8 Solver RADAU5

8.1 General information
Authors: E. Hairer and G. Wanner
last update: January 18, 2002
language: Fortran 77
availability: the code RADAU5 is freely available (in the public domain)
official link: http://www.unige.ch/~hairer/prog/stiff/radau5.f
problem type: ODEs and DAEs of index less than or equal to 3
IVP testset files: solver: radau5.f
driver: radau5d.f
auxiliary files: radaua.f (auxiliary linear algebra routines)

8.2 Numerical method
The code RADAU5 uses an implicit Runge-Kutta method (Radau IIa) of order 5 (three stages) with
step size control and continuous output. It is written for problems of the form $M y' = f(t, y)$ with
a possibly singular matrix $M$. It is therefore also suitable for the solution of differential-algebraic
problems.

8.3 Implementation details
Nonlinear systems are solved by a simplified Newton iteration. A similarity transformation on the
inverse of the Butcher array is performed in order to reduce the computational cost associated to
the solution of linear systems (see [HW96], page 121) so that, each time the Jacobian is updated, a
factorization of one real and one complex matrix of the same dimension as that of the continuous
problem is needed.

8.4 How to solve test problems with RADAU5
Compiling

    f90 -o dotest radau5d.f problem.f radau5.f radaua.f report.f,

will yield an executable dotest that solves the problem, of which the Fortran routines in the format
described in Section IV.3 are in the file problem.f.

As an example, we perform a test run, in which we solve problem HIRES. Figure I.8.1 shows what
one has to do.

References

[HW96] E. Hairer and G. Wanner. Solving Ordinary Differential Equations II: Stiff and Differential-
Test Set for IVP Solvers (release 2.3)

Solving Problem HIRES using RADAU5

User input:

give relative error tolerance: 1d-4

give absolute error tolerance: 1d-4

give initial stepsize: 1d-4

Numerical solution:

<table>
<thead>
<tr>
<th>solution component</th>
<th>scd</th>
<th>ignore</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mixed</td>
<td>abs</td>
</tr>
<tr>
<td>y( 1) = 0.748515248440879E-003</td>
<td>4.94</td>
<td>4.94</td>
</tr>
<tr>
<td>y( 2) = 0.1464912389469645E-003</td>
<td>5.65</td>
<td>5.65</td>
</tr>
<tr>
<td>y( 3) = 0.610142628653334E-004</td>
<td>5.67</td>
<td>5.67</td>
</tr>
<tr>
<td>y( 4) = 0.1196763210067838E-002</td>
<td>4.68</td>
<td>4.68</td>
</tr>
<tr>
<td>y( 5) = 0.273188990794999E-002</td>
<td>3.46</td>
<td>3.46</td>
</tr>
<tr>
<td>y( 6) = 0.7347017643277632E-002</td>
<td>2.96</td>
<td>2.96</td>
</tr>
<tr>
<td>y( 7) = 0.3074620885907540E-002</td>
<td>3.65</td>
<td>3.65</td>
</tr>
<tr>
<td>y( 8) = 0.2625379114092413E-002</td>
<td>3.65</td>
<td>3.65</td>
</tr>
</tbody>
</table>

used components for scd 8 8 8
scd of Y (maximum norm) 2.96 2.96 0.75
using mixed error yields mescd 2.96
using relative error yields scd 0.75

Integration characteristics:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>number of integration steps</td>
<td>38</td>
</tr>
<tr>
<td>number of accepted steps</td>
<td>31</td>
</tr>
<tr>
<td>number of f evaluations</td>
<td>295</td>
</tr>
<tr>
<td>number of Jacobian evaluations</td>
<td>20</td>
</tr>
<tr>
<td>number of LU decompositions</td>
<td>36</td>
</tr>
</tbody>
</table>

CPU-time used: 0.0010 sec

Figure I.8.1: Example of performing a test run, in which we solve problem HIRES with RADAU5. The experiment was done on an ALPHAserver DS20E, with a 667MHz EV67 processor. We used the Fortran 90 compiler f90 with the optimization flag -05.