The DAETS Differential-Algebraic Equation Solver

John Pryce[1] Ned Nedialkov[2]

[1]Dept of Information Systems, Cranfield U., UK
j.d.pryce@ntlworld.com

[2]Dept of Computing and Software, McMaster U., Canada
nedialk@mcmaster.ca

University of Louisiana at Lafayette, 8 May 2008
Taylor Model Workshop, Fields Institute, 22 May 2008

Contents

DAEs—what and why?

- (Explicit) ordinary differential equations (ODEs) specify derivative of a vector of state variables in terms of those variables, $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$
- Differential algebraic equation (DAE) system mixes purely algebraic equations with those about derivatives
- Any modeling of complex systems may give a DAE — eliminating algebraic eqns may be unnatural/expensive
- Ubiquitous in mechanical systems, control, chemical engineering, electrical circuit modeling, . . .

“DAEs are not ODEs” (Petzold)

- Compare Initial Value Problems (IVPs) for:

$$\begin{aligned} 0 &= x - g(t) \qquad & \epsilon\, z' &= x - g(t) \qquad & (g(t)\text{ given}) \\ x' &= y & x' &= y & \\ y' &= z & y' &= z & \end{aligned}$$

- For any $\epsilon \neq 0$, system on right is an ODE, with 3 degrees of freedom (DOF)—needs 3 IVs for unique solution
- System on left has zero DOF—unique solution

$$x = g(t), \qquad y = g'(t), \qquad z = g''(t)$$

- Funny features of DAE
  - Cause-effect reversal
  - Solution can be less smooth than driving function, instead of smoother

DAETS is a new kind of DAE solver

- Excellent at high-index DAEs
- Excellent for getting high accuracy
- Returns useful data about structure of problem
- Doesn't compete on speed at moderate accuracies
- . . . or on handling very large problems
- Infrastructure: FADBAD++, IPOPT

DAETS solves DAEs by Taylor series expansion

- DAETS (Differential Algebraic Equations by Taylor Series) solves DAE initial value problems, for state variables $x_j(t),\ j = 1, \dots, n$, of the general form

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

- Can be fully implicit
- $\mathrm{d}/\mathrm{d}t$ can appear anywhere in the expressions for f_i
- e.g. one of the equations could be

$$\frac{\big((x_1' \sin t)'\big)^2}{1 + (x_2')^2} + t^2 \cos x_2 = 0$$

DAETS solves high index problems

- Index ν measures how “hard” DAE is to solve
- For traditional methods, $\nu \geq 3$ considered hard
- DAETS based on structural analysis of DAE + automatic differentiation, so in principle unaffected by index
- Have solved artificial problems up to $\nu = 47$
  (Any physical sense? . . . is another matter)

Numerical method summary

- Start with code for the f_i that define the DAE
- Use AD (FADBAD++ package) to evaluate suitable derivatives $f_i^{(r)} = \dfrac{d^r f_i}{dt^r}$ at given $t = t_r$
- For each step:
- Equate these to zero "in batches" to get Taylor coefficients of (vector) solution $\mathbf{x}(t)$ at current $(t_{r-1}, \mathbf{x}_{r-1})$
- Sum Taylor series to get approximation $\mathbf{x}_r$ at $t_r = t_{r-1} + h$
- Project this $\mathbf{x}_r$ on DAE's constraints to get a consistent $\mathbf{x}_r$
- Repeat, to step along range in usual way

Numerical method, cont

- Before all this, do Structural Analysis:
  preprocess the DAE code to find the $2n$ integer offsets,
  one for each variable, one for each equation
- These prescribe the "batches" in the overall process of solving for TCs
- They imply the Initial Values data is not a flat vector unlike with most DAE solvers
- Following example illustrates

The notorious simple pendulum

Index 3 system with equations

$$
\begin{aligned}
0 &= f = x'' + x\lambda \\
0 &= g = y'' + y\lambda - G & G &= \text{gravity} \\
0 &= h = x^2 + y^2 - L^2 & L &= \text{length of pendulum.}
\end{aligned}
$$

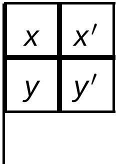
State variables $x(t)$, $y(t)$, $\lambda(t)$

Item	x	y	λ	f	g	h
Offset	2	2	0	0	0	2

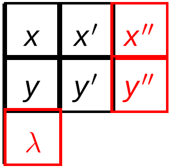
User needs offsets to understand IVs

- Offsets tell what initial values should be provided
- Offsets $\begin{array}{ccc} x & y & \lambda \\ 2 & 2 & 0 \end{array}$

  mean that IVs comprise values for $x, x'; \; y, y'$



- Except when DAETS sees DAE is non-linear in leading derivatives (here x'', y'', λ) it requires an extra set of derivatives
  E.g. if first equation were $0 = (x'')^3 + x\lambda$ then IVs must comprise $x, x', x''; \; y, y', y''; \; \lambda$



- Reason: to assure local uniqueness of solution

User needs offsets to understand constraints

Offsets tell what constraints the provided IVs must meet for consistency

Offsets

f	g	h
0	0	2

mean they must satisfy

in the linear case,
$h, h' = 0$:

$$0 = h = x^2 + y^2 - L^2$$
$$0 = h' = 2xx' + 2yy'$$

in the non-linear case
$f;\ g;\ h, h', h'' = 0$,
so in our example, add these:

$$0 = f = (x'')^3 + x\lambda$$
$$0 = g = y'' + y\lambda - G$$
$$0 = h'' = 2(xx'' + (x')^2 + yy'' + (y')^2)$$

Finding consistent point

- Solution $\mathbf{x}(t)$ must satisfy algebraic constraints for all t to be consistent
- Constraint can be obvious, as (for Pendulum) $h = 0$, or hidden, as $h' = 0$
- Finding initial consistent point can be hardest part of solving DAE
- Not built in to most solvers.
  Often, user has only a poor guess of required values
- But fits naturally into DAETS workflow.
  We formulate it as a nonlinear minimisation problem and give it to IPOPT package

Numerical results

- Accuracy comparisons on a standard test problem
- DAETS on a High-index problem
- Efficiency comparisons
- DAETS on a Continuation problem

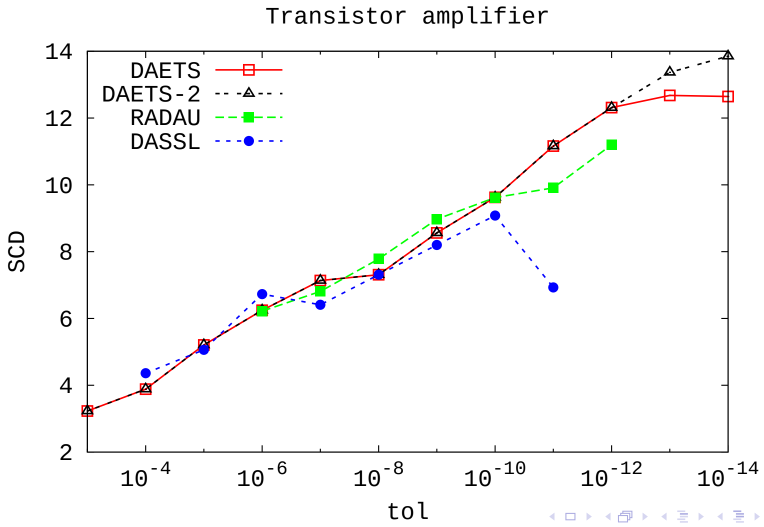
Plots of accuracy vs. tolerance

- Problem is Transistor Amplifier from *Test Set for Initial Value Problem Solvers*, Bari University, Italy
- Index 1 DAE of size $n = 8$
- "DAETS", "RADAU", "DASSL" curves compare with reference solution (at end of range) in Test Set documentation
- "DAETS-2" curve uses reference solution computed by DAETS with tol$= 10^{-16}$
- We plot "Significant Correct Digits" SCD

  $$= -\log_{10} \|\text{componentwise relative error at end of integration}\|$$

  as a function of tolerance

Plots of accuracy vs. tolerance



Comments on this experiment

- Even though only index-1, this problem is too much for DASSL at tolerances below 10^{-10}
- RADAU gets another 2 orders of accuracy, and DAETS probably another 3 orders beyond that
- Difference between DAETS and DAETS-2 curves shows DAETS's "reference solution" is better than Test Set's one (computed by PSIDE solver on Cray C90 in double precision, machine epsilon = 0.25e-28)

“Multi-pendula” — a class of high-index problems

- System is a “chain” of P simple pendula with coupling
- Pendulum 1 is as normal
- Tension in pendulum $(p-1)$ has a small effect on length of pendulum p, for $p = 2, \ldots, P$
- For $P = 2$

$$\begin{aligned} 0 &= x_1'' + \lambda_1 x_1 \\ 0 &= y_1'' + \lambda_1 y_1 - G \\ 0 &= x_1^2 + y_1^2 - L^2 \end{aligned} \quad \text{and} \quad \begin{aligned} 0 &= x_2'' + \lambda_2 x_2 \\ 0 &= y_2'' + \lambda_2 y_2 - G \\ 0 &= x_2^2 + y_2^2 - (L + c\lambda_1)^2 \end{aligned}$$

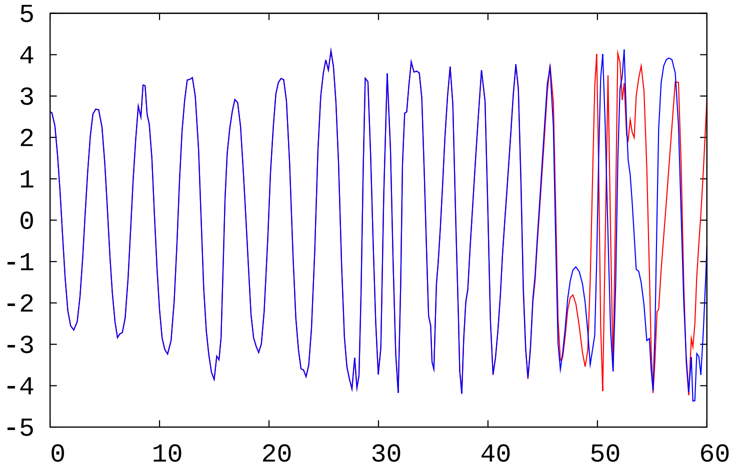
where c is a constant. (λ is essentially tension.)

Multi-pendula are high-index and chaotic

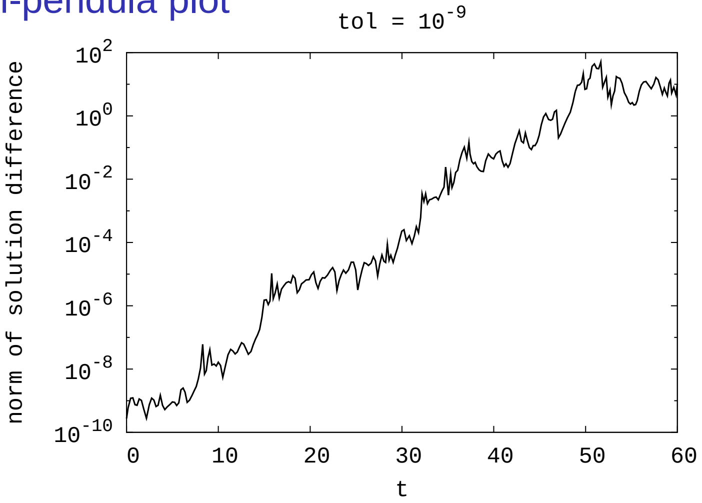
- Chain of length P has size $n = 3P$ and index $2P + 1$
- Not surprisingly shows chaotic behaviour for all $P \geq 2$
- DAETS has solved system for P up to 23 giving index 47.
- Here are sample solutions for $x_7(t)$ ($P = 7$, index 15) with two slightly differing sets of IVs

Multi-pendula plot

Seven pendula, tol = 10^{-9}, x_7 with two slightly differing ICs



Multi-pendula plot



Exponential divergence of nearby solutions suggests chaos

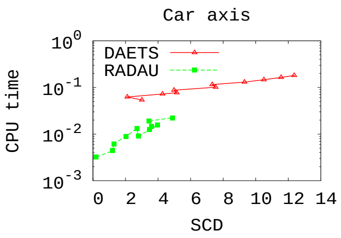
Efficiency experiments

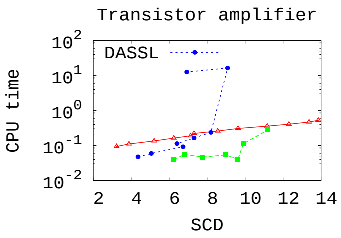
- For problems from ODE/DAE Test Set, plot CPU time vs. Significant Correct Digits SCD (defined above)
- Problems are

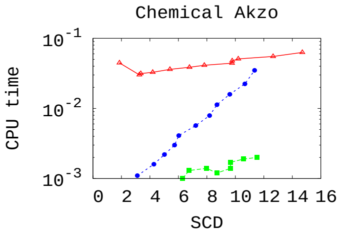
Car axis:	index-3 DAE, $n = 10$;
Transistor amplifier:	index-1 DAE, $n = 8$;
Chemical Akzo Nobel:	index-1 DAE, $n = 6$;
HIRES:	ODE, $n = 8$.

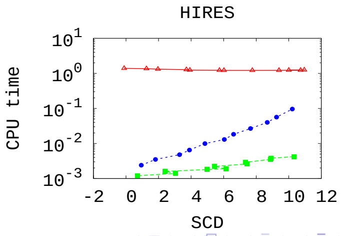
- Compare with DASSL and RADAU solvers

Efficiency: Work-Precision diagrams









Comments on work-precision data

These are work-precision diagrams as described in ODE/DAE Test Set for DAETS, DASSL, and RADAU on four problems.

- DASSL, RADAU much faster for low to medium precision
- Car axis (high index): DAETS keeps going up to 13 correct digits while DASSL & RADAU can only give about 5. Power of AD!
- HIRES: Weird behaviour for DAETS. Much more expensive than the others, BUT tighter tolerance means less work. Why?

Continuation problems

- No need for derivatives actually to be present — can solve n purely algebraic equations

$$\mathbf{f}(\lambda, \mathbf{x}) = \mathbf{0}$$

  to find $\mathbf{x} = (x_1, \ldots, x_n)$ as a function of λ
- To handle turning points, best use arc-length continuation. Treat λ and the x_i as all on same footing, define Euclidean arc-length s by

$$(\mathrm{d}\lambda/\mathrm{d}s)^2 + (\mathrm{d}x_1/\mathrm{d}s)^2 + \ldots + (\mathrm{d}x_n/\mathrm{d}s)^2 = 1,$$

  and find $(\lambda, \mathbf{x})$ as function of s
- Gives an index 1 DAE of size $n + 1$

Continuation is difficult

- Difficulty in typical applications is path tracking failure.
- Illustrate with problem from Layne Watson (1979).
  Solve $\mathbf{g}(\mathbf{x}) = \mathbf{x}$ (find fixed point) for $\mathbf{g} = (g_1, \ldots, g_n)$ where

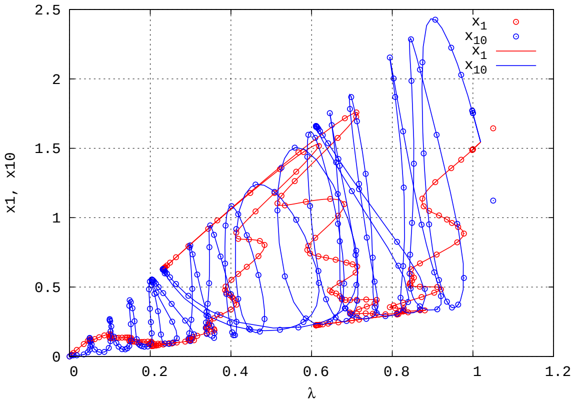
$$g_i(\mathbf{x}) = g_i(x_1, \ldots, x_n) = \exp(\cos(i \sum_{k=1}^{n} x_k)), \quad i = 1, \ldots, n.$$

- Many solutions. Hard for even n around 10.
- Formulate as

$$\mathbf{0} = \mathbf{f}(\lambda, \mathbf{x}) = \mathbf{x} - \lambda \mathbf{g}(\mathbf{x})$$

  and "continue" from $\lambda = 0$ (trivial solution)
  to $\lambda = 1$ (what we want to solve) using arc-length.

Two components of Layne Watson curve for $n = 10$



- Lots of turning points!
- Tracking failure is serious problem if step size h unlimited. Restricting $h \leq 0.3$ cured it.

How to improve the structural analysis

Chemical Akzo Nobel problem is an index-1 DAE with 6 variables and equations
Here is the solution scheme DAETS reports at present

```
>> showstruct(spsigmx('chemakzo'));
Stage -1:
  Solve nothing
     after giving IVs for x1 x2 x3 x4 x5
Stage 0:
  Solve f1 f2 f3 f4 f5 f6
     after giving IVs for x1' x2' x3' x4' x5' x6
```

...and so on, giving a complete scheme for generating Taylor coefficients

Improving the structural analysis

Dulmage Mendelsohn re-orders a matrix to block triangular
It can drastically reduce the size of the equations to solve:

```
>> showstruct(spsigmx('chemakzo'));
Stage -1:
  Solve nothing   after giving IVs for x1 x2 x3 x4 x5
Stage 0:
  Solve f6        automatically for x6
  Solve f5        after giving IVs for x5'
  Solve f4        automatically for x4'
  Solve f3        automatically for x3'
  Solve f2        after giving IVs for x2'
  Solve f1        automatically for x1'
```

Result: one nonlinear system of size 6×6 has become

- $4\times$linear systems of size 1×1, and
- $2\times$nonlinear systems of size 1×1

The same data in graphic form

Indices of variables

Indices of equations

	6	5	4	3	2	1	$c_i=$
6	0		0			0	0
5	0	1	0	0	0	0	0
4		0	1	0		0	0
3		0	0	1	0	0	0
2	0		0		1	0	0
1		0	0	0	0	1	0
$d_j=$	0	1	1	1	1	1	DOF=5 Index=1

Summary

- We now have a robust DAE code based on Structural Analysis theory that I began, with George Corliss, in 1996
- Excellent for high index and high accuracy
  and giving you information about DAE structure
- Current state:
  - Paper "Solving DAEs by Taylor Series III: the DAETS code" accepted by JNAIAM Jan 2008
  - User Guide: final touches May 2008
  - Distribution: Free demo version, $\leq$ 8 variables: from Ned.
    Commercial version: from Canada's Flintbox innovation portal. Licence levels \$199, \$399, \$599
    Binary library plus `C++` header files.
- Thanks to Ned Nedialkov as main software architect
  — he has lots yet to do

Example: simple pendulum—code for function

```
#include "DAEsolver.h"

template <typename T>
void fcn(int n,  T t, const T *z, T *f, void *p) {

   const double g = 9.8, L = 10.0;
// z[0], z[1], z[2] are x, y, lambda.

   f[0] = Diff(z[0],2) + z[0]*z[2];
   f[1] = Diff(z[1],2) + z[1]*z[2] - g;
   f[2] = sqr (z[0])   + sqr(z[1]) - sqr(L);
}
```

Example: simple pendulum—main program

```
int main() {
  const int n = 3;                      // size of the problem
  DAEsolver Solver(n, DAE_FCN(fcn)); // create a solver + analyse DAE
  Solver.printDAEinfo();                // print info about the DAE
  Solver.printDAEpointStructure();   // .. and more info
  DAEsolution x(Solver);                // create a DAE solution object
  x.setT(0.0);                          // initial value of t
  x.setX(0, 0,-1.0).setX(0, 1, 0.0); // .. and of x and x'
  x.setX(1, 0, 0.0).setX(1, 1, 1.0); // .. and of y and y'
  double tend = 100.0;
  DAEexitflag flag;
  Solver.integrate(x, tend, flag);   // integrate until tend
  if (flag!=success)
    printDAEexitflag(flag);             // check the exit flag
  x.printSolution();                    // print solution
  x.printStats();                       // print integration statistics
  return 0;
}
```

Pendulum output

```
johnpryce> pendulum
SIGNATURE MATRIX & OFFSETS:
      0   1   2  |c_i
   |-----------------
  0|  2   -   0* | 0
  1|  -   2*  0  | 0
  2|  0*  0   -  | 2
   |-----------------
d_j|  2   2   0

          DAE
             size..................3
             index ................3
             LINEAR
Initial values must be given for:
    variable    derivatives
        0       0       1
        1       0       1
        -
```

Pendulum output

```
t = 1.000000e+02
        x                x'               x''
--------------------------------------------------
0    8.037130e+00    6.453216e+00   -1.414013e+01
1    5.950171e+00   -8.716614e+00   -6.684351e-01
2    1.759350e+00
```

```
CPU TIME (sec)...........0.0527
NO STEPS.................388
   accepted..............388
   rejected..............0    * 0.00%
STEPSIZES
   smallest..............0.02
   largest ..............0.35
ORDER OF TAYLOR SERIES...15
TOLERANCE
   relative..............1.0e-12
   absolute..............1.0e-12
```