Analysis of Math003 case

# Reference Solution

This test case has an exact solution:

**Time [s] x1 x2 x3 x4**

0 0 0 0 0

0.99999999 0 0 0 0

1 1 0 3 0

1.416666657 1 0 3 2.5

1.41667 1 0 0 2.5

1.99999999 1 0 0 2.5

2 0 0 0 2.5

2.99999999 0 0 0 2.5

3 0 1 -3 2.5

3.83333333 0 1 -3 -2.5

3.83334 0 1 0 -2.5

3.99999999 0 1 0 -2.5

4 0 0 0 -2.5

4.99999999 0 0 0 -2.5

5 1 0 3 -2.5

5.83333333 1 0 3 2.5

5.83334 1 0 0 2.5

5.99999999 1 0 0 2.5

6 1 1 0 2.5

10 1 1 0 2.5

## Calculation with Modelica

The test case can be solved with Modelica alone, yielding the correct solution within the selected tolerances. Steps may not show as expected, depending on output frequency selection.

# Algorithm: Gauss-Jacobi (FMI v1, fixed step, no iteration)

Step size = 0.01 s, that means 1000 evals + 1 for the initial time point.

------------------------------------------------------------------------------

Wall clock time = 104.804 ms

------------------------------------------------------------------------------

Output writing = 54.309 ms

Master-Algorithm = 16.726 ms 1001

Convergence failures = 0

Error test time and failure count = 0.000 ms 0

------------------------------------------------------------------------------

Part1 doStep = 5.311 ms 1001

getState = 0.000 ms 0

setState = 0.000 ms 0

Part2 doStep = 5.475 ms 1001

getState = 0.000 ms 0

setState = 0.000 ms 0

Part3 doStep = 4.323 ms 1001

getState = 0.000 ms 0

setState = 0.000 ms 0

------------------------------------------------------------------------------

## Time event problem



Integration interval [0.99,1.00]:

* [t=0.99,1.00] 🡪 Part1 🡪 [x1=1, x2=0]
* [x1=**0**, x2=0] 🡪 Part2 🡪 [x3=0]
* [x1=1, x2=0,**x3=0**] 🡪 Part3 🡪 [x4=0]

Integration interval [1.0,1.01]:

* [t=1..1,01] 🡪 Part1 🡪 [x1=1, x2=0]
* [x1=1, x2=0] 🡪 Part2 🡪 [x3=1]
* [x1=1, x2=0,**x3=0**] 🡪 Part3 🡪 [x4=0]

Integration interval [1.01,1.02]:

* [t=1..1,01] 🡪 Part1 🡪 [x1=1, x2=0]
* [x1=1, x2=0] 🡪 Part2 🡪 [x3=1]
* [x1=1, x2=0,**x3=1**] 🡪 Part3 🡪 [x4=0.06]

Takes two evaluations to detect correct solution.

## State-Event problem



Integration interval [1.43,1.44]

Part3 🡪 [x4>2.5]

Integration interval [1.44,1.45]

Part3 🡪 [x4>2.5] (integration continued, since change in x3 has not been seen by Part3 yet)

# Algorithm: Gauss-Seidel (FMI v1, fixed step, no iteration)

------------------------------------------------------------------------------

Wall clock time = 100.327 ms

------------------------------------------------------------------------------

Output writing = 51.595 ms

Master-Algorithm = 16.195 ms 1001

Convergence failures = 0

Error test time and failure count = 0.000 ms 0

------------------------------------------------------------------------------

Part1 doStep = 5.139 ms 1001

getState = 0.000 ms 0

setState = 0.000 ms 0

Part2 doStep = 5.254 ms 1001

getState = 0.000 ms 0

setState = 0.000 ms 0

Part3 doStep = 4.180 ms 1001

getState = 0.000 ms 0

setState = 0.000 ms 0

------------------------------------------------------------------------------

## Time event problem:



Gauss-Seidel -> too early

## State event problem



# Algorithm: Gauss-Jacobi with error control

------------------------------------------------------------------------------

Wall clock time = 57.551 ms

------------------------------------------------------------------------------

Output writing = 9.770 ms

Master-Algorithm = 25.809 ms 193

Convergence failures = 0

Error test time and failure count = 12.750 ms 68

------------------------------------------------------------------------------

Part1 doStep = 8.273 ms 783

getState = 0.749 ms 193

setState = 0.995 ms 329

Part2 doStep = 10.464 ms 783

getState = 0.709 ms 193

setState = 0.971 ms 329

Part3 doStep = 6.010 ms 783

getState = 0.590 ms 193

setState = 0.787 ms 329

------------------------------------------------------------------------------

Less evals!



Problem: Step at t=3s not detected.



Richardson-Extrapolation (step-doubling error test) not fully sufficient when not iterating. In the example above, first interval and second interval are integrated with same input variables, hence same result.

# Algorithm: Gauss-Seidel with error control

------------------------------------------------------------------------------

Wall clock time = 36.067 ms

------------------------------------------------------------------------------

Output writing = 3.591 ms

Master-Algorithm = 19.393 ms 93

Convergence failures = 0

Error test time and failure count = 7.358 ms 27

------------------------------------------------------------------------------

Part1 doStep = 6.952 ms 360

getState = 0.406 ms 93

setState = 0.459 ms 147

Part2 doStep = 7.270 ms 360

getState = 0.388 ms 93

setState = 0.462 ms 147

Part3 doStep = 4.643 ms 360

getState = 0.292 ms 93

setState = 0.358 ms 147

------------------------------------------------------------------------------

Compared to Gauss-Jacobi, less evals = faster



Same problem with detecting changes as Gauss-Jacobi -> earlier detection of time-events leads to built-up of larger steps (hence less work in total), but around t=4s overshoots largely.

Conclusion: not applicable when not iterating.

# Algorithm: Gauss-Seidel with iteration and error control



Iteration count makes a difference, 2 iterations (one repeated step), 3 iterations (two times repeated), for higher iteration counts there are no difference in results in this case.

Problem: why is the solution not exactly obtained? Why still differences?

Reason: error control is only based on last value, not on slopes of variables in half-intervals. Hence, when first half interval is too long and overshoots a time/state-event, the result of the full step and the half-step will always be the same.



Example:

t = 0,967388 s, t + h/2 = 1,000639 s, t + h = 1,03389

At t+h/2 the time event has already occurred, and since the interval [t,t+h/2] is repeated, the integration of x4 with slope 2.5 commences already at t = 0,967388 s. For the full step the time event has also passed and integration with slope 2.5 also commences at t = 0,967388 s, leading to the same, yet wrong result.

Conclusion: error test should be complemented with slope comparison.