Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Test set for IVP solvers

W.M. Lioen, J.J.B. de Swart and W.A. van der Veen

Department of Numerical Mathematics

NM-R9615 November 30, 1996

Report NM-R9615
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Test Set for IVP Solvers

W. M. Lioen, J. J. B. de Swart & W. A. van der Veen

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

version August 15, 1996

Abstract

In this paper a collection of Initial Value test Problems for systems of Ordinary Differential Equations, Implicit Differential Equations and Differential-Algebraic Equations is presented. This test set is maintained by the project group for Parallel IVP Solvers of CWI, department of Numerical Mathematics. This group invites everyone to contribute new test problems to this test set. How new problems can be submitted can be found in this paper as well.

AMS Subject Classification (1991): Primary: 65Y20, Secondary: 65-04, 65C20, 65L05
CR Subject Classification (1991): G.1.7, G.4
Keywords & Phrases: test problems, software, IVP, IDE, ODE, DAE
Note: The idea to develop this test set was discussed at the workshop ODE to NODE, held in Geiranger, Norway, 19–22 June 1995.
Acknowlegdements: The maintenance of this test set is supported financially by STW (Dutch Foundation for Technical Sciences). Some problems presented here were contributed by M. Günther, B. Simeon (TH Darmstadt) and G. Denk (TU München). This work was sponsored by the Stichting Nationale Computerfaciliteiten (National Computing Facilities Foundation, NCF) for the use of supercomputer facilities, with financial support from the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (Netherlands Organization for Scientific Research, NWO).

Test problems collected so far:

1. **Chemical Akzo Nobel problem** (ODE of dimension 6)
2. **Problem HIRES** (ODE of dimension 8)
3. **Pollution Problem** (ODE of dimension 20)
4. **Ring Modulator** (ODE of dimension 15)
5. **Andrew's squeezing mechanism** (index 3 DAE of dimension 27)
6. **Transistor Amplifier** (index 1 DAE of dimension 8)
7. **Medical Akzo Nobel problem** (ODE of dimension 400)
8. **EMEP problem** (ODE of dimension 66)
9. **NAND gate** (index 0 IDE of dimension 14)
10. **Charge pump** (index 2 DAE of dimension 9)
11. **Wheelset** (index 2 IDE of dimension 17)
12. **Two bit adding unit** (index 1 DAE of dimension 350)
13. **Car Axis problem** (index 3 DAE of dimension 10)
14. **Fekete problem** (index 2 DAE of dimension 160)

I Introduction

In testing new codes for the numerical solution of Initial Value Problems, it would save a lot of time if one could give a generally accepted reference for test problems and use the same Fortran 77 codes of the test problems, instead of describing and programming the problems oneself. Moreover, if everyone would use the same source of the test problem, i.e. the same formulation, parameters, integration interval, initial values, way of programming, etc., the comparison between the results of several authors becomes much easier. This test set tries to fulfill these demands.

This set is meant to be a supplement to existing test sets, like NSDTST and STDTST by Enright & Pryce [EP87], and PADETEST by Bellen [Bel92]. Information on the numerical solution is presented by solving the problems with some well-known codes. Some problems were taken directly from industry, others from the literature. Especially, the standard work by Hairer & Wanner [HW91], in which a lot of problems arising in practice are brought together, turned out to be very useful. The cooperation with M. Günther, B. Simeon (TH Darmstadt) and G. Denk (TU München) has led to a few contributions to the test set as well.

The test set can be obtained in two ways:

1. via the WWW page with URL

   `http://www.cwi.nl/cwi/projects/IVPtestset.shtml` ,

2. via anonymous ftp at the site

   `ftp.cwi.nl` in the directory `pub/IVPtestset` .

Every test problem consists of a description (c.f. Section II) and a Fortran code (c.f. Section III). Both are obtainable via WWW or in the ftp directory mentioned above.

Drivers that make these codes suitable for runs with the codes RADAU5 by Hairer & Wanner [HW95], VODE by Brown, Hindmarsh and Byrne [BHB92] and DASSL by Petzold [Pet91] are available as well.

Section II gives information on the structure of the problem descriptions. It is followed by Section III on the format of the Fortran subroutines, here we also explain how to use the Fortran codes for running the test problems.

I.1 How to submit new test problems

In order to let this test set be a success, it is necessary that a lot of new test problems are contributed. On the other hand, to restrict the amount of time for the maintainers of the test set to incorporate new problems, it is important that the submissions are in a prescribed format. Firstly, every problem should have a PostScript file (preferably together with a LaTeX-file) with a description of the problem containing the 4 subsections mentioned in Section II. Secondly, a set of Fortran routines that are necessary for implementation has to be supplied in the format specified in Section III.

Submissions can be sent by e-mail to `IVPtestset@cwi.nl`.

I.2 People Involved

This test set is maintained by the project group for Parallel IVP Solvers of CWI, department of Numerical Mathematics. Members of this group are:

1.	P.J. van der Houwen	`senna@cwi.nl`
2.	W. Hoffmann[1]	`walter@fwi.uva.nl`
3.	B.P. Sommeijer	`bsom@cwi.nl`
4.	W.M. Lioen	`walter@cwi.nl`
5.	W.A. van der Veen	`wolter@cwi.nl`
6.	J.J.B. de Swart	`jacques@cwi.nl`

[1] University of Amsterdam

II Problem descriptions

Every problem description contains the following 4 subsections:

1. **General information**
   The problem identification is given. Is it an IDE, ODE or DAE, what is its dimension, its index? The contributor and any further relevant information are listed too. What is meant here by IDE, ODE, DAE and index, is explained in Section III.

2. **Mathematical description of the problem**
   All ingredients that are necessary for implementation are given in mathematical formulas.

3. **Origin of the problem**
   A brief description of the origin, in order to give a physical interpretation of the problem. References to the literature are given for further details.

4. **Numerical solution of the problem**
   This subsection consists of 4 subsubsections.

   (a) **Solution in the endpoint.** The values of the solution components in the endpoint are listed.

   (b) **Behaviour of the numerical solution.** This subsubsection presents plots of (some of) the components over (part of) the integration interval.

   (c) **Run characteristics.** Integration statistics of runs with DASSL, RADAU5 and VODE (if applicable) serve to give insight in the numerical difficulty of the problem. Specifications of the computer and Fortran 77 compiler used to perform the run, are included. The characteristics are in the following format:

   - *solver*
     The name of the numerical solver with which the run was performed.
   - *rtol*
     The user supplied relative error tolerance.
   - *atol*
     The user supplied absolute error tolerance.
   - *h0*
     The user supplied initial step size (if relevant).
   - *scd*
     The scd values denote the minimum number of significant correct digits in the numerical solution in the endpoint, i.e.

     $$\text{scd} := -\log_{10}(\text{max. norm of the relative error in the endpoint}).$$

     If some components of the solution vector are not taken into account for the computation of the scd value, or if the absolute error is computed instead of the relative error, then this is specified locally.
   - *steps*
     Total number of steps taken by the solver (including rejected steps due to error test failures and/or convergence test failures).
   - *accept*
     The number of accepted steps.
   - *# f* and *# Jac*
     The number of evaluations of the derivative function and its Jacobian, respectively.
   - *# LU*
     The number of LU-decompositions (if delivered by the code). The codes, except for RADAU5, count the LU-decompositions of systems of dimension d, where d

is the dimension of the test problem. RADAU5 uses the three-stage Radau IIA method. Every iteration of the inexact Newton process, used for solving systems of non-linear equations, requires the solution of a linear systems of dimension $3d$. This linear system is, by means of transformations, reduced to 2 linear systems of dimension d, one of which is complex. The decompositions of these 2 linear systems are counted by RADAU5 as 1 LU-decomposition.

- *CPU*
  The CPU time in seconds to perform the run on the specified computer. Since timings on most computers may depend on other processes (like e.g. daemons), the minimum of the CPU times of 100 runs is listed.

Apart from tests with publicly available, well-known codes, tests with PSODE are included. PSODE is a code developed at CWI for the parallel solution of ODEs. PSODE uses the 4-stage Radau IIA method and solves the 4 stages in parallel. The integration statistics listed in this Test Set refer to the implementation on a sequential computer. We included the speed-up factors obtained with PSODE on the Cray C98/4256 at SARA, using the tool ATExpert. Since PSODE is still under development, it is not yet publicly available. For more details on PSODE and its parallel implementation, we refer to [SB95] and [HS91].

(d) **Work-precision diagram.** For every relevant solver, a range of input tolerances and, if necessary, a range of initial stepsizes, were used to produce a plot of the resulting scd values against the number of CPU seconds needed for the run. The format of these diagrams is as in Hairer & Wanner [HW91, pp. 166–167, 324–325]. The range of tolerances and initial stepsizes is problem dependent and specified locally.

We want to emphasize that the reader should be careful with using these diagrams for a mutual comparison of the solvers. The diagrams just show the result of runs with the prescribed input on the specified computer. A more sophisticated setting of the input parameters, another computer or compiler, as well as another range of tolerances might change the diagrams considerably.

III Fortran codes for the problems

For every test problem, the file `problem.f` contains a set of Fortran subroutines defining the problem. We have categorized the test problems in three classes: IDEs, ODEs and DAEs.

In this test set, we call a problem an **IDE** (system of Implicit Differential Equations) if it is of the form

$$\begin{aligned} &G(t, y, y') = 0, \qquad t_{\text{begin}} \le t \le t_{\text{end}}, \\ &y, G(t, y, y') \in \mathbf{R}^d, \\ &y(t_{\text{begin}}) \text{ and } y'(t_{\text{begin}}) \text{ are given.} \end{aligned}$$

A problem is named an **ODE** (system of Ordinary Differential Equations), if it has the form

$$\begin{aligned} &y' = f(t, y), \qquad t_{\text{begin}} \le t \le t_{\text{end}}, \\ &y, f(t, y) \in \mathbf{R}^d, \\ &y(t_{\text{begin}}) \text{ is given,} \end{aligned}$$

whereas the label **DAE** is given to problems which can be cast in the form

$$\begin{aligned} &My' = f(t, y), \qquad t_{\text{begin}} \le t \le t_{\text{end}}, \\ &y, f(t, y) \in \mathbf{R}^d, \qquad M \in \mathbf{R}^{d \times d} \\ &y(t_{\text{begin}}) \text{ is given,} \end{aligned}$$

where M is a constant, possibly singular matrix. Connected to IDEs and DAEs is the concept of index. Here, we mean by the index of a problem the differential index as defined in [HW91]. Note that ODEs and DAEs are subclasses of IDEs.

Every class of problems corresponds to one format of the Fortran routines.

III.1 IDEs

The form for IDEs reads

$$\begin{aligned} &G(t, y, y') = 0, \qquad t_{\text{begin}} \le t \le t_{\text{end}}, \\ &y, G(t, y, y') \in \mathbf{R}^d, \\ &y(t_{\text{begin}}) \text{ and } y'(t_{\text{begin}}) \text{ are given.} \end{aligned}$$

The subroutines are:

1. ```
   subroutine prob(problm, neqn, tbegin, tend, ijac, mljac, mujac,
                   ind1, ind2, ind3)
   character*(*) problm
   integer neqn, ijac, mljac, mujac, ind1, ind2, ind3
   double precision tbegin, tend
   ```
   describes the problem

   `problm - character`
   : On exit, `problm` contains a character string uniquely identifying the problem. The first 8 characters (possibly blank padded) should be unique.

   `neqn - integer`
   : On exit, `neqn` contains d, the dimension of the problem.

   `tbegin - double precision`
   : On exit, `tbegin` contains t_{begin}, the begin point of the integration interval.

   `tend - double precision`
   : On exit, `tend` contains t_{end}, the end point of the integration interval.
   ```

`ijac - integer`
On exit, `ijac` contains a switch for the computation of the Jacobians $\partial G/\partial y$ and $\partial G/\partial y'$:

- `ijac` $= 0$: the Jacobians have to be computed internally by the solver, a dummy subroutine `jeval` is supplied;
- `ijac` $= 1$: the Jacobians are supplied by the subroutine `jeval`, defined below.

`mljac - integer`
On exit, `mljac` contains a switch for the structure of the Jacobians $\partial G/\partial y$ and $\partial G/\partial y'$:

- `mljac` $=$ `neqn`: the Jacobians are full matrices;
- $0 \leq$ `mljac` $<$ `neqn`: the Jacobians are band matrices, `mljac` is the lower bandwidth of the Jacobian matrices (`mljac` $\geq$ number of non-zero diagonals below the main diagonal).

`mujac - integer`
On exit, `mujac` contains the upper bandwidth of the Jacobian matrices $\partial G/\partial y$ and $\partial G/\partial y'$ (`mujac` $\geq$ number of non-zero diagonals above the main diagonal).
Need not be defined if `mljac` $=$ `neqn`.

The parameters `ind1`, `ind2` and `ind3` give information on the index of the variables. The right hand side function subroutine `feval` is written such that the index 1, 2, 3 variables appear in this order. The relation `ind1` $+$ `ind2` $+$ `ind3` $=$ `neqn` should hold.

`ind1 - integer`
On exit, `ind1` contains the number variables with index lower than 2 (`ind1` > 0 should hold).
For systems of index lower than 2 this equals `neqn`.

`ind2 - integer`
On exit, `ind2` contains the number of index 2 variables.
For systems of index lower than 2 this equals 0.

`ind3 - integer`
On exit, `ind3` contains the number of index 3 variables.
For systems of index lower than 2 this equals 0.

For the definition of index of a variable, we refer to [BCP89]. We remark that the differential index of the whole problem equals the maximum of the indices of all variables.

2. `subroutine init(neqn, y, dy, incon)`
`integer neqn, incon`
`double precision y(neqn), dy(neqn)`

returns the (possibly inconsistent) initial values $y(t_{\text{begin}})$ and $y'(t_{\text{begin}})$

`neqn - integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`y - double precision` array of `dimension` at least `(neqn)`
On exit, `y(i)` contains $y_i(t_{\text{begin}})$, $i = 1, \ldots, d$, the initial values of the solution.

`dy - double precision` array of `dimension` at least `(neqn)`
On exit,

- `dy(i)` contains $y_i'(t_{\text{begin}})$, $i = 1, \ldots, d$, consistent initial values of the derivative of the solution, if `incon` $= 1$.
- `dy(i)` contains 0 or an approximation to $y_i'(t_{\text{begin}})$, $i = 1, \ldots, d$, inconsistent initial values of the derivative of the solution, if `incon` $= 0$.

`incon - integer`
On exit, `incon` contains a switch for the consistency of the initial values t_{begin}, $y(t_{\text{begin}})$, and $y'(t_{\text{begin}})$:

- `incon` $= 0$: the initial values are possibly inconsistent: $G(t_{\text{begin}}, y(t_{\text{begin}}), y'(t_{\text{begin}})) \neq 0$. The solver has to compute $y'(t_{\text{begin}})$.
- `incon` $= 1$: the initial values are consistent: $G(t_{\text{begin}}, y(t_{\text{begin}}), y'(t_{\text{begin}})) = 0$.

3. `subroutine geval(neqn, t, y, dy, g)`
`integer neqn`
`double precision t, y(neqn), dy(neqn), g(neqn)`
evaluates the function G

`neqn - integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`t - double precision`
On entry, `t` must specify t, the value of the independent variable.
Unchanged on exit.

`y - double precision` array of `dimension` at least `(neqn)`
On entry, `y(i)` must specify $y_i($`t`$)$, $i = 1, \ldots, d$, the solution at $t =$ `t`.
Unchanged on exit.

`dy - double precision` array of `dimension` at least `(neqn)`
On entry, `dy(i)` must specify $y_i'($`t`$)$, $i = 1, \ldots, d$, the derivative of the solution at $t =$ `t`.
Unchanged on exit.

`g - double precision` array of `dimension` at least `(neqn)`
On exit, `g(i)` contains $G_i($`t`$, $`y`$, $`dy`$)$, $i = 1, \ldots, d$, the value of the function G at $t =$ `t`.

4. `subroutine jeval(neqn, t, y, dy, dgdy, dgddy, ldim)`
`integer neqn, ldim`
`double precision t, y(neqn), dy(neqn), dgdy(ldim,neqn), dgddy(ldim,neqn)`
evaluates the Jacobians $\partial G/\partial y$ and $\partial G/\partial y'$
(this routine is only called if `ijac` $= 1$; a dummy subroutine is supplied in the case `ijac` $= 0$)

`neqn - integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`t - double precision`
On entry, `t` must specify t, the value of the independent variable.
Unchanged on exit.

`y - double precision` array of `dimension` at least `(neqn)`
On entry, `y(i)` must specify $y_i($`t`$)$, $i = 1, \ldots, d$, the solution at $t =$ `t`.
Unchanged on exit.

`dy - double precision` array of `dimension` at least `(neqn)`
On entry, `dy(i)` must specify $y_i'($`t`$)$, $i = 1, \ldots, d$, the derivative of the solution at $t =$ `t`.
Unchanged on exit.

`dgdy - double precision` array of `dimension (ldim,p)` where `p` $\geq$ `neqn`
On exit,

- `dgdy(i,j)` contains $\partial G_i($`t`$, $`y`$, $`dy`$)/\partial y_j$ if the Jacobian is a full matrix (`mljac` $=$ `neqn`);
- `dgdy(i` $-$ `j` $+$ `mujac` $+ 1$`,j)` contains $\partial G_i($`t`$, $`y`$, $`dy`$)/\partial y_j$ if the Jacobian is a band matrix ($0 \leq$ `mljac` $<$ `neqn`) (LAPACK / LINPACK / BLAS storage).

`dgddy - double precision` array of `dimension (ldim,p)` where `p` $\geq$ `neqn`
On exit,

- `dgddy(i,j)` contains $\partial G_i(\mathtt{t},\mathtt{y},\mathtt{dy})/\partial y'_j$ if the Jacobian is a full matrix (`mljac` = `neqn`);
- `dgddy(i - j + mujac + 1,j)` contains $\partial G_i(\mathtt{t},\mathtt{y},\mathtt{dy})/\partial y'_j$ if the Jacobian is a band matrix ($0 \leq$ `mljac` $<$ `neqn`) (LAPACK / LINPACK / BLAS storage).

`ldim - integer`
On entry, `ldim` must specify the first dimension of the arrays `dgdy` and `dgddy` as declared in the calling (sub)program. If `mljac` = `neqn`, then the Jacobians are supposed to be full and the relation `ldim` $\geq$ `neqn` must hold. If $0 \leq$ `mljac` $<$ `neqn` then the Jacobians are taken as banded and the relation `ldim` $\geq$ `mljac` + `mujac` + 1 should hold.
Unchanged on exit.

5. `subroutine solut(neqn, y)`
   `integer neqn`
   `double precision y(neqn)`

   returns a reference solution in the endpoint $y(t_{\mathrm{end}})$

   `neqn - integer`
   On entry, `neqn` must specify d, the dimension of the problem.
   Unchanged on exit.

   `y - double precision` array of `dimension` at least `(neqn)`
   On exit, `y(i)` contains $y_i(t_{\mathrm{end}})$, $i = 1,\ldots,d$, the reference solution at $t = t_{\mathrm{end}}$, the endpoint.

If the index of the IDE is lower than 2, then the problem can be handled by DASSL. The file `dassld.f` contains a driver such that compiling

```
f77 dassld.f ddassl.f problem.f
```

yields an executable that solves a problem, of which the Fortran routines in the format above are in the file `problem.f`, with DASSL.
The auxiliary linear algebra routines used by DASSL are included in the driver. Unless stated otherwise, all input parameters are set to their default values.

III.2 ODEs

For the ODE case, the problem is written in the form

$$\begin{aligned} &y' = f(t,y), \qquad t_{\mathrm{begin}} \leq t \leq t_{\mathrm{end}},\\ &y, f(t,y) \in \mathbf{R}^d,\\ &y(t_{\mathrm{begin}}) \text{ is given.} \end{aligned}$$

The subroutines are:

1. `subroutine prob(problm, neqn, tbegin, tend, ijac, mljac, mujac)`
   `character*(*) problm`
   `integer neqn, ijac, mljac, mujac`
   `double precision tbegin, tend`

   describes the problem

   `problm - character`
   On exit, `problm` contains a character string uniquely identifying the problem.
   The first 8 characters (possibly blank padded) should be unique.

`neqn - integer`
On exit, `neqn` contains d, the dimension of the problem.

`tbegin - double precision`
On exit, `tbegin` contains t_{begin}, the begin point of the integration interval.

`tend - double precision`
On exit, `tend` contains t_{end}, the end point of the integration interval.

`ijac - integer`
On exit, `ijac` contains a switch for the computation of the Jacobian:

- `ijac` $= 0$: the Jacobian has to be computed internally by the solver, a dummy subroutine `jeval` is supplied;
- `ijac` $= 1$: the Jacobian is supplied by the subroutine `jeval`, defined below.

`mljac - integer`
On exit, `mljac` contains a switch for the structure of the Jacobian:

- `mljac` $=$ `neqn`: the Jacobian is a full matrix;
- $0 \leq$ `mljac` $<$ `neqn`: the Jacobian is a band matrix, `mljac` is the lower bandwidth of the Jacobian matrix (`mljac` $\geq$ number of non-zero diagonals below the main diagonal).

`mujac - integer`
On exit, `mujac` contains the upper bandwidth of the Jacobian matrix (`mujac` $\geq$ number of non-zero diagonals above the main diagonal). Need not be defined if `mljac` $=$ `neqn`.

2. `subroutine init(neqn, y)`
`integer neqn`
`double precision y(neqn)`

returns the initial value $y(t_{\text{begin}})$

`neqn - integer`
On entry, `neqn` must specify d, the dimension of the problem. Unchanged on exit.

`y - double precision` array of `dimension` at least `(neqn)`
On exit, `y(i)` contains $y_i(t_{\text{begin}})$, $i = 1, \ldots, d$, the initial values.

3. `subroutine feval(neqn, t, y, dy)`
`integer neqn`
`double precision t, y(neqn), dy(neqn)`

evaluates the right hand side function f (i.e. the derivative y')

`neqn - integer`
On entry, `neqn` must specify d, the dimension of the problem. Unchanged on exit.

`t - double precision`
On entry, `t` must specify t, the value of the independent variable. Unchanged on exit.

`y - double precision` array of `dimension` at least `(neqn)`
On entry, `y(i)` must specify $y_i($`t`$)$, $i = 1, \ldots, d$, the solution at $t =$ `t`. Unchanged on exit.

`dy - double precision` array of `dimension` at least `(neqn)`
On exit, `dy(i)` contains $f_i($`t`$,$ `y`$)$, $i = 1, \ldots, d$, the derivatives of the solution y at $t =$ `t`.

4. `subroutine jeval(neqn, t, y, jac, ldim)`
   `integer neqn, ldim`
   `double precision t, y(neqn), jac(ldim,neqn)`

   evaluates the Jacobian $\partial f/\partial y$
   (this routine is only called if `ijac` $= 1$; a dummy subroutine is supplied in the case `ijac` $= 0$)

   `neqn - integer`
   : On entry, `neqn` must specify d, the dimension of the problem.
   Unchanged on exit.

   `t - double precision`
   : On entry, `t` must specify t, the value of the independent variable.
   Unchanged on exit.

   `y - double precision` array of `dimension` at least `(neqn)`
   : On entry, `y(i)` must specify y_i(`t`), $i = 1, \ldots, d$, the solution at $t =$ `t`.
   Unchanged on exit.

   `jac - double precision` array of `dimension (ldim,p)` where `p` $\geq$ `neqn`
   : On exit,
   - `jac(i,j)` contains ∂f_i(`t`,`y`)$/\partial y_j$ if the Jacobian is a full matrix (`mljac` $=$ `neqn`);
   - `jac(i` $-$ `j` $+$ `mujac` $+ 1$, `j`) contains ∂f_i(`t`,`y`)$/\partial y_j$
     if the Jacobian is a band matrix ($0 \leq$ `mljac` $<$ `neqn`)
     (LAPACK / LINPACK / BLAS storage).

   `ldim - integer`
   : On entry, `ldim` must specify the first dimension of array `jac` as declared in the calling (sub)program. If `mljac` $=$ `neqn`, then the Jacobian is supposed to be full and the relation `ldim` $\geq$ `neqn` must hold. If $0 \leq$ `mljac` $<$ `neqn` then the Jacobian is taken as banded and the relation `ldim` $\geq$ `mljac` $+$ `mujac` $+ 1$ should hold.
   Unchanged on exit.

5. `subroutine solut(neqn, y)`
   `integer neqn`
   `double precision y(neqn)`

   returns a reference solution in the endpoint $y(t_{\mathrm{end}})$

   `neqn - integer`
   : On entry, `neqn` must specify d, the dimension of the problem.
   Unchanged on exit.

   `y - double precision` array of `dimension` at least `(neqn)`
   : On exit, `y(i)` contains $y_i(t_{\mathrm{end}})$, $i = 1, \ldots, d$, the reference solution at $t = t_{\mathrm{end}}$, the endpoint.

The files `rad5do.f`, `voded.f` and `dassldo.f` are drivers such that compiling

```
f77 rad5do.f radau5.f problem.f
f77 voded.f vode.f problem.f
f77 dassldo.f ddassl.f problem.f
```

yields executables that solve a problem, of which the Fortran routines in the format above are in the file `problem.f`, with RADAU5, VODE and DASSL, respectively.
The auxiliary linear algebra routines used by RADAU5, VODE and DASSL are included in the corresponding drivers. For RADAU5, the DECSOL routines are used. In `voded.f`, the input parameter `iwork(6)` is set equal to `1d6`, to allow VODE to make more f-evaluations. Unless stated otherwise, all input parameters are set to their default values.

III.3 DAEs

The form for DAEs reads

$$\begin{aligned} &My' = f(t,y), \qquad t_{\text{begin}} \le t \le t_{\text{end}}, \\ &y, f(t,y) \in \mathbf{R}^d, \qquad M \in \mathbf{R}^{d\times d} \\ &y(t_{\text{begin}}) \text{ is given,} \end{aligned}$$

where M is a constant, possibly singular matrix. The subroutines are:

1. `subroutine prob(problm, neqn, tbegin, tend, ijac, mljac, mujac, mlmas, mumas, ind1, ind2, ind3)`
   `integer neqn, ijac, mljac, mujac, mlmas, mumas, ind1, ind2, ind3`
   `double precision tbegin, tend`

   describes the problem

   `problm - character`
   : On exit, `problm` contains a character string uniquely identifying the problem. The first 8 characters (possibly blank padded) should be unique.

   `neqn - integer`
   : On exit, `neqn` contains d, the dimension of the problem.

   `tbegin - double precision`
   : On exit, `tbegin` contains t_{begin}, the begin point of the integration interval.

   `tend - double precision`
   : On exit, `tend` contains t_{end}, the end point of the integration interval.

   `ijac - integer`
   : On exit, `ijac` contains a switch for the computation of the Jacobian:
     - `ijac` $= 0$: the Jacobian has to be computed internally by the solver, a dummy subroutine `jeval` is supplied;
     - `ijac` $= 1$: the Jacobian is supplied by the subroutine `jeval`, defined below.

   `mljac - integer`
   : On exit, `mljac` contains a switch for the structure of the Jacobian:
     - `mljac` $=$ `neqn`: the Jacobian is a full matrix;
     - $0 \le$ `mljac` $<$ `neqn`: the Jacobian is a band matrix, `mljac` is the lower bandwidth of the Jacobian matrix (`mljac` $\ge$ number of non-zero diagonals below the main diagonal).

   `mujac - integer`
   : On exit, `mujac` contains the upper bandwidth of the Jacobian matrix (`mujac` $\ge$ number of non-zero diagonals above the main diagonal). Need not be defined if `mljac` $=$ `neqn`.

   `mlmas - integer`
   : On exit, `mlmas` contains a switch for the structure of the mass matrix M:
     - `mlmas` $=$ `neqn`: the mass matrix is a full matrix;
     - $0 \le$ `mlmas` $<$ `neqn`: the mass matrix is a band matrix, `mlmas` is the lower bandwidth of the mass matrix (`mlmas` $\ge$ number of non-zero diagonals below the main diagonal).

   `mumas - integer`
   : On exit, `mumas` contains the upper bandwidth of the mass matrix (`mumas` $\ge$ number of non-zero diagonals above the main diagonal). Need not be defined if `mlmas` $=$ `neqn`.

The parameters `ind1`, `ind2` and `ind3` give information on the index of the variables. The right hand side function subroutine `feval` is written such that the index 1, 2, 3 variables appear in this order. The relation `ind1` + `ind2` + `ind3` = `neqn` should hold.

`ind1 - integer`
On exit, `ind1` contains the number variables with index lower than 2 (`ind1` > 0 should hold).
For systems of index lower than 2 this equals `neqn`.

`ind2 - integer`
On exit, `ind2` contains the number of index 2 variables.
For systems of index lower than 2 this equals 0.

`ind3 - integer`
On exit, `ind3` contains the number of index 3 variables.
For systems of index lower than 2 this equals 0.

For the definition of index of a variable, we refer to [BCP89]. We remark that the differential index of the whole problem equals the maximum of the indices of all variables.

2. `subroutine init(neqn, y, dy, incon)`
`integer neqn, incon`
`double precision y(neqn), dy(neqn)`

returns the (possibly inconsistent) initial values $y(t_{\text{begin}})$ and $y'(t_{\text{begin}})$

`neqn - integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`y - double precision` array of `dimension` at least `(neqn)`
On exit, `y(i)` contains $y_i(t_{\text{begin}})$, $i = 1, \ldots, d$, the initial values of the solution.

`dy - double precision` array of `dimension` at least `(neqn)`
On exit,

- `dy(i)` contains $y_i'(t_{\text{begin}})$, $i = 1, \ldots, d$, consistent initial values of the derivative of the solution, if `incon` $= 1$.
- `dy(i)` contains 0 or an approximation to $y_i'(t_{\text{begin}})$, $i = 1, \ldots, d$, inconsistent initial values of the derivative of the solution, if `incon` $= 0$.

`incon - integer`
On exit, `incon` contains a switch for the consistency of the initial values t_{begin}, $y(t_{\text{begin}})$, and $y'(t_{\text{begin}})$:

- `incon` $= 0$: the initial values are possibly inconsistent:
  $My'(t_{\text{begin}}) \neq f(t_{\text{begin}}, y(t_{\text{begin}}))$. The solver has to compute $y'(t_{\text{begin}})$.
- `incon` $= 1$: the initial values are consistent: $My'(t_{\text{begin}}) = f(t_{\text{begin}}, y(t_{\text{begin}}))$.

3. `subroutine feval(neqn, t, y, dy)`
`integer neqn`
`double precision t, y(neqn), dy(neqn)`

evaluates the right hand side function f

`neqn - integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`t - double precision`
On entry, `t` must specify t, the value of the independent variable.
Unchanged on exit.

`y` - `double precision` array of `dimension` at least `(neqn)`
On entry, `y(i)` must specify $y_i(\mathtt{t})$, $i = 1, \ldots, d$, the solution at $t = \mathtt{t}$.
Unchanged on exit.

`dy` - `double precision` array of `dimension` at least `(neqn)`
On exit, `dy(i)` contains $f_i(\mathtt{t}, \mathtt{y})$, $i = 1, \ldots, d$.

4. `subroutine jeval(neqn, t, y, jac, ldim)`
`integer neqn, ldim`
`double precision t, y(neqn), jac(ldim,neqn)`

evaluates the Jacobian $\partial f / \partial y$
(this routine is only called if `ijac` $= 1$; a dummy subroutine is supplied in the case `ijac` $= 0$)

`neqn` - `integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`t` - `double precision`
On entry, `t` must specify t, the value of the independent variable.
Unchanged on exit.

`y` - `double precision` array of `dimension` at least `(neqn)`
On entry, `y(i)` must specify $y_i(\mathtt{t})$, $i = 1, \ldots, d$, the solution at $t = \mathtt{t}$.
Unchanged on exit.

`jac` - `double precision` array of `dimension (ldim,p)` where `p` $\geq$ `neqn`
On exit,

- `jac(i,j)` contains $\partial f_i(\mathtt{t}, \mathtt{y}) / \partial y_j$ if the Jacobian is a full matrix (`mljac` $=$ `neqn`);
- `jac(i` $-$ `j` $+$ `mujac` $+ 1$`,j)` contains $\partial f_i(\mathtt{t}, \mathtt{y}) / \partial y_j$
  if the Jacobian is a band matrix ($0 \leq$ `mljac` $<$ `neqn`)
  (LAPACK / LINPACK / BLAS storage).

`ldim` - `integer`
On entry, `ldim` must specify the first dimension of array `jac` as declared in the calling (sub)program. If `mljac` $=$ `neqn`, then the Jacobian is supposed to be full and the relation `ldim` $\geq$ `neqn` must hold. If $0 \leq$ `mljac` $<$ `neqn` then the Jacobian is taken as banded and the relation `ldim` $\geq$ `mljac` $+$ `mujac` $+ 1$ should hold.
Unchanged on exit.

5. `subroutine solut(neqn, y)`
`integer neqn`
`double precision y(neqn)`

returns a reference solution in the endpoint $y(t_{\mathrm{end}})$

`neqn` - `integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`y` - `double precision` array of `dimension` at least `(neqn)`
On exit, `y(i)` contains $y_i(t_{\mathrm{end}})$, $i = 1, \ldots, d$, the reference solution at $t = t_{\mathrm{end}}$, the endpoint.

6. `subroutine mas(neqn, am, ldim)`
`integer neqn, ldim`
`double precision am(ldim,neqn)`

returns the mass matrix M

`neqn` - `integer`
On entry, `neqn` must specify d, the dimension of the problem.
Unchanged on exit.

`am` - `double precision` array of `dimension (ldim,p)` where `p` $\geq$ `neqn`
On exit,

- `am(i,j)` contains M_{ij} if the mass matrix is a full matrix (`mlmas` = `neqn`);
- `am(i - j + mumas + 1,j)` contains M_{ij} if the mass matrix is a band matrix ($0 \leq$ `mlmas` $<$ `neqn`) (LAPACK / LINPACK / BLAS storage).

`ldim` - `integer`
On entry, `ldim` must specify the first dimension of array `am` as declared in the calling (sub)program. If `mlmas` = `neqn`, then the mass matrix is supposed to be full and the relation `ldim` $\geq$ `neqn` must hold. If $0 \leq$ `mlmas` $<$ `neqn` then the mass matrix is taken as banded and the relation `ldim` $\geq$ `mlmas` + `mumas` + 1 should hold.
Unchanged on exit.

If the index of the problem is lower than 4, then RADAU5 can solve problems of this type. The file `rad5da.f` contains a driver such that compiling

```
f77 rad5da.f radau5.f problem.f
```

yields an executable that solves a problem, of which the Fortran routines in the format above are in the file `problem.f`, with RADAU5. The DECSOL routines are used for the linear algebra and are included in the driver.

Since DAEs are a subset of the class of IDEs, DASSL can solve index 1 DAEs by setting

$$G(t, y, y') = My' - f(t, y).$$

The file `dasslda.f` contains a driver for DASSL. Compiling

```
f77 dasslda.f ddassl.f problem.f
```

yields an executable that solves a index 1 DAE, of which the Fortran routines are in the file `problem.f`, with DASSL.
The auxiliary linear algebra routines used by DASSL are included in the driver. Unless stated otherwise, all input parameters are set to their default values.

References

[BCP89] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial–Value Problems in Differential–Algebraic Equations*. North–Holland, New York – Amsterdam – London, 1989.

[Bel92] A. Bellen. PADETEST: a set of real-life test differential equations for parallel computing. Technical Report 103, Dipartimento di Scienze Matematiche, Università di Trieste, 1992.

[BHB92] Peter N. Brown, Alan C. Hindmarsh, and George D. Byrne. *VODE: A variable coefficient ODE solver*, August 1992. Available via WWW at URL http://www.netlib.org/ode/vode.f.

[EP87] W. H. Enright and J. D. Pryce. Two Fortran packages for assessing initial value methods. *ACM Transactions on Mathematical Software*, 13-I:1–27, 1987.

[HS91] P. J. van der Houwen and B. P. Sommeijer. Iterated Runge–Kutta methods on parallel computers. *SIAM J. Sci. Stat. Comput.*, 12:1000–1028, 1991.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems*. Springer-Verlag, 1991.

[HW95] E. Hairer and G. Wanner. *RADAU5*, September 1995. Available via WWW at URL ftp://ftp.unige.ch/pub/doc/math/stiff/radau5.f.

[Pet91] L. R. Petzold. *DASSL: A Differential/Algebraic System Solver*, June 1991. Available via WWW at URL http://www.netlib.org/ode/ddassl.f.

[SB95] J. J. B. de Swart and J. G. Blom. Experiences with sparse matrix solvers in parallel ODE software. Technical Report NM-R9520, CWI, Amsterdam, 1995. To appear in: Computers & mathematics with applications.

Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Test set for IVP solvers

W.M. Lioen, J.J.B. de Swart and W.A. van der Veen

Department of Numerical Mathematics

NM-R9615 November 30, 1996

Report NM-R9615
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

1 Chemical Akzo Nobel problem

1.1 General information

This IVP is a stiff system of 6 non-linear differential equations. It has been taken from [Sto95]. The parallel-IVP-algorithm group of CWI contributed this problem to the test set.

1.2 Mathematical description of the problem

The problem is of the form

$$\frac{dy}{dt} = f(y), \quad y(0) = y_0,$$

with

$$y \in \mathbf{R}^6, \quad 0 \leq t \leq 180.$$

The function f is defined by

$$f(y) = \begin{pmatrix} -2r_1 & +r_2 & -r_3 & -r_4 & & \\ -\frac{1}{2}r_1 & & & -r_4 & -\frac{1}{2}r_5 & +F_{in} \\ r_1 & -r_2 & +r_3 & & & \\ & -r_2 & +r_3 & -2r_4 & & \\ & r_2 & -r_3 & & +r_5 & \\ & & & & -r_5 & \end{pmatrix},$$

where the r_i and F_{in} are auxiliary variables, given by

$$\begin{aligned} r_1 &= k_1 \cdot y_1^4 \cdot y_2^{\frac{1}{2}}, \\ r_2 &= k_2 \cdot y_3 \cdot y_4, \\ r_3 &= \frac{k_2}{K} \cdot y_1 \cdot y_5, \\ r_4 &= k_3 \cdot y_1 \cdot y_4^2, \\ r_5 &= k_4 \cdot y_6^2 \cdot y_2^{\frac{1}{2}}, \\ F_{in} &= klA \cdot (\frac{p(O_2)}{H} - y_2). \end{aligned}$$

The values of the parameters k_1, k_2, k_3, k_4, K, klA, $p(O_2)$ and H are

$$\begin{aligned} k_1 &= 18.7, \\ k_2 &= 0.58, \\ k_3 &= 0.09, \\ k_4 &= 0.42, \\ K &= 34.4, \\ klA &= 3.3, \\ p(O_2) &= 0.9, \\ H &= 737. \end{aligned}$$

Finally, the initial vector y_0 is given by

$$y_0 = \begin{pmatrix} 0.437 \\ 0.00123 \\ 0 \\ 0 \\ 0 \\ 0.367 \end{pmatrix}.$$

1.3 Origin of the problem

The problem originates from Akzo Nobel Central Research in Arnhem, The Netherlands. It describes a chemical process, in which 2 species, MBT and CHA, are mixed, while oxygen is continuously added. The resulting species of importance is CBS. The reaction equations, as given by Akzo Nobel [CBS93], are

$$\begin{aligned}
2MBT + \frac{1}{2}O_2 &\xrightarrow{k_1} MBTS + H_2O, \\
CBS + MBT &\underset{k_2}{\overset{k_2/K}{\rightleftarrows}} MBTS + CHA, \\
MBT + 2CHA + O_2 &\xrightarrow{k_3} BT + \text{sulfate}, \\
MBT.CHA + \frac{1}{2}O_2 &\xrightarrow{k_4} CBS + H_2O, \\
MBT + CHA &\rightleftarrows MBT.CHA.
\end{aligned}$$

The last equation describes an equilibrium

$$Ks^1 = \frac{[MBT.CHA]}{[MBT] \cdot [CHA]},$$

while the others describe reactions, whose velocities are given by

$$\begin{aligned}
r_1 &= k_1 \cdot [MBT]^4 \cdot [O_2]^{\frac{1}{2}}, \\
r_2 &= k_2 \cdot [MBTS] \cdot [CHA], \\
r_3 &= \frac{k_2}{K} \cdot [MBT] \cdot [CBS], \\
r_4 &= k_3 \cdot [MBT] \cdot [CHA]^2, \\
r_5 &= k_4 \cdot [MBT.CHA]^2 \cdot [O_2]^{\frac{1}{2}},
\end{aligned}$$

respectively. Here the square brackets '[ ]' denote concentrations.

The inflow of oxygen per volume unit is denoted by F_{in}, and satisfies

$$F_{in} = klA \cdot (\frac{p(O_2)}{H} - [O_2]),$$

where klA is the mass transfer coefficient, H is the Henry constant and $p(O_2)$ is the partial oxygen pressure. $p(O_2)$ is assumed to be independent of $[O_2]$. The parameters k_1, k_2, k_3, k_4, K, klA, H and $p(O_2)$ are given constants[2].

The process is started by mixing 0.437 mol/liter $[MBT]$ with 0.367 mol/liter $[MBT.CHA]$. The concentration of oxygen at the beginning is 0.00123 mol/liter. Initially, no other species are present. The simulation is performed on the time interval [0, 180 minutes].

Identifying the concentrations $[MBT]$, $[O_2]$, $[MBTS]$, $[CHA]$, $[CBS]$, $[MBT.CHA]$ with $y_1, \ldots, y_6$, respectively, one easily arrives at the mathematical formulation of the preceding subsection.

1.4 Numerical solution of the problem

1.4.1 Solution at $t = 180$:

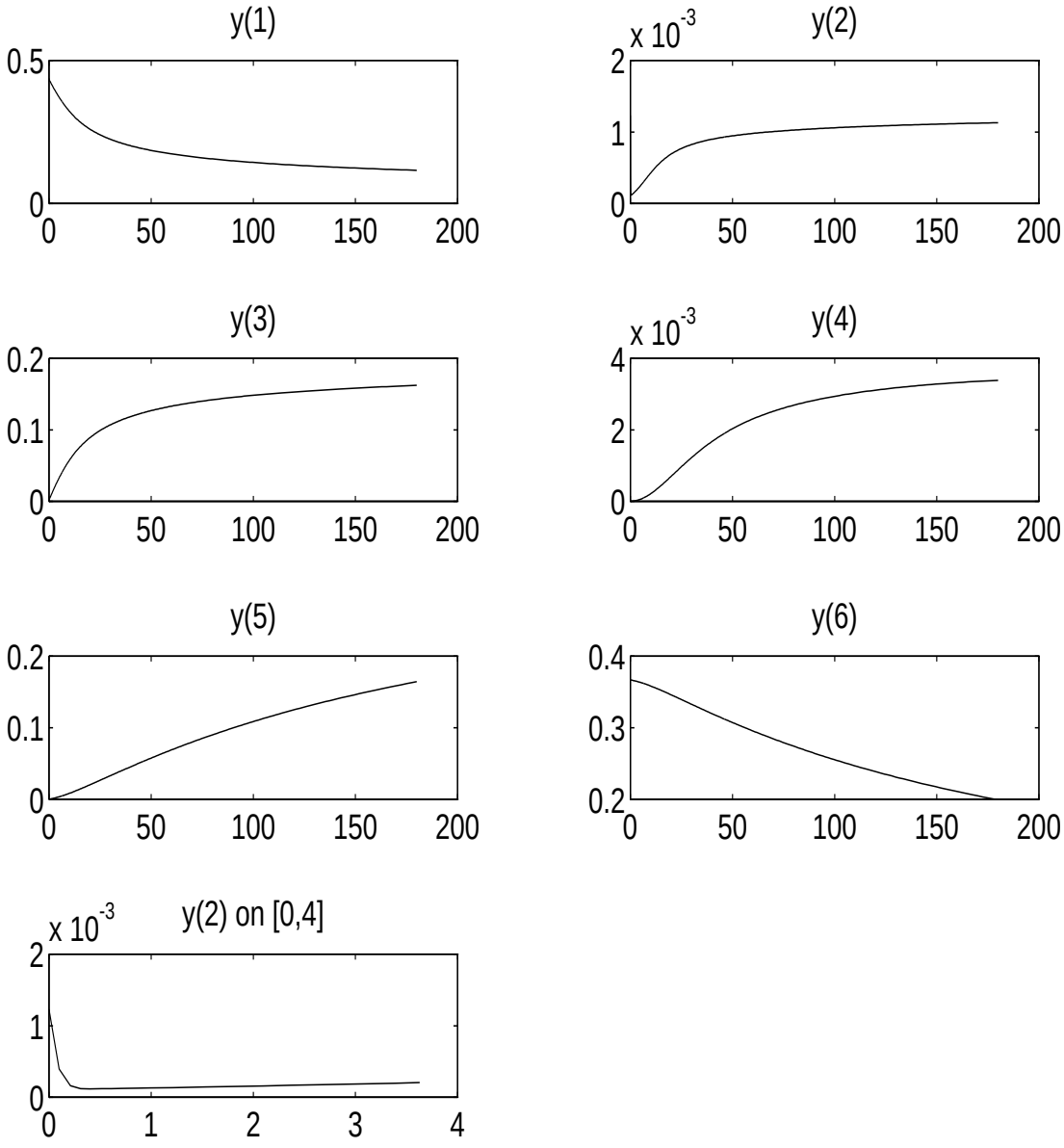
y_1	0.1161602274780192
y_2	$0.1119418166040848 \cdot 10^{-2}$
y_3	0.1621261719785814
y_4	$0.3396981299297459 \cdot 10^{-2}$
y_5	0.1646185108335055
y_6	0.1989533275954281

[1] This value plays a role in parameter estimation.

[2] Apart from H, which is generally known, all parameters have been estimated by W. Stortelder [Sto95].

1.4.2 Behaviour of the numerical solution

The following plots show the behaviour of the solution components:



1.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		3.98	46	44	72	13		0.02
	10^{-7}	10^{-7}		5.76	160	155	225	24		0.05
	10^{-10}	10^{-10}		8.00	396	391	474	32		0.10
RADAU5	10^{-7}	10^{-7}	10^{-7}	7.09	59	58	388	46	55	0.03
	10^{-10}	10^{-10}	10^{-10}	9.15	284	284	1595	69	109	0.11
VODE	10^{-4}	10^{-4}		2.45	64	63	92	2	17	0.01
	10^{-7}	10^{-7}		5.91	183	170	263	4	41	0.04
	10^{-10}	10^{-10}		7.87	367	358	450	7	44	0.07
PSODE	10^{-4}	10^{-4}	10^{-5}	4.94	25	24	612	3	92	0.04
	10^{-7}	10^{-7}	10^{-7}	7.71	70	68	1548	7	196	0.09
	10^{-10}	10^{-10}	10^{-10}	11.42	219	218	4550	5	248	0.25

The speed-up factors of PSODE on the Cray C98 are, in order of decreasing rtol: 1.8, 1.8, 1.9.

1.4.4 Work-precision diagram

In Figure 1 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol = $10^{-(4+m/4)}$, $m = 0, 1, 4, \ldots, 24$ for DASSL, $m = 6, \ldots, 24$ for RADAU5, $m = 0, \ldots, 24$ for VODE, $m = 1, \ldots, 24$ for PSODE; atol = rtol; h0 = rtol for RADAU5 and PSODE.

References

[CBS93] CBS-reaction-meeting Köln. Handouts, May 1993. Br/ARLO-CRC.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[Sto95] W. J. H. Stortelder, 1995. Private communication.

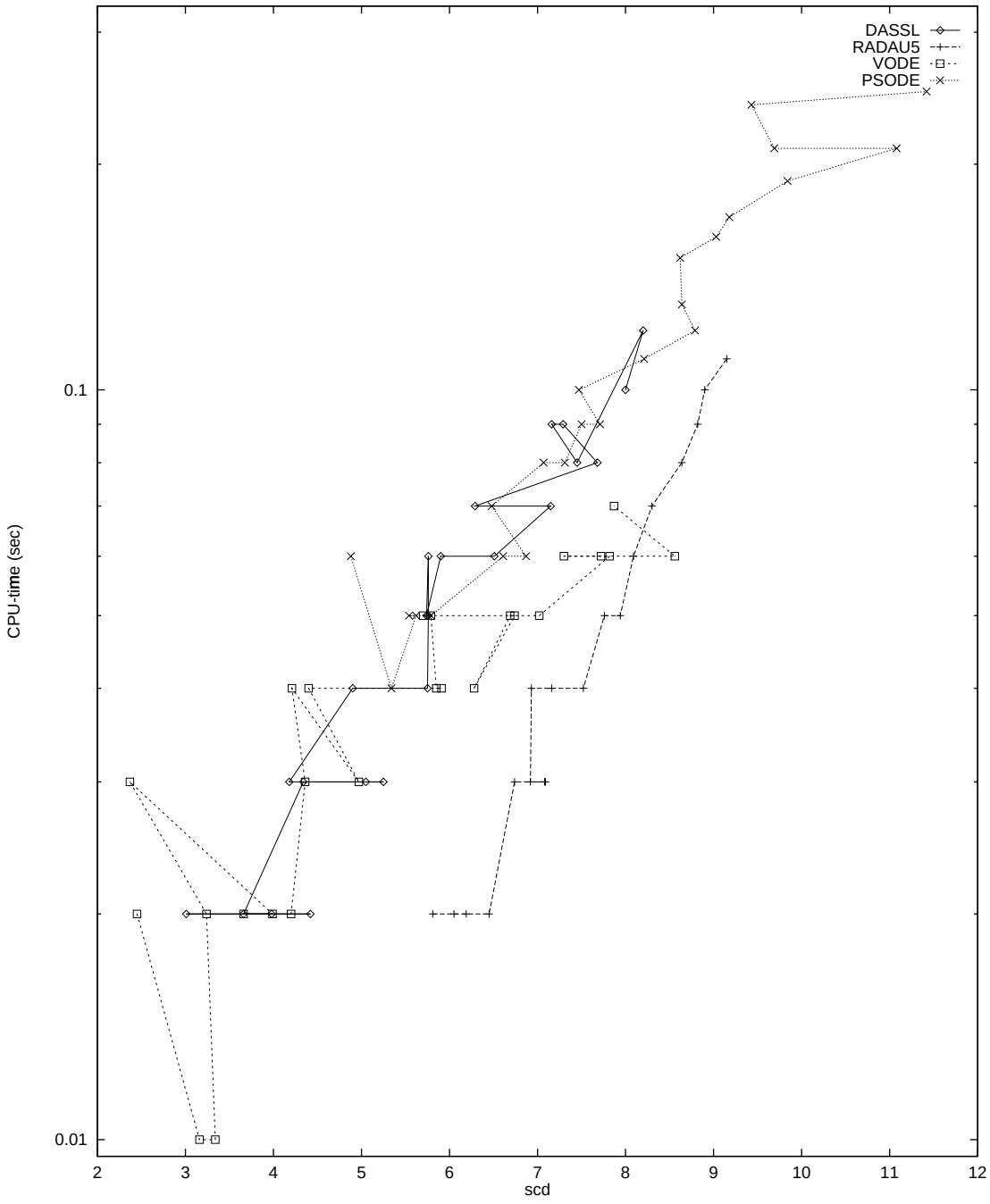


Figure 1: Work-precision diagram for Chemical Akzo Nobel problem

2 Problem HIRES

2.1 General information

This IVP is a stiff system of 8 non-linear differential equations. It was proposed by Schäfer in 1975 [Sch75]. The name HIRES was given by Hairer & Wanner [HW91]. It refers to 'High Irradiance RESponse', which is described by this IVPODE. The parallel-IVP-algorithm group of CWI contributed this problem to the test set.

2.2 Mathematical description of the problem

The problem is of the form

$$\frac{dy}{dt} = f(y), \quad y(0) = y_0,$$

with

$$y \in \mathbf{R}^8, \quad 0 \le t \le 321.8122.$$

The function f is defined by

$$f(y) = \begin{pmatrix} -1.71y_1 & +0.43y_2 & +8.32y_3 & +0.0007 & \\ 1.71y_1 & -8.75y_2 & & & \\ -10.03y_3 & +0.43y_4 & +0.035y_5 & & \\ 8.32y_2 & +1.71y_3 & -1.12y_4 & & \\ -1.745y_5 & +0.43y_6 & +0.43y_7 & & \\ -280y_6y_8 & +0.69y_4 & +1.71y_5 & -0.43y_6 & +0.69y_7 \\ 280y_6y_8 & -1.81y_7 & & & \\ -280y_6y_8 & +1.81y_7 & & & \end{pmatrix}.$$

The initial vector y_0 is given by $(1, 0, 0, 0, 0, 0, 0, 0.0057)^T$.

2.3 Origin of the problem

The problem originates from plant physiology, and is described in [Sch75]. It explains the "High Irradiance Responses" (HIRES) of Photomorphogenesis on the Basis of Phytochrome, by means of a chemical reaction involving 8 reactants. It has been promoted as a test problem by Gottwald in [Got77]. The reaction scheme is given below.

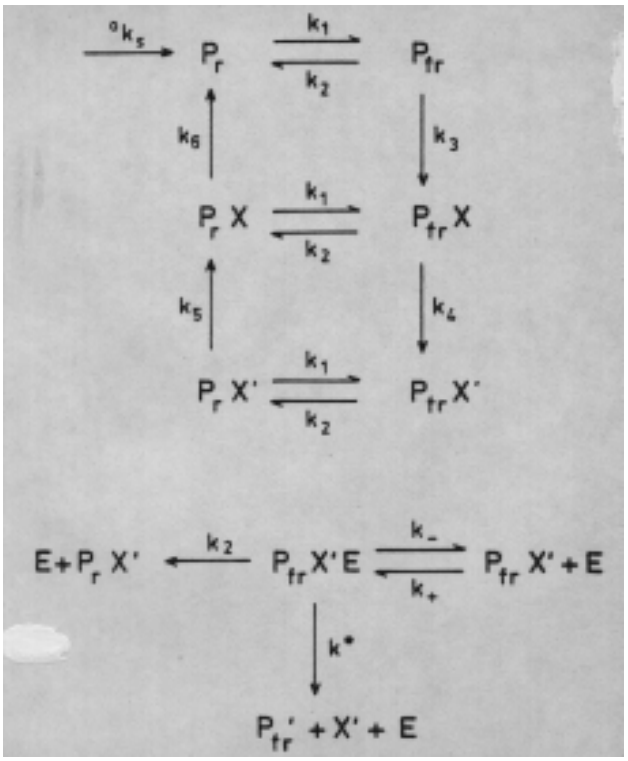


Figure 1: *Reaction scheme for HIRES (taken from [Got77])*

The values of the parameters were taken from [HW91]:

k_1	=	1.71	k_+	=	280
k_2	=	0.43	k_-	=	0.69
k_3	=	8.32	k^*	=	0.69
k_4	=	0.69	o_{k_s}	=	0.0007
k_5	=	0.035			
k_6	=	8.32			

Identifying P_r, P_{fr}, P_rX, $P_{fr}X$, P_rX', $P_{fr}X'$, $P_{fr}X'E$ and E with y_i, $i \in \{1, \ldots, 8\}$, respectively, the differential equations mentioned in Subsection 2.2 easily follow.

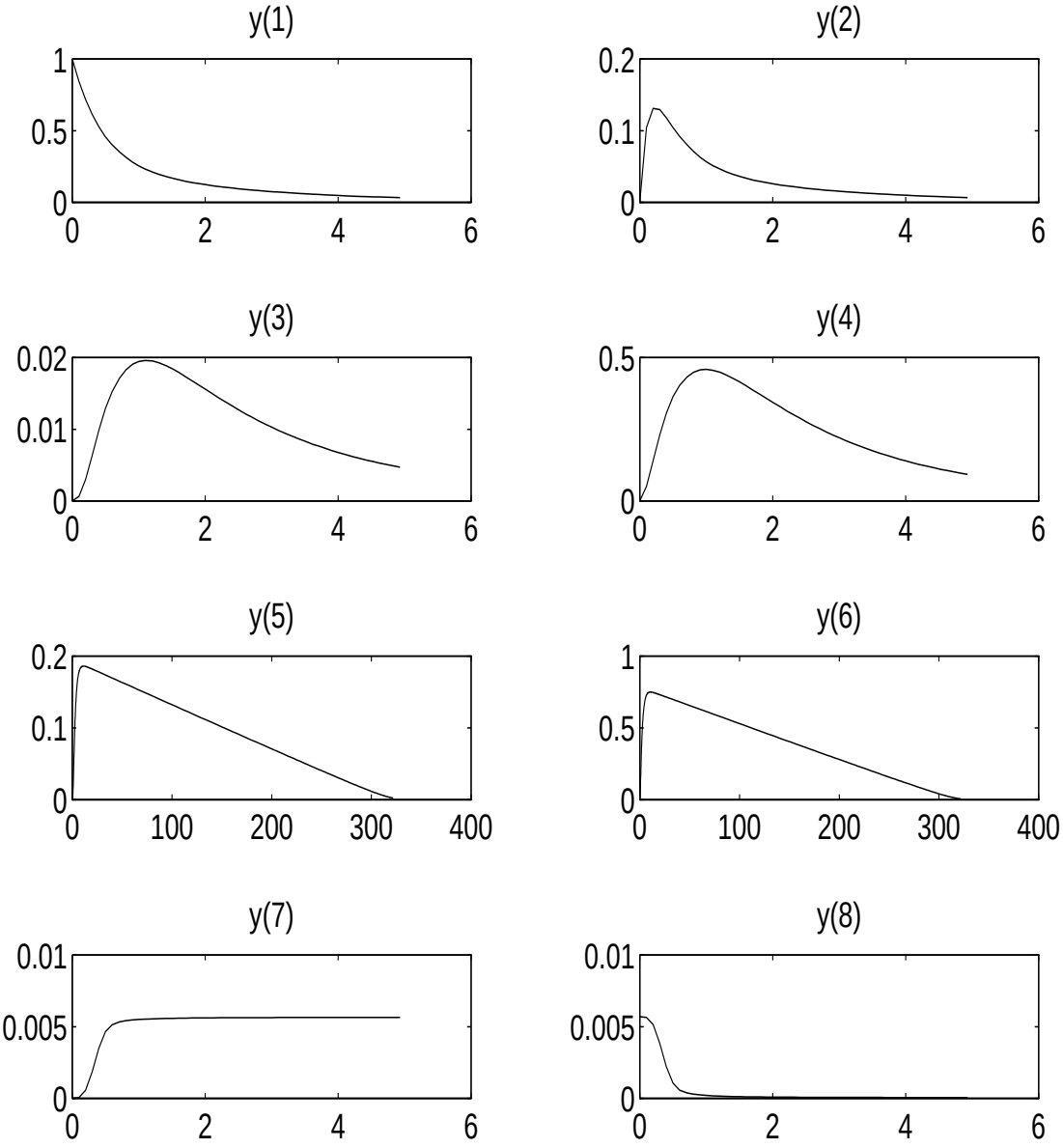
2.4 Numerical solution of the problem

2.4.1 Solution at $t = 321.8122$:

y_1	$0.7371312573325668 \cdot 10^{-3}$
y_2	$0.1442485726316185 \cdot 10^{-3}$
y_3	$0.5888729740967575 \cdot 10^{-4}$
y_4	$0.1175651343283149 \cdot 10^{-2}$
y_5	$0.2386356198831331 \cdot 10^{-2}$
y_6	$0.6238968252742796 \cdot 10^{-2}$
y_7	$0.2849998395185769 \cdot 10^{-2}$
y_8	$0.2850001604814231 \cdot 10^{-2}$

2.4.2 Behaviour of the numerical solution

The following plots show the behaviour of the solution (on different time intervals):



2.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		1.03	99	90	176	32		0.04
	10^{-7}	10^{-7}		3.36	311	307	459	40		0.10
	10^{-10}	10^{-10}		7.01	1077	1061	1493	47		0.31
RADAU5	10^{-4}	10^{-4}	10^{-7}	2.32	45	37	331	22	45	0.03
	10^{-7}	10^{-7}	10^{-9}	4.81	135	133	784	46	85	0.06
	10^{-10}	10^{-10}	10^{-11}	8.85	701	701	3752	140	223	0.27
VODE	10^{-4}	10^{-4}		1.33	131	129	191	10	24	0.03
	10^{-7}	10^{-7}		3.84	390	365	608	9	69	0.09
	10^{-10}	10^{-10}		6.18	880	827	1224	15	134	0.18
PSODE	10^{-4}	10^{-4}	10^{-7}	5.18	61	60	1540	17	232	0.12
	10^{-7}	10^{-7}	10^{-9}	8.31	203	181	5061	24	624	0.37
	10^{-10}	10^{-10}	10^{-11}	10.77	570	558	14022	66	984	0.94

The speed-up factors of PSODE on the Cray C98 are, in order of decreasing rtol: 2.2, 2.3, 2.2.

2.4.4 Work-precision diagram

In Figure 2 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol = $10^{-(4+m/4)}$, $m = 0, \ldots, 24$; atol = rtol; h0 = $10^{-2} \cdot$ rtol for RADAU5 and PSODE.

References

[Got77] B. A. Gottwald. MISS – ein einfaches Simulations-System für biologische und chemische Prozesse. *EDV in Medizin und Biologie*, 3:85–90, 1977.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[Sch75] E. Schäfer. A new approach to explain the 'high irradiance responses' of photomorphogenesis on the basis of phytochrome. *J. of Math. Biology*, 2:41–56, 1975.

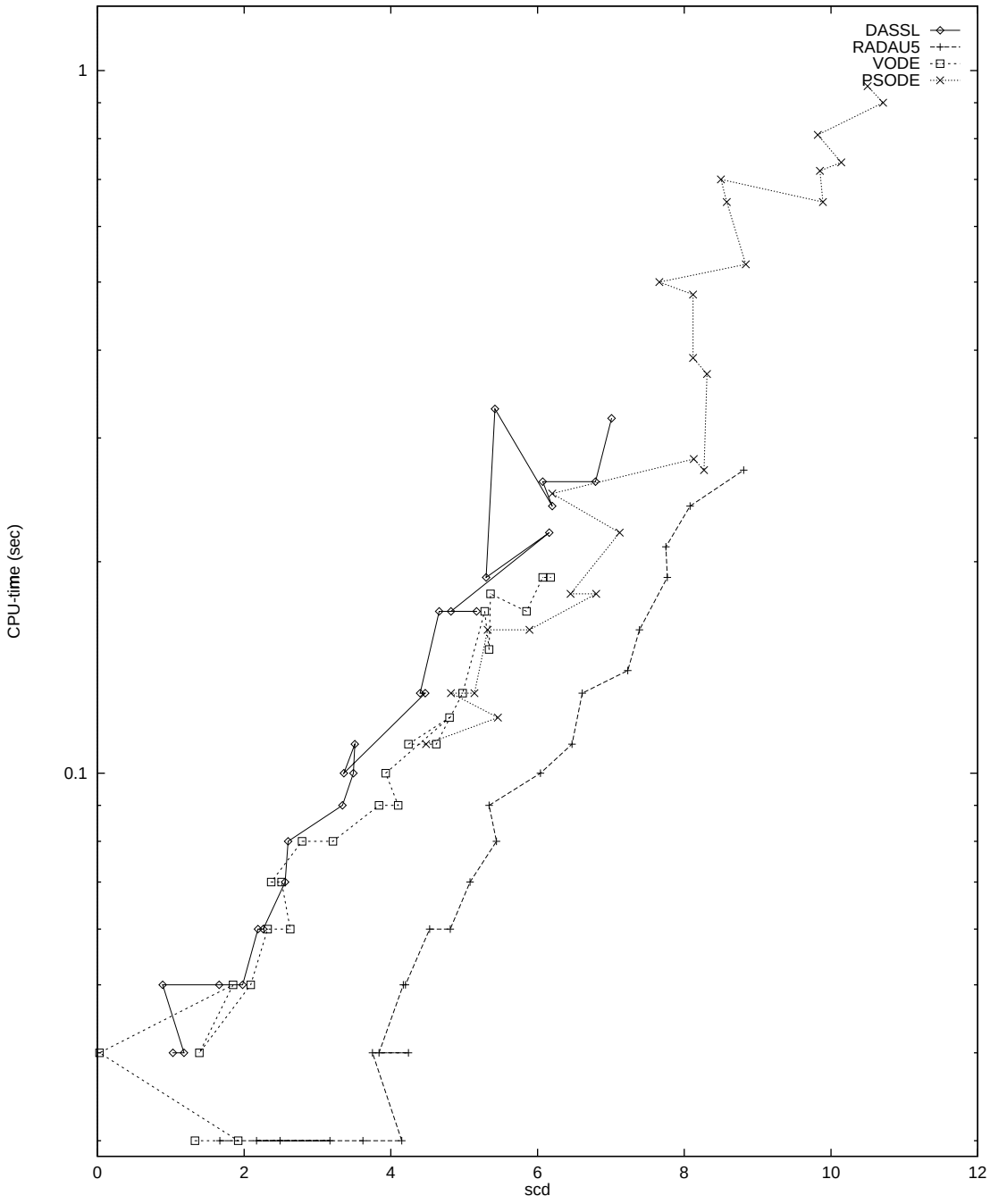


Figure 2: Work-precision diagram for Problem HIRES

3 Pollution problem

3.1 General information

This IVP is a stiff system of 20 non-linear differential equations. It is the chemical part of the air pollution model developed at The Dutch National Institute of Public Health and Environmental Protection (RIVM) and it is described by Verwer in [Ver94]. The parallel-IVP-algorithm group of CWI contributed this problem to the test set.

3.2 Mathematical description of the problem

The problem is of the form

$$\frac{dy}{dt} = f(y), \quad y(0) = y_0,$$

with

$$y \in \mathbf{R}^{20}, \quad 0 \leq t \leq 60.$$

The function f is defined by

$$f = \begin{pmatrix} -\sum_{j\in\{1,10,14,23,24\}} r_j + \sum_{j\in\{2,3,9,11,12,22,25\}} r_j \\ -r_2 - r_3 - r_9 - r_{12} + r_1 + r_{21} \\ -r_{15} + r_1 + r_{17} + r_{19} + r_{22} \\ -r_2 - r_{16} - r_{17} - r_{23} + r_{15} \\ -r_3 + 2r_4 + r_6 + r_7 + r_{13} + r_{20} \\ -r_6 - r_8 - r_{14} - r_{20} + r_3 + 2r_{18} \\ -r_4 - r_5 - r_6 + r_{13} \\ r_4 + r_5 + r_6 + r_7 \\ -r_7 - r_8 \\ -r_{12} + r_7 + r_9 \\ -r_9 - r_{10} + r_8 + r_{11} \\ r_9 \\ -r_{11} + r_{10} \\ -r_{13} + r_{12} \\ r_{14} \\ -r_{18} - r_{19} + r_{16} \\ -r_{20} \\ r_{20} \\ -r_{21} - r_{22} - r_{24} + r_{23} + r_{25} \\ -r_{25} + r_{24} \end{pmatrix},$$

where the r_i are auxiliary variables, given by

r_1	$=$	$k_1 \cdot y_1$	r_2	$=$	$k_2 \cdot y_2 \cdot y_4$
r_3	$=$	$k_3 \cdot y_5 \cdot y_2$	r_4	$=$	$k_4 \cdot y_7$
r_5	$=$	$k_5 \cdot y_7$	r_6	$=$	$k_6 \cdot y_7 \cdot y_6$
r_7	$=$	$k_7 \cdot y_9$	r_8	$=$	$k_8 \cdot y_9 \cdot y_6$
r_9	$=$	$k_9 \cdot y_{11} \cdot y_2$	r_{10}	$=$	$k_{10} \cdot y_{11} \cdot y_1$
r_{11}	$=$	$k_{11} \cdot y_{13}$	r_{12}	$=$	$k_{12} \cdot y_{10} \cdot y_2$
r_{13}	$=$	$k_{13} \cdot y_{14}$	r_{14}	$=$	$k_{14} \cdot y_1 \cdot y_6$
r_{15}	$=$	$k_{15} \cdot y_3$	r_{16}	$=$	$k_{16} \cdot y_4$
r_{17}	$=$	$k_{17} \cdot y_4$	r_{18}	$=$	$k_{18} \cdot y_{16}$
r_{19}	$=$	$k_{19} \cdot y_{16}$	r_{20}	$=$	$k_{20} \cdot y_{17} \cdot y_6$
r_{21}	$=$	$k_{21} \cdot y_{19}$	r_{22}	$=$	$k_{22} \cdot y_{19}$
r_{23}	$=$	$k_{23} \cdot y_1 \cdot y_4$	r_{24}	$=$	$k_{24} \cdot y_{19} \cdot y_1$
r_{25}	$=$	$k_{25} \cdot y_{20}$			

The values of the parameters k_j are

k_1	$= 0.350$	k_2	$= 0.266 \cdot 10^2$
k_3[1]	$= 0.123 \cdot 10^5$	k_4	$= 0.860 \cdot 10^{-3}$
k_5	$= 0.820 \cdot 10^{-3}$	k_6	$= 0.150 \cdot 10^5$
k_7	$= 0.130 \cdot 10^{-3}$	k_8	$= 0.240 \cdot 10^5$
k_9	$= 0.165 \cdot 10^5$	k_{10}	$= 0.900 \cdot 10^4$
k_{11}	$= 0.220 \cdot 10^{-1}$	k_{12}	$= 0.120 \cdot 10^5$
k_{13}	$= 0.188 \cdot 10$	k_{14}	$= 0.163 \cdot 10^5$
k_{15}	$= 0.480 \cdot 10^7$	k_{16}	$= 0.350 \cdot 10^{-3}$
k_{17}	$= 0.175 \cdot 10^{-1}$	k_{18}	$= 0.100 \cdot 10^9$
k_{19}	$= 0.444 \cdot 10^{12}$	k_{20}	$= 0.124 \cdot 10^4$
k_{21}	$= 0.210 \cdot 10$	k_{22}	$= 0.578 \cdot 10$
k_{23}	$= 0.474 \cdot 10^{-1}$	k_{24}	$= 0.178 \cdot 10^4$
k_{25}	$= 0.312 \cdot 10$		

Finally, the initial vector y_0 is given by

$$y_0 = (0, 0.2, 0, 0.04, 0, 0, 0.1, 0.3, 0.01, 0, 0, 0, 0, 0, 0, 0.007, 0, 0, 0)^T.$$

3.3 Origin of the problem

The problem is a chemical model consisting of 25 reactions and 20 reacting compounds. The reactions read:

[1] Notice that this constant has a typing error in [Ver94].

1.	NO2	→	NO+O3P
2.	NO+O3	→	NO2
3.	HO2+NO	→	NO2+OH
4.	HCHO	→	2 HO2+CO
5.	HCHO	→	CO
6.	HCHO+OH	→	HO2+CO
7.	ALD	→	MEO2+HO2+CO
8.	ALD+OH	→	C2O3
9.	C2O3+NO	→	NO2+MEO2+CO2
10.	C2O3+NO2	→	PAN
11.	PAN	→	C2O3+NO2
12.	MEO2+NO	→	CH3O+NO2
13.	CH3O	→	HCHO+HO2
14.	NO2+OH	→	HNO3
15.	O3P	→	O3
16.	O3	→	O1D
17.	O3	→	O3P
18.	O1D	→	2 OH
19.	O1D	→	O3P
20.	SO2+OH	→	SO4+HO2
21.	NO3	→	NO
22.	NO3	→	NO2+O3P
23.	NO2+O3	→	NO3
24.	NO3+NO2	→	N2O5
25.	N2O5	→	NO3+NO2

Writing down the reaction velocities r_j for every reaction equation and making the identification in the table below, one arrives at the system of differential equations given in the preceding subsection. The square brackets '[ ]' denote concentrations. Also listed are the concentrations at $t = 0$.

variable	species	initial value
y_1	[NO2]	0
y_2	[NO]	0.2
y_3	[O3P]	0
y_4	[O3]	0.04
y_5	[HO2]	0
y_6	[OH]	0
y_7	[HCHO]	0.1
y_8	[CO]	0.3
y_9	[ALD]	0.01
y_{10}	[MEO2]	0
y_{11}	[C2O3]	0
y_{12}	[CO2]	0
y_{13}	[PAN]	0
y_{14}	[CH3O]	0
y_{15}	[HNO3]	0
y_{16}	[O1D]	0
y_{17}	[SO2]	0.007
y_{18}	[SO4]	0
y_{19}	[NO3]	0
y_{20}	[N2O5]	0

The time interval [0,60] represents the behaviour of the reactants sufficiently.

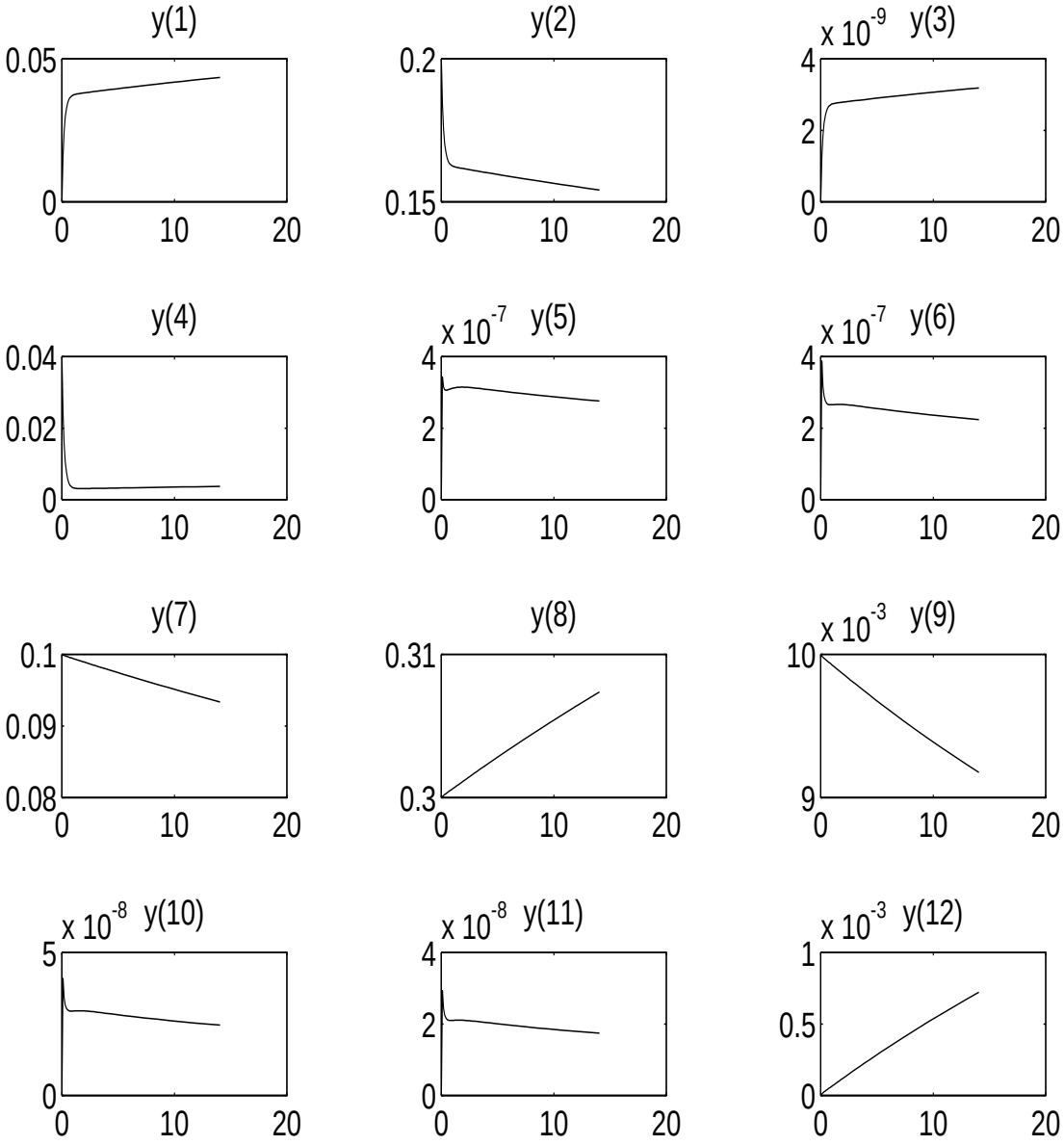
3.4 Numerical solution of the problem

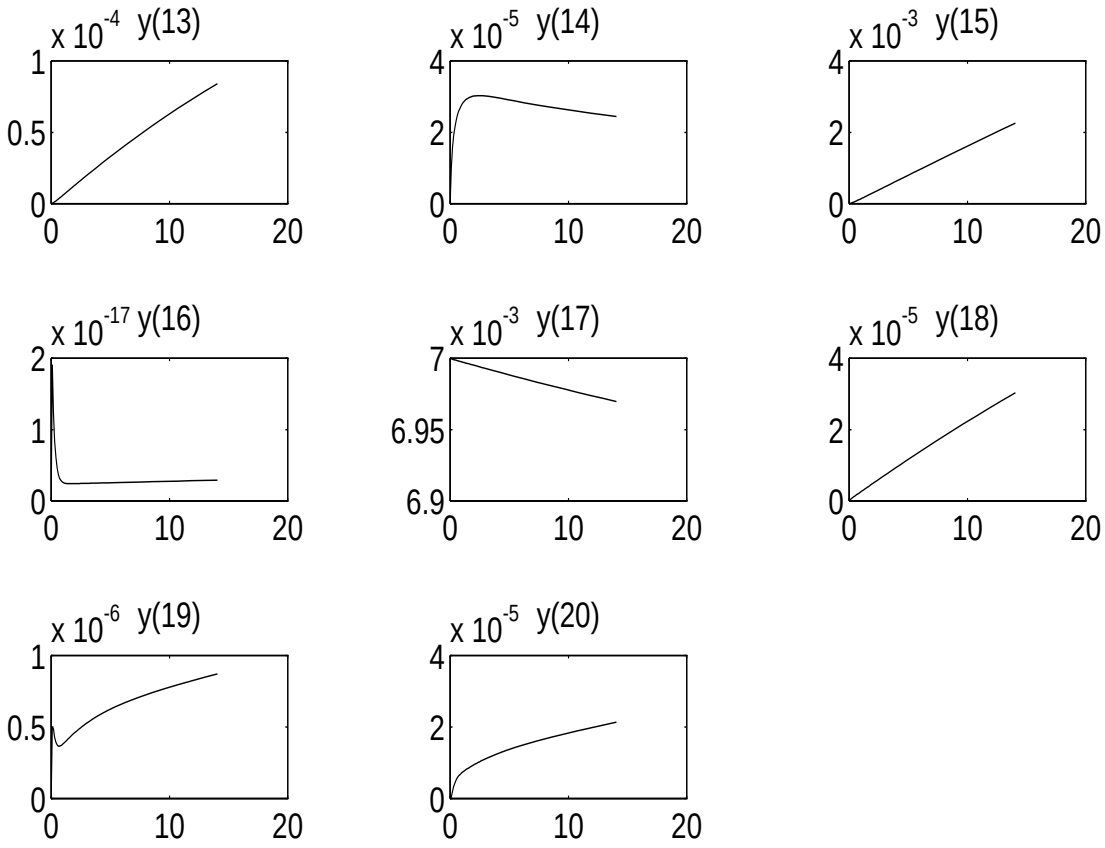
3.4.1 Solution at $t = 60$:

y_1	$0.5646255480022769 \cdot 10^{-1}$
y_2	0.1342484130422339
y_3	$0.4139734331099427 \cdot 10^{-8}$
y_4	$0.5523140207484359 \cdot 10^{-2}$
y_5	$0.2018977262302196 \cdot 10^{-6}$
y_6	$0.1464541863493966 \cdot 10^{-6}$
y_7	$0.7784249118997964 \cdot 10^{-1}$
y_8	0.3245075353396018
y_9	$0.7494013383880406 \cdot 10^{-2}$
y_{10}	$0.1622293157301561 \cdot 10^{-7}$
y_{11}	$0.1135863833257075 \cdot 10^{-7}$
y_{12}	$0.2230505975721359 \cdot 10^{-2}$
y_{13}	$0.2087162882798630 \cdot 10^{-3}$
y_{14}	$0.1396921016840158 \cdot 10^{-4}$
y_{15}	$0.8964884856898295 \cdot 10^{-2}$
y_{16}	$0.4352846369330103 \cdot 10^{-17}$
y_{17}	$0.6899219696263405 \cdot 10^{-2}$
y_{18}	$0.1007803037365946 \cdot 10^{-3}$
y_{19}	$0.1772146513969984 \cdot 10^{-5}$
y_{20}	$0.5682943292316392 \cdot 10^{-4}$

3.4.2 Behaviour of the numerical solution

The following plots show the behaviour of the solution components on the interval [0,12]:





3.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		1.98	36	35	57	14		0.03
	10^{-7}	10^{-7}		4.13	135	135	192	23		0.09
	10^{-10}	10^{-10}		5.55	365	362	497	40		0.23
RADAU5	10^{-4}	10^{-4}	10^{-5}	2.41	25	20	164	15	24	0.06
	10^{-7}	10^{-7}	10^{-7}	4.94	44	44	272	19	37	0.09
	10^{-10}	10^{-10}	10^{-10}	8.53	207	207	1125	26	76	0.29
VODE	10^{-4}	10^{-4}		1.65	55	55	106	5	17	0.04
	10^{-7}	10^{-7}		3.64	149	149	210	4	26	0.07
	10^{-10}	10^{-10}		4.72	375	357	528	7	60	0.18
PSODE	10^{-4}	10^{-4}	10^{-5}	5.58	34	34	834	7	136	0.24
	10^{-7}	10^{-7}	10^{-7}	7.52	90	90	1982	6	244	0.51
	10^{-10}	10^{-10}	10^{-10}	10.66	283	283	6183	5	324	1.34

The speed-up factors of PSODE on the Cray C98 are, in order of decreasing rtol: 2.8, 2.8, 2.7.

3.4.4 Work-precision diagram

In Figure 1 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol = $10^{-(4+m/4)}$, $m = 0, \ldots, 24$; atol = rtol; h0 = rtol for RADAU5 and PSODE.

References

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[Ver94] J. G. Verwer. Gauss-Seidel iteration for stiff ODEs from chemical kinetics. *SIAM J. Sci. Comput.*, 15(5):1243–1259, 1994.

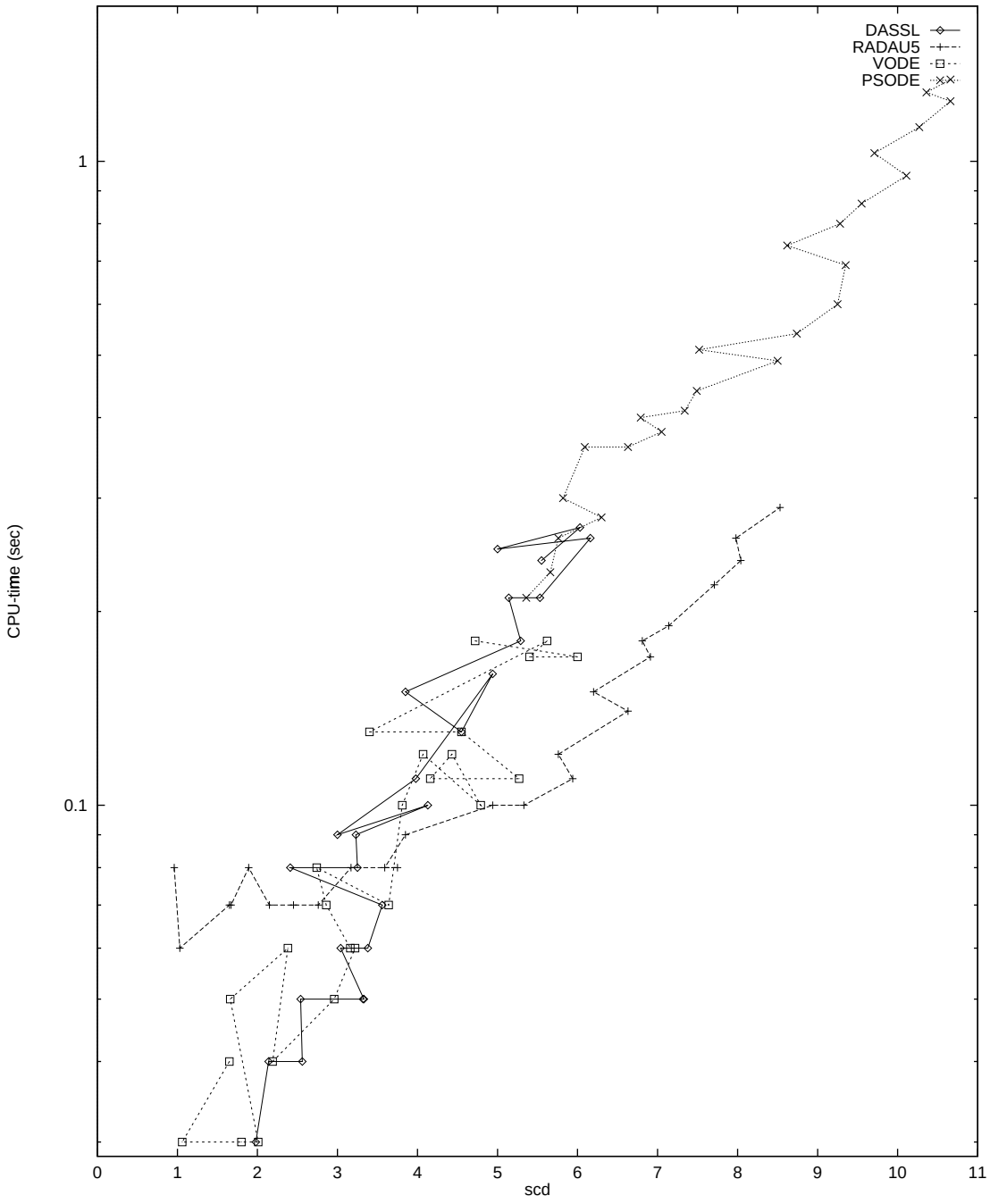


Figure 1: Work-precision diagram for Pollution Problem

4 Ring modulator

4.1 General information

The type of the problem depends on the parameter C_s. If $C_s \neq 0$, then it is a stiff system of 15 non-linear ordinary differential equations. For $C_s = 0$ we have a DAE of index 2, consisting of 11 differential equations and 4 algebraic equations. It has been taken from [KRS91], where the approach of Horneber [Hor76] is followed. The parallel-IVP-algorithm group of CWI contributed this problem to the test set.

4.2 Mathematical description of the problem

For the ODE case, the problem is of the form

$$\frac{dy}{dt} = f(t,y), \quad y(0) = y_0,$$

with

$$y \in \mathbf{R}^{15}, \quad 0 \leq t \leq 10^{-3}.$$

The function f is defined by

$$f(t,y) = \begin{pmatrix} C^{-1}(y_8 - 0.5y_{10} + 0.5y_{11} + y_{14} - R^{-1}y_1) \\ C^{-1}(y_9 - 0.5y_{12} + 0.5y_{13} + y_{15} - R^{-1}y_2) \\ C_s^{-1}(y_{10} - q(U_{D1}) + q(U_{D4})) \\ C_s^{-1}(-y_{11} + q(U_{D2}) - q(U_{D3})) \\ C_s^{-1}(y_{12} + q(U_{D1}) - q(U_{D3})) \\ C_s^{-1}(-y_{13} - q(U_{D2}) + q(U_{D4})) \\ C_p^{-1}(-R_p^{-1}y_7 + q(U_{D1}) + q(U_{D2}) - q(U_{D3}) - q(U_{D4})) \\ -L_h^{-1}y_1 \\ -L_h^{-1}y_2 \\ L_{s2}^{-1}(0.5y_1 - y_3 - R_{g2}y_{10}) \\ L_{s3}^{-1}(-0.5y_1 + y_4 - R_{g3}y_{11}) \\ L_{s2}^{-1}(0.5y_2 - y_5 - R_{g2}y_{12}) \\ L_{s3}^{-1}(-0.5y_2 + y_6 - R_{g3}y_{13}) \\ L_{s1}^{-1}(-y_1 + U_{in1}(t) - (R_i + R_{g1})y_{14}) \\ L_{s1}^{-1}(-y_2 - (R_c + R_{g1})y_{15}) \end{pmatrix}. \tag{1}$$

The auxiliary functions $U_{D1}, U_{D2}, U_{D3}, U_{D4}, q, U_{in1}$ and U_{in2} are given by

$$\begin{aligned} U_{D1} &= y_3 - y_5 - y_7 - U_{in2}(t), \\ U_{D2} &= -y_4 + y_6 - y_7 - U_{in2}(t), \\ U_{D3} &= y_4 + y_5 + y_7 + U_{in2}(t), \\ U_{D4} &= -y_3 - y_6 + y_7 + U_{in2}(t), \\ q(U) &= \gamma(e^{\delta U} - 1), \\ U_{in1}(t) &= 0.5\sin(2000\pi t), \\ U_{in2}(t) &= 2\sin(20000\pi t). \end{aligned}$$

The values of the parameters C, C_s, C_p, R, R_p, L_h, L_{s1}, L_{s2}, L_{s3}, R_{g1}, R_{g2}, R_{g3}, R_i, R_c, γ and δ are:

$$\begin{aligned} C &= 1.6 \cdot 10^{-8}, \\ C_s &= \begin{cases} 10^{-9} & \Rightarrow \text{ ODE-system}, \\ 0 & \Rightarrow \text{ DAE-system}, \end{cases} \\ C_p &= 10^{-8}, \end{aligned}$$

$$
\begin{aligned}
R &= 25000, \\
R_p &= 50, \\
L_h &= 4.45, \\
L_{s1} &= 0.002, \\
L_{s2} &= 5 \cdot 10^{-4}, \\
L_{s3} &= 5 \cdot 10^{-4}, \\
R_{g1} &= 36.3, \\
R_{g2} &= 17.3, \\
R_{g3} &= 17.3, \\
R_i &= 50, \\
R_c &= 600, \\
\gamma &= 40.67286402 \cdot 10^{-9}, \\
\delta &= 17.7493332.
\end{aligned}
$$

Finally, the initial vector y_0 is given by

$$y_0 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)^T.$$

4.3 Origin of the problem

The problem originates from electrical circuit analysis. It describes the behaviour of a ring modulator. A ring modulator mixes a low-frequency signal with a high-frequency signal. The diagram of a ring modulator is given below.

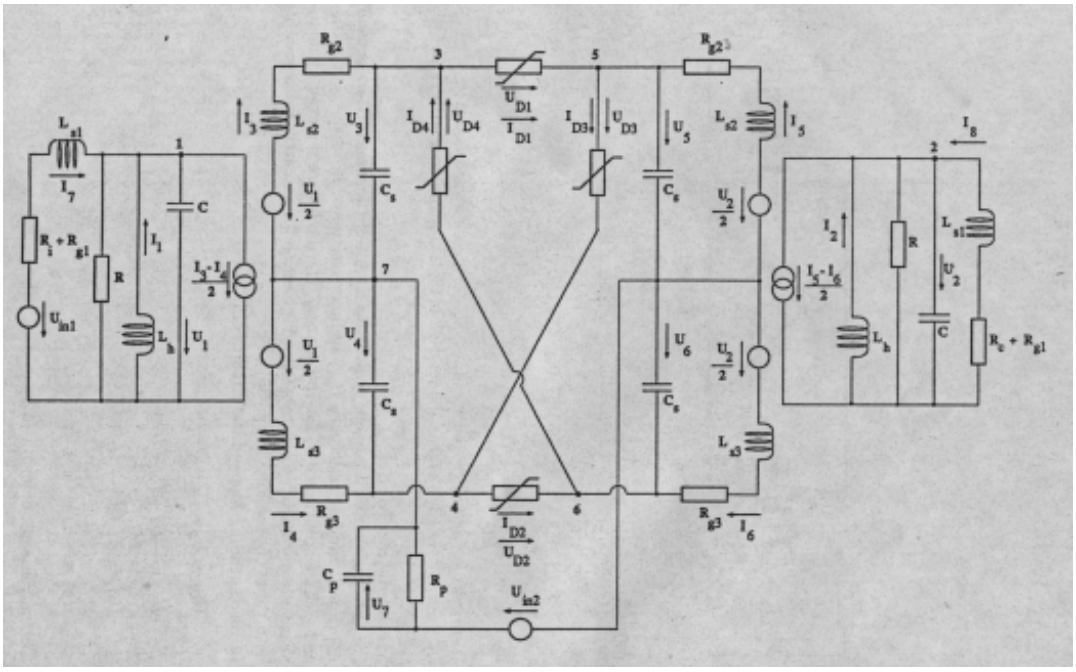


Figure 1: *Circuit diagram for Ring Modulator*

Every capacity in the diagram leads to a differential equation:

$$C\dot{U} = I.$$

Applying Kirchhoff's Current Law yields the following differential equations:

$$\begin{aligned}
C\dot{U}_1 &= I_1 - 0.5I_3 + 0.5I_4 + I_7 - R^{-1}U_1,\\
C\dot{U}_2 &= I_2 - 0.5I_5 + 0.5I_6 + I_8 - R^{-1}U_2,\\
C_s\dot{U}_3 &= I_3 - q(U_{D1}) + q(U_{D4}),\\
C_s\dot{U}_4 &= -I_4 + q(U_{D2}) - q(U_{D3}),\\
C_s\dot{U}_5 &= I_5 + q(U_{D1}) - q(U_{D3}),\\
C_s\dot{U}_6 &= -I_6 - q(U_{D2}) + q(U_{D4}),\\
C_p\dot{U}_7 &= -R_p^{-1}U_7 + q(U_{D1}) + q(U_{D2}) - q(U_{D3}) - q(U_{D4}),
\end{aligned}$$

where U_{D1}, U_{D1}, U_{D1} and U_{D1} stand for:

$$\begin{aligned}
U_{D1} &= U_3 - U_5 - U_7 - U_{in2},\\
U_{D2} &= -U_4 + U_6 - U_7 - U_{in2},\\
U_{D3} &= U_4 + U_5 + U_7 + U_{in2},\\
U_{D4} &= -U_3 - U_6 + U_7 + U_{in2}.
\end{aligned}$$

The diode function q is given by

$$q(U) = \gamma(e^{\delta U} - 1),$$

where γ and δ are fixed constants.

Every inductor leads to a differential equation as well:

$$L\dot{I} = U.$$

Hence, we obtain another 8 differential equations for the 8 inductors:

$$\begin{array}{rcrcrcl}
L_h\dot{I}_1 &=& -U_1, & & & & \\
L_h\dot{I}_2 &=& -U_2, & & & & \\
L_{s2}\dot{I}_3 &=& 0.5U_1 & - & U_3 & - & R_{g2}I_3, \\
L_{s3}\dot{I}_4 &=& -0.5U_1 & + & U_4 & - & R_{g3}I_4, \\
L_{s2}\dot{I}_5 &=& 0.5U_2 & - & U_5 & - & R_{g2}I_5, \\
L_{s3}\dot{I}_6 &=& -0.5U_2 & + & U_6 & - & R_{g3}I_6, \\
L_{s1}\dot{I}_7 &=& -U_1 & + & U_{in1}, & - & (R_i + R_{g1})I_7, \\
L_{s1}\dot{I}_8 &=& -U_2, & & & - & (R_c + R_{g1})I_8.
\end{array}$$

Initially all voltages and currents are zero.

Identifying the voltages with $y_1, \ldots, y_7$ and the currents with $y_8, \ldots, y_{15}$, we obtain the 15 differential equations mentioned before.

4.4 Numerical solution of the problem

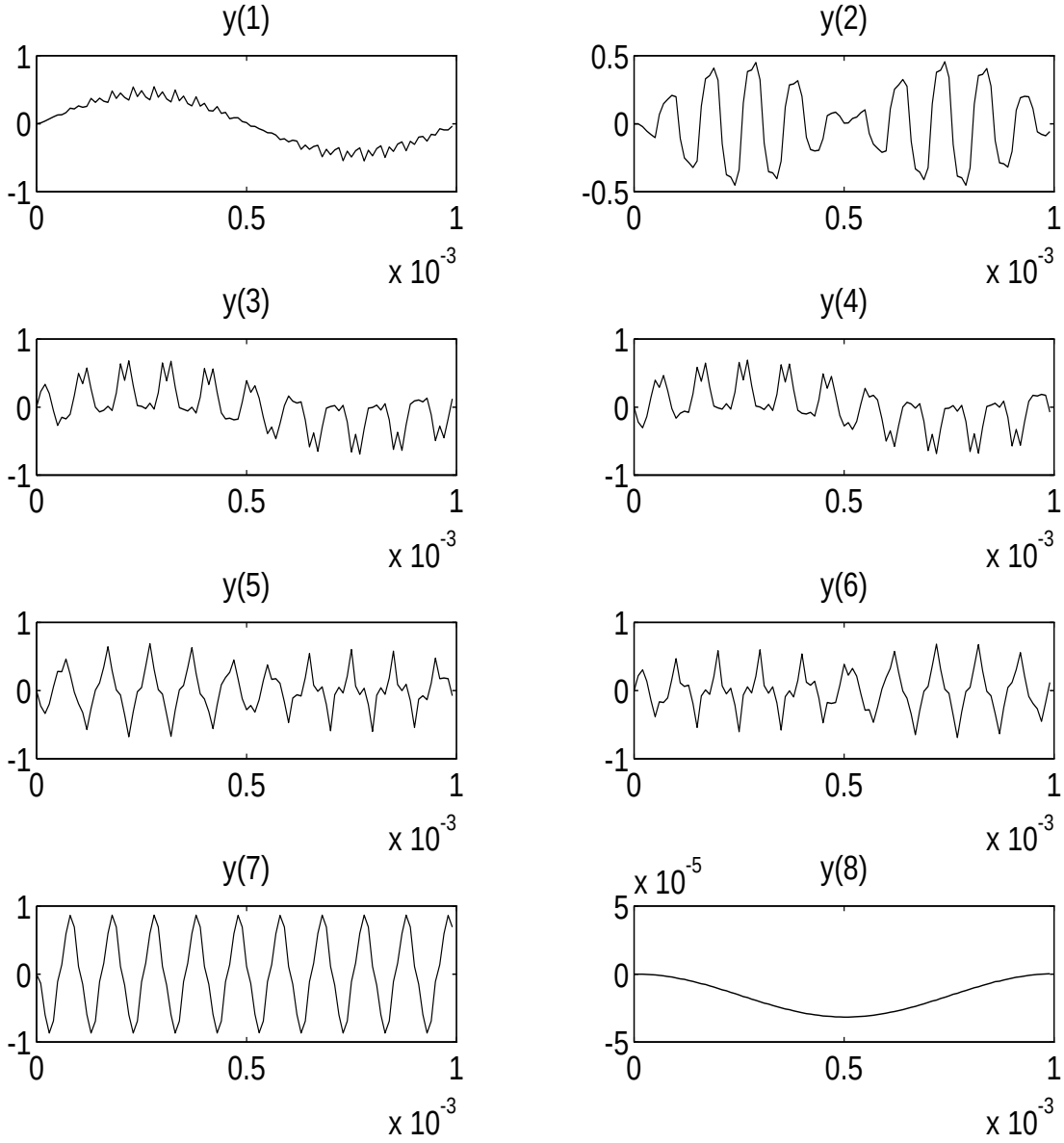
All the tests concern the ODE-case in which only 1 ring modulator is involved (i.e. $C_s = 10^{-9}$ and $N = 1$).

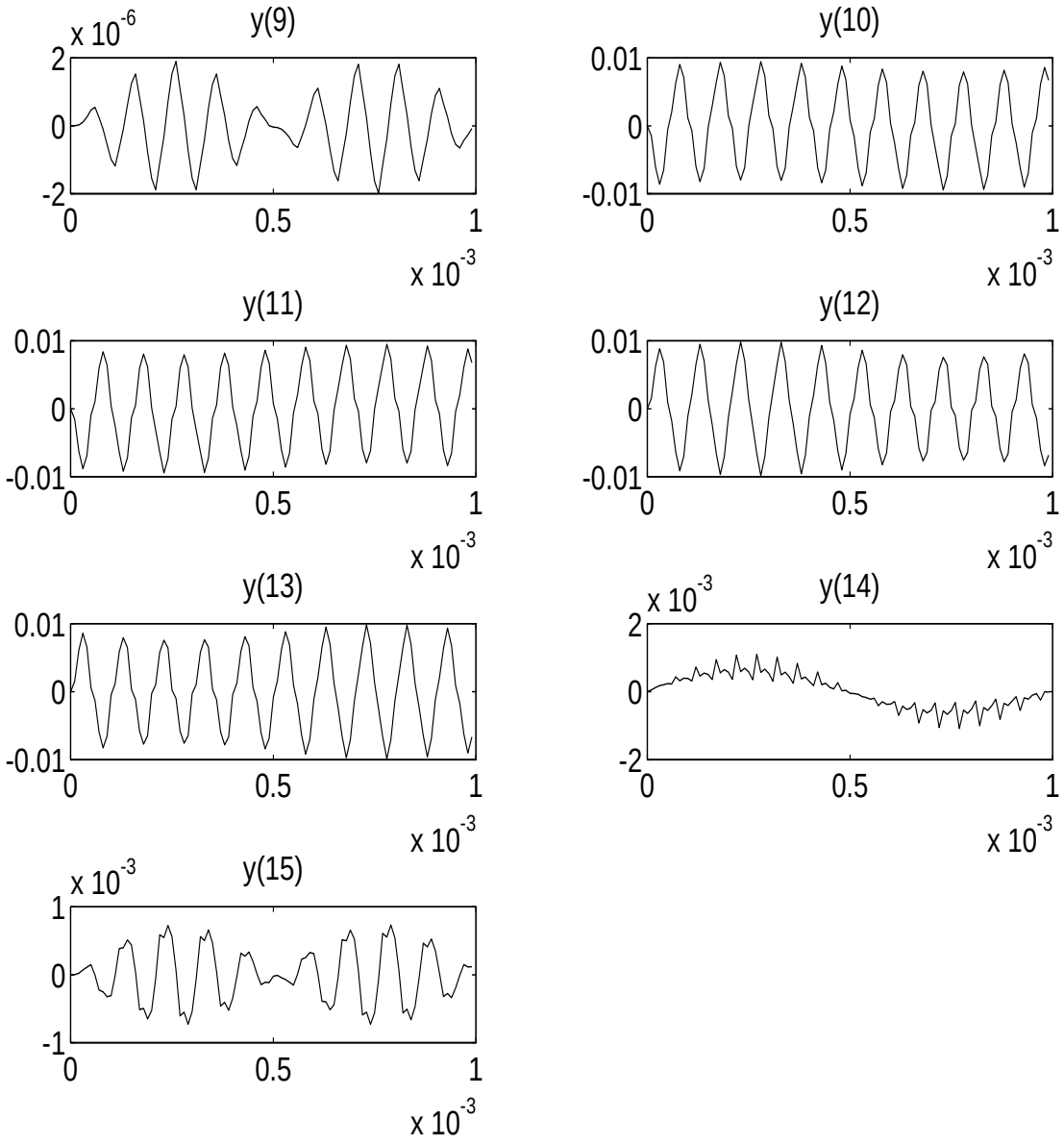
4.4.1 Solution at $t = 10^{-3}$:

y_1	$-0.17079903291846 \cdot 10^{-1}$
y_2	$-0.66609789784834 \cdot 10^{-2}$
y_3	0.27531919254370
y_4	-0.39115731811511
y_5	-0.38851730770493
y_6	0.27795920295388
y_7	0.11146002811043
y_8	$0.29791296267403 \cdot 10^{-6}$
y_9	$-0.31427403451731 \cdot 10^{-7}$
y_{10}	$0.70165883118556 \cdot 10^{-3}$
y_{11}	$0.85207537676917 \cdot 10^{-3}$
y_{12}	$-0.77741454302426 \cdot 10^{-3}$
y_{13}	$-0.77631966493048 \cdot 10^{-3}$
y_{14}	$0.78439425971261 \cdot 10^{-4}$
y_{15}	$0.25232278361831 \cdot 10^{-4}$

4.4.2 Behaviour of the numerical solution

The following plots show the behaviour of the solution components:





4.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		0.77	5989	5759	10230	442		3.51
	10^{-7}	10^{-7}		3.63	18987	18746	28016	546		9.98
RADAU5	10^{-4}	10^{-4}	10^{-6}	2.89	2562	2119	15747	1076	2161	3.44
	10^{-7}	10^{-7}	10^{-8}	5.99	11158	10699	58768	2309	5989	11.71
VODE	10^{-4}	10^{-4}		0.50	7326	6837	11712	250	1191	2.97
	10^{-7}	10^{-7}		3.43	17977	16899	25937	329	2213	6.83
PSODE	10^{-4}	10^{-4}	10^{-6}	3.75	2537	2190	58638	422	7516	11.18
	10^{-7}	10^{-7}	10^{-8}	7.70	8506	7384	172496	420	19836	32.23

The speed-up factors of PSODE on the Cray C98 are, in order of decreasing rtol: 2.8, 2.7.

4.4.4 Work-precision diagram

In Figure 2 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol $= 10^{-(4+m/8)}$, $m = 0, 2, \ldots, 24$ for VODE and $m = 0, \ldots, 24$ otherwise; atol = rtol; h0 $= 10^{-2} \cdot$ rtol for RADAU5 and PSODE.

References

[Hor76] E. H. Horneber. *Analyse nichtlinearer RLCÜ-Netzwerke mit Hilfe der gemischten Potentialfunktion mit einer systematischen Darstellung der Analyse nichtlinearer dynamischer Netzwerke.* PhD thesis, Universität Kaiserslautern, 1976.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[KRS91] W. Kampowski, P. Rentrop, and W. Schmidt. *Classification and Numerical Simulation of Electric Circuits.* Math. Inst. Tech. Univ. München, 1991.

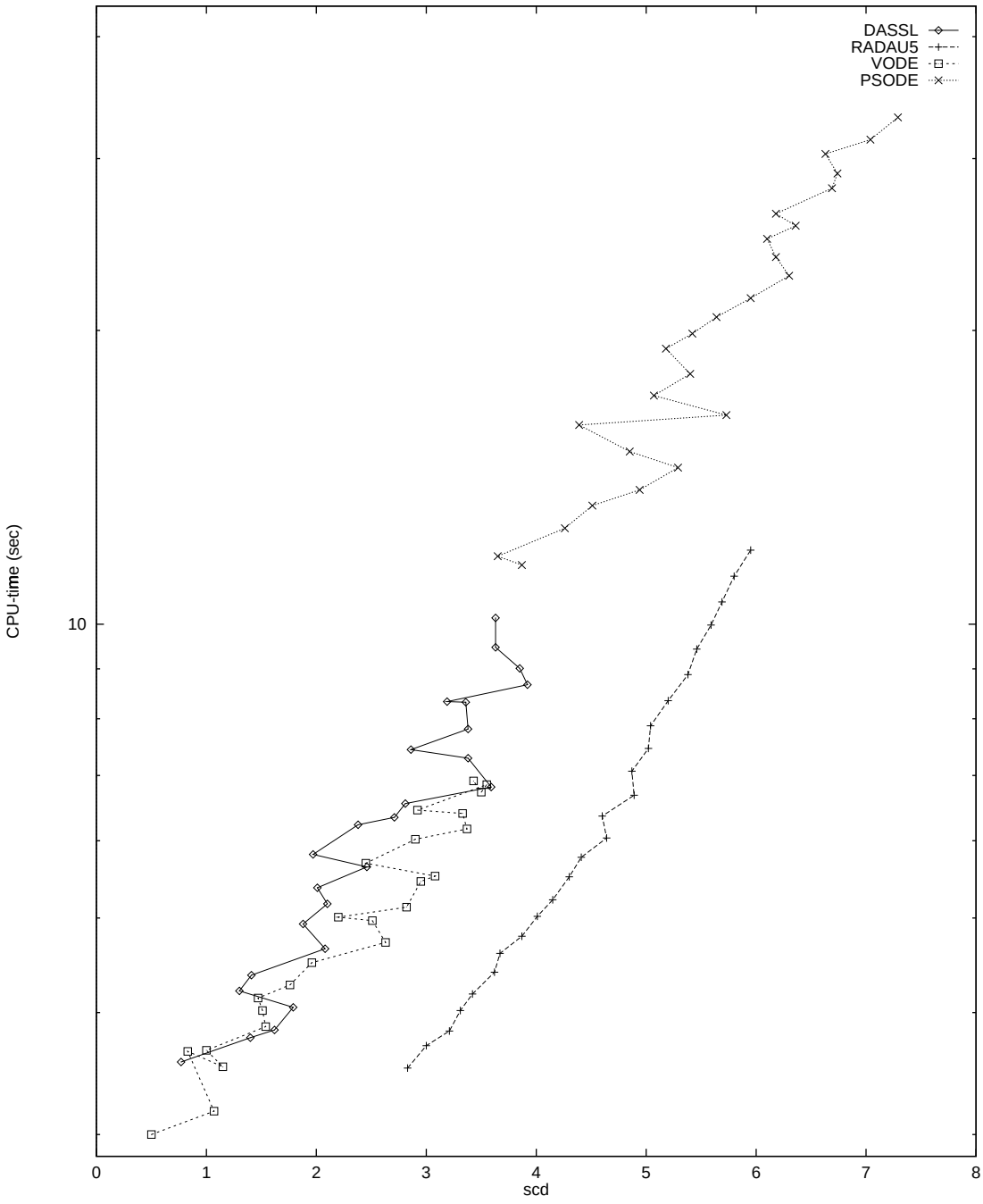


Figure 2: Work-precision diagram for Ring Modulator

5 Andrew's squeezing mechanism

5.1 General information

The problem is a non-stiff second order DAE of index 3, consisting of 7 differential and 6 algebraic equations. It has been promoted as a test problem by Giles [Gil78] and Manning [Man81]. The formulation here corresponds to the one presented in Hairer & Wanner [HW91]. The parallel-IVP-algorithm group of CWI contributed this problem to the test set.

5.2 Mathematical description of the problem

The problem is of the form

$$\begin{aligned} M(q)\ddot{q} &= f(q,\dot{q}) - G^T(q)\lambda, \\ 0 &= g(q), \end{aligned}$$

with initial conditions

$$q(0) = q_0,\ \dot{q}(0) = \dot{q}_0,\ \ddot{q}(0) = \ddot{q}_0,\ \lambda(0) = \lambda_0.$$

Here,

$$\begin{aligned} 0 &\le t \le 0.03, \\ q &\in \mathbf{R}^7, \\ \lambda &\in \mathbf{R}^6, \\ M &: \mathbf{R}^7 \to \mathbf{R}^{7\times 7}, \\ f &: \mathbf{R}^{14} \to \mathbf{R}^7, \\ g &: \mathbf{R}^7 \to \mathbf{R}^6, \\ G &= \frac{dg}{dq}. \end{aligned}$$

The function $M(q) = (M_{ij}(q))$ is given by:

$$\begin{aligned} M_{11}(q) &= m_1 \cdot ra^2 + m_2(rr^2 - 2da \cdot rr \cdot \cos q_2 + da^2) + I_1 + I_2, \\ M_{21}(q) &= M_{12}(q) = m_2(da^2 - da \cdot rr \cdot \cos q_2) + I_2, \\ M_{22}(q) &= m_2 \cdot da^2 + I_2, \\ M_{33}(q) &= m_3(sa^2 + sb^2) + I_3, \\ M_{44}(q) &= m_4(e - ea)^2 + I_4, \\ M_{54}(q) &= M_{45}(q) = m_4((e - ea)^2 + zt(e - ea)\sin q_4) + I_4, \\ M_{55}(q) &= m_4(zt^2 + 2zt(e - ea)\sin q_4 + (e - ea)^2) + m_5(ta^2 + tb^2) + I_4 + I_5, \\ M_{66}(q) &= m_6(zf - fa)^2 + I_6, \\ M_{76}(q) &= M_{67}(q) = m_6((zf - fa)^2 - u(zf - fa)\sin q_6) + I_6, \\ M_{77}(q) &= m_6((zf - fa)^2 - 2u(zf - fa)\sin q_6 + u^2) + m_7(ua^2 + ub^2) + I_6 + I_7, \\ M_{ij}(q) &= 0 \quad \text{for all other cases.} \end{aligned}$$

The function $f = (f_i(q,\dot{q}))$ reads:

$$\begin{aligned} f_1(q,\dot{q}) &= mom - m_2 \cdot da \cdot rr \cdot \dot{q}_2(\dot{q}_2 + 2\dot{q}_1)\sin q_2, \\ f_2(q,\dot{q}) &= m_2 \cdot da \cdot rr \cdot \dot{q}_1^2 \cdot \sin q_2, \\ f_3(q,\dot{q}) &= F_x(sc \cdot \cos q_3 - sd \cdot \sin q_3) + F_y(sd \cdot \cos q_3 + sc \cdot \sin q_3), \\ f_4(q,\dot{q}) &= m_4 \cdot zt(e - ea)\dot{q}_5^2 \cdot \cos q_4, \end{aligned}$$

$$
\begin{aligned}
f_5(q,\dot{q}) &= -m_4 \cdot zt(e-ea)\dot{q}_4(\dot{q}_4+2\dot{q}_5)\cos q_4,\\
f_6(q,\dot{q}) &= -m_6 \cdot u(zf-fa)\dot{q}_7^2 \cdot \cos q_6,\\
f_7(q,\dot{q}) &= m_6 \cdot u(zf-fa)\dot{q}_6(\dot{q}_6+2\dot{q}_7)\cos q_6.
\end{aligned}
$$

F_x and F_y are defined by:

$$
\begin{aligned}
F_x &= F(xd-xc),\\
F_y &= F(yd-yc),\\
F &= -c_0(L-l_0)/L,\\
L &= \sqrt{(xd-xc)^2+(yd-yc)^2},\\
xd &= sd\cdot\cos q_3 + sc\cdot\sin q_3 + xb,\\
yd &= sd\cdot\sin q_3 - sc\cdot\cos q_3 + yb.
\end{aligned}
$$

The function $g=(g_i(q))$ is given by:

$$
\begin{aligned}
g_1(q) &= rr\cdot\cos q_1 - d\cdot\cos(q_1+q_2) - ss\cdot\sin q_3 - xb,\\
g_2(q) &= rr\cdot\sin q_1 - d\cdot\sin(q_1+q_2) + ss\cdot\cos q_3 - yb,\\
g_3(q) &= rr\cdot\cos q_1 - d\cdot\cos(q_1+q_2) - e\cdot\sin(q_4+q_5) - zt\cdot\cos q_5 - xa,\\
g_4(q) &= rr\cdot\sin q_1 - d\cdot\sin(q_1+q_2) + e\cdot\cos(q_4+q_5) - zt\cdot\sin q_5 - ya,\\
g_5(q) &= rr\cdot\cos q_1 - d\cdot\cos(q_1+q_2) - zf\cdot\cos(q_6+q_7) - u\cdot\sin q_7 - xa,\\
g_6(q) &= rr\cdot\sin q_1 - d\cdot\sin(q_1+q_2) - zf\cdot\sin(q_6+q_7) + u\cdot\cos q_7 - ya.
\end{aligned}
$$

The constants arising in these formulas are given by:

m_1	=	0.04325	I_1	=	$2.194\cdot10^{-6}$	ss	=	0.035
m_2	=	0.00365	I_2	=	$4.410\cdot10^{-7}$	sa	=	0.01874
m_3	=	0.02373	I_3	=	$5.255\cdot10^{-6}$	sb	=	0.01043
m_4	=	0.00706	I_4	=	$5.667\cdot10^{-7}$	sc	=	0.018
m_5	=	0.07050	I_5	=	$1.169\cdot10^{-5}$	sd	=	0.02
m_6	=	0.00706	I_6	=	$5.667\cdot10^{-7}$	ta	=	0.02308
m_7	=	0.05498	I_7	=	$1.912\cdot10^{-5}$	tb	=	0.00916
xa	=	−0.06934	d	=	0.028	u	=	0.04
ya	=	−0.00227	da	=	0.0115	ua	=	0.01228
xb	=	−0.03635	e	=	0.02	ub	=	0.00449
yb	=	0.03273	ea	=	0.01421	zf	=	0.02
xc	=	0.014	rr	=	0.007	zt	=	0.04
yc	=	0.072	ra	=	0.00092	fa	=	0.01421
c_0	=	4530	l_0	=	0.07785	mom	=	0.033

The initial values are

$$
\begin{aligned}
q_0 &= \begin{pmatrix}
-0.061713890014276449635894845800 1\\
0\\
0.455279819163070380255912382449\\
0.222668390165885884674473185609\\
0.487364979543842550225598953530\\
-0.222668390165885884674473185609\\
1.230547444549821192497350155 68
\end{pmatrix},\\
\dot{q}_0 &= (0,0,0,0,0,0,0)^T,
\end{aligned}
$$

$$\ddot{q}_0 = \begin{pmatrix} 14222.443919954113870591162588 7 \\ -10666.832939965585402943371941 5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$\lambda_0 = \begin{pmatrix} 98.566870396241089605765498217 0 \\ -6.122688344255662655031143931 22 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

5.3 Origin of the problem

The problem describes the motion of 7 rigid bodies connected by joints without friction. See [Gil78] and [Man81] for more details.

5.4 Numerical solution of the problem

To show the behaviour of the solution we use the RADAU5 code of Hairer & Wanner [HW91]. Hence, we transform the problem to the form

$$\tilde{M}\frac{dy}{dt} = \tilde{f}(y), \quad y(0) = y_0,$$

with

$$y = \begin{pmatrix} q \\ \dot{q} \\ \ddot{q} \\ \lambda \end{pmatrix}, \quad \tilde{M} = \begin{pmatrix} I & O & O & O \\ O & I & O & O \\ O & O & O & O \\ O & O & O & O \end{pmatrix} \quad \text{and} \quad \tilde{f} = \begin{pmatrix} \dot{q} \\ \ddot{q} \\ M(q)\ddot{q} - f(q,\dot{q}) + G^T(q)\lambda \\ g(q) \end{pmatrix}.$$

This is the index 3 formulation. Replacing the function $g(q)$ arising in $\tilde{f}$ by $G(q)\dot{q}$ or $g_{qq}(q)(\dot{q},\dot{q}) + G(q)\ddot{q}$ yields the index 2 and index 1 formulation, respectively.

The Jacobian was approximated by

$$\begin{pmatrix} O & I & O & O \\ O & O & I & O \\ O & O & M & G^T \\ G & O & O & O \end{pmatrix}, \begin{pmatrix} O & I & O & O \\ O & O & I & O \\ O & O & M & G^T \\ O & G & O & O \end{pmatrix}, \begin{pmatrix} O & I & O & O \\ O & O & I & O \\ O & O & M & G^T \\ O & O & G & O \end{pmatrix},$$

for the index 3, 2 and 1 formulation, respectively. That is, the derivatives of $f(q,\dot{q})$ as well as those of $M(q)$ and $G(q)$ are neglected. Note that the evaluation of such a Jacobian is for free.

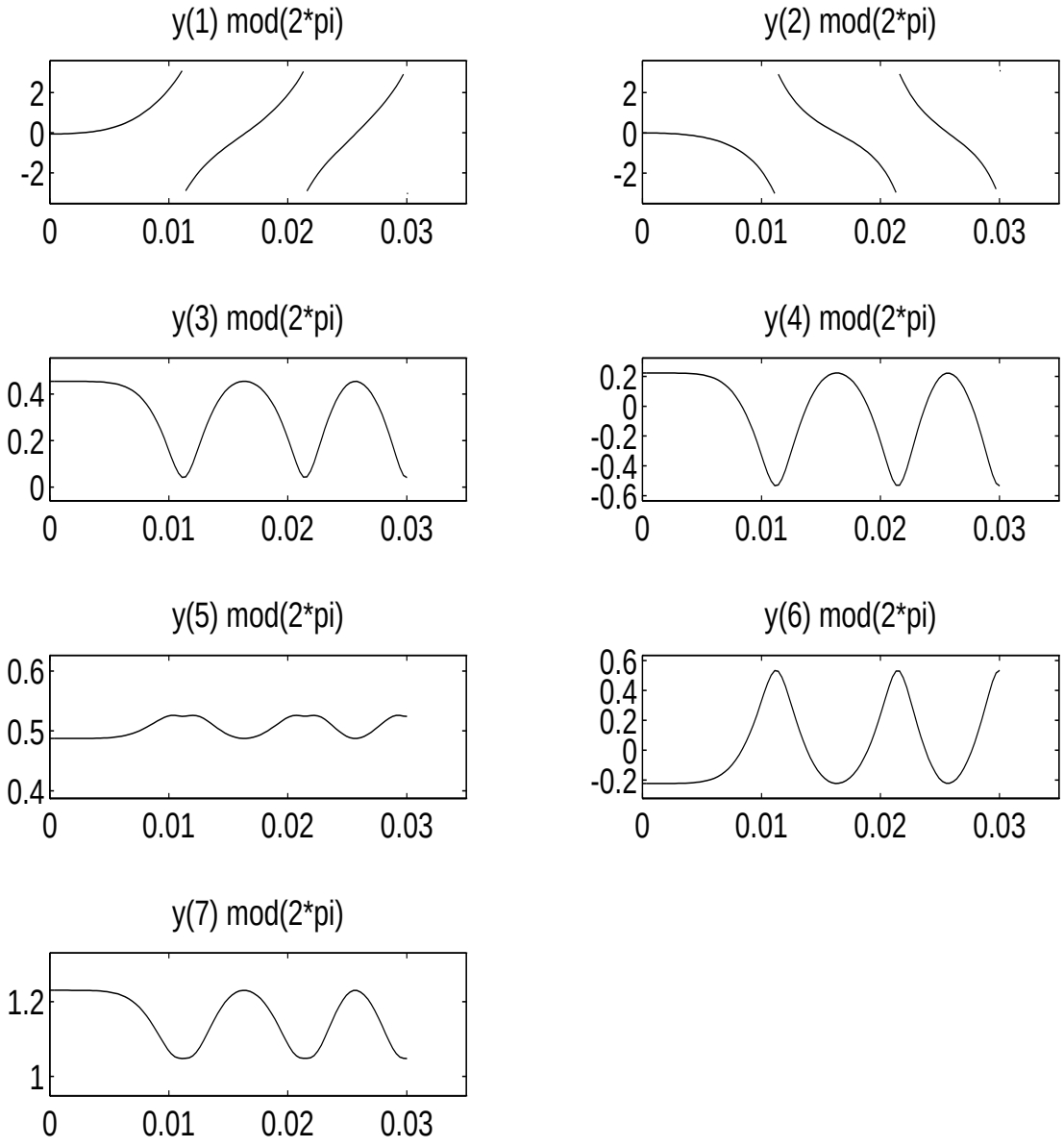
The RADAU5 input parameters IWORK(5), IWORK(6) and IWORK(7) were set equal to 7, 7 and 13 respectively. This means that the first 7 variables are of index 1, the next 7 of index 2 and the last 13 of index 3. See [HLR80] for a justification.

5.4.1 Values of the first 7 components at $t = 0.03$:

y_1	$0.1581077 \cdot 10^2$
y_2	$-0.1575637 \cdot 10^2$
y_3	$0.4082224 \cdot 10^{-1}$
y_4	-0.5347301
y_5	0.5244100
y_6	0.5347301
y_7	$0.1048081 \cdot 10$

5.4.2 Behaviour of the solution

The following plots show the behaviour of the first 7 solution components mod2π:



5.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. They correspond to the index 3 formulation.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
RADAU5	10^{-4}	10^{-4}	10^{-4}	2.39	80	54	703	53	80	0.50
	10^{-7}	10^{-7}	10^{-7}	3.83	171	162	1431	154	170	1.16

5.4.4 Work-precision diagram

In Figure 1 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol $= 10^{-(4+m/8)}$, $m = 0, \ldots, 24$; atol = rtol; h0 = rtol.

References

[Gil78] D. R. A. Giles. An algebraic approach to A-stable linear multistep-multiderivative integration formulas. *BIT*, 14:382–406, 1978.

[HLR80] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge–Kutta Methods.* Lecture Notes in Mathematics 1409. Springer-Verlag, 1980.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[Man81] D. W. Manning. *A computer technique for simulating dynamic multibody systems based on dynamic formalism.* PhD thesis, Univ. Waterloo, Ontario, 1981.

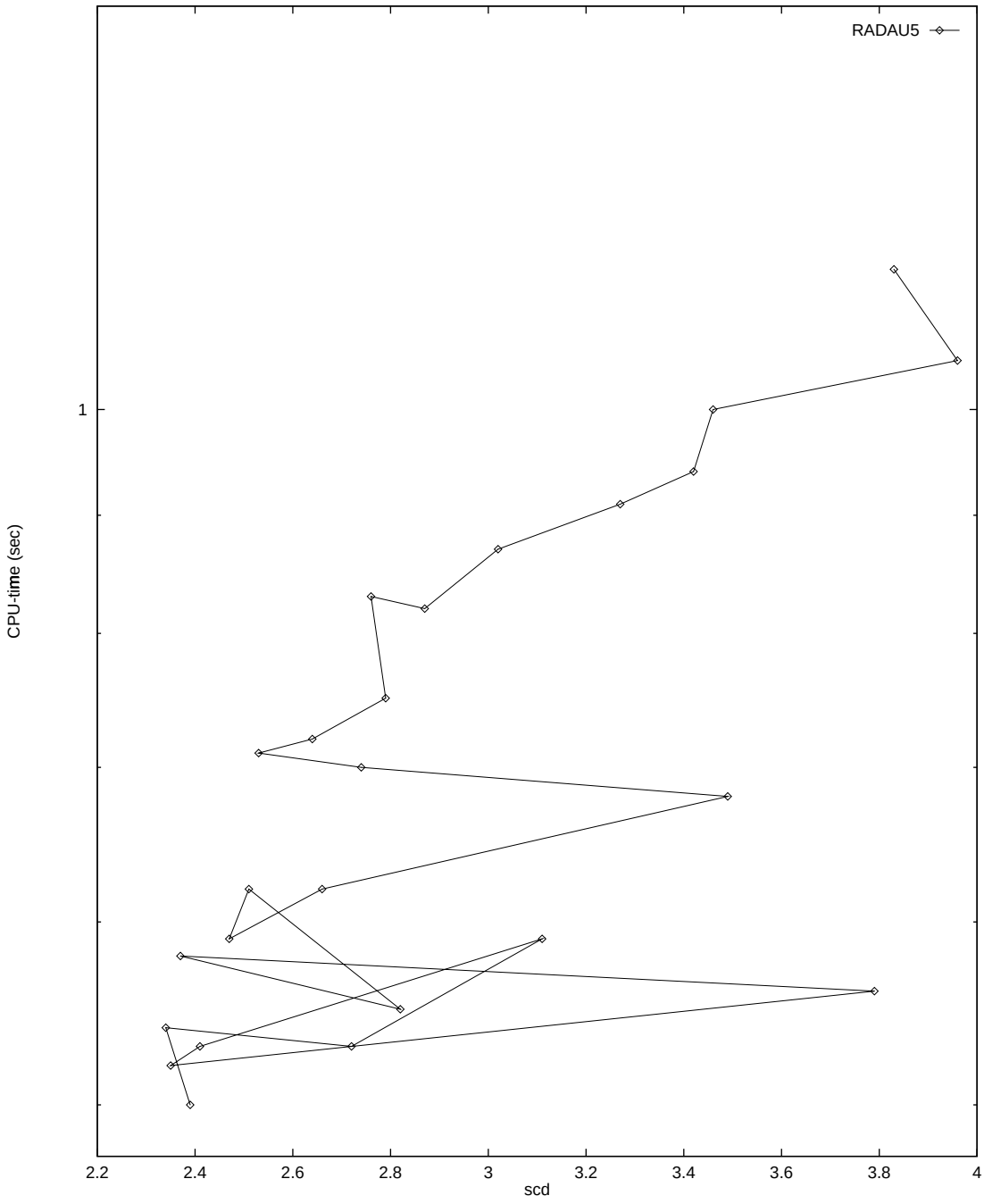


Figure 1: Work-precision diagram for Andrew's squeezing mechanism

6 Transistor amplifier

6.1 General information

The problem is a stiff DAE of index 1 consisting of 8 equations and is of the form $My' = f(y)$ with M a matrix of rank 5. P. Rentrop has received it from K. Glashoff & H.J. Oberle and has documented it in [RRS89]. The formulation presented here has been taken from [HLR80]. The parallel-IVP-algorithm group of CWI contributed this problem to the test set.

6.2 Mathematical description of the problem

The problem is of the form

$$G(t, y, y') = 0, \;\; y(0) = y_0, \; y'(0) = y'_0,$$

with

$$y \in \mathbf{R}^8, \;\; 0 \le t \le 0.2.$$

The function G is defined by

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

where the matrix M is given by

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

and the function f by

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

where g and U_e are auxiliary functions given by

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

The values of the technical parameters are:

$$\begin{aligned} U_b &= 6, \\ U_F &= 0.026, \\ \alpha &= 0.99, \\ \beta &= 10^{-6}, \\ R_0 &= 1000, \\ R_k &= 9000 \quad \text{for} \quad k = 1, \ldots, 9, \\ C_k &= k \cdot 10^{-6} \quad \text{for} \quad k = 1, \ldots, 5. \end{aligned}$$

Consistent initial values at t=0 are

$$\begin{aligned} y_1(0) &= 0, & y_1'(0) &= 51.338775 \\ y_2(0) &= U_b/(\tfrac{R_2}{R_1}+1), & y_2'(0) &= y_1'(0) \\ y_3(0) &= y_2(0), & y_3'(0) &= -y_2(0)/(C_2 \cdot R_3) \\ y_4(0) &= U_b, & y_4'(0) &= -24.9757667 \\ y_5(0) &= U_b/(\tfrac{R_6}{R_5}+1), & y_5'(0) &= y_4'(0) \\ y_6(0) &= y_5(0), & y_6'(0) &= -y_5(0)/(C_4 \cdot R_7) \\ y_7(0) &= U_b, & y_7'(0) &= -10.00564453 \\ y_8(0) &= 0, & y_8'(0) &= y_7'(0) \end{aligned}$$

The initial values $y_1'(0)$, $y_4'(0)$ and $y_7'(0)$ were determined numerically.

6.3 Origin of the problem

The problem originates from electrical circuit analysis. It is a model for the transistor amplifier. The diagram of the circuit is given below:

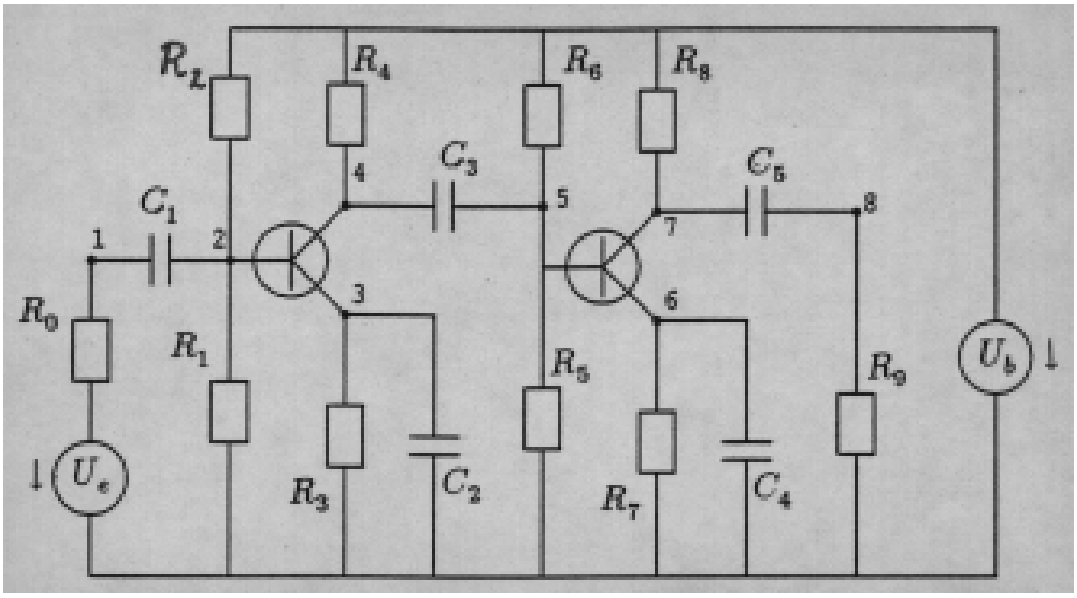


Figure 1: *Circuit diagram of Transistor Amplifier (taken from [HLR80])*

Here U_e is the initial signal and U_8 is the amplified exit voltage. To formulate the governing equations, Kirchoff's Current Law is used in each numbered node. This law states that the total sum of all currents entering a node must be zero. All currents passing through the circuit components can be expressed in terms of the unknown voltages $U_1, \ldots, U_8$. Consider for instance node 1. The current I_{C_1} passing through capacitor C_1 is given by

$$I_{C_1} = \frac{d}{dt}(C_1(U_2 - U_1)),$$

and the current I_{R_0} passing through the resistor R_0 by

$$I_{R_0} = \frac{U_e - U_1}{R_0}.$$

Here, the currents are directed towards node 1 if the current is positive. A similar derivation for the other nodes gives the system:

$$\begin{aligned}
\text{node 1:} \quad & \tfrac{d}{dt}(C_1(U_2 - U_1)) + \tfrac{U_e(t)}{R_0} - \tfrac{U_1}{R_0} && = 0,\\
\text{node 2:} \quad & \tfrac{d}{dt}(C_1(U_1 - U_2)) + \tfrac{U_b}{R_2} - U_2(\tfrac{1}{R_1} + \tfrac{1}{R_2}) + (\alpha - 1)g(U_2 - U_3) && = 0,\\
\text{node 3:} \quad & -\tfrac{d}{dt}(C_2 U_3) + g(U_2 - U_3) - \tfrac{U_3}{R_3} && = 0,\\
\text{node 4:} \quad & -\tfrac{d}{dt}(C_3(U_4 - U_5)) + \tfrac{U_b}{R_4} - \tfrac{U_4}{R_4} - \alpha g(U_2 - U_3) && = 0,\\
\text{node 5:} \quad & \tfrac{d}{dt}(C_3(U_4 - U_5)) + \tfrac{U_b}{R_6} - U_5(\tfrac{1}{R_5} + \tfrac{1}{R_6}) + (\alpha - 1)g(U_5 - U_6) && = 0,\\
\text{node 6:} \quad & -\tfrac{d}{dt}(C_4 U_6) + g(U_5 - U_6) - \tfrac{U_6}{R_7} && = 0,\\
\text{node 7:} \quad & -\tfrac{d}{dt}(C_5(U_7 - U_8)) + \tfrac{U_b}{R_8} - \tfrac{U_7}{R_8} - \alpha g(U_5 - U_6) && = 0,\\
\text{node 8:} \quad & -\tfrac{d}{dt}(C_5(U_7 - U_8)) + \tfrac{U_8}{R_9} && = 0,
\end{aligned}$$

where

$$g(U_i - U_j) = \beta(e^{\frac{U_i - U_j}{U_F}} - 1)$$

is a simple model of the transistors. The initial signal $U_e(t)$ is

$$U_e(t) = 0.1\sin(200\pi t).$$

To arrive at the mathematical formulation of the preceding subsection, one just has to identify U_i with y_i.

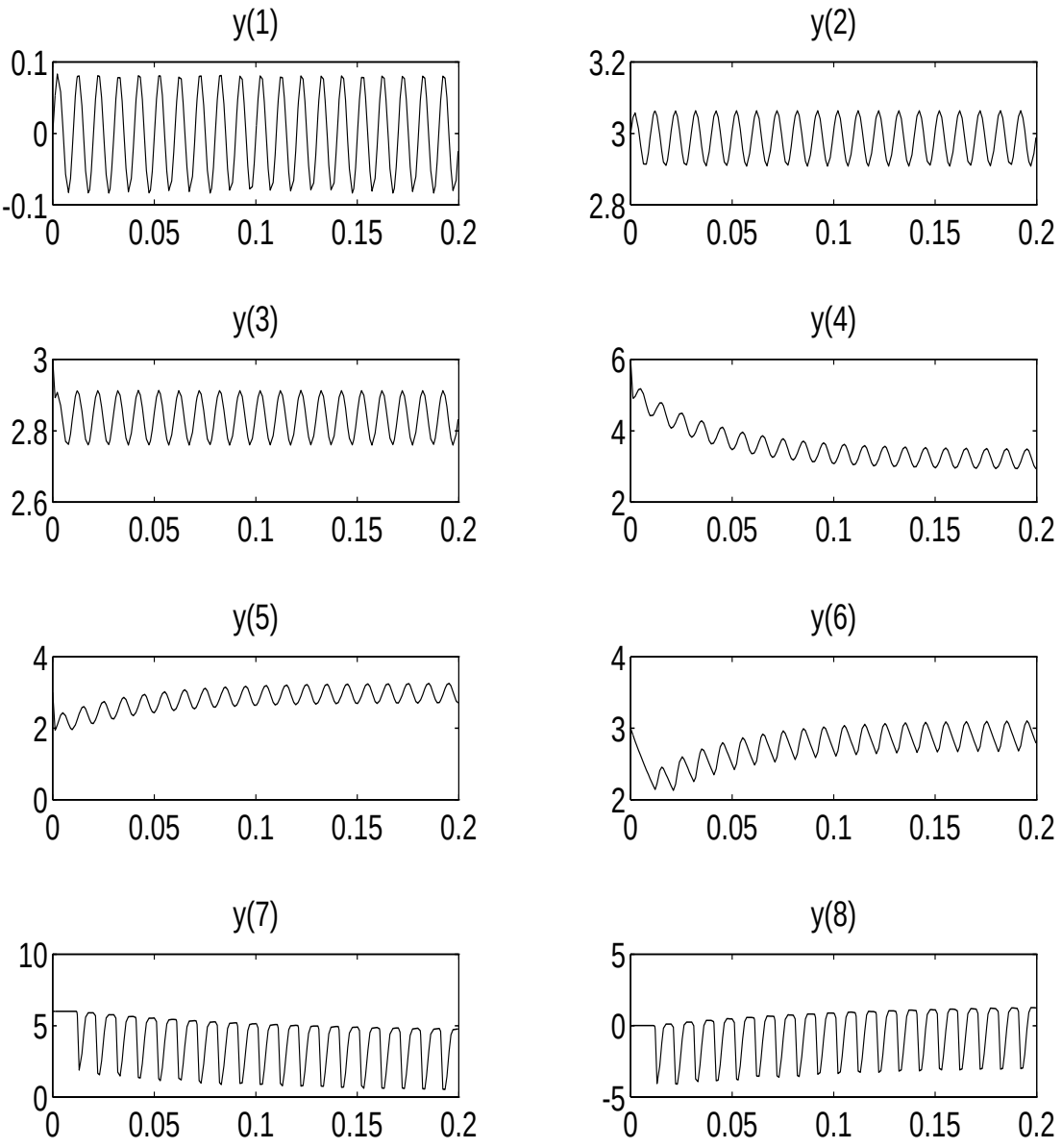
6.4 Numerical solution of the problem

6.4.1 Solution at $t = 0.2$:

y_1	$-0.556214501 \cdot 10^{-2}$
y_2	$0.3006522473125 \cdot 10$
y_3	$0.284995878984 \cdot 10$
y_4	$0.29264225362 \cdot 10$
y_5	$0.27046178656 \cdot 10$
y_6	$0.2761837776452 \cdot 10$
y_7	$0.4770927631 \cdot 10$
y_8	$0.1236995867 \cdot 10$

6.4.2 Behaviour of the numerical solution

The following plots show the behaviour of the solution components:



6.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		2.91	6003	2340	18200	7212		3.97
	10^{-7}	10^{-7}		4.56	33008	6531	115181	52725		25.32
RADAU5	10^{-7}	10^{-7}	10^{-9}	8.35	3196	3066	25021	3047	3187	2.34

6.4.4 Work-precision diagram

In Figure 2 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol $= 10^{-(4+m/8)}$, $m = 0, \ldots, 24$ for DASSL and $m = 8, \ldots, 24$ for RADAU5; atol = rtol; h0 $= 10^{-2} \cdot$ rtol for RADAU5.

References

[HLR80] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge–Kutta Methods.* Lecture Notes in Mathematics 1409. Springer-Verlag, 1980.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[RRS89] P. Rentrop, M. Roche, and G. Steinebach. The application of Rosenbrock-Wanner type methods with stepsize control in differential-algebraic equations. *Numer. Math.*, 55:545–563, 1989.

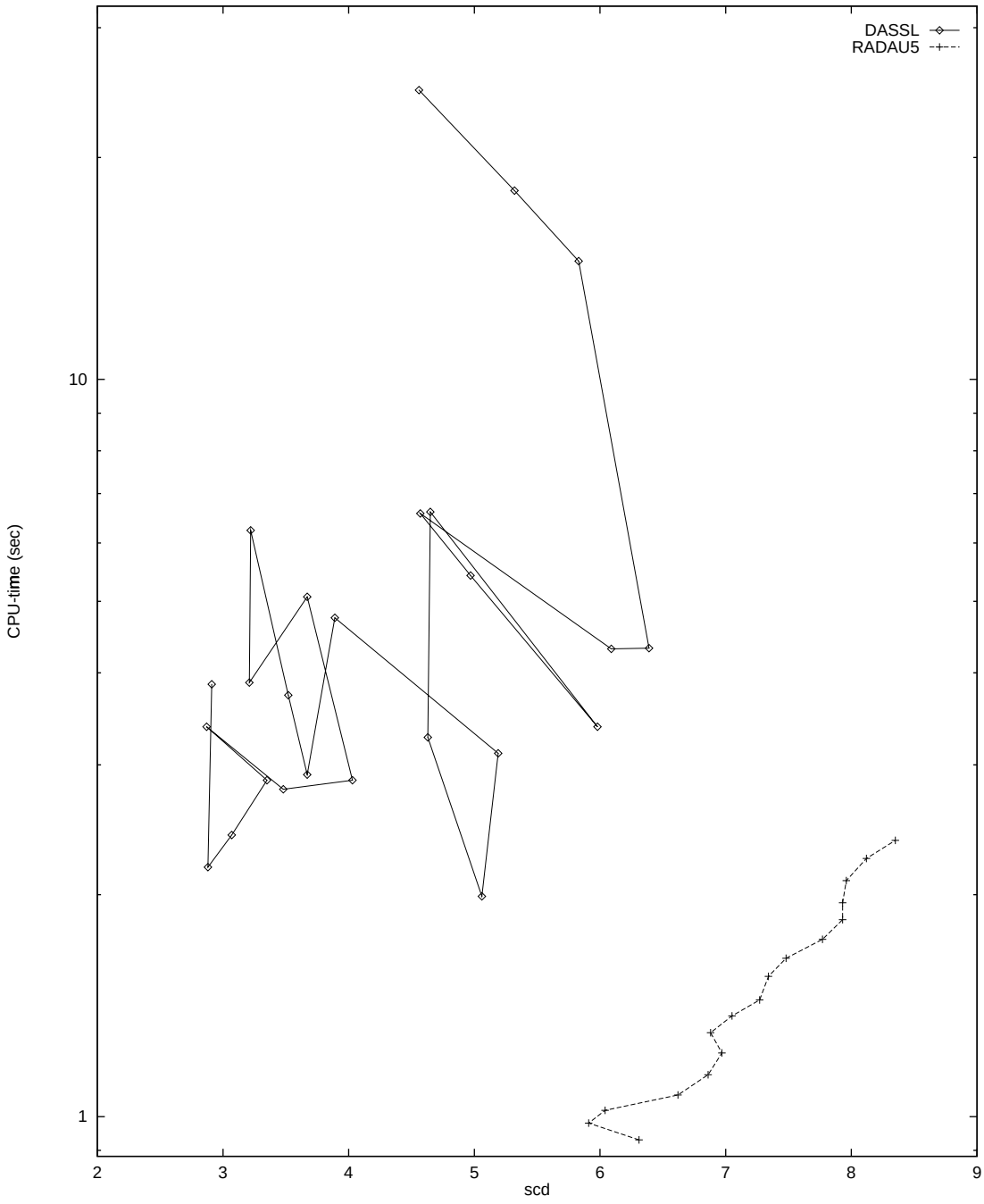


Figure 2: Work-precision diagram for Transistor Amplifier

7 Medical Akzo Nobel problem

7.1 General information

The problem consists of 2 partial differential equations. Semi-discretisation of this system yields a stiff ODE. The parallel-IVP-algorithm group of CWI contributed this problem to the test set in collaboration with R. van der Hout from Akzo Nobel Central Research.

7.2 Mathematical description of the problem

The problem is of the form

$$\frac{dy}{dt} = f(t,y), \quad y(0) = g,$$

with

$$y \in \mathbf{R}^{2N}, \quad 0 \le t \le 20.$$

Here, the integer N is an user-supplied parameter. The function f is given by

$$\begin{aligned} f_{2j-1} &= \alpha_j \frac{y_{2j+1} - y_{2j-3}}{2\Delta\zeta} + \beta_j \frac{y_{2j-3} - 2y_{2j-1} + y_{2j+1}}{(\Delta\zeta)^2} - k\, y_{2j-1} y_{2j}, \\ f_{2j} &= -k\, y_{2j} y_{2j-1}, \end{aligned}$$

where

$$\begin{aligned} \alpha_j &= \frac{2(j\Delta\zeta - 1)^3}{c^2}, \\ \beta_j &= \frac{(j\Delta\zeta - 1)^4}{c^2}. \end{aligned}$$

Here, j ranges from 1 to N, $\Delta\zeta = \frac{1}{N}$, $y_{-1}(t) = \phi(t)$, $y_{2N+1} = y_{2N-1}$ and $g \in \mathbf{R}^{2N}$ is given by

$$g = (0, v_0, 0, v_0, \ldots, 0, v_0)^T.$$

For the function ϕ we chose

$$\phi(t) = \begin{cases} 2 & \text{for} \quad t \in (0,5], \\ 0 & \text{for} \quad t \in (5,20]. \end{cases}$$

Suitable values for the parameters k, v_0 and c are 100, 1 and 4, respectively.

7.3 Origin of the problem

The Akzo Nobel research laboratories formulated this problem in their study of the penetration of radio-labeled antibodies into tumorous tissue [Hou94]. This study was carried out for diagnostic as well as therapeutic purposes.

Let us consider a reaction diffusion system in one spatial dimension:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - kuv, \tag{1}$$

$$\frac{\partial v}{\partial t} = -kuv, \tag{2}$$

which originates from the chemical reaction

$$A + B \xrightarrow{k} C.$$

Here A reacts with a substrate B and k denotes the rate constant. The concentrations of A and B are denoted by u and v, respectively. In the derivation of the equations (1) and (2) it was assumed that the reaction is governed by mass action kinetics and in addition that the chemical A is mobile while B is immobile.

Consider a clean semi-infinite slab, in which the substrate B is uniformly distributed. When the slab is exposed at its surface to the chemical A, this chemical starts to penetrate into the slab.

To model this penetration, the equations (1) and (2) are considered in the strip

$$S_T = \{(x,t) : 0 < x < \infty,\ 0 < t < T\} \text{ for some } T,$$

along with the following initial and boundary conditions:

$$u(x,0) = 0,\ \ v(x,0) = v_0 \ \text{ for } \ x > 0,$$

where v_0 is a constant, and

$$u(0,t) = \phi(t) \ \text{ for } \ 0 < t < T.$$

In order to solve the problem numerically, we transform the variable x in such a way that the semi-infinite slab is transformed into a finite one. A suitable transformation is provided by the following special family of Möbius transformations:

$$\zeta = \frac{x}{x+c}, \text{ with } c > 0.$$

Each transformation in this class transforms S_T into the slab:

$$\{(\zeta,t) : 0 < \zeta < 1,\ 0 < t < T\}.$$

In terms of ζ the problem now reads:

$$\frac{\partial u}{\partial t} = \frac{(\zeta-1)^4}{c^2}\frac{\partial^2 u}{\partial \zeta^2} + \frac{2(\zeta-1)^3}{c^2}\frac{\partial u}{\partial \zeta} - kuv, \tag{3}$$

$$\frac{\partial v}{\partial t} = -kuv, \tag{4}$$

with initial conditions

$$u(\zeta,0) = 0,\ \ v(\zeta,0) = v_0 \ \text{ for } \ \zeta > 0, \tag{5}$$

and boundary conditions

$$u(0,t) = \phi(t),\ \ \frac{\partial u}{\partial \zeta}(1,t) = 0 \ \text{ for } \ 0 < t < T. \tag{6}$$

The last boundary condition is derived from $\frac{\partial u}{\partial x}(\infty,t) = 0$.

The system consisting of (3), (4), (5) and (6) will be written as a system of ordinary differential equations by using the method of lines, i.e. by discretizing the spatial derivatives. We use the uniform grid $\{\zeta_j\}_{j=1,\ldots,N}$ defined by:

$$\zeta_j = j \cdot \Delta\zeta, \quad j = 1,\ldots,N, \quad \Delta\zeta = \frac{1}{N}.$$

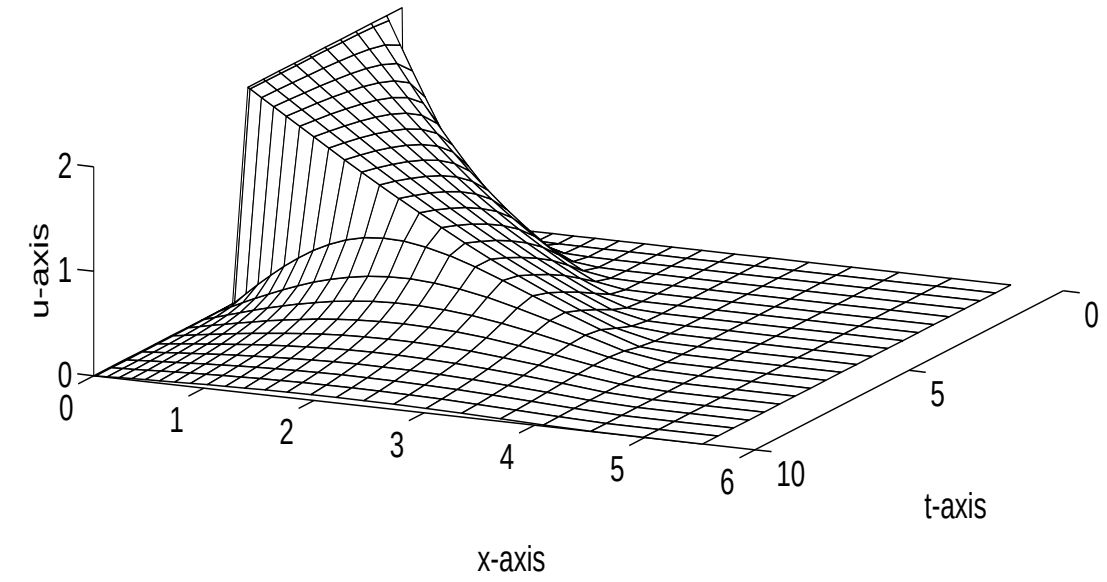
Let u_j and v_j denote the approximations of $u(\zeta_j, t)$ and $v(\zeta_j, t)$, respectively. Obviously, u_j and v_j are functions of t. In terms of the function u_j, our choices for the discretisation of the spatial first and second order derivatives read

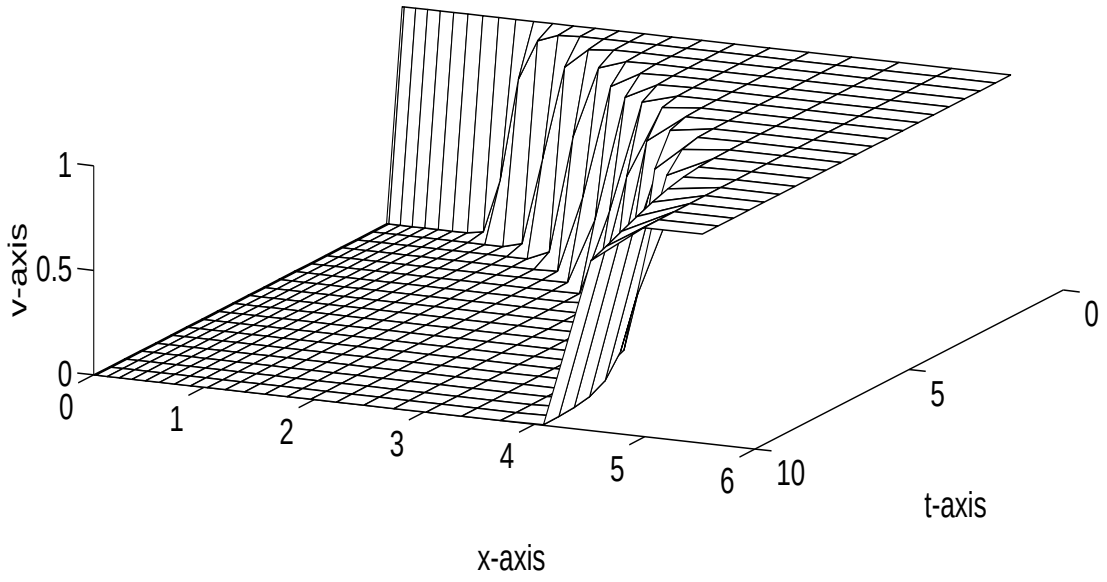
$$\frac{\partial u_j}{\partial \zeta} = \frac{u_{j+1} - u_{j-1}}{2\Delta\zeta} \quad \text{and} \quad \frac{\partial^2 u_j}{\partial \zeta^2} = \frac{u_{j-1} - 2u_j + u_{j+1}}{(\Delta\zeta)^2},$$

respectively, where $j = 1, \ldots, N$. Suitable values for u_0 and u_{N+1} are obtained from the boundary conditions. They are given by $u_0 = \phi(t)$ and $u_{N+1} = u_N$.

Defining $y(t)$ by $y = (u_1, v_1, u_2, v_2, \ldots, u_N, v_N)^T$, and choosing $T = 20$, this semi-discretised problem is precisely the ODE described in Subsection 7.2.

To give an idea of the solution to the PDE (3) – (6) we give plots of u and v as function of x and t:





From these plots we nicely see that injection of chemical A destroys (temporarily) B.

7.4 Numerical solution of the problem

The numerical experiments were done for the case $N = 200$.

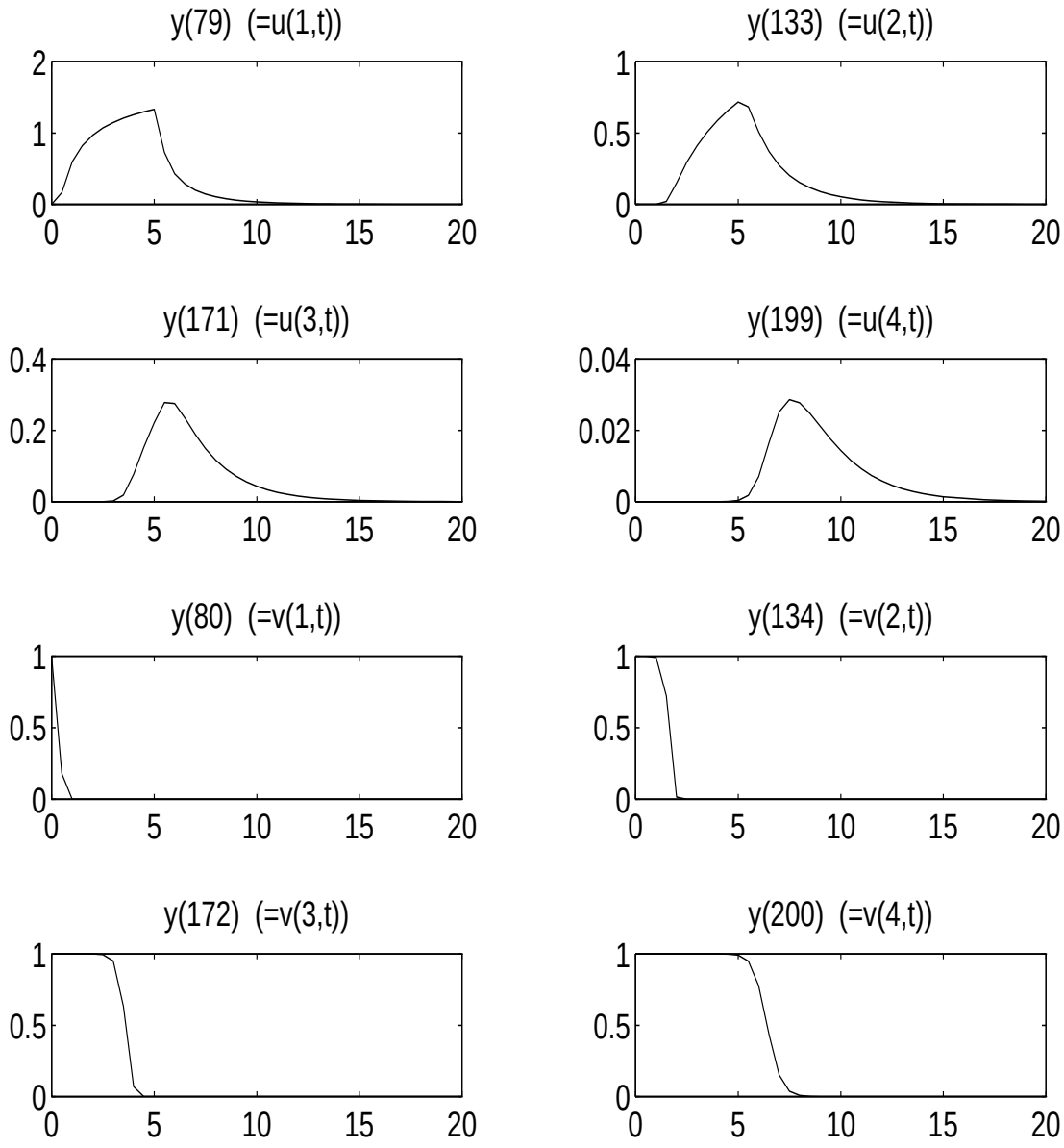
7.4.1 Solution at $t = 20$:

We only give the value of some components of y in the endpoint. For the complete reference solution we refer to the Fortran subroutine `solut`. The components listed in the Table below correspond to the values of u and v in $x = 1, 2.4, 4.0$ and 6.0.

y_{79}	$=$	$0.23399422318936 \cdot 10^{-3}$	y_{80}	$=$	0.00000000000000
y_{149}	$=$	$0.35956160302716 \cdot 10^{-3}$	y_{150}	$=$	0.00000000000000
y_{199}	$=$	$0.11737412926802 \cdot 10^{-3}$	y_{200}	$=$	$0.61908071460151 \cdot 10^{-5}$
y_{239}	$=$	$0.68600948191191 \cdot 10^{-11}$	y_{240}	$=$	0.99999973258552

7.4.2 Behaviour of the numerical solution

The following plots show the behaviour of the solution components y_i for $i \in \{79, 80, 133, 134, 171, 172, 199, 200\}$, which correspond to approximations of the PDE solutions u and v on the grid lines $x = 1$, $x = 2$, $x = 3$ and $x = 4$:



7.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. Since some solution components are zero, the scd values listed here denote the absolute precision.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		3.35	386	372	604	59		3.66
	10^{-7}	10^{-7}		5.55	1322	1303	1799	77		11.31
RADAU5	10^{-4}	10^{-4}	10^{-9}	4.16	124	110	841	74	118	2.53
	10^{-7}	10^{-7}	10^{-11}	7.22	546	525	3306	338	419	9.79
VODE	10^{-4}	10^{-4}		2.88	414	395	615	10	77	2.54
	10^{-7}	10^{-7}		4.78	1025	993	1262	18	128	5.59
PSODE	10^{-4}	10^{-4}	10^{-9}	6.08	240	231	5447	15	412	20.41
	10^{-7}	10^{-7}	10^{-11}	8.73	819	807	18059	10	512	63.60

The speed-up factors of PSODE on the Cray C98 are, in order of decreasing rtol: 3.4, 3.4.

7.4.4 Work-precision diagram

In Figure 1 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol $= 10^{-(4+m/8)}$, $m = 0, \ldots, 24$; atol = rtol; h0 $= 10^{-5} \cdot$ rtol for RADAU5 and PSODE.

References

[Hou94] R. van der Hout, 1994. Private communication.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

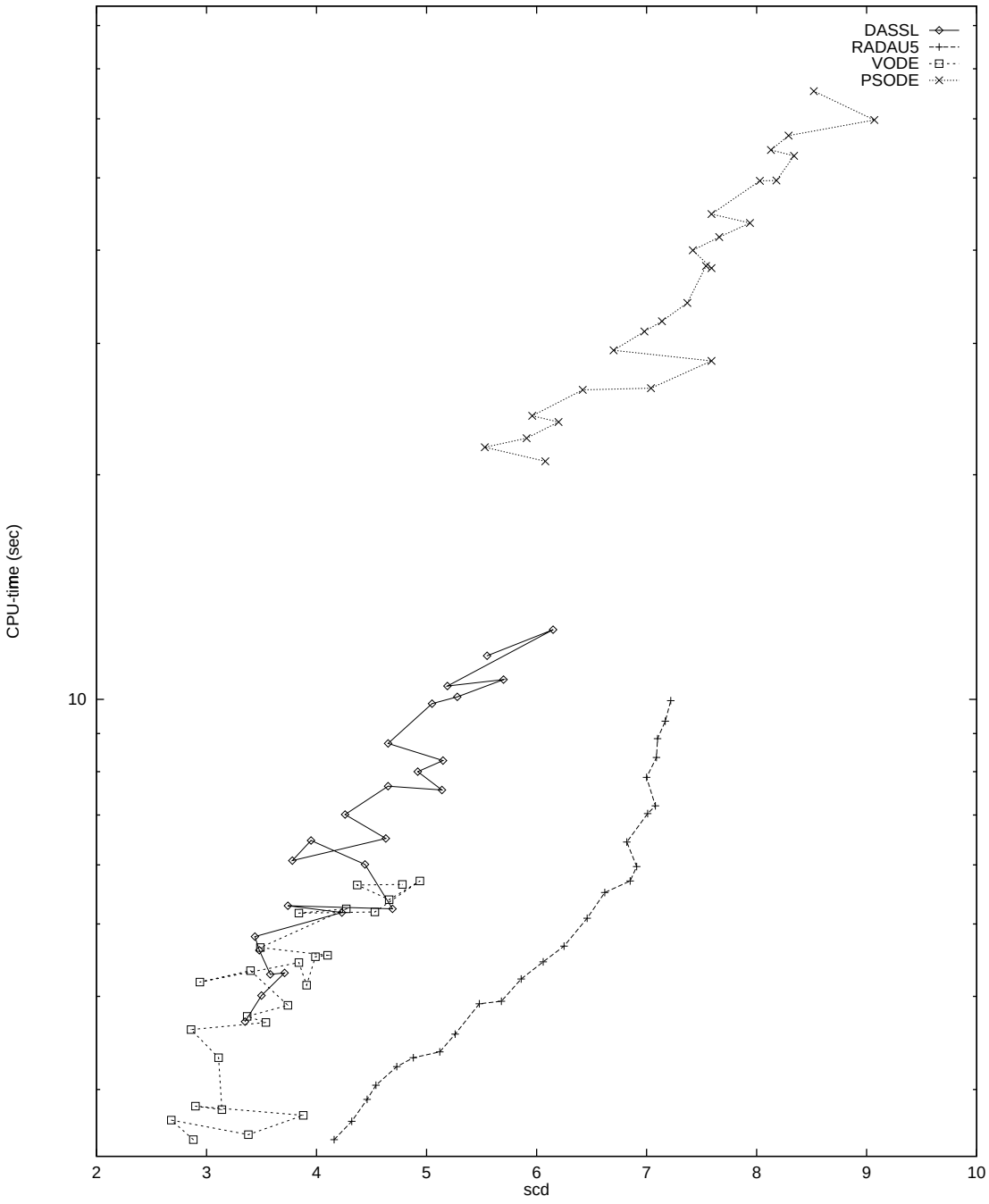


Figure 1: Work-precision diagram for Medical Akzo Nobel problem

8 EMEP problem

8.1 General information

The problem is a stiff system of 66 ordinary differential equations. The 'Mathematics and the Environment' project group at CWI contributed this problem to the test set.

8.2 Mathematical description of the problem

The problem is of the form

$$\frac{dy}{dt} = f(t,y), \quad y(0) = g,$$

with

$$y \in \mathbf{R}^{66}, \quad 14400 \leq t \leq 417600.$$

The initial vector $g = (g_i)$ is given by

$$g_i = \begin{cases} 1.0 \cdot 10^9 & \text{for} \quad i = 1 \\ 5.0 \cdot 10^9 & \text{for} \quad i \in \{2,3\} \\ 3.8 \cdot 10^{12} & \text{for} \quad i = 4 \\ 3.5 \cdot 10^{13} & \text{for} \quad i = 5 \\ 1.0 \cdot 10^7 & \text{for} \quad i \in \{6,7,\ldots,13\} \\ 5.0 \cdot 10^{11} & \text{for} \quad i = 14 \\ 1.0 \cdot 10^2 & \text{for} \quad i \in \{15,16,\ldots,37\} \\ 1.0 \cdot 10^{-3} & \text{for} \quad i = 38 \\ 1.0 \cdot 10^2 & \text{for} \quad i \in \{39,40,\ldots,66\} \end{cases}$$

The function f is too voluminous to be described here. We refer to the Fortran subroutine `feval` and to [VS94] to get more insight in the function.

8.3 Origin of the problem

The problem is the chemistry part of the EMEP MSC-W ozone chemistry model, which is in development at the Norwegian Meteorological Institute in Oslo, Norway. About 140 reactions with a total of 66 species are involved. Below we give the correspondence between the solution vector y and the chemical species.

$y = ($	NO,	NO2,	SO2,	CO,	CH4,	C2H6,
	NC4H10,	C2H4,	C3H6,	OXYL,	HCHO,	CH3CHO,
	MEK,	O3,	HO2,	HNO3,	H2O2,	H2,
	CH3O2,	C2H5OH,	SA,	CH3O2H,	C2H5O2,	CH3COO,
	PAN,	SECC4H,	MEKO2,	R2OOH,	ETRO2,	MGLYOX,
	PRRO2,	GLYOX,	OXYO2,	MAL,	MALO2,	OP,
	OH,	OD,	NO3,	N2O5,	ISOPRE,	NITRAT,
	ISRO2,	MVK,	MVKO2,	CH3OH,	RCO3H,	OXYO2H,
	BURO2H,	ETRO2H,	PRRO2H,	MEKO2H,	MALO2H,	MACR,
	ISNI,	ISRO2H,	MARO2,	MAPAN,	CH2CCH3,	ISONO3,
	ISNIR,	MVKO2H,	CH2CHR,	ISNO3H,	ISNIRH,	MARO2H $)^T$.

The integration interval covers 112 hours. Rate coefficients are often variable. E.g., photolysis rates obviously depend on solar elevation and cloudiness, and undergo a discontinuity at sunrise and sunset, as can be seen from the plots in the next section. The unit of the species is number of molecules per cm^3, the time t is in seconds. The test problem corresponds to the rural case in [VS94].

A more elaborate description of the model can be found in [VS94], [Sim93] and [SASJ93].

8.4 Numerical solution of the problem

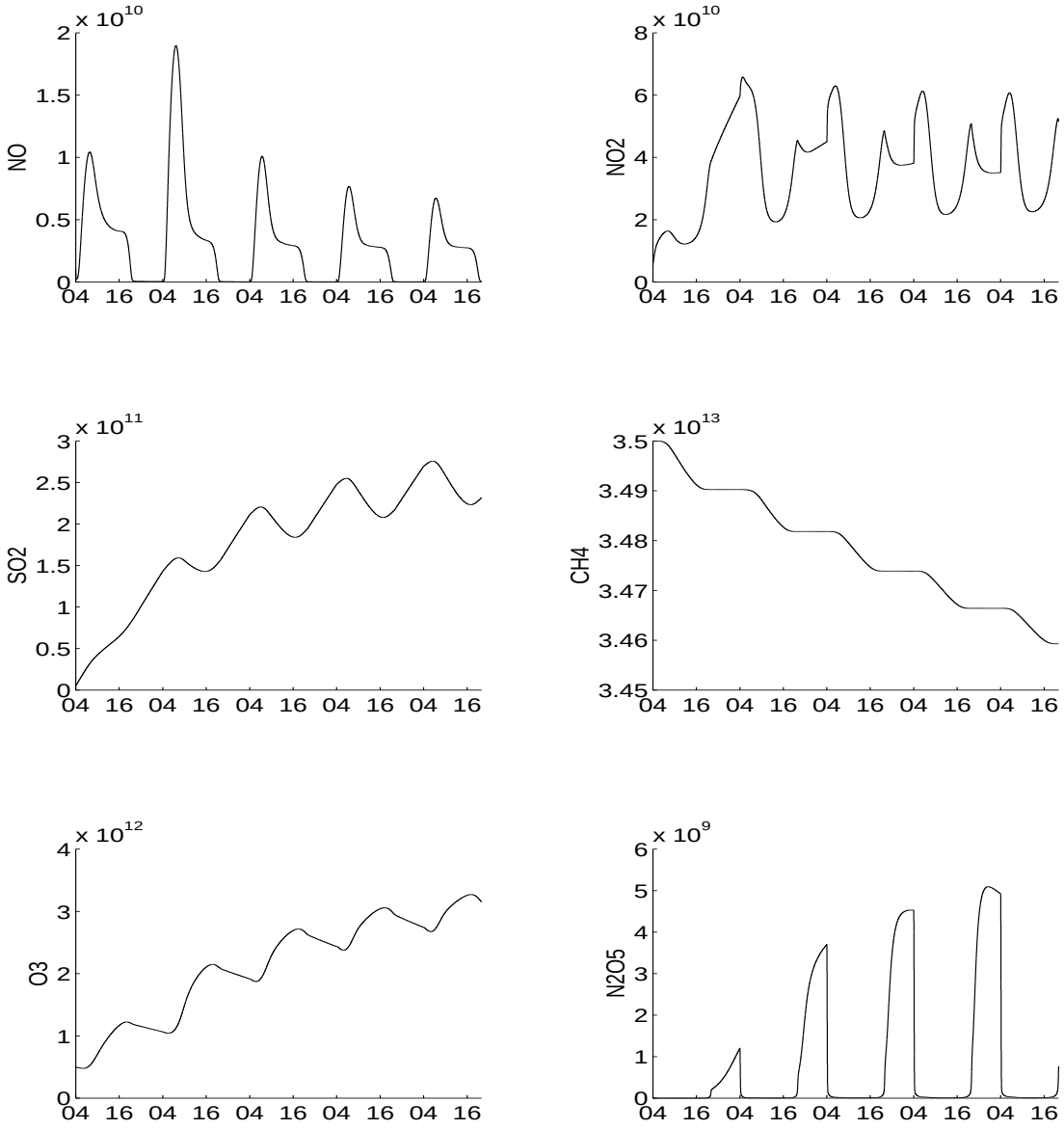
8.4.1 Solution at the endpoint

We only give the value at the endpoint $t = 417600$ of the components of y corresponding to NO, NO2, SO2, CH4, O3 and N2O5 (i.e. y_1, y_2, y_3, y_5, y_{14} and y_{40}). For the complete reference solution we refer to the Fortran subroutine `solut`.

NO	=	$0.25645805093601 \cdot 10^{8}$	NO2	=	$0.51461347708556 \cdot 10^{11}$
SO2	=	$0.23156799577319 \cdot 10^{12}$	CH4	=	$0.34592853260350 \cdot 10^{14}$
O3	=	$0.31503085853931 \cdot 10^{13}$	N2O5	=	$0.76845966195032 \cdot 10^{9}$

8.4.2 Behaviour of the numerical solution

We restrict ourselves to plots of the same species as mentioned in Section 8.4.1. The values at the horizontal axis denote the time t in hours modulo 24 hours.



8.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. Since components y_{36} and y_{38} are relatively very small and physically unimportant, we did not include these components in the computation of the scd value.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-2}	1		1.65	668	609	1380	161		4.53
	10^{-4}	1		3.24	1824	1727	3303	257		9.61
	10^{-6}	1		5.17	4356	4185	7340	432		20.06
RADAU5	10^{-2}	1	10^{-7}	2.74	303	217	3011	201	300	15.20
	10^{-4}	1	10^{-7}	3.16	615	496	5157	439	593	29.11
	10^{-6}	1	10^{-7}	5.20	1547	1395	10592	1243	1464	69.21
VODE	10^{-2}	1		1.00	731	680	1331	72	204	4.27
	10^{-4}	1		2.58	2015	1908	3373	66	316	8.67
	10^{-6}	1		4.57	4082	3816	6195	87	651	16.87
PSODE	10^{-2}	10^{-2}	10^{-7}	2.40	301	298	8890	139	1308	27.76
	10^{-4}	10^{-4}	10^{-7}	4.78	530	499	15967	193	2048	46.48

The speed-up factors of PSODE on the Cray C98 are, in order of decreasing rtol: 3.7, 3.6.

8.4.4 Work-precision diagram

In Figure 1 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol $= 10^{-(2+m/8)}$, $m = 0, \ldots, 16$ for PSODE and $m = 0, \ldots, 32$ otherwise; atol = rtol for PSODE and atol = 1 otherwise; h0 $= 10^{-7}$ for RADAU5 and PSODE.

References

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[SASJ93] D. Simpson, Y. Andersson-Skold, and M. E. Jenkin. Updating the chemical scheme for the EMEP MSC-W model: Current status. Report EMEP MSC-W Note 2/93, The Norwegian Meteorological Institute, Oslo, 1993.

[Sim93] D. Simpson. Photochemical model calculations over Europe for two extended summer periods: 1985 and 1989. model results and comparisons with observations. *Atmospheric Environment*, 27A:921–943, 1993.

[VS94] J. G. Verwer and D. Simpson. Explicit methods for stiff ODEs from atmospheric chemistry. Report NM-R9409, CWI, Amsterdam, 1994.

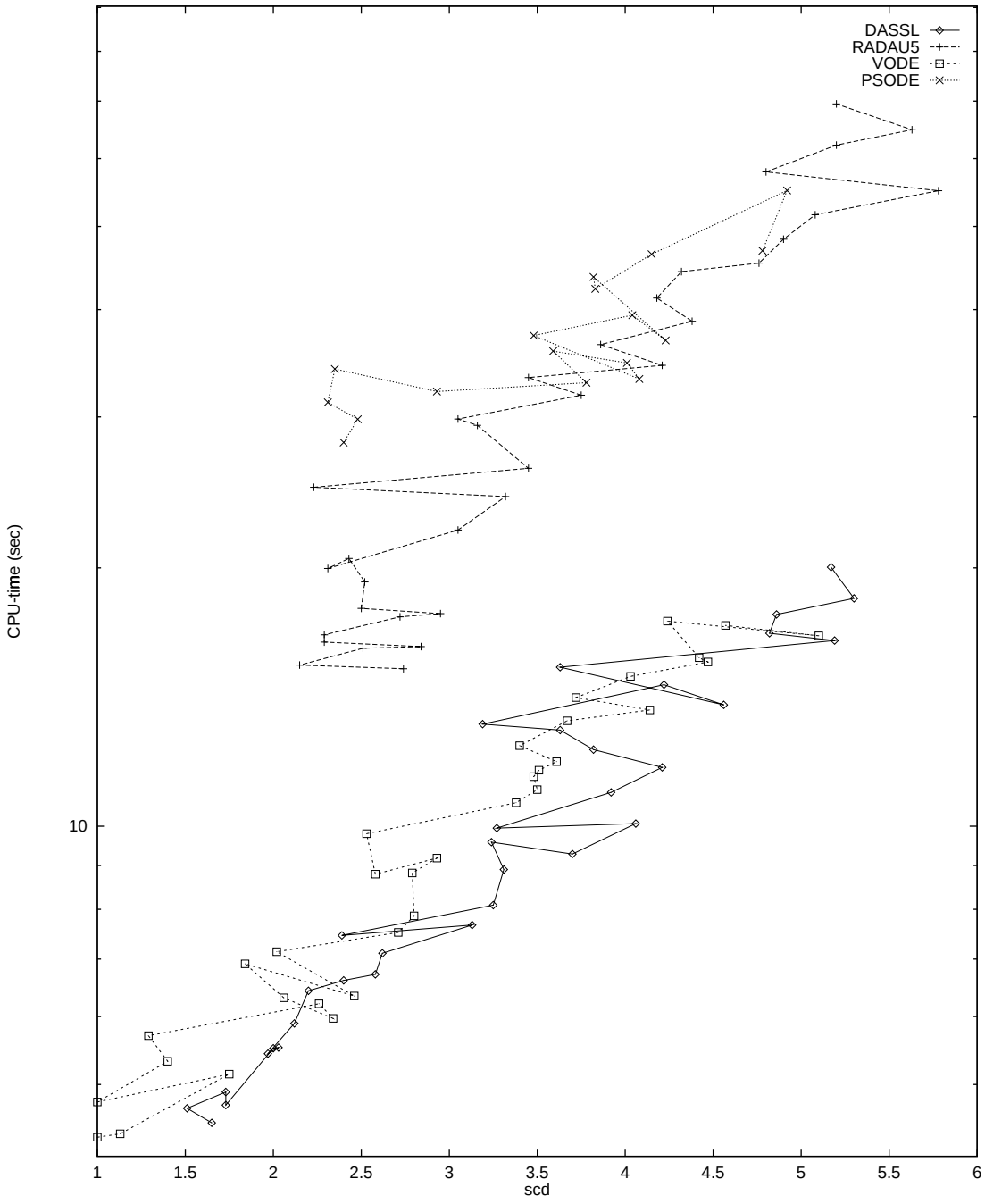


Figure 1: Work-precision diagram for EMEP problem

9 NAND gate

9.1 General information

The problem is a system of 14 stiff implicit ordinary differential equations. It has been contributed by Michael Günther and Peter Rentrop [GR95].

9.2 Mathematical description of the problem

The problem is of the form:

$$C(y(t))\,\frac{dy}{dt} = f(t, y(t)), \qquad y(0) = y_0,$$

with

$$y \in \mathbf{R}^{14}, \qquad 0 \le t \le 80.$$

The equations are given by:

$$\begin{aligned}
C_{GS}\cdot(\dot y_5 - \dot y_1) &= i^D_{DS}(y_2 - y_1, y_5 - y_1, y_3 - y_5, y_5 - y_2, y_4 - V_{DD}) \\
&\quad + \frac{y_1 - y_5}{R_{GS}} && (1\text{-}1)\\
C_{GD}\cdot(\dot y_5 - \dot y_2) &= -i^D_{DS}(y_2 - y_1, y_5 - y_1, y_3 - y_5, y_5 - y_2, y_4 - V_{DD}) \\
&\quad + \frac{y_2 - V_{DD}}{R_{GD}} && (1\text{-}2)\\
C_{BS}(y_3 - y_5)\cdot(\dot y_5 - \dot y_3) &= \frac{y_3 - V_{BB}}{R_{BS}} - i^D_{BS}(y_3 - y_5) && (1\text{-}3)\\
C_{BD}(y_4 - V_{DD})\cdot(-\dot y_4) &= \frac{y_4 - V_{BB}}{R_{BD}} - i^D_{BD}(y_4 - V_{DD}) && (1\text{-}4)
\end{aligned}$$

$$\begin{aligned}
&C_{GS}\cdot\dot y_1 + C_{GD}\cdot\dot y_2 + C_{BS}(y_3 - y_5)\cdot\dot y_3 \\
&\quad - (C_{GS} + C_{GD} + C_{BS}(y_3 - y_5) + C_5)\cdot\dot y_5 - C_{BD}(y_9 - y_5)\cdot(\dot y_5 - \dot y_9) = \\
&\quad \frac{y_5 - y_1}{R_{GS}} + i^D_{BS}(y_3 - y_5) + \frac{y_5 - y_7}{R_{GD}} + i^E_{BD}(y_9 - y_5) && (1\text{-}5)
\end{aligned}$$

$$\begin{aligned}
C_{GS}\cdot\dot y_6 &= -i^E_{DS}(y_7 - y_6, V_1(t) - y_6, y_8 - y_{10}, V_1(t) - y_7, y_9 - y_5) \\
&\quad + \quad C_{GS}\cdot\dot V_1(t) - \frac{y_6 - y_{10}}{R_{GS}} && (1\text{-}6)\\
C_{GD}\cdot\dot y_7 &= i^E_{DS}(y_7 - y_6, V_1(t) - y_6, y_8 - y_{10}, V_1(t) - y_7, y_9 - y_5) \\
&\quad + \quad C_{GD}\cdot\dot V_1(t) - \frac{y_7 - y_5}{R_{GD}} && (1\text{-}7)\\
C_{BS}(y_8 - y_{10})\cdot(\dot y_8 - \dot y_{10}) &= -\frac{y_8 - V_{BB}}{R_{BS}} + i^E_{BS}(y_8 - y_{10}) && (1\text{-}8)\\
C_{BD}(y_9 - y_5)\cdot(\dot y_9 - \dot y_5) &= -\frac{y_9 - V_{BB}}{R_{BD}} + i^E_{BD}(y_9 - y_5) && (1\text{-}9)
\end{aligned}$$

$$\begin{aligned}
&C_{BS}(y_8 - y_{10})\cdot(\dot y_8 - \dot y_{10}) - C_{BD}(y_{14} - y_{10})\cdot(\dot y_{10} - \dot y_{14}) + C_{10}\cdot\dot y_{10} = \\
&\qquad \frac{y_{10} - y_6}{R_{GS}} + i^E_{BS}(y_8 - y_{10}) + \frac{y_{10} - y_{12}}{R_{GD}} + i^E_{BD}(y_{14} - y_{10}) && (1\text{-}10)
\end{aligned}$$

$$
\begin{aligned}
C_{GS}\cdot\dot{y}_{11} &= -i^E_{DS}(y_{12}-y_{11},V_2(t)-y_{11},y_{13},V_2(t)-y_{12},y_{14}-y_{10}) \\
&+ \quad C_{GS}\cdot\dot{V}_2(t)-\frac{y_{11}}{R_{GS}} \qquad (1\text{-}11)\\
C_{GD}\cdot\dot{y}_{12} &= i^E_{DS}(y_{12}-y_{11},V_2(t)-y_{11},y_{13},V_2(t)-y_{12},y_{14}-y_{10}) \\
&+ \quad C_{GD}\cdot\dot{V}_2(t)-\frac{y_{12}-y_{10}}{R_{GD}} \qquad (1\text{-}12)\\
C_{BS}(y_{13})\cdot\dot{y}_{13} &= -\frac{y_{13}-V_{BB}}{R_{BS}}+i^E_{BS}(y_{13}) \qquad (1\text{-}13)\\
C_{BD}(y_{14}-y_{10})\cdot(\dot{y}_{14}-\dot{y}_{10}) &= -\frac{y_{14}-V_{BB}}{R_{BS}}+i^E_{BD}(y_{14}-y_{10}) \qquad (1\text{-}14)
\end{aligned}
$$

The functions C_{BD} and C_{BS} read

$$
C_{BD}(U)=C_{BS}(U)=\begin{cases} C_0\cdot\left(1-\frac{U}{\phi_B}\right)^{-\frac{1}{2}} & for \quad U\le 0\\ C_0\cdot\left(1+\frac{U}{2\cdot\phi_B}\right) & for \quad U>0\end{cases}
$$

with $C_0=0.24\cdot10^{-4}$ and $\phi_B=0.87$.
The functions i^D_{BS} and i^E_{BS} have the same form denoted by i_{BS}. The only difference between them is that the constants used in i_{BS} depend on the superscript D and E. The same holds for the functions $i^{D/E}_{BD}$, $i^{D/E}_{DS}$.
The functions i_{BS}, i_{BD} and i_{DS} are defined by

$$
i_{BS}(U_{BS}) = \begin{cases} -i_S\cdot\left(\exp(\frac{U_{BS}}{U_T})-1\right) & \text{for} \quad U_{BS}\le 0\\ 0 & \text{for} \quad U_{BS}>0\end{cases} \qquad (2)
$$

$$
i_{BD}(U_{BD}) = \begin{cases} -i_S\cdot\left(\exp(\frac{U_{BD}}{U_T})-1\right) & \text{for} \quad U_{BD}\le 0\\ 0 & \text{for} \quad U_{BD}>0\end{cases} \qquad (3)
$$

$$
i_{DS}(U_{DS},U_{GS},U_{BS},U_{GD},U_{BD}) =
$$

$$
\begin{cases} GDS_+(U_{DS},U_{GS},U_{BS}) & \text{for} \quad U_{DS}>0\\ 0 & \text{for} \quad U_{DS}=0\\ GDS_-(U_{DS},U_{GD},U_{BD}) & \text{for} \quad U_{DS}<0\end{cases} \qquad (4)
$$

where

$$
GDS_+(U_{DS},U_{GS},U_{BS}) =
$$

$$
\begin{cases} 0 & \text{for} \quad U_{GS}-U_{TE}\le 0\\ -\beta\cdot(1+\delta\cdot U_{DS})\cdot & \text{for} \quad 0<U_{GS}-U_{TE}\le U_{DS}\\ \cdot(U_{GS}-U_{TE})^2 & \\ -\beta\cdot U_{DS}\cdot(1+\delta\cdot U_{DS})\cdot & \text{for} \quad 0<U_{DS}<U_{GS}-U_{TE}\\ \cdot[2\cdot(U_{GS}-U_{TE})-U_{DS}] & \end{cases} \qquad (5)
$$

$$
\text{with} \qquad U_{TE} = U_{T0}+\gamma\cdot\left(\sqrt{\Phi-U_{BS}}-\sqrt{\Phi}\right)
$$

$$GDS\_(U_{DS}, U_{GD}, U_{BD}) =$$

$$\begin{cases} 0 & \text{for} \quad U_{GD} - U_{TE} \leq 0 \\ \beta \cdot (1 - \delta \cdot U_{DS}) \cdot & \text{for} \quad 0 < U_{GD} - U_{TE} \leq -U_{DS} \\ \cdot (U_{GD} - U_{TE})^2 & \\ -\beta \cdot U_{DS} \cdot (1 - \delta \cdot U_{DS}) \cdot & \text{for} \quad 0 < -U_{DS} < U_{GD} - U_{TE} \\ \cdot [2 \cdot (U_{GD} - U_{TE}) + U_{DS}] & \end{cases} \tag{6}$$

$$\text{with} \qquad U_{TE} \quad = \quad U_{T0} + \gamma \cdot \left(\sqrt{\Phi - U_{BD}} - \sqrt{\Phi} \right)$$

The constants used in the definition of i_{BS}, i_{BD} and i_{DS} carry a superscript D or E. Using for example the constants with superscript E in the functions i_{BS} yields the function i_{BS}^E. These constants are shown in the following table.

	E	D
i_S	10^{-14}	10^{-14}
U_T	25.85	25.85
U_{T0}	0.2	-2.43
β	$1.748 \cdot 10^{-3}$	$5.35 \cdot 10^{-4}$
γ	0.035	0.2
δ	0.02	0.02
Φ	1.01	1.28

The other constants are given by

$$\begin{gathered} V_{BB} = -2.5, \\ V_{DD} = 5, \\ C_5 = C_{10} = 0.5 \cdot 10^{-4}, \\ R_{GS} = R_{GD} = 4, \\ R_{BS} = R_{BD} = 10, \\ C_{GS} = C_{GD} = 0.6 \cdot 10^{-4}. \end{cases} \end{gathered}$$

The functions $V_1(t)$ and $V_2(t)$ are

$$V_1(t) = \begin{cases} 20 - tm & \text{if} \quad 15 < tm \leq 20 \\ 5 & \text{if} \quad 10 < tm \leq 15 \\ tm - 5 & \text{if} \quad 5 < tm \leq 10 \\ 0 & \text{if} \quad tm \leq 5 \end{cases}$$

with $tm = t \bmod 20$ and

$$V_2(t) = \begin{cases} 40 - tm & \text{if} \quad 35 < tm \leq 40 \\ 5 & \text{if} \quad 20 < tm \leq 35 \\ tm - 15 & \text{if} \quad 15 < tm \leq 20 \\ 0 & \text{if} \quad tm \leq 15 \end{cases}$$

with $tm = t \bmod 40$.
The initial values are given by

$$\begin{gathered} y_1 = y_2 = y_5 = y_7 = 5.0, \\ y_3 = y_4 = y_8 = y_9 = y_{13} = y_{14} = V_{BB} = -2.5, \\ y_6 = y_{10} = y_{12} = 3.62385, \\ y_{11} = 0. \end{gathered}$$

Remark: in this description the unit of time is the nanosecond, while in the report [GR95] the unit of time is the second.

9.3 Origin of the problem

The NAND gate in Figure 1 consists of two n-channel enhancement MOSFETs (ME), one n-channel depletion MOSFET (MD) and two load capacitances C_5 and C_{10}. MOSFETs are special transistors. They have four terminals: the drain, the bulk, the source and the gate. The gate voltages of both enhancement transistors are controlled by two voltage sources V_1 and V_2.

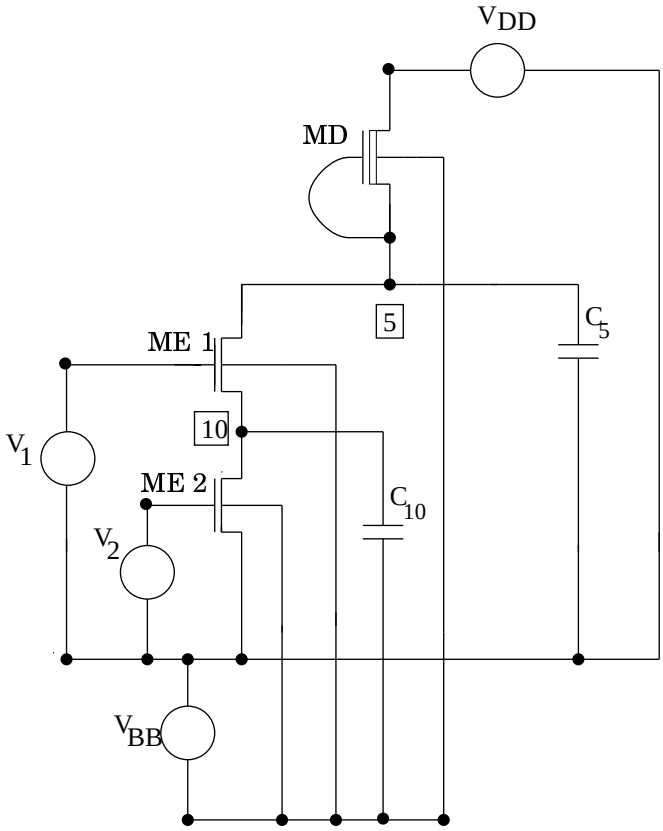


Figure 1: *Circuit diagram of the NAND gate (taken from [GR95])*

		V2	
		LOW	HIGH
V1	LOW	HIGH	HIGH
	HIGH	HIGH	LOW

Figure 2: *Response of the NAND gate*

Depending on the input voltages, the NAND gate generates a response at node 5 as shown in Figure 2. If we represent the logical values 1 and 0 by high respectively low voltage levels, we see that the NAND gate executes the *Not AND* operation. This behaviour is easily explained: If V_1 respectively V_2 is low, then the corresponding enhancement transistors locks; the voltage at node 5 is high at $V_{DD} = 5V$ due to MD. If both V_1 and V_2 exceed a given threshold voltage U_T, then a

drain current through both enhancement transistors occurs. The MOSFETs open and the voltage at node 5 breaks down. The response is low.

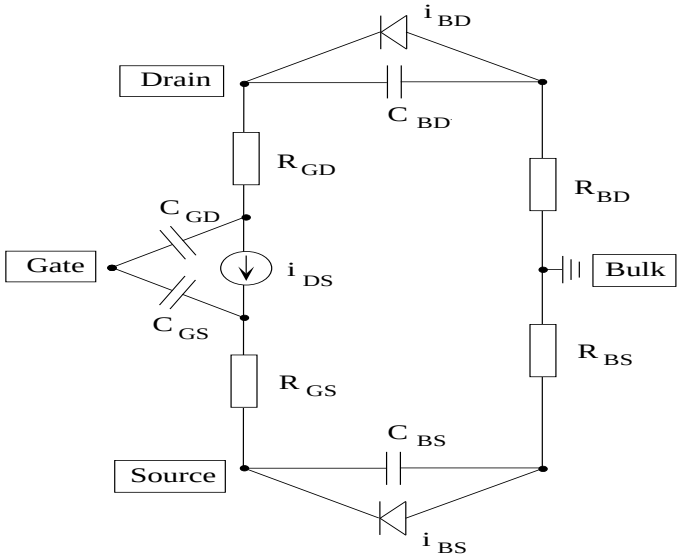


Figure 3: *Companion model of a MOSFET (taken from [GR95])*

In the circuit analysis the three MOSFETs are replaced by the circuit shown in Figure 3. Here, the well-known companion model of Shichmann and Hodges [SH68] is used. The characteristics of the circuit elements can differ depending on the MD or ME case. This circuit has four internal nodes corresponding to the drain, the bulk, the source and the gate. The static behaviour of the transistor is described by the drain current i_{DS}. To include secondary effects, load capacitances like R_{GS}, R_{GD}, R_{BS}, and R_{BD} are introduced. The so-called *pn*-junction between source and bulk is modelled by the diode i_{BS} and the non-linear capacitance C_{BS}. Analogously, i_{BD} and C_{BD} model the *pn*-junction between bulk and diode. Linear gate capacitances C_{GS} and C_{GD} are used to describe the intrinsic charge flow effects roughly.

To formulate the circuit equations, we note that the circuit consists of 14 nodes. These 14 nodes are the nodes 5 and 10 and the 12 internal nodes of the three transistors. For every node a variable is introduced that represents the voltage in that node. In terms of these voltages the circuit equations are formulated by using the Kirchoff Current Law (KCL) along with the transistor model shown in Figure 3. The differential equations given in the previous section result from applying KCL to the following nodes:

equations	nodes
1–4	internal nodes MD-transistor
5	node 5
6–9	internal nodes ME1-transistor
10	node 10
11–14	internal nodes ME2-transistor

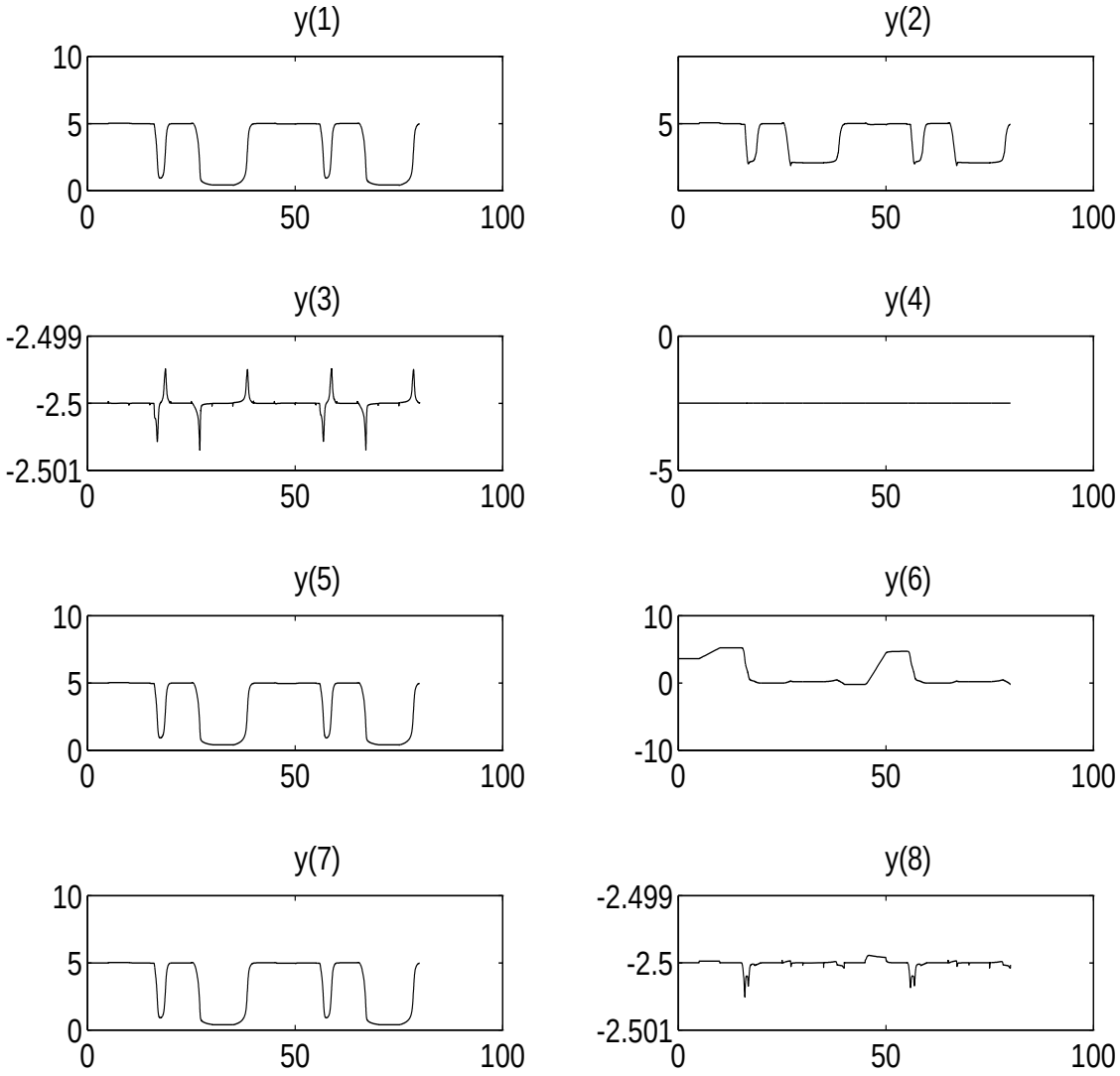
9.4 Numerical solution of the problem

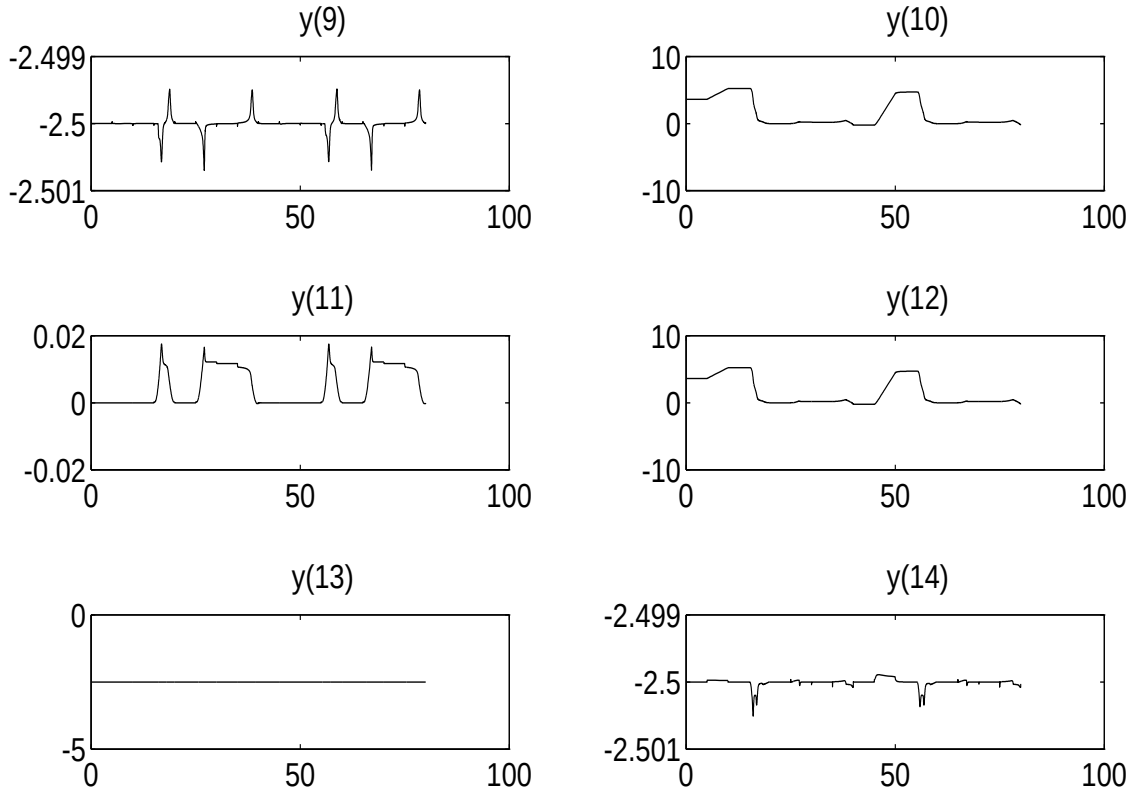
9.4.1 Solution at t=80:

y_1	4.971088699357187
y_2	4.999752103929143
y_3	−2.499998781491202
y_4	−2.499999999999975
y_5	4.970837023267885
y_6	−0.2091214033360282
y_7	4.970593243271932
y_8	−2.500077409198804
y_9	−2.499998781491201
y_{10}	−0.2090289585389384
y_{11}	$-2.3999997751031701 \cdot 10^{-4}$
y_{12}	−0.2091214033360281
y_{13}	−2.499999999999991
y_{14}	−2.500077409198804

9.4.2 Behaviour of the numerical solution

The following plots show the behaviour of the solution components on the interval [0,80]:





9.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		0.56	894	744	1668	323		1.79
	10^{-7}	10^{-7}		3.66	3805	3414	6039	914		5.84

9.4.4 Work-precision diagram

In Figure 4 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol = $10^{-(4+m/8)}$, $m = 0, \ldots, 24$; atol = rtol.

References

[GR95] M. Günther and P. Rentrop. The NAND-gate – a benchmark for the numerical simulation of digital circuits. Technical Report 1740, Techn. Hochschule Darmstadt, Fachbereich Mathematik, Darmstadt, 1995. To appear in: Proceedings of the Cauer Symposium, Berlin, VDE-Verlag.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[SH68] H. Shichman and D. A. Hodges. Insulated-gate field-effect transistor switching circuits. *IEEE J. Solid State Circuits*, SC-3:285–289, 1968.

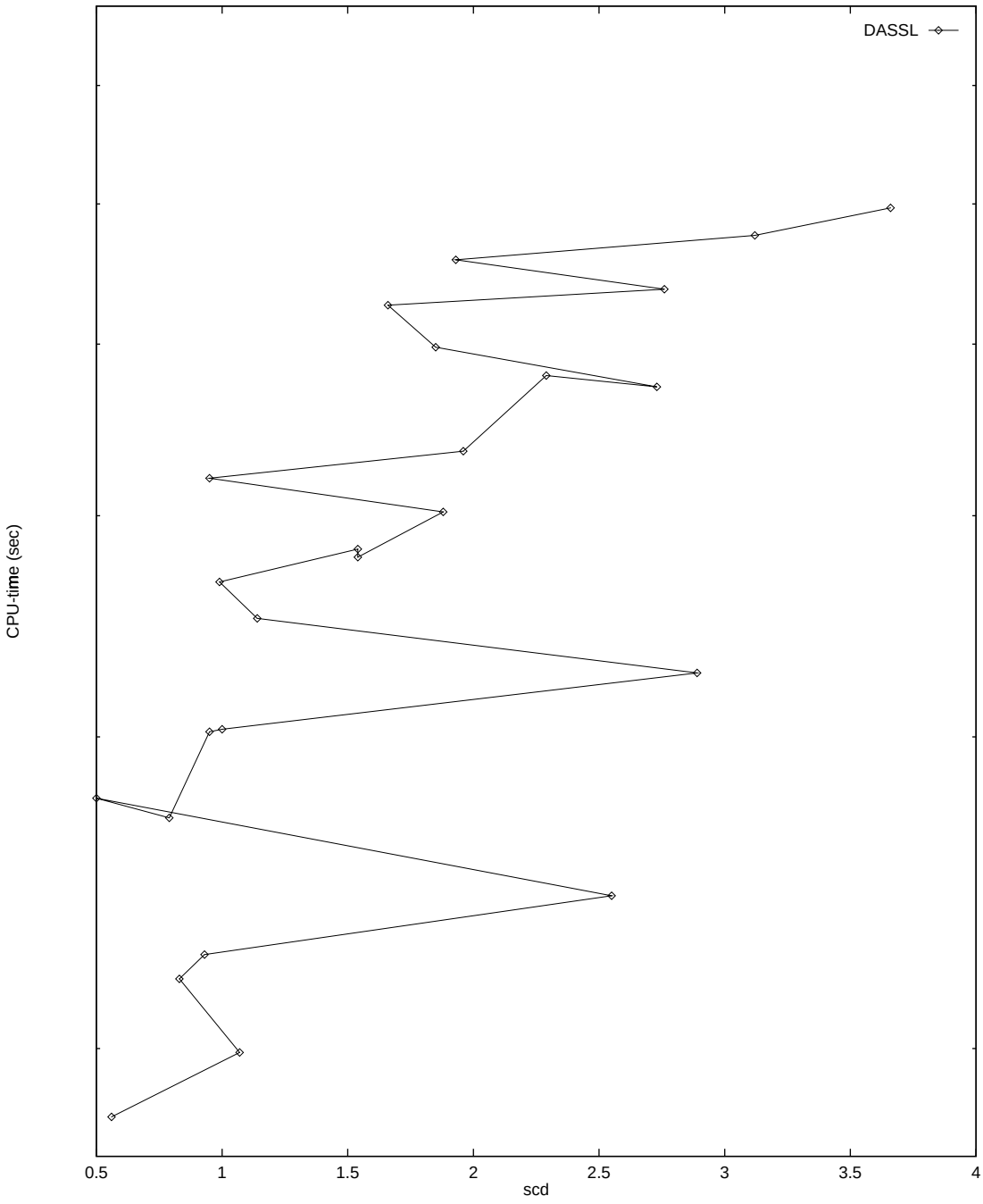


Figure 4: Work-precision diagram for NAND gate

10 Charge pump

10.1 General information

The problem is a stiff DAE of index 2, consisting of 3 differential and 6 algebraic equations. It has been contributed by Michael Günther, Georg Denk and Uwe Feldmann [GDF95].

10.2 Mathematical description

The problem is of the form

$$M\frac{dy}{dt} - f(t, y(t)) = 0, \qquad y(0) = y_0,$$

with

$$y \in \mathbf{R}^9, \qquad 0 \le t \le 1.2 \cdot 10^{-6}.$$

The matrix M is the zero matrix except for the the minor $M_{1..3,1..5}$, that is given by

$$M_{1..3,1..5} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

The function f is defined by

$$f(t,y) = \begin{pmatrix} -y_9 \\ 0 \\ 0 \\ -y_6 + V_{in}(t) \\ y_1 - Q_G(v) \\ y_2 - C_S \cdot y_7 \\ y_3 - Q_S(v) \\ y_4 - C_D \cdot y_8 \\ y_5 - Q_D(v) \end{pmatrix},$$

with $v := (v_1, v_2, v_3) = (y_6, y_6 - y_7, y_6 - y_8)$, $C_D = 0.4 \cdot 10^{-12}$ and $C_S = 1.6 \cdot 10^{-12}$. The functions Q_G, Q_S and Q_D are given by:

1. if $v_1 \le V_{FB} := U_{T0} - \gamma\sqrt{\Phi} - \Phi$

$$\begin{aligned} Q_G(v) &= C_{ox}(v_1 - V_{FB}) \\ Q_S(v) &= Q_D(v) = 0 \end{aligned}$$

   with $C_{ox} = 4 \cdot 10^{-12}$, $U_{T0} = 0.2$, $\gamma = 0.035$ and $\Phi = 1.01$.

2. if $v_1 > V_{FB}$ and $v_2 \le U_{TE} := U_{T0} + \gamma(\sqrt{\Phi + v_1 - v_2} - \sqrt{\Phi})$

$$\begin{aligned} Q_G(v) &= C_{ox}\gamma(\sqrt{(\gamma/2)^2 + v_1 - V_{FB}} - \gamma/2) \\ Q_S(v) &= Q_D(v) = 0. \end{aligned}$$

3. if $v_1 > V_{FB}$ and $v_2 > U_{TE}$

$$\begin{aligned} Q_G(v) &= C_{ox}\left[\frac{2}{3}(U_{GDT} + U_{GST} - \frac{U_{GDT}U_{GST}}{U_{GDT} + U_{GST}}) + \gamma\sqrt{\Phi - U_{BS}}\right] \\ Q_S(v) &= Q_D(v) = -\frac{1}{2}(Q_G - C_{ox}\gamma\sqrt{\Phi - U_{BS}})). \end{aligned}$$

Here, U_{BS}, U_{GST} and U_{GDT} are given by

$$\begin{aligned} U_{BS} &= v_2 - v_1, \\ U_{GST} &= v_2 - U_{TE}, \\ U_{GDT} &= \begin{cases} v_3 - U_{TE} & \text{for} \quad v_3 > U_{TE}, \\ 0 & \text{for} \quad v_3 \le U_{TE}. \end{cases} \end{aligned}$$

The function $V_{in}(t)$ is defined using $\tau = (10^8 \cdot t) \bmod 120$ by

$$V_{in}(t) = \begin{cases} 0 & \text{if} \quad \tau < 50 \\ 20(\tau - 50) & \text{if} \quad 50 \le \tau < 60 \\ 20 & \text{if} \quad 60 \le \tau < 110 \\ 20(120 - \tau) & \text{if} \quad \tau \ge 110. \end{cases}$$

Finally, the initial value y_0 reads

$$y_0 = (0, 0, 0, 0, 0, 0, 0, 0, 0)^T.$$

10.3 Origin of the problem

The Charge-pump circuit shown in Figure 1 consists of two capacitors and an n-channel MOS-transistor. The nodes gate, source, gate, and drain of the MOS-transistor are connected with the nodes 1, 2, 3, and Ground, respectively. In formulating the circuit equations, the transistor is replaced by four non-linear current sources in each of the connecting branches. They model the transistor.

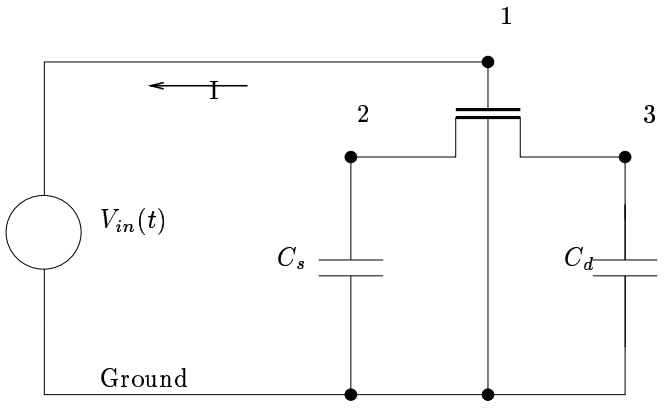


Figure 1: *Circuit diagram of Charge-pump circuit (taken from [GDF95])*

After inserting the transistor model in the circuit, we get the final circuit, that is the circuit shown in Figure 1, where:

- The transistor has been removed and is replaced by a solid line between the nodes 2 and 3. The point where the lines 2–3 and 1–Ground cross each other becomes a node, which will be denoted by T.
- There are current sources between nodes 1 and T, between 2 and T and between 3 and T. There is also a current source between the ground and node T, but as the node Ground does not enter the circuit equations, it will not be discussed. The currents produced by these sources are written as the derivatives of charges: current from 1 to T: Q'_G, from T to 2: Q'_S and from T to 3: Q'_D. Here, the functions Q_G, Q_S and Q_D depend on the voltage drops U_1, $U_1 - U_2$ and $U_1 - U_3$, where U_i denotes the potential in node i.

The unknowns in the circuit are given by:

- The charges produced by the current sources: Y_{T1}, Y_{T2}, Y_{T3}. They are aliases for respectively Q_G, Q_S and Q_D. Consequently, Y'_{Ti} is the current between node T and node i.
- The charges Y_S and Y_D in the capacitors C_S and C_D.
- Potentials in nodes 1 to 3: U_1, U_2, U_3.
- The current through the voltage source $V_{in}(t)$: I.

In terms of these physical variables, the vector y introduced earlier reads

$$y = (Y_{T1}, Y_S, Y_{T2}, Y_D, Y_{T3}, U_1, U_2, U_3, I)^T.$$

Now, the following equations hold

$$\begin{aligned} Y'_{T1} &= -I, \\ Y'_S + Y'_{T2} &= 0, \\ Y'_D + Y'_{T3} &= 0, \\ U_1 &= V_{in}(t). \end{aligned}$$

The charges depend on the potentials and are given by

$$\begin{aligned} Y_{T1} &= Q_G(U_1, U_1 - U_2, U_1 - U_3), \\ Y_S &= C_S \cdot U_2, \\ Y_{T2} &= Q_S(U_1, U_1 - U_2, U_1 - U_3), \\ Y_D &= C_D \cdot U_3, \\ Y_{T3} &= Q_D(U_1, U_1 - U_2, U_1 - U_3). \end{aligned}$$

The functions Q_G, Q_S and Q_D are given in the previous section.
Remark: the potential U_1 is known. Here, it is treated as an unknown in order to keep the formulation general and leaving open the possibility to extend the circuit. In addition, removing U_1 by hand contradicts a CAD approach in circuit simulation.

10.4 Numerical solution of the problem

Michael Günther, Georg Denk and Uwe Feldmann [GDF95] provided the source code defining the problem.

Their implementation is written for use with DASSL. To deal with the discontinuities and the index 2 character, they included a special DASSL driver (`pumpdriv.f`). At the points, where the derivative of $V_{in}(t)$ is discontinuous, DASSL is restarted in order to prevent loss of accuracy. Since y_9 is an index 2 variable, the error control for this variable has been switched off by setting atol(9)=rtol(9)=1000. For this problem, it would have been possible to include the index 2 variable in the error control, but this would require a modification of the DASSL code. In the future we hope to include results obtained by RADAU5.

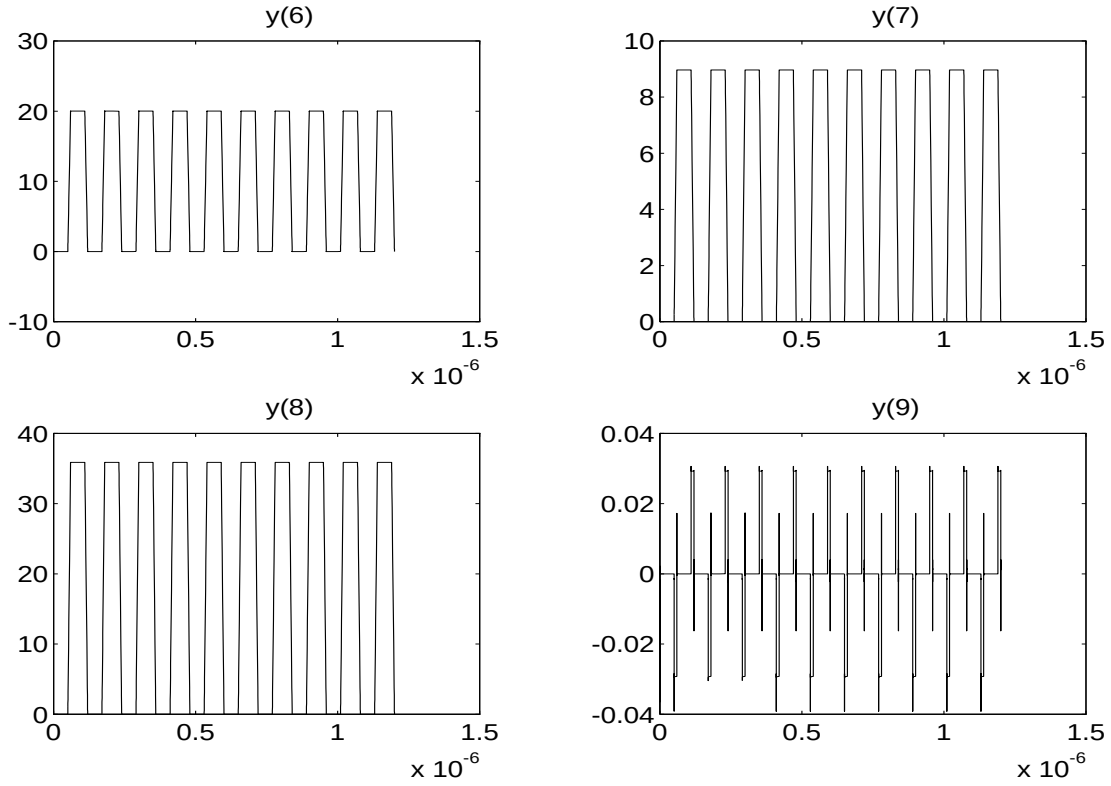
10.4.1 Solution at the endpoint

Until further notice the ninth component is ignored because it is neglected in the error control. At the endpoint all components are zero except for the first one, which is given by $y_1 = 1.26283 \cdot 10^{-13}$. We remark that the magnitude of this component is at most 10^{-10}.

10.4.2 Behaviour of the numerical solution

Only the last four components have been plotted, since they are the physically important quantities. The other five components refer to charge flows inside the transistor, which are quantities the user is not interested in. These components have a similar behaviour as the components 6, 7 and 8, but their magnitude is at most 10^{-10}.

Since there is no error control for the variable y_9, we prescribed a maximal value for the stepsize. By comparing plots of y_9 obtained by varying this maximal value, we were able to come up with a sufficiently accurate solution, that is shown in the last plot.



10.4.3 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. The various components differ enormously in magnitude. Therefore, the parameters atol and rtol were chosen to be component-dependent and are given by

$$\begin{aligned} \text{atol}(i) &= \text{Tol} \cdot 10^{-6} \quad \text{for} \quad i = 1, \ldots, 5, \\ \text{atol}(i) &= \text{Tol} \quad \text{for} \quad i = 6, \ldots, 8, \\ \text{atol}(9) &= 1000, \\ \text{rtol}(i) &= \text{Tol} \quad \text{for} \quad i = 1, \ldots, 8, \\ \text{rtol}(9) &= 1000. \end{aligned}$$

Since the components $y_2, y_3, \ldots, y_8$ are exactly zero in the endpoint and the error control of the ninth component was neglected, the computation of the number of significant correct digits (scd), involves only the first component y_1.

solver	Tol	scd	steps	accept	# f	# Jac	CPU
DASSL	10^{-2}	0.14	4594	3636	8369	4662	0.56
	10^{-4}	3.80	14933	11257	33593	15019	1.06

10.4.4 Work-precision diagram

In Figure 2 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: Tol $= 10^{-(2+m/8)}$, $m = 0, \ldots, 16$.

References

[GDF95] M. Günther, G. Denk, and U. Feldmann. How models for MOS transistors reflect charge distribution effects. Technical Report 1745, Technische Hochschule Darmstadt, Fachbereich Mathematik, Darmstadt, 1995. To appear in: Mathematical Modelling of Systems.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

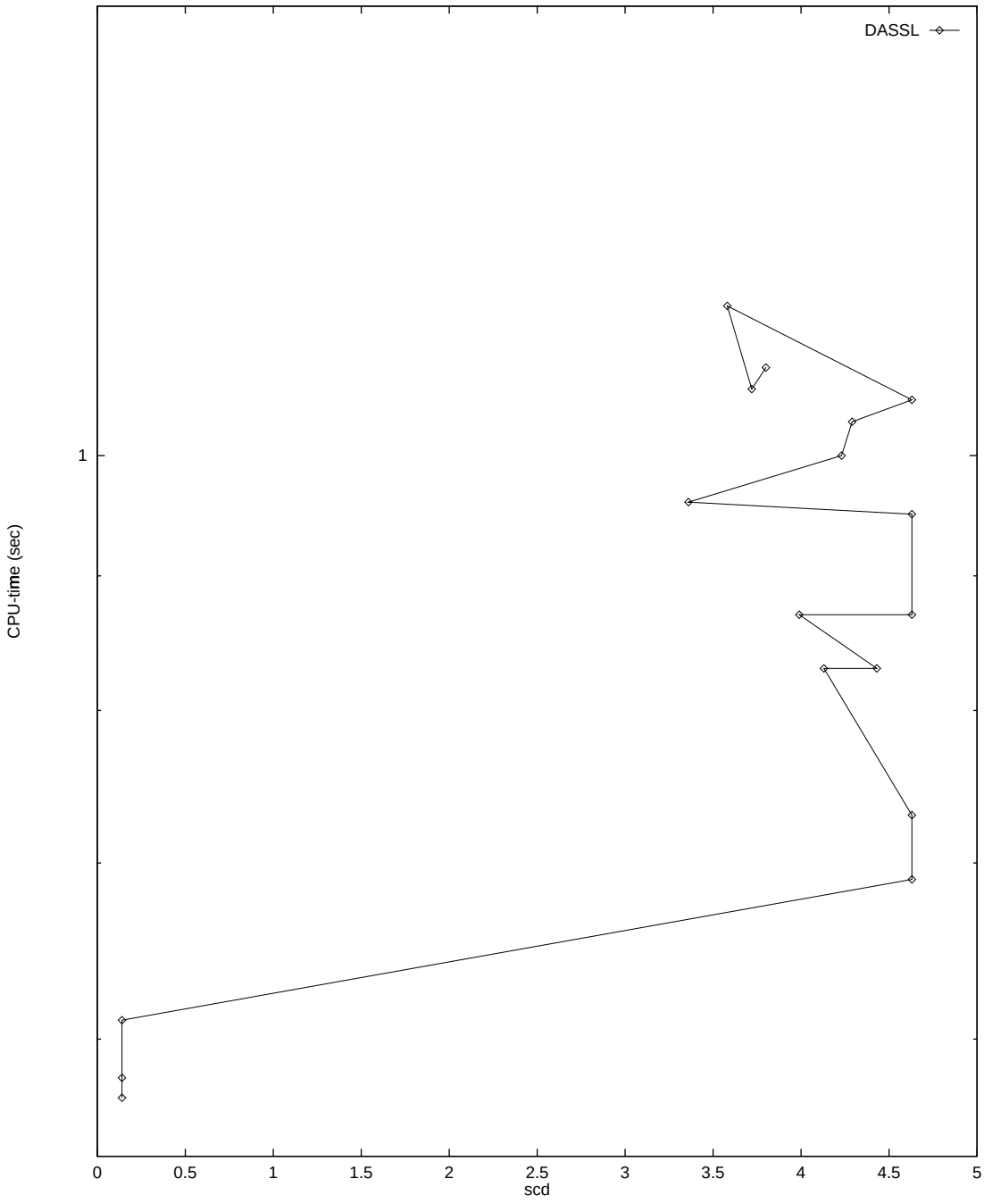


Figure 2: Work-precision diagram for Charge pump

11 Wheelset

Contributed by Bernd Simeon, Claus Führer, Peter Rentrop, Nov. 1995.

11.1 General Information

The wheelset example shows some typical properties of simulation problems in contact mechanics, i.e., friction, contact conditions, stiffness, etc.. This problem is originally described by a system of differential-algebraic equations (DAEs) of index 3 with additional index 1 equations. An index reduced formulation as an index 2 DAE with additional index 1 equations is provided also. The differential-algebraic system consists of 17 equations. Test results are based on the index-2 formulation.

Comments to `bernd.simeon@mathematik.th-darmstadt.de`
`claus@dna.lth.se`

Reference: B. Simeon, C. Führer, P. Rentrop: [SFR91]
Differential-Algebraic Equations in Vehicle System Dynamics, Surv. Math. Ind. I:1–37 (1991)

11.2 Mathematical description of the problem

The wheelset problem is described by the DAE

$$\begin{aligned} \dot{p} &= v && (1)\\ M(p)\dot{v} &= f(p,q,v,\beta,N(p,q,\lambda)) - (\partial g_1(p,q)/\partial p)^T \lambda && (2)\\ \dot{\beta} &= d(p,q,v,\beta,N(p,q,\lambda)) && (3)\\ 0 &= g_1(p,q) && (4)\\ 0 &= g_2(p,q) && (5) \end{aligned}$$

with	p	$n_p = 5$	position variables
	v	$n_v = 5$	velocity variables
	β	$n_\beta = 1$	variable of first order dynamics
	q	$n_q = 4$	contact variables
	λ	$n_\lambda = 2$	Lagrange multipliers

The integration interval is from 0 to 10 [s].
(1) – (3) stand for the kinematic and dynamic equations with positive definite mass matrix $M(p)$. The 2 index-3 constraint equations g_1 describe the contact condition for wheel and rail while the additional 4 index-1 constraints g_2 express that wheel and rail may not intersect. The equations are given in detail in the next subsections.

11.2.1 Differential equations

The position coordinates are defined as (cf. Sect. 3)

$$p := \begin{pmatrix} x \\ y \\ z \\ \theta \\ \varphi \end{pmatrix} \qquad \begin{matrix} \text{lateral displacement} \\ \text{vertical displacement} \\ \text{longitudinal displacement} \\ \text{yaw angle} \\ \text{roll angle} \end{matrix}$$

and the contact variables as $q^T := (\ \psi_L \quad \xi_L \quad \psi_R \quad \xi_R\ )$ with

$$\begin{aligned} \xi_{L|R} &:= \text{coordinate of the contact point left/right,} \\ \psi_{L|R} &:= \text{shift angle left/right.} \end{aligned}$$

The first three equations in (2) yield the momentum equations:

$$\begin{aligned}
m_R\,\ddot{x} &= m_R\left(2\,v_0\,\kappa\,\cos\alpha\,\dot{z} + v_0^2\,\kappa\,\cos\alpha\,(1+\kappa\,(x\cos\alpha - y\sin\alpha))\right)\\
&\quad + T_{L_1} + T_{R_1} + Q_1 - m_R\,\tilde{g}\,\sin\alpha - b_{1,1}\,\lambda_1 - b_{1,2}\,\lambda_2 - 2\,c_x\,x\\
m_R\,\ddot{y} &= -m_R\left(2\,v_0\,\kappa\,\sin\alpha\,\dot{z} + v_0^2\,\kappa\,\sin\alpha\,(1+\kappa\,(x\cos\alpha - y\sin\alpha))\right)\\
&\quad + T_{L_2} + T_{R_2} + Q_2 - m_R\,\tilde{g}\,\cos\alpha - b_{2,1}\,\lambda_1 - b_{2,2}\,\lambda_2\\
m_R\,\ddot{z} &= m_R\left(-2\,v_0\,\kappa\,(\dot{x}\cos\alpha - \dot{y}\sin\alpha) + v_0^2\,\kappa^2\,z\right)\\
&\quad + T_{L_3} + T_{R_3} + Q_3 + F_A(t) - b_{3,1}\,\lambda_1 - b_{3,2}\,\lambda_2
\end{aligned}$$

where $b_{i,j}$ denotes the (i,j) element of the constraint Jacobian $\partial g_1(p,q)/\partial p$, see below. The next three equations yield the spin equations:

$$\begin{aligned}
&I_2\,\ddot{\theta}\,\cos\varphi =\\
&\quad -I_2\left(-\dot{\theta}\,\dot{\varphi}\sin\varphi + v_0\,\kappa(\,\dot{\varphi}(\sin\alpha\cos\theta\cos\varphi + \cos\alpha\sin\varphi) - \dot{\theta}\sin\alpha\sin\theta\sin\varphi)\right)\\
&\quad -I_1\,(\omega_0+\beta)\,(\dot{\varphi} - v_0\kappa\sin\theta\sin\alpha)\\
&\quad -(I_1 - I_2)\left(\dot{\theta}\,\sin\varphi - v_0\,\kappa\,(\cos\theta\cos\varphi\sin\alpha + \sin\varphi\cos\alpha)\right)(\dot{\varphi} - v_0\,\kappa\,\sin\alpha\sin\theta)\\
&\quad +\Big[\,-(\xi_L\sin\theta + R(\xi_L)\sin\psi_L\cos\theta\cos\varphi)\,T_{L_1}\\
&\qquad -R(\xi_L)\sin\psi_L\sin\varphi\,T_{L_2}\\
&\qquad +(-\xi_L\cos\theta + R(\xi_L)\sin\psi_L\sin\theta\cos\varphi)\,T_{L_3}\Big]\\
&\quad +\Big[\quad \text{corresponding terms of the right side}\quad\Big]\\
&\quad -\cos\theta\,\sin\varphi\,M_1 + \cos\varphi\,M_2 + \sin\theta\,\sin\varphi\,M_3 - b_{4,1}\,\lambda_1 - b_{4,2}\,\lambda_2
\end{aligned}$$

$$\begin{aligned}
&I_2\,\ddot{\varphi} =\\
&\quad I_2\,\dot{\theta}\,v_0\,\kappa\,\sin\alpha\,\cos\theta\\
&\quad +I_1\,(\omega_0+\beta)\left(\dot{\theta}\cos\varphi + v_0\,\kappa\,(\cos\theta\sin\varphi\sin\alpha - \cos\varphi\cos\alpha)\right)\\
&\quad +(I_1 - I_2)\Big(\dot{\theta}\,\sin\varphi - v_0\,\kappa(\cos\theta\cos\varphi\sin\alpha + \sin\varphi\cos\alpha)\Big)\\
&\qquad\qquad \Big(\dot{\theta}\,\cos\varphi + v_0\,\kappa(\cos\theta\sin\varphi\sin\alpha - \cos\varphi\cos\alpha)\Big)\\
&\quad +\Big[\,-(\xi_L\cos\theta\sin\varphi - R(\xi_L)\cos\psi_L\cos\theta\cos\varphi)\,T_{L_1}\\
&\qquad +(\xi_L\cos\varphi + R(\xi_L)\cos\psi_L\sin\varphi)\,T_{L_2}\\
&\qquad +(\xi_L\sin\theta\sin\varphi - R(\xi_L)\cos\psi_L\sin\theta\cos\varphi)\,T_{L_3}\Big]\\
&\quad +\Big[\quad \text{corresponding terms of the right side}\quad\Big]\\
&\quad +\sin\theta\,M_1 + \cos\theta\,M_3 - b_{5,1}\,\lambda_1 - b_{5,2}\,\lambda_2
\end{aligned}$$

$$\begin{aligned}
&I_1\,(\dot{\beta} + \ddot{\theta}\,\sin\varphi) =\\
&\quad -I_1\left(\dot{\theta}\,\dot{\varphi}\cos\varphi - v_0\,\kappa\,(\dot{\varphi}(\cos\alpha\cos\varphi - \sin\alpha\cos\theta\sin\varphi) - \dot{\theta}\sin\alpha\sin\theta\cos\varphi)\right)\\
&\quad +\Big[\,-R(\xi_L)\,(\cos\psi_L\sin\theta + \sin\psi_L\cos\theta\sin\varphi)\,T_{L_1} + R(\xi_L)\,\sin\psi_L\,\cos\varphi\,T_{L_2}\\
&\qquad -R(\xi_L)\,(\cos\psi_L\cos\theta - \sin\psi_L\sin\theta\sin\varphi)\,T_{L_3}\Big]\\
&\quad +\Big[\quad \text{corresponding terms of the right side}\quad\Big]\\
&\quad +\cos\theta\,\cos\varphi\,M_1 + \sin\varphi\,M_2 - \sin\theta\,\cos\varphi\,M_3 + L_A(t)\,.
\end{aligned}$$

The forces Q and moments M of the wagon body (cf. Sect. 3) satisfy the following equations:

$$\begin{aligned}
Q_1 &= \frac{m_A\,\tilde{g}}{\cos\alpha}\left(\frac{v_0^2\,\kappa}{\tilde{g}} - \tan\alpha\right) && \text{(lateral force)}\\
Q_2 &= -m_A\,\tilde{g}\,\cos\alpha\,\left(\frac{v_0^2\,\kappa}{\tilde{g}}\,\tan\alpha + 1\right) && \text{(vertical force)}\\
Q_3 &= -2\,c_z\,z && \text{(longitudinal force)}\\
M_1 &= 0\\
M_2 &= Q_3\,x_l && \text{(yaw moment)}\\
M_3 &= -h\,Q_1 && \text{(roll moment)}\\
0 &= \cos\theta\,M_1 - \sin\theta\,M_3 && \text{(no pitch moment)}
\end{aligned}$$

The creep forces $T_{L_{1,2,3}}$ and $T_{R_{1,2,3}}$ of the left and right contact point are obtained via the transformation

$$\begin{pmatrix} T_{L|R_1}\\ T_{L|R_2}\\ T_{L|R_3}\end{pmatrix} = \begin{pmatrix} \sin\theta & \cos\theta\cos\Delta_{L|R} & \mp\cos\theta\sin\Delta_{L|R}\\ 0 & \pm\sin\Delta_{L|R} & \cos\Delta_{L|R}\\ \cos\theta & -\sin\theta\cos\Delta_{L|R} & \pm\sin\theta\sin\Delta_{L|R}\end{pmatrix}\begin{pmatrix} T_{1_{L|R}}\\ T_{2_{L|R}}\\ 0\end{pmatrix}$$

where $T_{1_{L|R}}$ and $T_{2_{L|R}}$ denote the creep forces with respect to the local reference frame of the contact point and $\pm$ stands for the left and right side, respectively. The creep forces are approximated by

$$T_{1_{L|R}} := -\mu N_{L|R}\tanh\left(\frac{GC_{11}c^2}{\mu N_{L|R}}\nu_1\right) \tag{6}$$

$$T_{2_{L|R}} := -\mu N_{L|R}\tanh\left(\frac{GC_{22}c^2}{\mu N_{L|R}}\nu_2 + \frac{GC_{23}c^3}{\mu N_{L|R}}\varphi_3\right) \tag{7}$$

and corrected by

$$\text{if}\quad T_1^2 + T_2^2 > (\mu N)^2\ \text{, then}$$

$$\tilde{T}_1 := \frac{T_1}{\sqrt{T_1^2+T_2^2}}\,\mu N \qquad \text{and} \qquad \tilde{T}_2 := \frac{T_2}{\sqrt{T_1^2+T_2^2}}\,\mu N. \tag{8}$$

The constant parameters

$$\mu, G, C_{11}, C_{22}, C_{23}$$

(friction coefficient, glide modul, Kalker coefficients) are listed below. For the computation of c (size of contact ellipse) and for alternative creep force models see [Jas87].

The normal forces N are given by

$$\begin{aligned}
\begin{pmatrix} N_L\\ N_R\end{pmatrix} &= \gamma\begin{pmatrix} \cos\Delta_R & -\sin\Delta_R\\ -\cos\Delta_L & -\sin\Delta_L\end{pmatrix}\begin{pmatrix} b_{1,1} & b_{1,2}\\ b_{2,1} & b_{2,2}\end{pmatrix}\begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix}\\
\text{where } \gamma &:= \frac{1}{\sin\Delta_L\cos\Delta_R + \sin\Delta_R\cos\Delta_L}\,.
\end{aligned}$$

$\Delta_{L|R}$ denotes the contact angles and is defined as

$$\begin{aligned}
\tan\Delta_L &= \frac{(R'(\xi_L)\cos\varphi - \sin\varphi\cos\psi_L)\cos\theta + \sin\psi_L\sin\theta}{-R'(\xi_L)\sin\varphi - \cos\psi_L\cos\varphi}\,;\\
\tan\Delta_R &= \frac{(R'(\xi_R)\cos\varphi - \sin\varphi\cos\psi_R)\cos\theta + \sin\psi_R\sin\theta}{+R'(\xi_R)\sin\varphi + \cos\psi_R\cos\varphi}\,.
\end{aligned}$$

For the creepages we have the relations

$$\begin{aligned}
\nu_1 &= \frac{1}{v_{roll}}\left(\sin\theta v_{r1} + \cos\theta v_{r3}\right)\\
\nu_2 &= \frac{1}{v_{roll}}\left(\cos\theta\cos\Delta_{L|R} v_{r1} \pm \sin\Delta_{L|R} v_{r2} - \sin\theta\cos\Delta_{L|R} v_{r3}\right)\\
\varphi_3 &= \frac{1}{v_{roll}}\left(\mp\sin\Delta_{L|R}(\omega+\beta-v_0\,\kappa\,\sin\alpha) + \cos\Delta_{L|R}(\dot\theta - v_0\,\kappa\cos\alpha)\right)
\end{aligned}$$

where $v_{r1,2,3}$ (relative velocity at the contact point) and v_{roll} (rolling velocity) are given by (correspondingly for the right side)

$$\begin{aligned}
v_{r1} &= \dot x - \dot\theta(R(\xi_L)(\sin\theta\sin\varphi\cos\psi_L + \cos\theta\sin\psi_L) + \xi_L\sin\theta\cos\varphi)\\
&\quad -\dot\varphi\cos\theta(\xi_L\sin\varphi - R(\xi_L)\cos\varphi\cos\psi_L)\\
&\quad +(\omega_0+\beta)R(\xi_L)(-\sin\theta\cos\psi_L - \sin\varphi\cos\theta\sin\psi_L)\\
&\quad +v_0\kappa\cos\alpha(R(\xi_L)(\sin\theta\sin\varphi\cos\psi_L + \cos\theta\sin\psi_L) + \xi_L\sin\theta\cos\varphi - z)\\
v_{r2} &= \dot y + \dot\varphi(\xi_L\cos\varphi + R(\xi_L)\sin\varphi\cos\psi_L) + (\omega_0+\beta)R(\xi_L)\cos\varphi\sin\psi_L\\
&\quad +v_0\kappa\sin\alpha(z - \xi_L\sin\theta\cos\varphi - R(\xi_L)(\sin\theta\sin\varphi\cos\psi_L + \cos\theta\sin\psi_L))\\
v_{r3} &= \dot z + v_0 + v_0\kappa(x\cos\alpha - y\sin\alpha)\\
&\quad -\dot\theta(\xi_L\cos\theta\cos\varphi + R(\xi_L)(\cos\theta\sin\varphi\cos\psi_L - \sin\theta\sin\psi_L))\\
&\quad +\dot\varphi\sin\theta(\xi_L\sin\varphi - R(\xi_L)\cos\varphi\cos\psi_L)\\
&\quad +(\omega+\beta)R(\xi_L)(\sin\theta\sin\varphi\sin\psi_L - \cos\theta\cos\psi_L)\\
&\quad -v_0\kappa\sin\alpha(\xi_L\sin\varphi - R(\xi_L)\cos\varphi\cos\psi_L)\\
&\quad +v_0\cos\alpha(\xi_L\cos\theta\cos\varphi + R(\xi_L)(\cos\theta\sin\varphi\cos\psi_L - \sin\theta\sin\psi_L))
\end{aligned}$$

and

$$v_{roll} = \frac{1}{2}\left\| \begin{pmatrix} -2\dot x + 2v_0\kappa z\cos\alpha \\ -2\dot y - 2v_0\kappa z\sin\alpha \\ -2\dot z - 2v_0 - 2v_0\kappa(x\cos\alpha - y\sin\alpha)\end{pmatrix} + \begin{pmatrix} v_{r1}\\ v_{r2}\\ v_{r3}\end{pmatrix}\right\|_2 .$$

11.2.2 Constraints

The index-3 constraints g_1 read

$$\begin{pmatrix} G(\hat\xi_L) - y - \xi_L\sin\varphi + R(\xi_L)\cos\varphi\cos\psi_L \\ G(\hat\xi_R) - y - \xi_R\sin\varphi + R(\xi_R)\cos\varphi\cos\psi_R\end{pmatrix} = 0 \tag{9}$$

with profile functions R (wheel) and G (rail),see Fig. 1,

$$R(\xi) = \rho_0 + \tan\delta_0\,(a_0 - |\xi|) \qquad \text{for } a_0 - \Delta a < |\xi| < b_2\,;$$

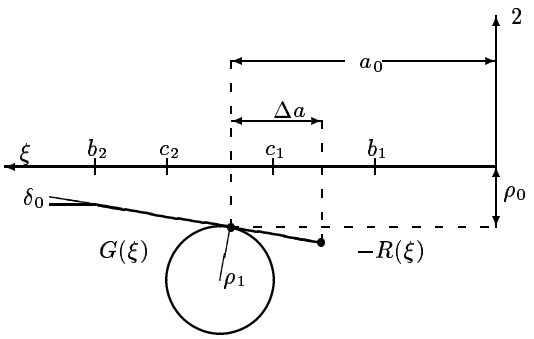
$$G(\hat\xi) = \sqrt{\rho_1^2 - \left(|\hat\xi| - a_0 - \rho_1\sin\delta_0\right)^2} - \rho_0 - \cos\delta_0\,\rho_1 \qquad \text{for } c_1 < |\hat\xi| < c_2\,.$$

ξ stands for the left or right coordinate $\xi_{L/R}$, respectively, and $\hat\xi$ is defined by

$$\hat\xi_{L|R} := x + \xi_{L|R}\cos\theta\cos\varphi + R(\xi_{L|R})\left(\cos\theta\sin\varphi\cos\psi_{L|R} - \sin\theta\sin\psi_{L|R}\right).$$

The index-1 constraints g_2 read

$$G'(\hat\xi_L)\left(R'(\xi_L)\sin\varphi + \cos\varphi\cos\psi_L\right) + R'(\xi_L)\cos\theta\cos\varphi$$



ρ_0 : nominal rolling radius	a_0 : nominal gauge/2
ρ_1 : radius track	b_1, b_2 : wheel boundaries
δ_0 : angle of wheel cone/2	c_1, c_2 : track boundaries

Figure 1: Profile functions (left side)

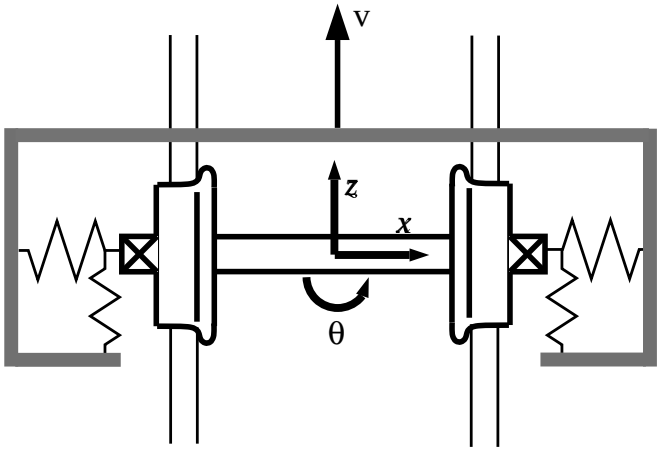
$$
\begin{aligned}
-\cos\theta\sin\varphi\cos\psi_L + \sin\theta\sin\psi_L &= 0 \\
R'(\xi_L)\sin\theta\cos\varphi - \sin\theta\sin\varphi\cos\psi_L - \cos\theta\sin\psi_L &= 0 \\
G'(\hat{\xi}_R)\left(R'(\xi_R)\sin\varphi + \cos\varphi\cos\psi_R\right) + R'(\xi_R)\cos\theta\cos\varphi & \qquad (10) \\
-\cos\theta\sin\varphi\cos\psi_R + \sin\theta\sin\psi_R &= 0 \\
R'(\xi_R)\sin\theta\cos\varphi - \sin\theta\sin\varphi\cos\psi_R - \cos\theta\sin\psi_R &= 0
\end{aligned}
$$

where $G'(\hat{\xi}_{L|R}) := \frac{d}{d\hat{\xi}_{L|R}} G(\hat{\xi}_{L|R})\,, R'(\xi_{L|R}) := \frac{d}{d\xi_{L|R}} R(\xi_{L|R})$.

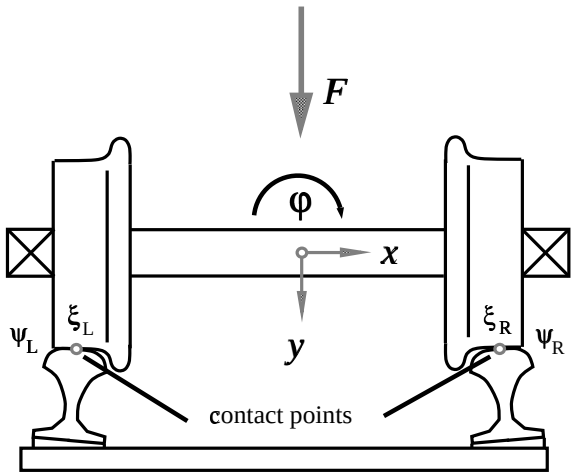
11.2.3 List of parameters

The following data is according to [Jas90] where a hardware bogie model scaled 1:4 is investigated.

	Parameter	Unit	
m_R	mass wheelset	kg	16.08
$\tilde{g}$	gravity constant	m/s^2	9.81
v_0	nominal velocity	m/s	30.0
F_A	propulsion force	N	0
L_A	propulsion moment	kg m^2	0
ω_0	nominal angular velocity	1/s	v_0/ρ_0
I_1	lateral moment of inertia	kg m^2	0.0605
I_2	vertical moment of inertia	kg m^2	0.366
m_A	mass of wagon body	kg	0.0
h	height of wagon body	m	0.0
c_x	spring constant	N/m	6400.0
c_z	spring constant	N/m	6400.0
x_l	width of wheelset/2	m	0.19
δ_0	cone angle/2	rad	0.0262
ρ_0	nominal radius	m	0.1
a_0	gauge/2	m	0.1506
ρ_1	radius track	m	0.06



(a)



(b)

Figure 2: The wheelset and the track. (a) View from above, (b) lateral cross section.

μ	friction coefficient		0.12
G	glide modul	N/m^2	$7.92 \cdot 10^{10}$
C_{11}	Kalker coefficient		4.72772197
C_{22}	Kalker coefficient		4.27526987
C_{23}	Kalker coefficient		4.97203505

11.3 Origin of the problem

The motion of a simple wheelset on a rail track exhibits a lot of the difficulties which occur in the simulation of contact problems in mechanics. The state space form approach for this class of problems requires simplifications and table look ups in order to eliminate the nonlinear constraints. The above example provides thus an alternative by using the DAE approach.

Figure 2 shows the mechanical model. The coordinates p denote the displacements and rotations of the wheelset with respect to the reference frame which is centered in the middle of the track. The wheelset is subjected to

- the gravity and centrifugal forces;

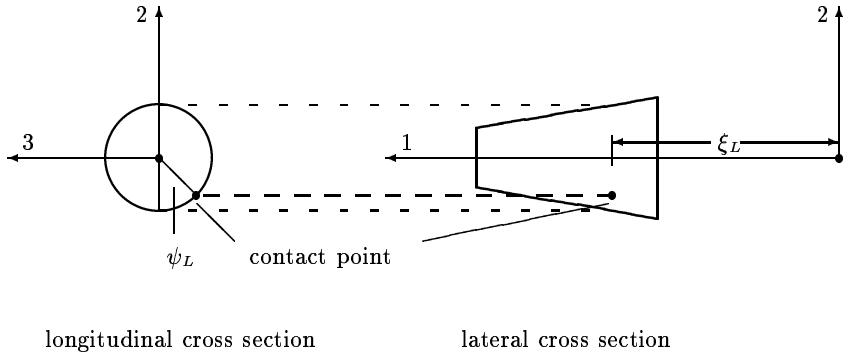


Figure 3: Shift angle and coordinate of contact point on the left side

- creep forces in the contact points of wheel and rail;
- forces of the wagon body, which is represented by a frame connected to the wheelset via springs and dampers and proceeding with constant speed v_0;
- constraint forces which enforce the contact of wheel and rail on both sides.

We are particularly interested in a complete and correct formulation of the nonlinear constraint equations. An elimination of the constraints without severe simplifications or the introduction of tables for the dependent variables is impossible. In this example thus a reduction to state space form involves various obstacles, whereas the DAE-formulation is straightforward.

By means of the profile functions R and G which describe the cross sections of wheel and rail depending on the contact points we first express the constraint equations as $0 = g_1$, see Figure 3. This enforces that the contact points of wheel and rail coincide on both sides. Additionally, we have to guarantee that wheel and rail do not intersect, which is accomplished by the conditions $0 = g_2$. Note that $\partial g_2/\partial q$ is regular, which means that we can apply formally the implicit function theorem to eliminate the additional contact variables q.

The constraint Jacobian follows from

$$0 = d/dt\, g_1(p,q) = \frac{\partial g_1}{\partial p}\dot{p} + \frac{\partial g_1}{\partial q}\dot{q} = \frac{\partial g_1}{\partial p}\dot{p} - \frac{\partial g_1}{\partial q}\left(\frac{\partial g_2}{\partial q}\right)^{-1}\frac{\partial g_2}{\partial p}\dot{p}$$

For this class of contact problems, however, it can be shown that $\partial g_1/\partial q\,\dot{q} \equiv 0$ whence the expression for the constraint Jacobian simplifies to $\partial g_1/\partial p$. The equations of motion of the wheelset are then derived by applying the formalism of Newton and Euler.

Remarks

- $N(p,q,\lambda) \in \mathbf{R}^2$ denotes the normal forces which act in the contact points. They are necessary to evaluate the creep forces.
- $\beta \in \mathbf{R}$ denotes the deviation of the angular velocity and is given by an additional differential equation.

11.4 Numerical solution of the problem

11.4.1 Solution at the endpoint:

y_1	$0.86355386965811 \cdot 10^{-2}$
y_2	$0.13038281022727 \cdot 10^{-4}$
y_3	$-0.93635784016818 \cdot 10^{-4}$
y_4	$-0.13642299804033 \cdot 10^{-1}$
y_5	$0.15292895005422 \cdot 10^{-2}$
y_6	$-0.76985374142666 \cdot 10^{-1}$
y_7	$-0.25151106429207 \cdot 10^{-3}$
y_8	$0.20541188079539 \cdot 10^{-2}$
y_9	-0.23904837703692
y_{10}	$-0.13633468454173 \cdot 10^{-1}$
y_{11}	-0.24421377661131
y_{16}	$-0.10124044903201 \cdot 10^{-1}$
y_{17}	$-0.56285630573753 \cdot 10^{-2}$
y_{14}	$0.37839614386969 \cdot 10^{-3}$
y_{15}	0.14173214964613
y_{12}	$-0.33666751972196 \cdot 10^{-3}$
y_{13}	-0.15949425684022

In the following, we investigate the dynamic behaviour of the wheelset model. Starting with an initial deflection in lateral direction (x-direction), the motion of the wheelset is simulated when running along a straight track. In [Jas90], a limit cycle was observed for this problem and the model data given above. This type of limit cycle, the so-called hunting motion, is a well known phenomenon in railway vehicle dynamics.

Fig. 4 shows the result of a simulation with DASSL applied to the index-2 formulation of the problem. The results are in good agreement with those given in [Jas90] obtained by a state space form approach and with measurements on a hardware model.

11.4.2 Statistic data of the simulation run

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

Solver: DASSL, latest release obtained from NETLIB, revised 910624.

atol=rtol=10^{10} for λ to exclude the Lagrange multipliers from error control.

Interval mode output after each $\Delta t = 0.1[s]$.

Initial conditions: corresponding to a lateral deflection of $x = 1.49[mm]$, see the Fortran routine.

A Jacobian of the problem is not supplied, DASSL used finite differences.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-4}	10^{-4}		0.16	5107	4265	10414	1396		12.84
	10^{-5}	10^{-5}		1.41	8671	7412	16237	1817		18.50
	10^{-6}	10^{-6}		2.27	14190	12485	24939	2704		28.18

11.4.3 Work-precision diagram

In Figure 5 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol = $10^{-(4+m/8)}$, $m = 0, \ldots, 16$; atol = rtol.

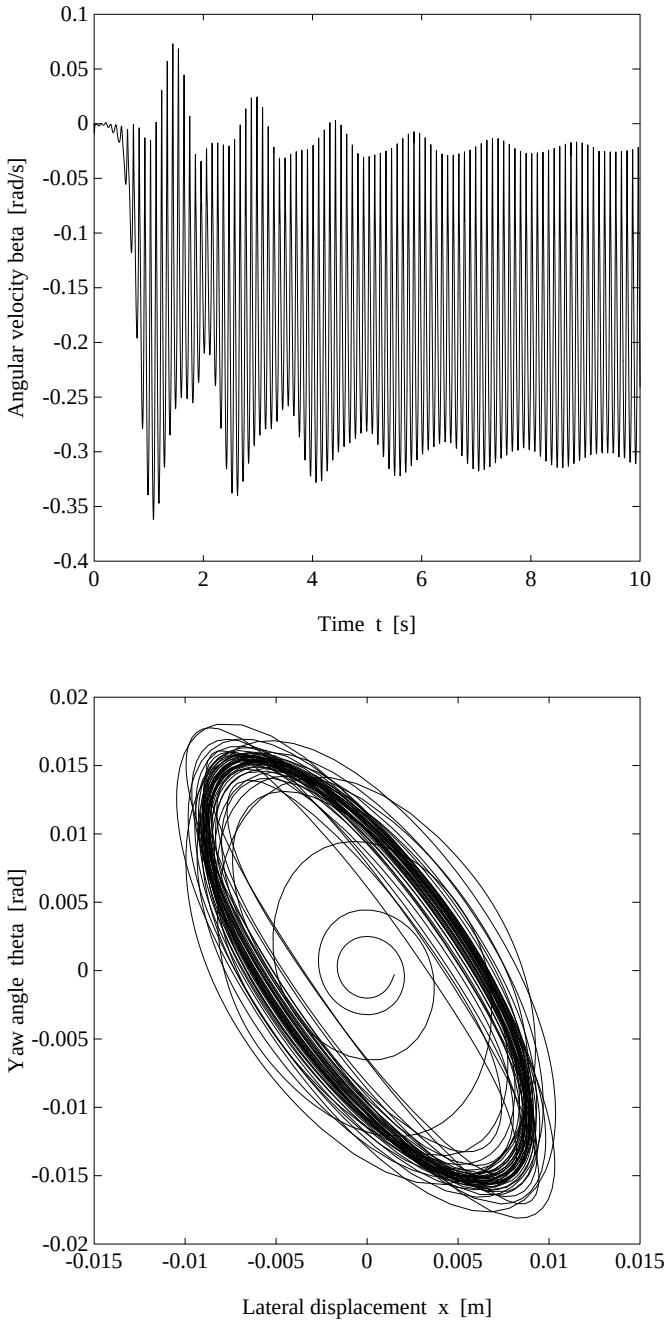


Figure 4: Limit cycle or 'hunting motion' of wheelset.

References

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[Jas87] A. Jaschinski. Anwendung der Kalkerschen Rollreibungstheorie zur dynamischen Simulation von Schienenfahrzeugen. Technical Report DFVLR 87-07, 1987.

[Jas90] A. Jaschinski. *On the Application of Similarity Laws to a Scaled Railway Bogie Model.* PhD thesis, Technische Universiteit Delft, 1990.

[SFR91] B. Simeon, C. Führer, and P. Rentrop. Differential-algebraic equations in vehicle system dynamics. *Surv. Math. Ind.*, 1:1–37, 1991.

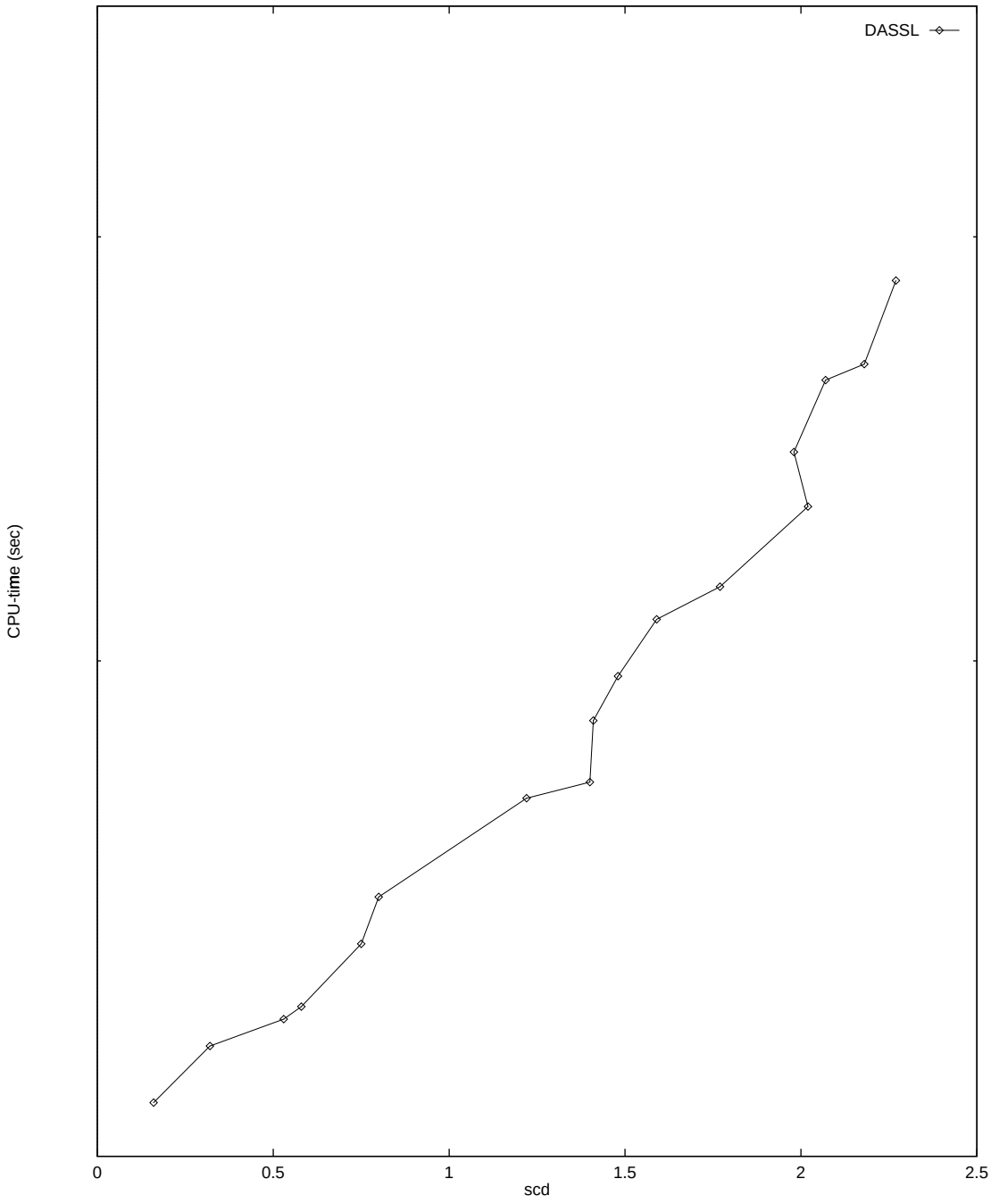


Figure 5: Work-precision diagram for Wheelset

Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Test set for IVP solvers

W.M. Lioen, J.J.B. de Swart and W.A. van der Veen

Department of Numerical Mathematics

NM-R9615 November 30, 1996

Report NM-R9615
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

12 The two bit adding unit

12.1 General Information

The problem is a stiff DAE of index 1, consisting of 175 differential equations and 175 algebraic equations. It has been contributed by M. Günther [Gün95, Gün96].

12.2 Mathematical description of the problem

The problem is of the form

$$\begin{aligned} \frac{dy}{dt} &= f(t,x), \\ 0 &= y - g(x), \end{aligned}$$

with

$$y(0) = y_0, \quad x(0) = x_0, \quad y, x \in \mathbf{R}^{175}, \quad 0 \le t \le 320.$$

The precise definition of the functions $f(t,x)$, $g(x)$ and the initial values y_0 and x_0 can be found in the code.

The differential-algebraic system is not very smooth: g is only a $\mathbf{C}^1$-function, while f is only $\mathbf{C}^0$.

12.3 Origin of the problem

The two bit adding unit computes the sum of two 2-base numbers (each two digits long) and a carry bit. These numbers are fed into the circuit in the form of input signals. As a result the circuit gives their sum coded as three output signals.

The two bit adding unit circuit is a digital circuit. These circuits are used to compute boolean expressions. This is accomplished by associating voltages with boolean variables. By convention the boolean is true if the voltage exceeds $2V$, and false if it is lower than $0.8V$. In between the boolean is undefined. Using CMOS technique, however, sharper bounds are possible for the representation of booleans.

Digital circuits that compute elementary logical operations are called gates. An example of a gate is the NAND gate of test problem 9. This circuit is used to compute the logical expression $\neg(V_1 \wedge V_2)$, where V_1 and V_2 are the booleans that are fed into the circuit as input signals.

The two bit adding unit is depicted in Figure 1. In this figure the symbols '&', '≥ 1' and a little white circle respectively stand for the AND, OR and NOT gate. A number of input signals and output signals enter and leave the circuit. Each signal is described by a time-dependent voltage and the boolean it represents. For these two quantities we shall use one symbol: the symbol of this boolean variable. Which one of the two quantities is meant by the symbol, is always clear from the context. With this convention, the input signals are referred to by the boolean variable they represent.

The circuit is designed to perform the addition

$$A_1\ A_0 + B_1\ B_0 + C_{in} = C\ S_1\ S_0.$$

The input signals representing the two numbers and the carry bit C_{in} are fed into the circuit at the nodes indicated by $\overline{\text{A0}}$, $\overline{\text{A1}}$, $\overline{\text{B0}}$, $\overline{\text{B1}}$ and Cin. They represent respectively the boolean variables $\bar{A}_0$, $\bar{A}_1$, $\bar{B}_0$, $\bar{B}_1$ and C_{in}. Here, a bar denotes the logical inversion. The output signals are delivered by the nodes indicated by S0, S1 and $\overline{\text{C}}$. They represent the boolean variables S_0, S_1 and $\bar{C}$.

In Figure 1, a number of boxes are drawn using dotted lines. Each of them represents one of the following gates: the NOR (first box to the left in the top-row), the ORANI gate (the box besides S_1), the NAND (the box besides the ORANI gate) and the ANDOI(the box at the bottom). The circuit diagram of the NAND-gate is given in test problem 9. For the circuit diagrams of the NOR, ANDOI and ORANI gate see Figures 2, 3 and 4. The logical expressions that are computed by them are given below.

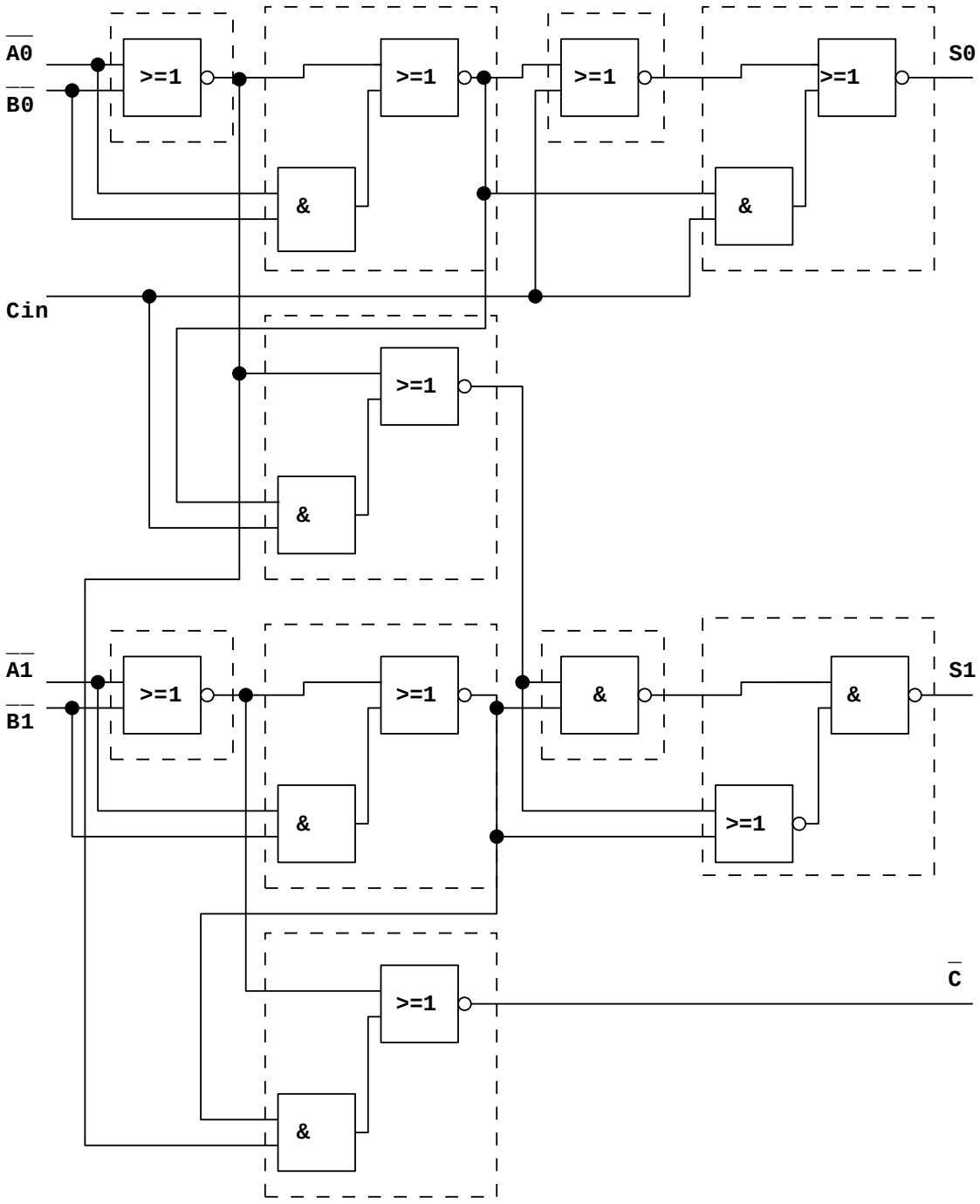


Figure 1: *Circuit diagram of the two bit adder (taken from [Gün95])*

Name	logical expression	# nodes	# times
NOR	$\neg(V_1 \vee V_2)$	$3 \cdot 4 + 1 = 13$	3
NAND	$\neg(V_1 \wedge V_2)$	$3 \cdot 4 + 2 = 14$	1
ANDOI	$\neg(V_1 \vee (V_2 \wedge V_3))$	$4 \cdot 4 + 2 = 18$	5
ORANI	$\neg(V_1 \wedge (V_2 \vee V_3))$	$4 \cdot 4 + 2 = 18$	1

The fourth column lists the number of times the gate occurs in the big circuit. The third column tabulates the number of nodes in the gate. These nodes consist of two types. The first type of nodes consists of the internal nodes of the transistors due to the MOS transistor model of Shichman and Hodges [SH68]. Each transistor has four internal nodes that are also the links between transistor and the rest of the circuit. The second type of nodes comprises the usual nodes that are used to link circuit components together. These nodes are indicated by a number placed inside a square. To prevent any misunderstanding, we remark that the big dots in Figures 2–4 do not represent nodes.

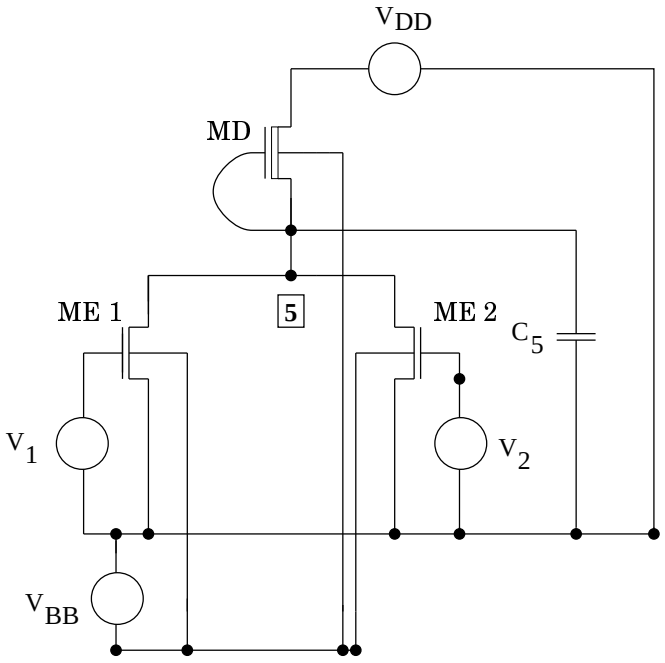


Figure 2: *Circuit diagram of the NOR gate (taken from [Gün95])*

The connection of a gate with the rest of the circuit consists of the input nodes and the output node of the gate. The input signals enter the gate at the nodes with symbol V_1, V_2 and V_3. The output signal leaves the gate from one of the numbered nodes. To ensure stability of the circuit, such an output node is always connected to a capacitance (we refer to the Fortran driver: `CLOAD` denoting the value of a load capacitance for the logical gates, and `COUT` for the output nodes S_0, S_1 and $\bar{C}$). Finally, three enhancement transistors are coupled with the ANDOI gate at the bottom for a correct treatment of C_{in}. This yields 12 internal nodes and two additional nodes, because the three transistors are coupled in series. Counting all nodes we have $3{\cdot}13{+}1{\cdot}14{+}5{\cdot}18{+}1{\cdot}18{+}14 = 175$ nodes.

Applying Kirchoff's law to all nodes yields a system of 175 equations. This system is an integral form DAE of the special form

$$A \cdot \dot{q}(V) = f(t, V).$$

The function q is a generally nonlinear function of node potentials V, which describes the charges stored in all charge storing elements [GDF96]. Assembling the charge flow at each node by an incidence matrix A, the dynamic part $A \cdot \dot{q}(V)$ equals the contribution of static currents denoted

by $f(t,V)$. If all load capacitances at the output nodes are nonzero, then the integral form DAE has differential index 0. If only one of the load capacitances equals zero, the generalized capacitance matrix $A \cdot \partial q(V)/\partial V$ is singular, yielding a system of differential index 1. This shows the regularization effects by applying additional capacitances. In the code, we use `CLOAD=0` and `COUT=2.0`.

To make this problem suitable for DASSL and RADAU5 the variable $Q = A \cdot q(V)$ of assembled charges is introduced leading to

$$\begin{aligned} \dot{Q} &= f(t,V), \\ 0 &= Q - Aq(V). \end{aligned}$$

This transformation of the integral form DAE into a linearly implicit system raises the differential index by one. However, in the case of singular load capacitances, no higher index effects are detected in the sense of an appropriate perturbation index [Gün96].

Some of the 175 variables have a special meaning. These are the voltage variables of the nodes that deliver the output signals. The output signals S_0, S_1 and $\bar{C}$ are given by respectively the variables x_{49}, x_{130} and x_{148}. Only these variables are of interest to the engineer.

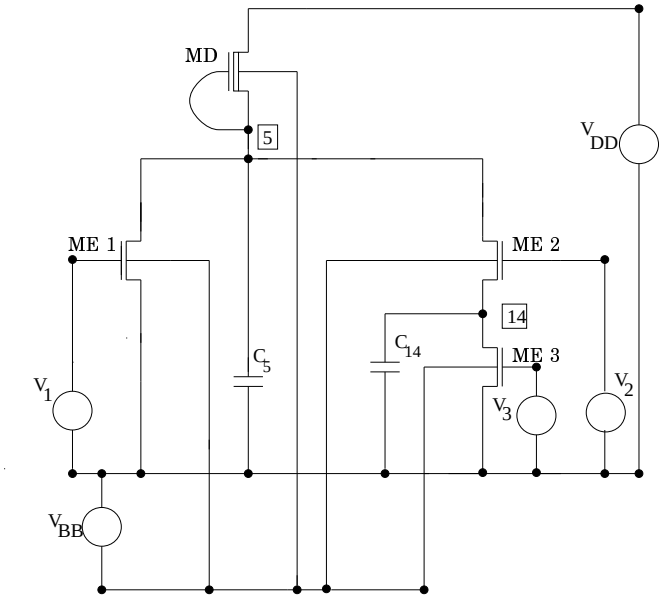


Figure 3: *Circuit diagram of the ANDOI gate (taken from [Gün95])*

In the next section we shall see the two bit adder in operation. Every 10 units of time the addition

$$A_1\ A_0 + B_1\ B_0 + C_{in} = C\ S_1\ S_0,$$

is carried out. The numbers that are added are represented by the input signals depicted in the Figures 5–7. The outcome of the addition is represented by output signals given in Figure 8. Often the output signals need time to adjust to changes in the input signal. Therefore, only during certain periods the sum is correctly represented by the output signals. The two bit adding unit has been designed in such a way that after each 10 units of time the output signal represents the sum correctly.

To see the two bit adding unit performing an addition let us see what happens at $t = 200$. Then the input signals read:

$$\bar{A}_0 = 0,\ \bar{A}_1 = 1,\ \bar{B}_0 = 0, \bar{B}_1 = 0,\ C_{in} = 1,$$

and the output signals are

$$S_0 = 1,\ S_1 = 0,\ \bar{C} = 0.$$

Recall, that a bar denotes the logical inverse. Clearly, the addition 01+11+1=101 has been carried out.

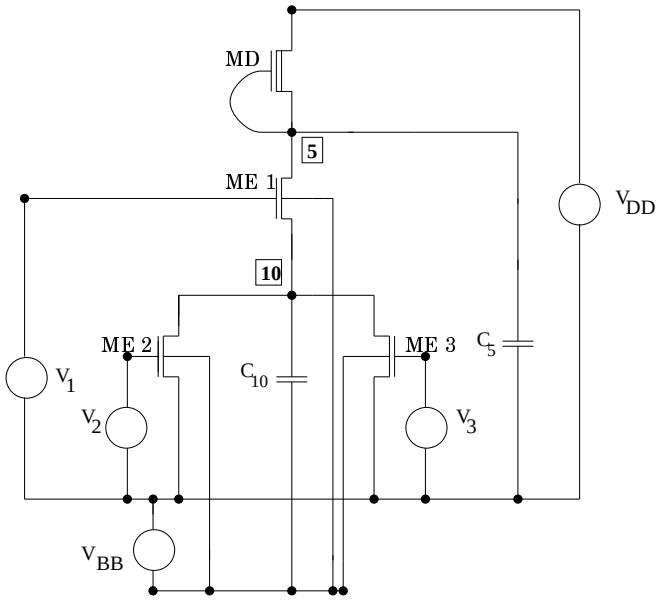


Figure 4: *Circuit diagram of the ORANI gate (taken from [Gün95])*

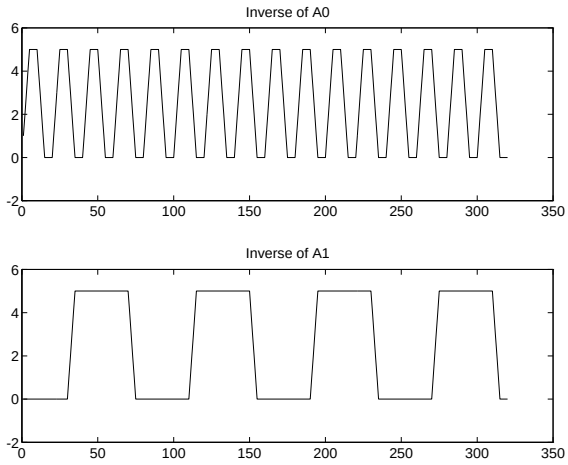


Figure 5: *The input signals* $\bar{A}_0$ *and* $\bar{A}_1$

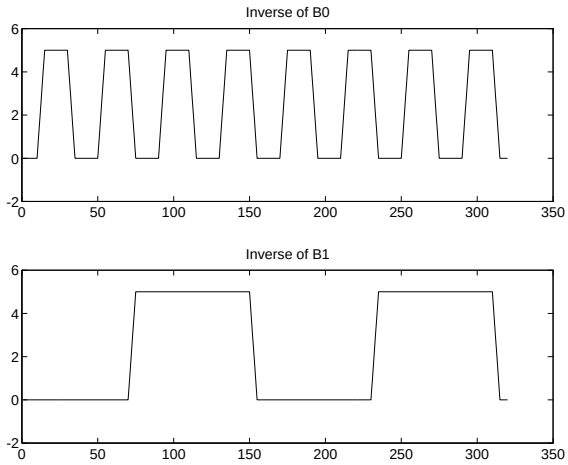


Figure 6: *The input signals* $\bar{B}_0$ *and* $\bar{B}_1$

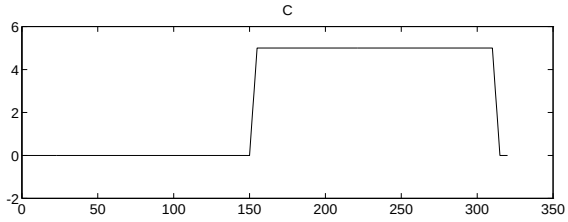


Figure 7: *The input signal* C

12.4 Numerical solution of the problem

M. Günther provided the source code defining the problem. In applying DASSL and RADAU5 the composite vector $(y^T(t), x^T(t))^T$ is used.
Remark: M. Günther also wrote a special purpose solver called CHORAL (CHarge-ORiented ALgorithm, [Gün95, Gün96]) for integrating equations of the form

$$\begin{aligned} \frac{dy}{dt} &= f(t,x), \\ 0 &= y - q(x). \end{aligned}$$

Most equations occurring in circuit analysis are of this form. In these equations the variables y and x represent respectively (assembled) charges and voltages. CHORAL is based on Rosenbrock-Wanner methods, while the special structure of the problem is exploited. The code eliminates the y variables, reducing the linear algebra work to solving systems of order 175 instead of 350. Correspondingly, a step size prediction and error control based directly on node potentials and currents is offered. For more information see

`http://www.mathematik.th-darmstadt.de/~guenther/Welcome.html`.

12.4.1 Solution at $t = 320$

For the solution at $t = 320$ we refer to the code. The voltages of the output signals are given by:

x_{49}	0.20404214956152
x_{130}	$0.49972184367837 \cdot 10$
x_{148}	0.20390916167369

12.4.2 Behaviour of the numerical solution

In Figure 8, we give plots of the voltages of the output signals x_{49}, x_{130} and x_{148}.

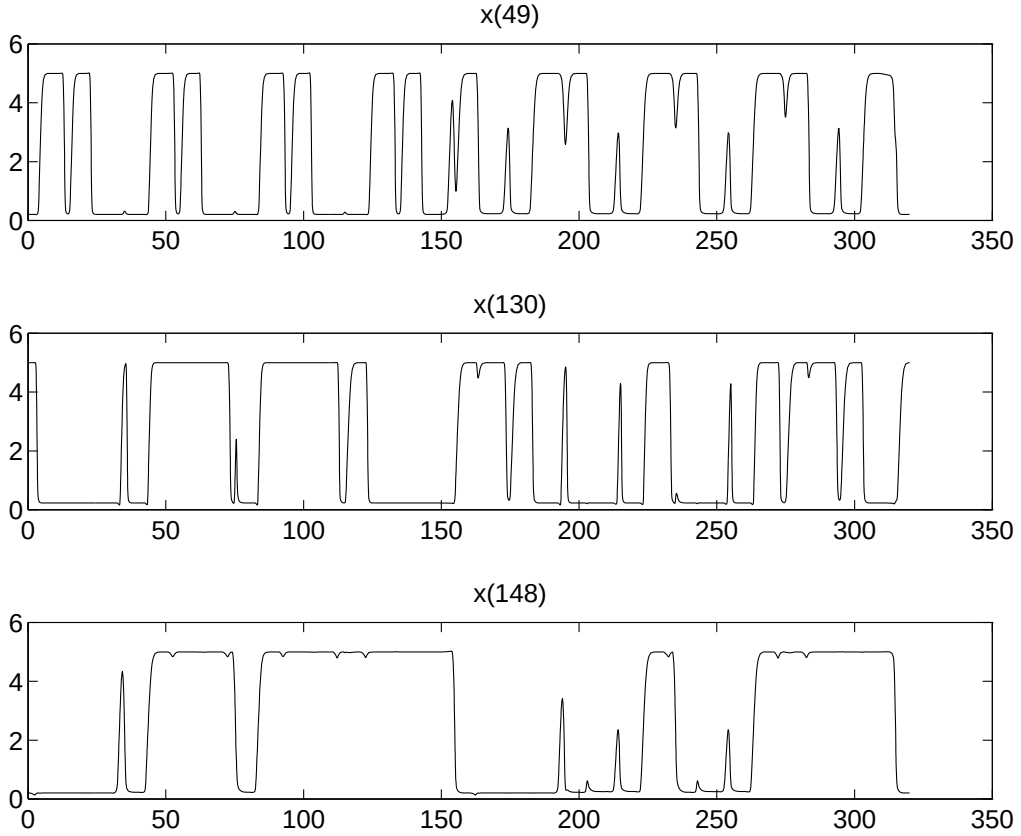


Figure 8: *The output signals S_0, S_1 and $\bar{C}$*

12.4.3 Run characteristics

To comply with the format of the standard drivers we applied DASSL and RADAU5 using error control for both the x and y. It would be more natural to use local error control only for the x variable because they are physically relevant, whereas y was introduced to make the problem suitable for DASSL and RADAU5. In computing the *scd* values only the output signals x_{49}, x_{130} and x_{148} were considered.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
DASSL	10^{-2}	10^{-2}		2.42	1423	1246	3148	521		587.47
	10^{-4}	10^{-4}		3.53	5465	5005	9526	892		1131.85
RADAU5	10^{-2}	10^{-2}	10^{-1}	3.68	1112	698	9722	692	1106	1320.04
	10^{-4}	10^{-4}	10^{-3}	4.28	2189	1609	17645	1442	2138	2602.30

12.4.4 Work-precision diagram

In Figure 9 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol $= 10^{-(2+m/8)}$, $m = 0, \ldots, 16$; atol = rtol; h0 $= 10 \cdot$ rtol for RADAU5.

References

[GDF96] M. Günther, G. Denk, and U. Feldmann. Modeling and simulating charge sensitive circuits. To appear in: Mathematical Modeling of Systems, 1996.

[Gün95] M. Günther. Ladungsorientierte Rosenbrock-Wanner-Methoden zur numerischen Simulation digitaler Schaltungen. Technical Report 168, Fortschritt-Berichte VDI Reihe 20, Düsseldorf, 1995.

[Gün96] M. Günther. Simulating digital circuits numerically – a charge-oriented ROW approach. Submitted for publication in Numer. Math., 1996.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[SH68] H. Shichman and D. A. Hodges. Insulated-gate field-effect transistor switching circuits. *IEEE J. Solid State Circuits*, SC-3:285–289, 1968.

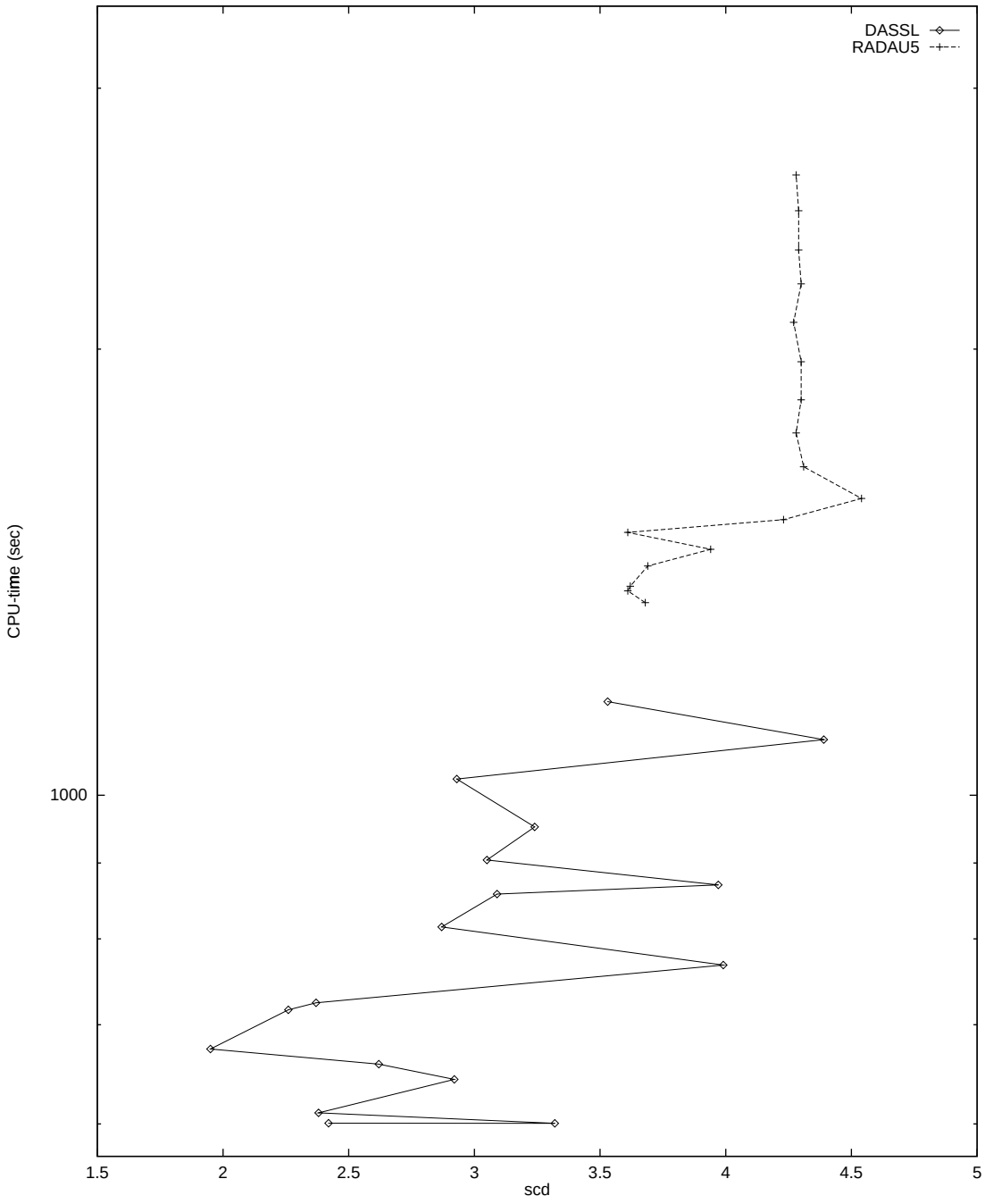


Figure 9: Work-precision diagram for The two bit adding unit

Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Test set for IVP solvers

W.M. Lioen, J.J.B. de Swart and W.A. van der Veen

Department of Numerical Mathematics

NM-R9615 November 30, 1996

Report NM-R9615
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

13 The car axis problem

13.1 General information

The problem is a stiff DAE of index 3, consisting of 8 differential and 2 algebraic equations. It has been taken from [Sch94]. Since not all initial conditions were given, we have chosen a consistent set of initial conditions.

13.2 Mathematical description of the problem

The problem is of the form

$$\begin{aligned} u' &= v, \\ Kv' &= f(u,\lambda), \\ 0 &= g(u). \end{aligned}$$

The equations are given by:

$$\begin{aligned}
\frac{dx_1}{dt} &= v_1 \\
\frac{dy_1}{dt} &= v_2 \\
\frac{dx_2}{dt} &= v_3 \\
\frac{dy_2}{dt} &= v_4 \\
\varepsilon^2\frac{M}{2}\frac{dv_1}{dt} &= (l_0 - \sqrt{(x_1-p(t))^2+(y_1-q(t))^2})\frac{x_1-p(t)}{\sqrt{(x_1-p(t))^2+(y_1-q(t))^2}} - 2\lambda_2(x_1-x_2) \\
\varepsilon^2\frac{M}{2}\frac{dv_2}{dt} &= (l_0 - \sqrt{(x_1-p(t))^2+(y_1-q(t))^2})\frac{y_1-q(t)}{\sqrt{(x_1-p(t))^2+(y_1-q(t))^2}} - \varepsilon^2\frac{M}{2} - 2\lambda_2(y_1-y_2) \\
\varepsilon^2\frac{M}{2}\frac{dv_3}{dt} &= (l_0 - \sqrt{x_2^2+y_2^2})\frac{x_2}{\sqrt{x_2^2+y_2^2}} - p(t)\lambda_1 + 2\lambda_2(x_1-x_2) \\
\varepsilon^2\frac{M}{2}\frac{dv_4}{dt} &= (l_0 - \sqrt{x_2^2+y_2^2})\frac{y_2}{\sqrt{x_2^2+y_2^2}} - \varepsilon^2\frac{M}{2} - q(t)\lambda_1 + 2\lambda_2(y_1-y_2) \\
0 &= p(t)x_2 + q(t)y_2 \\
0 &= (x_1-x_2)^2 + (y_1-y_2)^2 - l^2
\end{aligned}$$

The constants read

$$M = 10, \quad \varepsilon = 10^{-2}, \quad l = 1, \quad l_0 = 0.5.$$

The functions $p(t)$ and $q(t)$ are defined by

$$\begin{aligned} q(t) &= r\sin(\omega t), \\ p(t) &= \sqrt{l^2 - q^2(t)}, \end{aligned}$$

where the constants are given by $r = 0.1$ and $\omega = 10$. The initial conditions were chosen to be

$x_1 = 1$	$x_1' = -0.5$
$x_2 = 0$	$x_2' = -0.5$
$y_1 = 0.5$	$y_1' = 0$
$y_2 = 0.5$	$y_2' = 0$
$\lambda_1 = 0$	$\lambda_2 = 0$

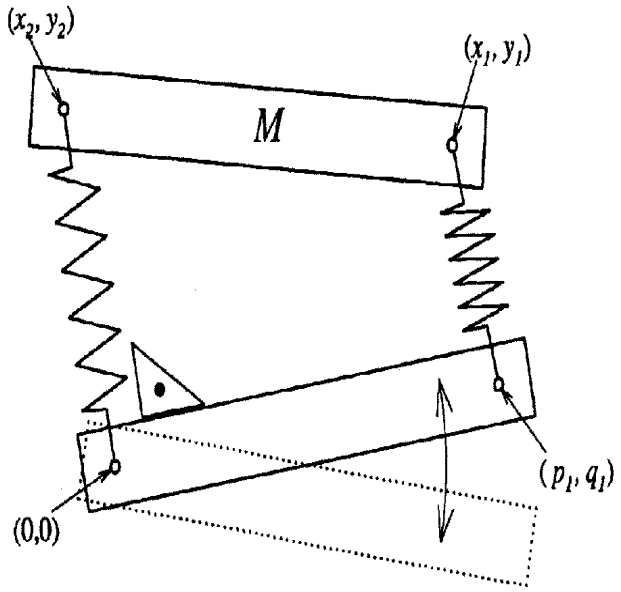


Figure 1: *Model of the car axis (taken from [Sch94])*

13.3 Origin of the problem

The problem models the axis of a car as depicted in Figure 1. In this model, the left wheel (at the origin $(0,0)$) rolls on a flat surface and the right wheel moves up and down in a sinusoidal way. Denoting the coordinates of the right wheel by $(p(t), q(t))$, we suppose that

$$\begin{aligned} q(t) &= r\sin(\omega t), \\ p(t) &= \sqrt{l^2 - q^2(t)}. \end{aligned}$$

Note that in Figure 1 the coordinates $(p(t), q(t))$ are denoted by $(p_1(t), q_1(t))$. This parameterization describes the situation where the right wheel rolls over equidistant hills of height r. The movement of the lower axis between $(0,0)$ and $(p(t), q(t))$ is carried over to the upper axis between (x_2, y_2) and (x_1, y_1) by two massless stiff springs with Hooke's constant $\frac{1}{\varepsilon^2}$ and length l_0. The movement of the mechanism has two constraints. First, the distance between (x_1, y_1) and (x_2, y_2) must remain constant and second that the left spring remains orthogonal to the lower axis. The equations for the motion of the points (x_1, y_1) and (x_2, y_2) are obtained by using Lagrangian mechanics. Scaling the Lagrange multipliers by ε^2 yields the 10 equations given in the previous section.

13.4 Numerical solution of the problem

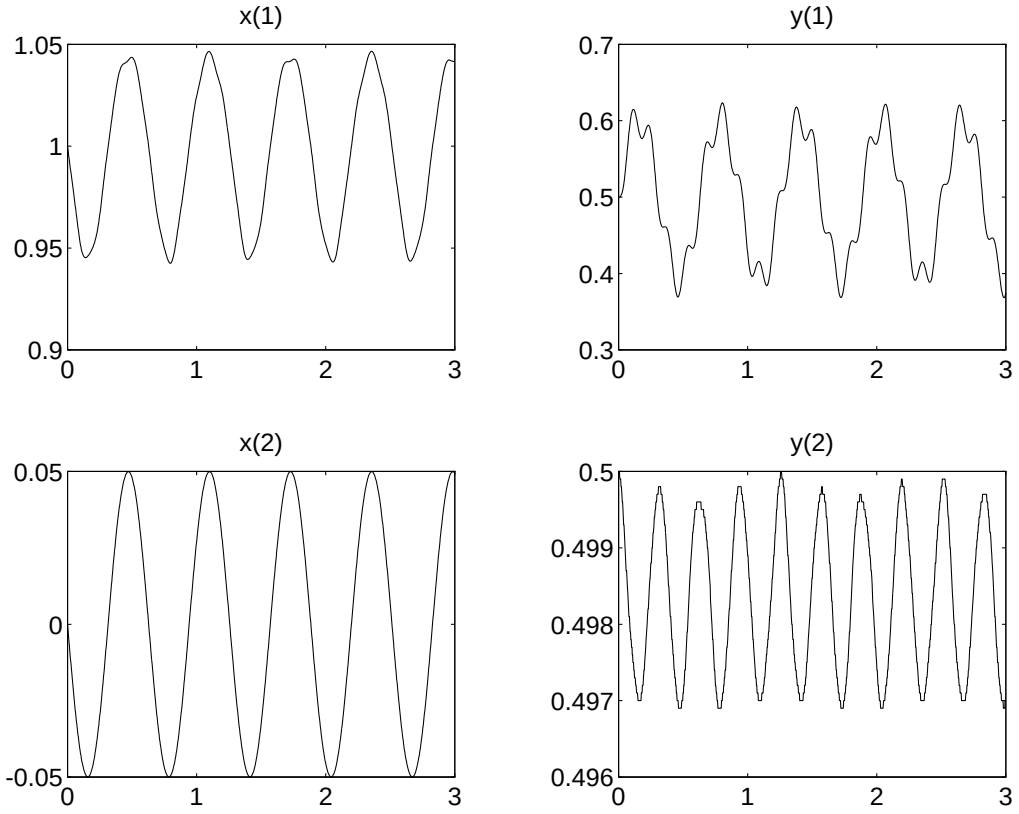
The code was run with RADAU5 only. Other codes cannot handle the problem, because the index is 3.

13.4.1 Solution at the endpoint

y_1	$0.10417425246881 \cdot 10$
y_2	0.37391102581670
y_3	$0.49345578498447 \cdot 10^{-1}$
y_4	0.49698946094432
y_5	$0.17556820846051 \cdot 10^{-1}$
y_6	0.77034106299374
y_7	$-0.77058369940007 \cdot 10^{-1}$
y_8	$0.74468523096869 \cdot 10^{-2}$
y_9	$0.47368244493243 \cdot 10^{-2}$
y_{10}	$0.11046501158750 \cdot 10^{-2}$

13.4.2 Behaviour of the numerical solution

The following plots show the motion of the upper car axis.



13.4.3 Run characteristics of RADAU5

We have refrained from supplying the Jacobian analytically, therefore RADAU5 was run using a finite difference approximation. The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
RADAU5	10^{-4}	10^{-4}	10^{-4}	0.55	116	115	952	113	116	0.15
	10^{-7}	10^{-7}	10^{-7}	2.39	581	580	4015	557	576	0.68
	10^{-10}	10^{-10}	10^{-10}	4.80	3189	3188	21431	2698	2821	3.49

13.4.4 Work-precision diagram

In Figure 2 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol = $10^{-(4+m/4)}$, $m = 0, \ldots, 24$; atol = rtol; h0 = rtol.

References

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[Sch94] S. Schneider. *Intégration de systèmes d'équations différentielles raides et différentielles-algébriques par des méthodes de collocations et méthodes générales linéaires.* PhD thesis, Université de Genéve, 1994.

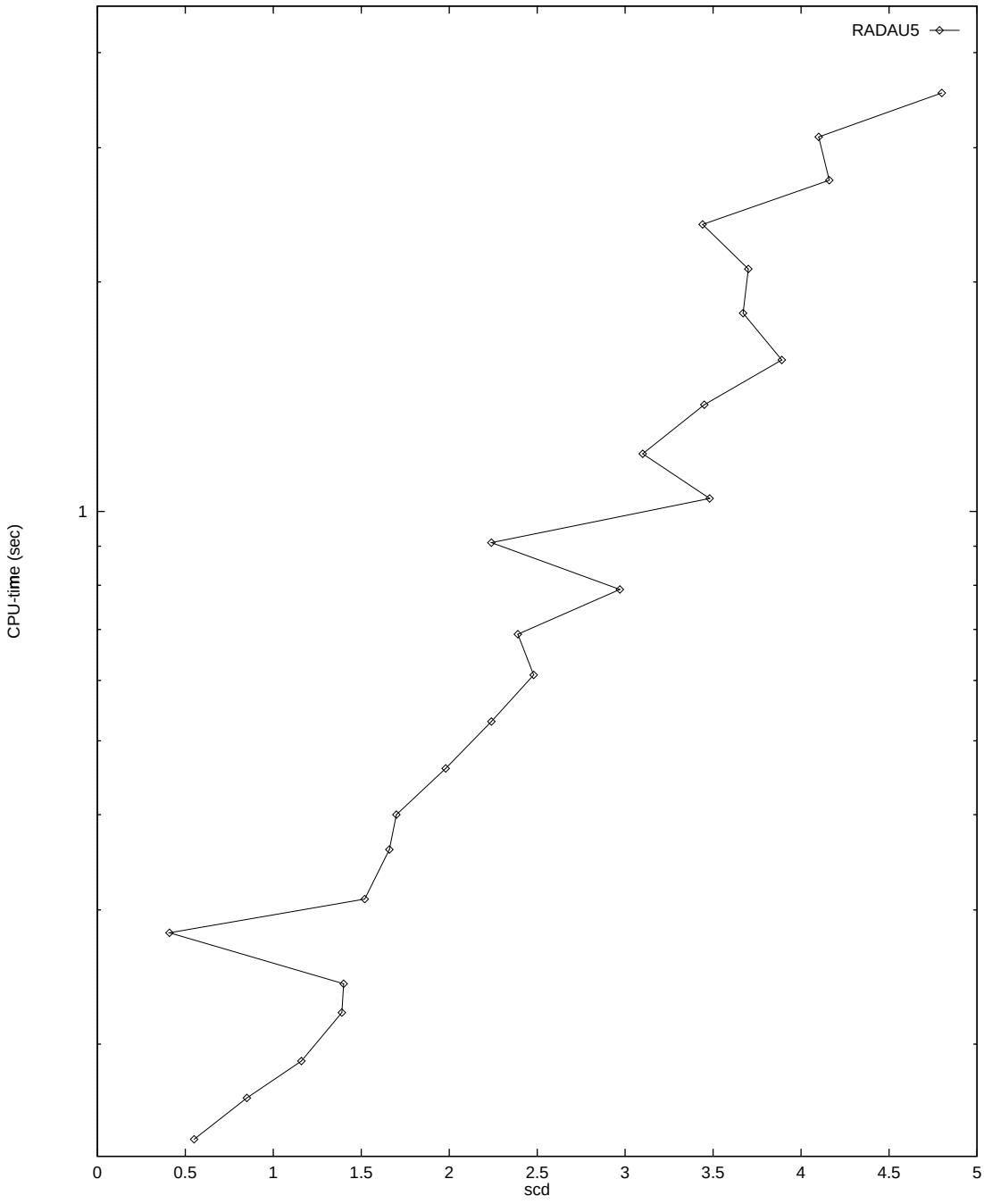


Figure 2: Work-precision diagram for Car Axis problem

Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Test set for IVP solvers

W.M. Lioen, J.J.B. de Swart and W.A. van der Veen

Department of Numerical Mathematics

NM-R9615 November 30, 1996

Report NM-R9615
ISSN 0169-0388

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

14 Fekete problem

14.1 General information

The problem is an index 2 DAE from mechanics. The dimension is $8N$, where N is an user supplied integer. The numerical tests shown here correspond to $N = 20$. The problem is of interest for the computation of the elliptic Fekete points [Par95]. The parallel-IVP-algorithm group of CWI contributed this problem to the test set, in collaboration with W. J. H. Stortelder.

14.2 Mathematical description of the problem

The problem is of the form

$$M\frac{dy}{dt} = f(y), \quad y(0) = y_0, \tag{1}$$

with

$$y \in \mathbf{R}^{8N}, \quad 0 \le t \le t_{\text{end}}.$$

Here, $t_{\text{end}} = 1000$ and M is the (constant) mass matrix given by

$$M = \begin{pmatrix} I_{6N} & 0 \\ 0 & 0 \end{pmatrix},$$

where I_{6N} is the identity matrix of dimension $6N$. Since the function f is too voluminous to be printed here, we refer to the Fortran codes and the next section for its definition.

The initial vector y_0 is given by $(y_{0,i})$, where

$$\begin{array}{lcll}
y_{0,i} & = & 0 & \text{for} \quad i = 1, 2 \\
y_{0,3} & = & 1 & \\
y_{0,3(j-1)+1} & = & \cos(2\pi(j-1)/(N-1)) & \left.\begin{array}{l}\\\\\end{array}\right\} \quad \text{for} \quad j = 2, \ldots, N \\
y_{0,3(j-1)+2} & = & \sin(2\pi(j-1)/(N-1)) & \\
y_{0,3(j-1)+3} & = & 0 & \\
y_{0,i} & = & 0 & \text{for} \quad i = 3N+1, \ldots, 6N \\
y_{0,6N+j} & = & \frac{1}{2}\langle p_j(0), \widehat{f}_j \rangle & \text{for} \quad j = 1, \ldots, N \\
y_{0,i} & = & 0 & \text{for} \quad i = 7N+1, \ldots, 8N.
\end{array}$$

Here,

$$p_j = \begin{pmatrix} y_{3(j-1)+1} \\ y_{3(j-1)+2} \\ y_{3(j-1)+3} \end{pmatrix}, \quad \widehat{f}_j = \begin{pmatrix} f_{3N+3(j-1)+1}((p(0), 0, \ldots, 0)^T) \\ f_{3N+3(j-1)+2}((p(0), 0, \ldots, 0)^T) \\ f_{3N+3(j-1)+3}((p(0), 0, \ldots, 0)^T) \end{pmatrix}, \tag{2}$$

and $p = (y_1, y_2, \ldots, y_{3N})^T$.

14.3 Origin of the problem

This problem is of interest for the computation of the elliptic Fekete points. Let us define the unit sphere in $\mathbf{R}^3$ by $\mathcal{S}^2$ and for any configuration $x := (x_1, x_2, \ldots, x_N)^T$ of points $x_i \in \mathcal{S}^2$, the function

$$V(x) := \prod_{i<j} \|x_i - x_j\|_2.$$

We denote the value of x for which V reaches its global maximum by $\widehat{x} = (\widehat{x}_1, \ldots, \widehat{x}_N)$. The points $\widehat{x}_1, \widehat{x}_2, \ldots, \widehat{x}_N$ are called the elliptic Fekete points of order N. For example, for $N = 4$, the points of the optimal solution form a tetrahedron. But, in case of 8 points, intuition fails; the elliptic Fekete points do not form a cube in this case. A cube where, for example, the upper plane is rotated over 45° with respect to the bottom plane, gives already a larger value of V. It turns out (see e.g. [Par95]) that $\widehat{x}$ is difficult to compute as solution of an global optimization problem.

For reasons that will become clear later, we differentiate $\log(V)$ with respect to x_k and apply the method of Lagrange multipliers to arrive at

$$\nabla_k \log(V(x)) \mid_{x=\widehat{x}} = \sum_{j \neq k} \frac{\widehat{x}_k - \widehat{x}_j}{\|\widehat{x}_k - \widehat{x}_j\|_2^2} = \zeta_k \widehat{x}_k, \tag{3}$$

where the ζ_k are Lagrange multipliers.

We now discuss the Fekete points from another point of view. Consider on $\mathcal{S}^2$ a number of N particles. We invoke a repulsive force on the particles, by which the particles will start to move away from each other. We compute the configuration of the particles as function of time, given that the particles cannot leave the unit sphere.

We denote the position in Cartesian coordinates of particle i at time t by $p_i(t)$ and the configuration of N points at time t by $p(t) = (p_1(t), \ldots, p_N(t))^T$. If we impose an adhesion force on the particles, then the particles will reach a stationary configuration at $t = t_{\text{stat}}$, which will be denoted by $\widehat{p} := (\widehat{p}_1, \widehat{p}_2, \ldots, \widehat{p}_N)$, where $\widehat{p}_i := p_i(t_{\text{stat}})$. The repulsive force on particle i caused by particle j is defined by

$$F_{ij} = \frac{p_i - p_j}{\|p_i - p_j\|_2^\gamma}. \tag{4}$$

Note that the choice $\gamma = 3$ can be interpreted as an electrical force working on particles with unit charge. If we choose $\gamma = 2$, then the energy of configuration $p(t)$ is given by

$$E(p(t)) = \sum_{i<j} \|p_i(t) - p_j(t)\|_2^{-1}.$$

Since the energy has a local minimum in the final configuration $\widehat{p}$, we know that

$$\nabla_k E(p) \mid_{p=\widehat{p}} = \sum_{j \neq k} \frac{\widehat{p}_k - \widehat{p}_j}{\|\widehat{p}_k - \widehat{p}_j\|_2^2} = \xi_k \widehat{p}_k, \tag{5}$$

where the ξ_k represent Lagrange multipliers. Comparing (3) and (5) tells us that computing $\widehat{p}$ for $\gamma = 2$ gives the elliptic Fekete points. For more details on elliptic Fekete points, we refer to [Par95], [SS93] and [SSP96].

We now explain how the computation of $\widehat{p}$ can be obtained as solution of an index 2 DAE. The configuration $p(t)$ p satisfies the equations

$$\begin{aligned} p' &= q, & (6)\\ q' &= g(p,q) + G^T(p)\lambda, & (7)\\ 0 &= \phi(p), & (8) \end{aligned}$$

where q is the velocity vector, $G = \partial\phi/\partial p$ and $\lambda \in \mathbf{R}^N$. The function $\phi : \mathbf{R}^{3N} \to \mathbf{R}^N$ is the constraint, which states that the particles cannot leave the unit sphere:

$$\phi_i(p) = p_{i,1}^2 + p_{i,2}^2 + p_{i,3}^2 - 1.$$

The function $g : \mathbf{R}^{6N} \to \mathbf{R}^{3N}$ is given by $g = (g_i)$, $i = 1, \ldots, N$, where

$$g_i(p,q) = \sum_{j \neq i} F_{ij}(p) + A_i(q),$$

where F_{ij} is given by (4). The function A_i is the adhesion force working on particle i, given by the formula

$$A_i = -\alpha q_i.$$

Here, α is valued 0.5. Without this adhesion force, the particles would periodically oscillate.

The term $G^T(p)\lambda$ in (7) represents the normal force which keeps the particle on $\mathcal{S}^2$.

The equations (6), (7), (8) form a differential algebraic system of index 3. To arrive at a more stable formulation of the problem, we stabilize the constraint (see [BCP89, p. 153]) by replacing (6) by

$$p' = q + G^T(p)\mu, \tag{9}$$

where $\mu \in \mathbf{R}^N$, and appending the differentiated constraint

$$0 = G(p)q. \tag{10}$$

The system (9), (7), (8), (10) is now of index 2; the variables p and q are of index 1, the variables λ and μ of index 2. We cast the system in the form (1) by setting $y = (p, q, \lambda, \mu)^T$ and $f(y) = f(p, q, \lambda, \mu) = (q + G^T\mu, g + G^T\lambda, \phi, Gq)^T$, where p_i is in Cartesian coordinates.

The initial positions $p_i(0)$ are all located on the intersection of $\mathcal{S}^2$ and the horizontal plane through the origin, except for the first particle, which is initially in $(0, 0, 1)$. Choosing $q(0) = 0$ yields $\mu(0) = 0$ and $\phi_i'(0) = \langle 2p_i(0), q_i(0)\rangle = 0$. Consequently,

$$\begin{aligned} \phi_i''(0) &= \langle 2p_i(0), q_i'(0)\rangle \\ &= \langle 2p_i(0), g_i(p(0), q(0)) + 2\lambda_i(0)p_i(0)\rangle. \end{aligned}$$

Requiring $\phi_i''(0) = 0$ gives

$$\lambda_i(0) = -\frac{\langle p_i(0), g_i(p(0), q(0))\rangle}{2\langle p_i(0), p_i(0)\rangle} = -\frac{1}{2}\langle p_i(0), g_i(p(0), q(0))\rangle.$$

For $N \leq 20$, $t_{\text{stat}} \leq 1000$, therefore we chose $t_{\text{end}} = 1000$.

14.4 Numerical solution of the problem

All the tests concern the case with $N = 20$. Solving the problem numerically leads to a phenomenon that one might call numerical bifurcation. Assume that two particles p_i and p_j are close to each other at time t_1 with $p_{i,1}(t_1) > p_{j,1}(t_1)$. It may happen that the numerical integration method applied with error tolerance τ computes a new stepsize h_τ such that $p_{i,1}(t + h_\tau) > p_{j,1}(t + h_\tau)$, whereas this method applied with error tolerance $\tilde{\tau}$ results in a stepsize $h_{\tilde{\tau}}$ for which $p_{i,1}(t + h_{\tilde{\tau}}) < p_{j,1}(t + h_{\tilde{\tau}})$. This means that for different error tolerances, the numerical integration method may compute paths of particles that differ significantly. The occurrence of this phenomenon is irrespective of the scale of the error tolerance. However, modulo rotations, the final configuration is the same within the accuracy of the numerical integration method for different values of the error tolerance. This leads us to measuring the precision of the numerical solution by computing the accuracy of the quantity V, defined by

$$V(y(t)) := \prod_{i<j} \|p_i - p_j\|_2.$$

Here, p_j is the same as in (2). Consequently, the scd value stands for

$$\text{scd} = -\log_{10}((V(y_{\text{num}}(t_{\text{end}})) - V(y_{\text{ref}}(t_{\text{end}})))/V(y_{\text{ref}}(t_{\text{end}}))),$$

where y_{num} is the numerical solution and y_{ref} the reference solution.

14.4.1 Solution at $t = 1000$:

The value of $V(y_{\text{ref}})$ in t_{end} reads $0.2862435499558360 \cdot 10^{24}$.

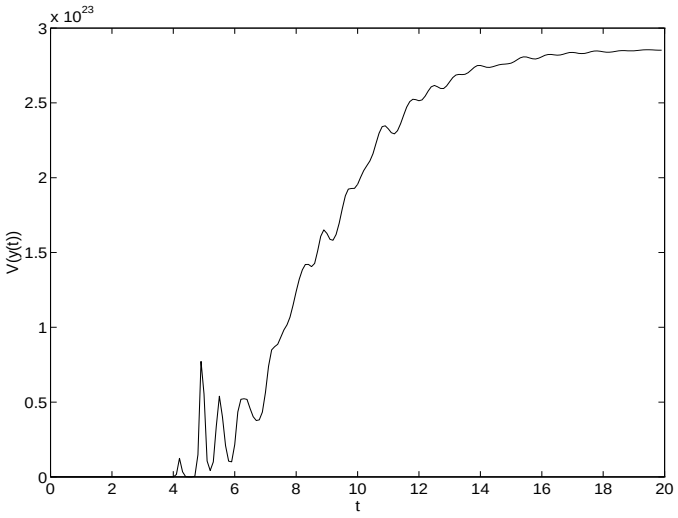
14.4.2 Run characteristics

The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`.

solver	rtol	atol	h0	scd	steps	accept	# f	# Jac	# LU	CPU
RADAU5	10^{-2}	10^{-2}	10^{-2}	5.73	63	53	632	50	63	60.93
	10^{-3}	10^{-3}	10^{-3}	7.66	84	79	701	76	84	68.63
	10^{-4}	10^{-4}	10^{-4}	8.23	146	138	1063	131	146	108.20

14.4.3 Behaviour of the numerical solution

The following plot shows the behaviour of $V(y_{\mathrm{ref}}(t))$ on [0,20], as computed by RADAU5 with rtol = atol = h0 = 10^{-10}:



14.4.4 Work-precision diagram

In Figure 1 we present a work-precision diagram (cf. [HW91, pp. 166–167, 324–325]). The runs were performed on a SGI workstation, an *Indy* with a 100 MHz R4000SC processor, using the Fortran 77 compiler with optimization: `f77 -O`. We used: rtol = $10^{-(2+m/16)}$, $m = 0, \ldots, 32$; atol = rtol; h0 = rtol.

References

[BCP89] K. E. Brenan, S. L. Campbell, and L. R. Petzold. *Numerical Solution of Initial–Value Problems in Differential–Algebraic Equations.* North–Holland, New York – Amsterdam – London, 1989.

[HW91] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-algebraic Problems.* Springer-Verlag, 1991.

[Par95] P. M. Pardalos. An open global optimization problem on the unit sphere. *Journal of Global Optimization*, 6:213, 1995.

[SS93] M. Schub and S. Smale. Complexity of Bezout's theorem III. Condition number and packing. *Journal of Complexity*, 9:4–14, 1993.

[SSP96] W. J. H. Stortelder, J. J. B. de Swart, and J. D. Pinter. Computation of the elliptic Fekete points. *In preparation*, 1996.

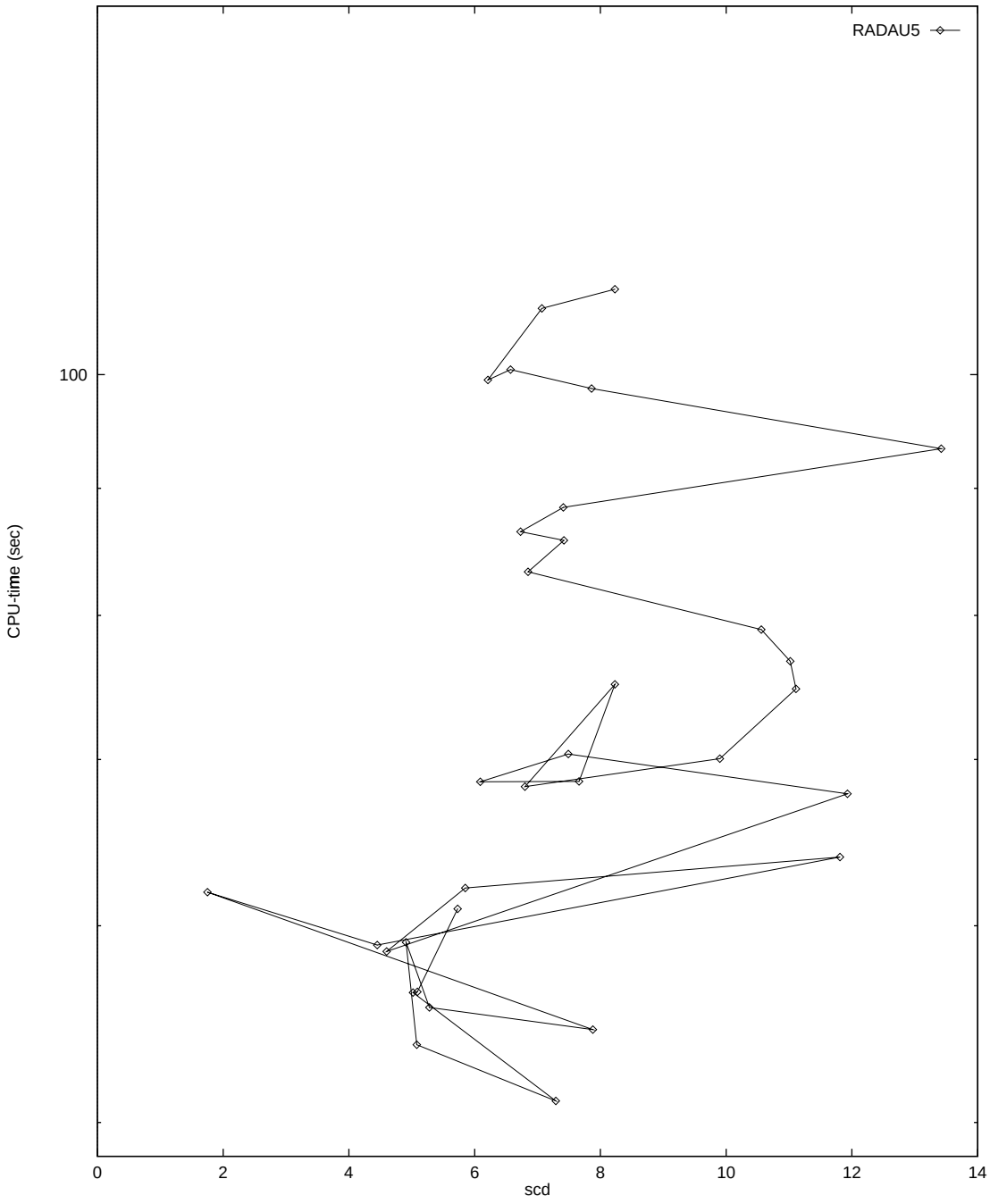


Figure 1: Work-precision diagram for Fekete Problem