{J}$** , where

$$\mathbf{J}_{ij} = \begin{cases} \frac{\partial f_i}{\partial x_j^{(\sigma_{ij})}} & \text{if } d_j - c_i = \sigma_{ij} \\ 0 & \text{otherwise} \end{cases}$$

Informally, a **consistent initial point** is the values of an appropriate set of the  $x_j$  and derivatives of them, at a time  $t$ , that specify a unique solution

5. If there is a consistent point of the DAE at which the System Jacobian  $\mathbf{J}$  is nonsingular, then

- DAE is solvable in a neighbourhood of this point
- method shows how to reduce the DAE to an ODE system
- or alternatively **solve by Taylor series**

## Example: simple pendulum

$$0 = f_1 = x'' + x\lambda$$

$$0 = f_2 = y'' + y\lambda - g$$

$$0 = f_3 = x^2 + y^2 - L^2$$

$g > 0$  and  $L > 0$  are constants,  $\lambda$  is a Lagrange multiplier

Constraints:

$$x^2 + y^2 - L^2 = 0 \quad (\text{obvious})$$

$$xx' + yy' = 0 \quad (\text{hidden})$$

$$\Sigma = \begin{array}{c} f_1 \\ f_2 \\ f_3 \\ d_j \end{array} \begin{array}{c} x \\ y \\ \lambda \\ c_i \end{array} \begin{pmatrix} 2 & -\infty & 0 \\ -\infty & 2 & 0 \\ 0 & 0 & -\infty \\ 2 & 2 & 0 \end{pmatrix} \begin{array}{c} 0 \\ 0 \\ 2 \end{array} \quad \mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x''} & 0 & \frac{\partial f_1}{\partial \lambda} \\ 0 & \frac{\partial f_2}{\partial y''} & \frac{\partial f_2}{\partial \lambda} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & 0 \end{bmatrix}$$

HVTs: (1,1), (2,3), (3,2) and (1,3), (2,2), (3,1)

## Computing Taylor Coefficients

We want to compute the  $x_{j,r}$  (up to some order) in the series expansion

$$x_j(t) = \sum_{r \geq 0} x_{j,r} (t - t^*)^r, \quad j = 1, \dots, n$$

Substituting into  $f_i$ ,

$$f_i(t) = \sum_{r \geq 0} f_{i,r} (t - t^*)^r, \quad i = 1, \dots, n$$

We want  $f_{i,r} = 0$

Each  $f_{i,r}$  is an expression in a finite number of the  $x_{j,r}$

## Solution Process for Taylor Coefficients

Denote  $k_d = -\max_j d_j$

For each stage  $k = k_d, k_d + 1, \dots, p$

$$\begin{array}{l} \text{solve } \{ f_{i,k+c_i} = 0 : \text{ all } i \text{ s.t. } k + c_i \geq 0 \} \\ \text{for } \{ x_{j,k+d_j} : \text{ all } j \text{ s.t. } k + d_j \geq 0 \} \end{array}$$

Example:

		$k$						
		$\mathbf{c, d}$	$-2$	$-1$	$0$	$1$	$\dots$	$p$
$f_1 = x'' + x\lambda$	$\mathbf{0}$				$f_{1,0}$	$f_{1,1}$	$\dots$	$f_{1,p}$
$f_2 = y'' + y\lambda - g$	$\mathbf{0}$				$f_{2,0}$	$f_{2,1}$	$\dots$	$f_{2,p}$
$f_3 = x^2 + y^2 - L^2$	$\mathbf{2}$		$f_{3,0}$	$f_{3,1}$	$f_{3,2}$	$f_{3,3}$	$\dots$	$f_{3,p+2}$
$x$	$\mathbf{2}$		$x_0$	$x_1$	$x_2$	$x_3$	$\dots$	$x_{p+2}$
$y$	$\mathbf{2}$		$y_0$	$y_1$	$y_2$	$y_3$	$\dots$	$y_{p+2}$
$\lambda$	$\mathbf{0}$				$\lambda_0$	$\lambda_1$	$\dots$	$\lambda_p$

Given an initial guess for  $(x, x', y, y') = (x_0, x_1, y_0, y_1)$ ,  
solve

stage	eqns	vars
-2	$0 = f_{3,0} = x_0^2 + y_0^2 - L^2$	$x_0, y_0$
-1	$0 = f_{3,1} = 2x_0x_1 + 2y_0y_1$	$x_1, y_1$
0	$0 = f_{1,0} = 2x_2 + x_0\lambda_0$ $0 = f_{2,0} = 2y_2 + y_0\lambda_0 - g$ $0 = f_{3,2} = 2x_0x_2 + 2y_0y_2 + x_1^2 + y_1^2$	$x_2, y_2, \lambda_0$
$\vdots$	$\vdots$	

- Stages  $k \leq 0$  comprise the projection on the constraint manifold
- The systems are generally nonlinear for  $k \leq 0$ , and underdetermined for  $k < 0$
- Stages  $k > 0$ : the systems are linear with a diagonally scaled  $\mathbf{J}$

# The Integration Process

Implemented in the C++ code DAETC by NN

- A. Input: DAE, initial values, integration interval, tolerance(s).
- B. Initialization
  1. generate the  $\Sigma$  matrix
  2. solve a linear assignment problem to compute HVT and offsets
  3. find a consistent initial point
- C. If a consistent initial point is computed, then, on each integration step:
  1. generate Taylor coefficients
  2. compute a Taylor series solution
  3. if an error tolerance is not satisfied, reduce the stepsize and goto C2
  4. project the Taylor series solution
  5. if an error tolerance for the projection is not satisfied, half the stepsize and do C2 and C4.

## Software

DAETC — DAE solver

Designed as a set of C++ classes

≈ 5,500 lines of C/C++ code

DAETC builds on

- LAP [R. Jonker and A. Volgenant] for solving linear assignment problems (C)
- FADBAD++ [C. Bendsten and O. Stauning] for generating Taylor coefficients and Jacobians of Taylor coefficients (C++)
- IPOPT [A. Wächter] for finding a consistent initial point and projecting on each integration step (FORTRAN 77)
- LAPACK for solving linear systems (FORTRAN 77)

# Nonsingularity Implies Validity

## What is the problem?

The method computes derivative-order of variables in equations “formally” without simplification.

For example, it believes

$x$  appears to derivative-order 1 in  $xy + x' - x'$

or in

$$(xy)' - x'y$$

etc.

Computed  $\sigma_{ij}$  can be  $>$  true  $\sigma_{ij}$

Finding a true value symbolically is *undecidable* in general

What can “wrong  $\sigma_{ij}$ ” do to the DAE solution process?

At first sight, it seems that wrong  $\sigma_{ij}$  could destroy it completely

In fact, no problem because . . .

**Theorem 1** *If a signature matrix is computed as  $\tilde{\Sigma} \geq \Sigma$ , one of two alternatives must occur:*

(i)  $\text{Val}(\tilde{\Sigma}) = \text{Val}(\Sigma)$

*Then, the corresponding (canonical) offsets may be different*

*The method computes the correct Taylor coefficients, but in a possibly different sequence from that used for  $\Sigma$*

*The method computes the correct System Jacobian  $\tilde{\mathbf{J}}$  for the sequence used*

(ii)  $\text{Val}(\tilde{\Sigma}) > \text{Val}(\Sigma)$

*Then, the method computes a structurally singular System Jacobian  $\tilde{\mathbf{J}}$*

*Therefore, the method fails, and we know that it fails*

Here,  $\text{Val}(\Sigma) = \sum_{(i,j) \in T} \sigma_{ij}$ , where  $T$  is a highest-value transversal

## Example

Two independent size 2 systems: partially implicit ODE for  $v, w$  and index-2 DAE for  $x, y$

	$v$	$w$	$x$	$y$	$c_i$	possible eqns
$f_1$	$1^*$	—	—	—	0	$0 = v' - g_1(t, v)$
$f_2$	1	$1^*$	—	—	0	$0 = w' - g_2(t, v, w, v')$
$f_3$	—	—	$0^*$	—	1	$0 = x - g_3(t)$
$f_4$	—	—	1	$0^*$	0	$0 = y - g_4(t, x, x')$
$d_j$	1	1	1	0		

DOF=2, Index=2

$$\mathbf{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{\partial g_2}{\partial v'} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{\partial g_4}{\partial x'} & 1 \end{bmatrix}$$

## Variant 1

Add spurious dependence of  $f_2$  on  $v''$

$$\begin{array}{r}
 f_1 \\
 f_2 \\
 f_3 \\
 f_4 \\
 d_j
 \end{array}
 \begin{array}{ccccc}
 v & w & x & y & c_i \\
 \left( \begin{array}{cccc}
 1^* & - & - & - \\
 \mathbf{2} & 1^* & - & - \\
 - & 1 & 0^* & - \\
 - & - & 1 & 0^*
 \end{array} \right) & & & & \\
 \mathbf{2} & 1 & 1 & 0
 \end{array}
 \left| \begin{array}{l}
 \mathbf{1} \\
 0 \\
 1 \\
 0
 \end{array} \right.
 \quad
 \left. \begin{array}{l}
 0 = w' - g_2(\dots) \\
 +v'' - v''
 \end{array} \right.$$

DOF=2, Index=2

Same HVT, different offsets  $\rightarrow$  different solution sequence (for same TCs), and now  $\tilde{\mathbf{J}}_{2,1} = 0$

$$\tilde{\mathbf{J}} = \begin{bmatrix}
 1 & 0 & 0 & 0 \\
 \mathbf{0} & 1 & 0 & 0 \\
 0 & 0 & 1 & 0 \\
 0 & 0 & -\frac{\partial g_4}{\partial x'} & 1
 \end{bmatrix}$$

## Variant 2

Instead, add spurious dependence of  $f_1$  on  $x$  and  $f_3$  on  $w'$

$$\begin{array}{r}
 f_1 \\
 f_2 \\
 f_3 \\
 f_4 \\
 d_j
 \end{array}
 \begin{array}{ccccc}
 v & w & x & y & c_i \\
 \left( \begin{array}{cccc}
 1^* & - & 0^\circ & - \\
 1^\circ & 1^* & - & - \\
 - & 1^\circ & 0^* & - \\
 - & - & 1 & 0^{*\circ}
 \end{array} \right) & & & & \begin{array}{c}
 1 \\
 1 \\
 1 \\
 0
 \end{array}
 \end{array}
 \left| \begin{array}{l}
 0 = v' - g_1(t, v) + x - x \\
 \\
 0 = x - g_3(t) + w' - w' \\
 \\
 \\
 \end{array}
 \right.$$

$$\text{DOF}=2, \text{ Index}=2$$

Creates 2nd HVT of value 2 marked  $\circ$

Still gives valid solution scheme, and the System Jacobians is as for the “true DAE”

### Variant 3

Change  $f_1$  to have spurious dependence on  $y$  instead

$$\begin{array}{r}
 f_1 \\
 f_2 \\
 f_3 \\
 f_4 \\
 d_j
 \end{array}
 \begin{array}{ccccc}
 v & w & x & y & c_i \\
 \left( \begin{array}{cccc}
 1 & - & - & 0^* \\
 1^* & 1 & - & - \\
 - & 1^* & 0 & - \\
 - & - & 1^* & 0
 \end{array} \right) & 0 & 0 & 0 & 0
 \end{array}
 \left| \begin{array}{l}
 0 = v' - g_1(t, v) + y - y \\
 \\
 0 = x - g_3(t) + w' - w' \\
 \\
 \\
 \end{array} \right.$$

Creates new transversal, value 3

Now

$$3 = \text{Val}(\tilde{\Sigma}) > \text{Val}(\Sigma) = 2,$$

and the method does not give a valid solution scheme

Indeed

$$\tilde{\mathbf{J}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{\partial g_2}{\partial v'} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Structurally singular so we know method fails

# Numerical Experience

## Index-5, Two-pendula Example

An artificial problem to show we can solve high-index DAEs

$$\begin{array}{ll} 0 & = x'' + x\lambda & 0 & = u'' + u\kappa \\ 0 & = y'' + y\lambda - g & 0 & = v'' + v\kappa - g \\ 0 & = x^2 + y^2 - L^2 & 0 & = u^2 + v^2 - (L + c\lambda)^2 \end{array}$$

$g, L, c$  are constants

$x(t), y(t), u(t), v(t)$  are the state variables

$\lambda(t), \kappa(t)$  are Lagrange multipliers

## The C++ Definition

```
#define sqr(Y) ((Y)*(Y))

template <typename T>
void TwoPendula( T *f, const T *Y, const T & t )
{
    double g, L, c;
    g = 1.0;
    L = 1.0;
    c = 0.1;

    // x = Y[0], y = Y[1], lambda = Y[2]
    // u = Y[3], v = Y[4], kappa = Y[5]

    f[0] = diff(Y[0],2) + Y[0]*Y[2];
    f[1] = diff(Y[1],2) + Y[1]*Y[2] - g;
    f[2] =  sqr(Y[0])    + sqr(Y[1]) - sqr(L);

    f[3] = diff(Y[3],2) + Y[3]*Y[5];
    f[4] = diff(Y[4],2) + Y[4]*Y[5] - g;
    f[5] =  sqr(Y[3])    + sqr(Y[4]) - sqr(L+c*Y[2]);
}
```

## Solution Plots

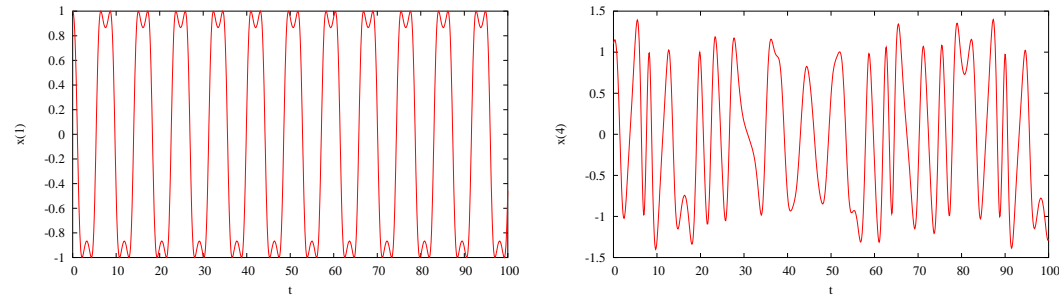


Figure 1:  $x(1) = x$ ,  $x(4) = u$  versus  $t$

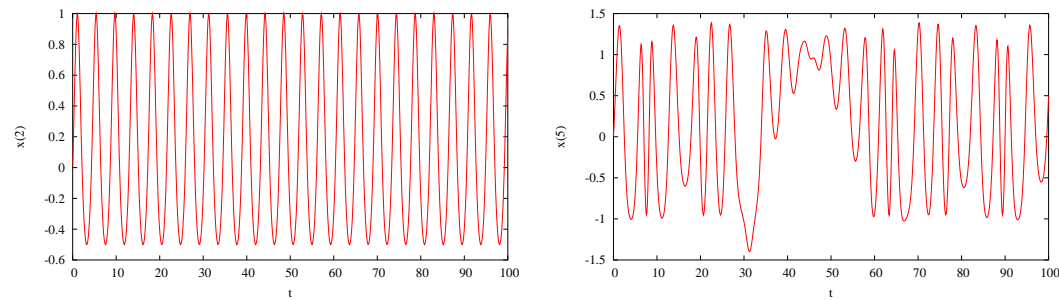


Figure 2:  $x(2) = y$ ,  $x(5) = v$  versus  $t$

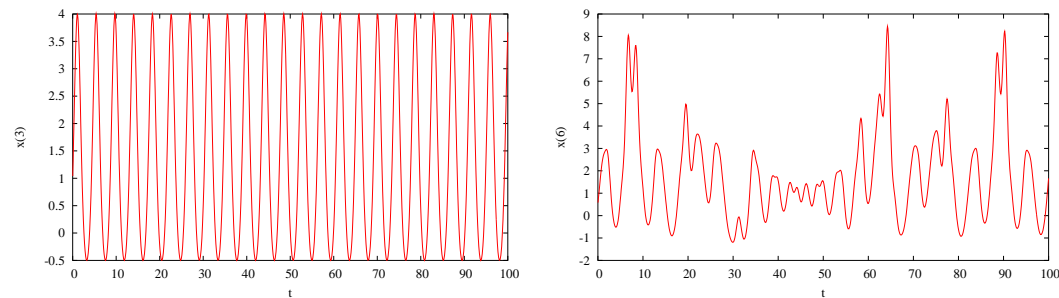


Figure 3:  $x(3) = \lambda$ ,  $x(6) = \kappa$  versus  $t$

## Output

The program first prints the result of its structural analysis.

A dash denotes  $-\infty$

# Signature Matrix

	0	1	2	3	4	5	c_j
	-----						
0	2*	-	0	-	-	-	2
1	-	2	0*	-	-	-	2
2	0	0*	-	-	-	-	4
3	-	-	-	2	-	0*	0
4	-	-	-	-	2*	0	0
5	-	-	0	0*	0	-	2
	-----						
d_j	4	4	2	2	2	0	

# Structural Index 5

# Degrees of Freedom 4

## # Solution Scheme

Stage	Functions	Variables
-4	-	$x(0, 0)$
	-	$x(1, 0)$
	$f(2, 0)$	-
	-	-
	-	-
	-	-
-3	-	$x(0, 1)$
	-	$x(1, 1)$
	$f(2, 1)$	-
	-	-
	-	-
	-	-
-2	$f(0, 0)$	$x(0, 2)$
	$f(1, 0)$	$x(1, 2)$
	$f(2, 2)$	$x(2, 0)$
	-	$x(3, 0)$
	-	$x(4, 0)$
	$f(5, 0)$	-

---

-1	f(0, 1)	x(0, 3)
	f(1, 1)	x(1, 3)
	f(2, 3)	x(2, 1)
	-	x(3, 1)
	-	x(4, 1)
	f(5, 1)	-

---

0	f(0, 2)	x(0, 4)
	f(1, 2)	x(1, 4)
	f(2, 4)	x(2, 2)
	f(3, 0)	x(3, 2)
	f(4, 0)	x(4, 2)
	f(5, 2)	x(5, 0)

---

Solution at  $t = 100$

$$\begin{aligned}
 x(0,0) &= -4.576627e-01 \\
 x(0,1) &= 1.482003e+00 \\
 x(0,2) &= 1.678422e+00 \\
 x(0,3) &= -4.387699e+00 \\
 x(0,4) &= -1.604256e+01
 \end{aligned}$$

$$x(1,0) = 8.891259e-01$$

$$x(1,1) = 7.628362e-01$$

$$x(1,2) = -2.260760e+00$$

$$x(1,3) = -4.832381e+00$$

$$x(1,4) = 1.082985e+01$$

$$x(2,0) = 3.667378e+00$$

$$x(2,1) = 2.288509e+00$$

$$x(2,2) = -6.782281e+00$$

$$x(3,0) = -1.275677e+00$$

$$x(3,1) = 2.528904e-01$$

$$x(3,2) = 2.121749e+00$$

$$x(4,0) = 4.905315e-01$$

$$x(4,1) = 1.295300e+00$$

$$x(4,2) = 1.841314e-01$$

$$x(5,0) = 1.663234e+00$$

```
CPU time (sec).....14.09
CPU time/step.....0.02
Steps.....819
  accepted.....819 (100.0%)
  rejected.....0 (0.0%)
  failed projections.....0 (0.0%)
```

## Car Axis, Index-3 DAE

Car Axis problem from the CWI Test Set of Lioen & de Swart (1999)

A simple model of a car axis going over a bumpy road.

“It is a **stiff DAE of index 3**, . . . of the form”

$$\begin{aligned} Kp'' &= f(t, p, \lambda) \\ 0 &= \phi(t, p) \end{aligned}$$

with  $p$  of dimension 4,  $\lambda$  of dimension 2

It is stated as a first-order system of 10 equations

We solve it directly, as a second-order system of 6 equations

## Solution Plots

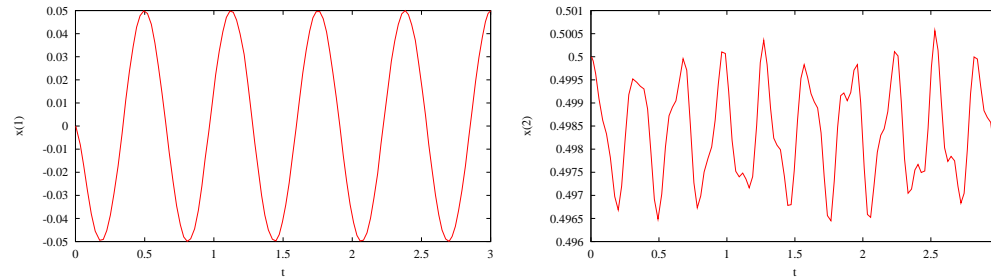


Figure 4:  $x(1) = x_l$ ,  $x(2) = y_l$  versus  $t$

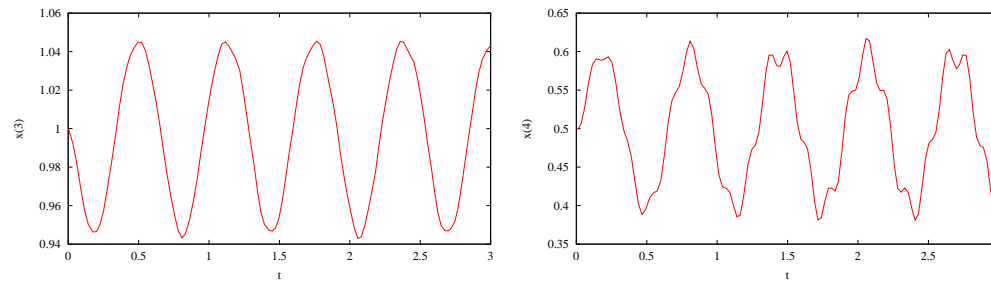


Figure 5:  $x(3) = x_r$ ,  $x(4) = y_r$  versus  $t$

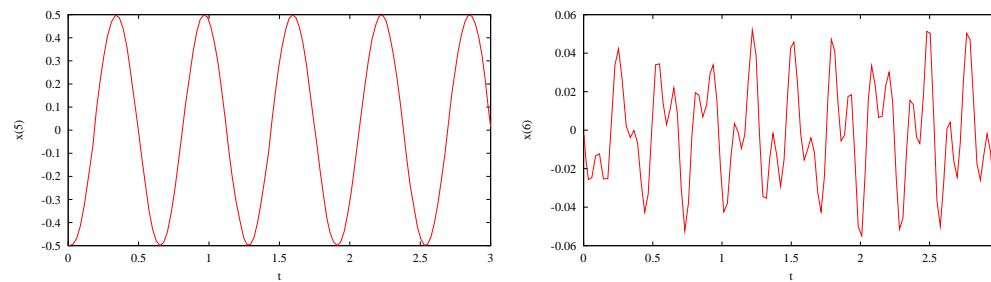


Figure 6:  $x(5) = \lambda_1$ ,  $x(6) = \lambda_2$  versus  $t$

Reference solution is computed with PSIDE on **CRAY C90**  
 using  $A_{tol} = R_{tol} = 10^{-16}$

SCD — Significant Correct Digits

$SCD := -\log_{10}(\text{max. norm of rel. error at endpoint})$

Tol	SCD	CPU (s)
$10^{-4}$	-0.28	0.30
$10^{-7}$	2.27	0.83
$10^{-10}$	4.86	2.63
$10^{-16}$	<b>7-8 ?</b>	<b>154.11</b>

## DAETC Results

PC with 1 GHz Athlon

Reference solution with  $A_{tol} = R_{tol} = 10^{-16}$

Fixed order 11, variable stepsize control

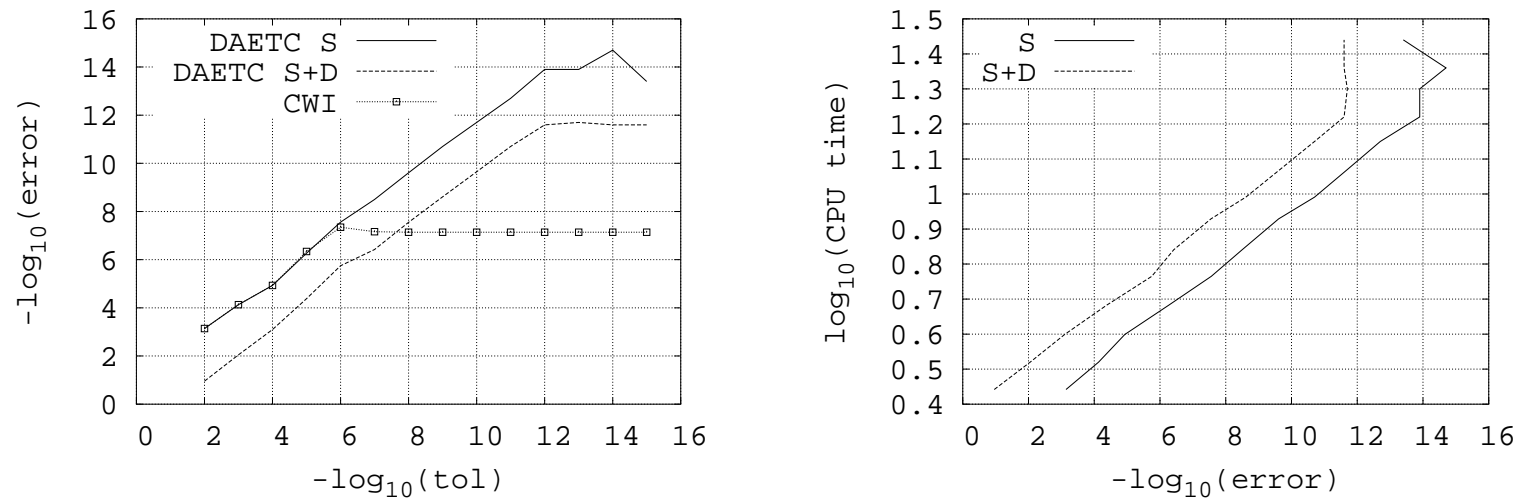


Figure 7: S: solution from DAETC; S+D: solution and derivatives from DAETC; CWI: reference solution from the CWI test set

The PSIDE ref. solution has 7.14 SCD, computed in  $\approx 154$  seconds

We achieve 8 SCD in  $\approx 7$  seconds

## Summary of Numerical Experience

We have solved over 20 problems with DAETC, and we are working through the CWI Test Set

- We can handle “uniformly” DAEs, ODEs and pure algebraic systems
- We can solve a problem directly, without converting to a first-order, lower-index form
- We expect to be competitive when
  - a problem is of too high an index for existing solvers and/or
  - high-accuracy is required
- The present, explicit method is not efficient on very stiff problems

## The Ring Modulator Problem

A model of a circuit from the CWI Test Set. Size  $n = 15$

The type of problem depends on a parameter  $C_s$ :

$C_s \neq 0$                     ODE

$C_s = 0$     index-2 DAE

Very stiff and highly oscillatory

In the original formulation, the computation results in a structurally singular System Jacobian, and the method fails

Replacing eqn 5 by (eqn 4) + (eqn 5) results in a nonsingular System Jacobian, and the method succeeds

*Given a DAE system for which SA fails, can we transform the DAE automatically (symbolically) such that SA succeeds?*

# To Be Done

## Strategic

### Interplay between structural analysis & methodology

Large systems are mostly generated automatically by software. Various methodologies exist, e.g. for applying Kirchoff's laws to circuits

If SA "works" ( $\mathbf{J}$  nonsingular) then we can handle high index etc.

Important to know what methodologies are "SA friendly"

## Nice Theory

We are developing an Hermite–Obreschkoff method: needed to handle stiff problems

## Technical

g-stop (event location) facility, variable order, ...