Testing of Rosenbrock Solvers for EXA2Green

Teresa Beck, Joseph Charles

May 27, 2015

In point 4 of the COSMO-ART roadmap M13-M36, we had promised an *"Integration of algorithmic changes, which were demonstrated in the PRACE 2IP WP8 to improve KPP solver performance (up to a factor 2 for the ODE solver)."* This document first outlines the fundamentals of the new solver, which will be denoted by "rosenbrock_posdef_h211b_qssa". Then, technical details concerning the integration of the solver into KPP-2.2.1 are explained. The following section presents tests for the new solver. Different configurations of the solver are being compared with respect to the speedup and the accuracy of the solution. All tests are carried out by means of a zero-dimensional box-model. The document closes with conclusions and an estimation of the new solver's potential E2S improvement for the extended testbed.

1 Basics of rosenbrock_posdef_h211b_qssa

In early 2013, G. Fanourgakis, J. Lelieveld and D. Taraborelli had announced a new integrator, that combines the quasi-steady-state approach with the Rosenbrock integrator and a new time-stepping algorithm. G. Fanourgakis noted that the "*scheme is very efficient and significantly reduces the number of iterations that the Rosenbrock implit ODE solver requires. At the same time the accuracy of the integrator remains almost unaffected.*"

The new integrator combines several features:

1. **Time-step control:** A new adaptive time-stepping algorithm for the integration of ODEs, as recommended by Söderlind [3].
2. **Stiffness reduction:**
   (a) Quasi-Steady-State Approximation (QSSA)
   (b) A linear interpolation of the rate coefficients.
3. **Positive definition**: Artificial preservation of positivity to improve stability.

1.1 Time-step control

Classic Rosenbrock solvers usually determine the time step size h_{n+1} by means of

$$h_{n+1} = h_n(\epsilon/r_n)^{1/k},$$

with the tolerance parameter ϵ, the error estimate r_n, $k = p+1$ and p the order of convergence. The new integrator uses a more sophisticated step size controller algorithm H211b, as introduced by Söderlind [3]. Different than the one mentioned before, it also makes use of the error estimates of the previous time steps. The size of the timesteps is determined by

$$h_{n+1} \quad = \quad h_n(\epsilon/r_n)^{1/(bk)}(\epsilon/r_{n-1})^{1/(bk)}(h_n/h_{n-1})^{-1/b}.$$

According to Söderlin [3], the optimal parameters are $b = 1$ and $k = 2$. Note, that this new time-step control is already implemented in the baseline.

1.2 Stiffness reduction

1.2.1 Quasi-Steady-State Approximation (QSSA)

A reduction fot the stiffness of the set of ODEs is accomplished via the simplest QSSA formula,

$$\begin{aligned} P(t,y) - D(t,y)\,y &= 0 \\ \Rightarrow y &= P(t,y)/D(t,y), \end{aligned}$$

i.e. for some species, we assume that the amount being destructed $D(t,y)\ y$ equals the amount being constructed $P(t,y)$ at some time. After an induction periof of τ_{ind}, those species are assumed to be in a quasi-stationary state. Sportisse and Djouad [4] have shown, that the error of using a QSSA is acceptable, if $|(P-L)/(P+L)| \leq 10^{-2}$. Turanyi et al. [6] have further shown, that the time within the QSSA species reach a quasi steationary steady state τ_{ind} is estimated to be about 10 times the longest QSSA species lifetime, τ^{max}_{QSSA}. The QSSA option can be turned on/off in ICNTRL(6). The respective threshold value for QSSA species lifetime τ^{max}_{QSSA} can be adjusted in RCNTRL(10).

```
!    ICNTRL(6)  -> 0 no QSSA
!                  1 QSSA
[...]
!    RCNTRL(10) -> thres_tau, threshold value for QSSA species lifetime
```

Further, undocumented options for QSSA seem to exist, as can be seen later in the code:

```
!~~~> Choice of QSSA
    IF (ICNTRL(6) == 0) THEN                [...no QSSA...]
ELSEIF (ICNTRL(6) == 1) THEN ! DAE QSSA            [...]
ELSEIF (ICNTRL(6) == 2) THEN ! Plain QSSA          [...]
ELSEIF (ICNTRL(6) == 3) THEN ! Iterated QSSA       [...]
ELSEIF (ICNTRL(6) == 4) THEN ! Extrapolated QSSA   [...]
ELSEIF (ICNTRL(6) == 5) THEN ! Extrapolated QSSA   [...]
```

1.2.2 Linear interpolation of rate coefficients

A second step to reduce stiffness is aimed at by using a linear interpolation scheme to smoothly vary the rate coefficients. It has been seen, that the abrupt change in the photolysis rate coefficients during night/day transitions significantly increase the stiffness of the ODEs. A linear interpolation scheme has been added to interpolate rate coefficients during the time splitting interval, which eventually may lead to a decrease in stiffness.

In the code, this option is controlled by an integer called i_do_lin_interp. This function can only be called in the non-autonomous mode, i.e. if $f = f(t,y)$ depends on t.

```
IF (.NOT.Autonomous) THEN
if (i_do_lin_interp==1) then
!CALL Update_SUN()
        CALL Update_RCONST()
        else if (i_do_lin_interp==2) then
            do i=1, Nreact
             x = rconst_dt(i) - rconst_prev(i)
             rconst(i) = rconst_prev(i) + x * delta/period
            enddo
endif
END IF
```

1.3 "Positive definite" option

Last, an intransparant "positive definite" option has been added, so that enables an artificial preservation of positivity of the solution. Typically, this approach, is encountered as "clipping", i.e. setting negative concentration values to zero. For stability reasons, an artificial preservation of positivity is advantageous. However, this approach is not mass-conserving. More elaborate techniques ensure positivity through additional postprocessing steps, such as a projection onto a non-negative simplex. Methods favoring positivity (as introduced by Sandu [2]) present computationally less costly alternatives.

Details (with missleading comments...) can be found in the code:

```
! positive definite following a suggestion from Adrian
! see "MAX(Ynew,ZERO)" for details
[...]
  IF (posdef==1) THEN
      Y = MAX(Ynew,ZERO) ! new value is positive definite:
  ENDIF
[...]
  IF (posdef==2) THEN
      Y = MAX(Ynew,ZERO) ! new value is positive definite:
  ENDIF
```

This option can be turned on/off in ICNTRL(5).

```
!    ICNTRL(5)  -> 0 no posdef, 1 posdef out of time loop,
!                  2 posdef every substeps
```

2 Integration into KPP-2.2.1

The original rosenbrock_posdef_h211b_qssa solver had been designed for KPP version 2.1. However, for the COSMO-ART baseline we had decided to stick to KPP version 2.2.1. For the sake of comparability, the original code of the new solver was slightly adapted and integrated into KPP version 2.2.1. The modified code is held in the repository's subfolder kpp-2.2.1_with_Prace_Integrator.

Following modifications to kpp-2.2.1 had to be made:

- **int/**
  - add rosenbrock_posdef_h211b_qssa.f90
  - add rosenbrock_posdef_h211b_qssa.def
- **int/rosenbrock_posdef_h211b_qssa.f90:**
  - rename variable lin_interp into i_do_lin_interp
- **src/gen.c**:
  - add function GenerateFun_Split()

3 Tests

All tests were carried out by means of the 0-dim box-model using the repository revision number 220. The utilized chemical mechanism is RADM-KA (Regional Acid Deposition Model Version Karlsruhe), a slightly amended version of the chemical mechanism RADM2 by Stockwell et al. [5]. RADM-KA comprises 193 reactions and 82 chemical species, whereas one species is held constant. This mechanism was chosen in accordance to the baseline scenario, the collaborators had agreed upon in March 2014.

The tests are set up on a three-dimensional discrete computing domain with $66 \times 56 \times 31$ grid cells, i.e. a total of 114 576 independent 0-dim boxes are being solved. The initial conditions for each box were extracted from realistic COSMO-ART model runs from a computing domain of the same dimension, centered above Paris at high noon (12am), April 13th, 2010. Initial conditions for all species therefore vary for each of the 114 576 0-dim boxs, but all represent a polluted atmosphere above Paris. The model supports variable photolysis conditions, which allows to locally adapt the rate constants. Temperature is kept constant over all boxes. The total integration time is set to one hour, with integrator restarts at every 120 seconds, again in accordance with the baseline. In total, the function INTEGRATE is therefore called 30 times.

In the tests presented here, no external disturbances due to advection, diffusion or further effects are taken into account, as it would be the case in a real air quality model. In fact, only the very first initial conditions represent a realistic set-up with perturbed initial conditions, as those initial values were extracted from realistic, three-dimensional air quality model runs. Since in our tests, the chemical species are not exposed to any perturbation throughout the simulation, they equilibrate with elapsing time. Hence, the systems will be less stiff in the following 29 INTEGRATE function calls. However, the number of time-steps necessary for the solution of the chemical kinetics is strongly influenced by the degree of stiffness. Therefore, only the measurements for the very first call of the INTEGRATE function, can give a qualitative estimate for performance improvements for future three-dimensional runs within COSMO-ART.

The simulation is being monitored by means of 8 key chemical species (NO, NO2, NO3, HO, H2O2, HCHO, OP1, O3), and observed at the gridpoint located right above Paris.

The source code for the simulation of the chemical kinetics was generated by KPP. To this end, first a KPP executable was produced in bin/kpp. After the Fortran code was generated by KPP, backup files containing modifications needed for the benchmarking (tuning of parameters, statistics output) of the different solvers are copied to the current directory. All tests are run with scalar relative and absolute tolerances of rtol $=10^{-2}$ and atol $=10^{-2}$ on one node of Piz Daint at CSCS (Intel 8-core Xeon E5-2670 processor).

Reference Solution As a reference solution, we took a run with the Rosenbrock4 solver using KPP-2.2.3 (as it corresponds to the solver used in the original COSMO-ART baseline) and scalar and absolute tolerances of rtol $=10^{-2}$ and atol $=10^{-2}$.

Accuracy We estimate the accuracy of the final solution by means of the relative L^2-error to the reference solution $\tilde{c}$ at final integration time,

$$\mathrm{err}_{\mathrm{rel}} = \sqrt{\frac{\sum_{i=0}^{N} \left(\tilde{c}_i(T) - c_i(T)\right)^2}{\sum_{i=0}^{N} \tilde{c}_i^2}},$$

$$\mathrm{err}_{\mathrm{rel}} = \sqrt{\sum_{i=0}^{N} \left(\frac{\tilde{c}_i(T) - c_i(T)}{\tilde{c}_i}\right)^2},$$

$$\mathrm{err}_{\mathrm{rel}} = \max \sum_{i=0}^{N} \left(\frac{\tilde{c}_i(T) - c_i(T)}{\tilde{c}_i}\right)^2,$$

with $T = 3600$ sec. The current way to calculate the relative error is not ideal as we consider only a portion of the full population of chemical species (NO, NO2, NO3, HO, H2O2, HCHO, OP1, O3). However we take into account the most critical chemical species, which are the ones that typically have the biggest error contributions, so the approximation is fine.

Speedup As an estimate for the serial speedup, we investigate the accumulated **number of timesteps** and the **elapsed time** for a) the first function call of the function “INTEGRATE” and b) an average over all function calls “INTEGRATE” and c) in total over all 30 timesteps. It has to be emphasized, that measure a) will give a realistic hint on the estimated speed-up within a three dimensional simulation in COSMO-ART.

```
!~~~> Time loop
T = TSTART
kron: DO WHILE (T < TEND)

   nbit = nbit + 1

   ISTATS(:) = 0

   TIME = T

   WRITE(6,990) (T-TSTART)/(TEND-TSTART)*100, T,        &
        ( TRIM(SPC_NAMES(MONITOR(i))),           &
        VARTOT(MONITOR(i),idim_spot,jdim_spot,kdim_spot)/CFACTOR, i=1,NMONITOR )
   CALL SaveData()

   ! get energy counter at startup
   CALL energy(ETS_init)
   CALL device_energy(device_ETS_init)

   TTS_init = MPI_WTIME()

   CHI = CHIBE( T/60.0d+00, 48.0d+00, 2.0d+00, TAG )
]
   CALL GetMass( C, DVAL )

   !CALL Update_SUN()
   CALL Update_RCONST()

   CALL INTEGRATE( TIN = T, TOUT = T+DT, ISTATUS_U = ISTATE, &
        RSTATUS_U = RSTATE, ICNTRL_U = ICNTRL, RCNTRL_U = RCNTRL)

   TTS_final = MPI_WTIME()

   ! get energy counter at end
   CALL energy(ETS_final)
   CALL device_energy(device_ETS_final)

   TTS = TTS + (TTS_final - TTS_init)
   ETS = ETS + (ETS_final - ETS_init)
   device_ETS = device_ETS + (device_ETS_final - device_ETS_init)

   IF (nbit == 1) THEN
      TTS_first_call = TTS
      ETS_first_call = ETS
      device_ETS_first_call = device_ETS
      tsteps_first_call = ISTATS(3)
   END IF

   global_ISTATS(:) = global_ISTATS(:) + ISTATS(:)

   T = RSTATE(1)

END DO kron
!~~~> End Time loop
```

3.1 Ros4 implementations for different KPP versions

In the first tests, [ref0], [ref1] and [ref2], the KPP-2.2.1, KPP-2.2.3 and KPP-2.2.1_with_Prace_Integrator implementations of Ros4 were checked. The parameters for the tests can be seen in table 1, the results in table 7.

In case of the Prace implementation, no additional options such as QSSA, posdef and linear interpolation were enabled. The results illustrate, that the pure implementation of Ros4 already uses a different time step control mechanism. So the results [ref2] differ to those of [ref0], especially in the number of timesteps per function call. At this point, it also gets clear, that the time measurements we take, are not linear with the

	[ref0]	[ref1]	[ref2]
KPP version	2.2.1	2.2.3	2.2.1_with_Prace_Integrator
ATOL(:)	1.0d-2		
RTOL(:)	1.0d-2		
ICNTRL(1)	0		
ICNTRL(2)	1		
ICNTRL(3)	3		
ICNTRL(4:20)	0		
RCNTRL(:)	0.0		

Table 1: Parameters for tests [ref0]-[ref2].

number of time steps, taken: the 176 timesteps taken in [ref0] need even more time, than the 329 timesteps token in [ref2].

3.2 Tests of Rodas3 rosenbrock_posdef_h211b_qssa

In the next set of tests, the new integrator rosenbrock_posdef_h211b_qssa was tested with varying parameters.

3.2.1 [timestep_<b>_<k>]

In these tests, the new time stepping control mechanism is tested only using Ros4, with no linear interpolation, no posdef option and no QSSA. According to Söderlind [3], an optimal choice is given by $b = 1$ and $k = 2$. As mentioned before, this new control mechanism is already implemented in the baseline in KPP-2.2.1. There $b = 1$ and $k = 4$ are also chosen for Ros4. Parameters and results can be seen in table 2 and 7.

3.2.2 [linint<0:2>]

The tests [linint0], [linint1] and [linint2] test all options of using linear interpolation using Ros4, with no posdef option and no QSSA. Parameters and results can be seen in table 3 and 7.

3.2.3 [posdef<1:2>]

The tests [posdef1] - [posdef2] test all options of positive definition using Ros4, no QSSA and no linear interpolation. Note, that is absolutely intransparant, what this option actually does! Parameters and results can be seen in table 4 and 7.

3.2.4 [QSSA<1:6>_<τ_{QSSA}^{max}>]

All options for QSSA using Ros4, no posdef option, and no linear interpolation are tested. The parameter τ_{QSSA}^{max} is varied from 10^{-9}, 10^{-6}, 10^{-3} and 10^{0}. Parameters and results can be seen in table 5 and 7.

3.2.5 [opt_<int>_<mode>]

The parameters for these tests are chosen in accordance to the results, presented on a poster by Taraborelli et al. [1]: "the optimum for speedup and accuracy has been found with Rosenbrock-Rodas3 in the autonomous mode, $\tau_{QSSA}^{max} = 1s$ and the H211b controller with $b = 1$ and $k = 2$. As it has been seen in the previous QSSA tests, best choices for ICNTRL(6) seem to be 2 or 3 - we now choose 2. We both tested Ros4 and Rodas3 in combination with above mentioned settings. Parameters and results can be seen in table 6 and 7.

	[timestep_1_2]	[timestep_1_4]
KPP version	2.2.1_with_Prace_Integrator	
ATOL(:)	1.0d-2	
RTOL(:)	1.0d-2	
ICNTRL(2)	1	
ICNTRL(3)	3	
ICNTRL(1, 4:20)	0	
RCNTRL(1:7,10:20)	0.0	
RCNTRL(8)	1	1
RCNTRL(9)	2	4

Table 2: Parameters for tests [timestep_<b>_<k>].

	[linint1]	[linint2]
KPP version	2.2.1_with_Prace_Integrator	
ATOL(:)	1.0d-2	
RTOL(:)	1.0d-2	
ICNTRL(2)	1	
ICNTRL(3)	3	
ICNTRL(1, 4:20)	0	
RCNTRL(:)	0.0	
I_DO_LIN_INTERP	1	2

Table 3: Parameters for tests [linint0]-[linint2].

	[posdef1]	[posdef2]
KPP version	2.2.1_with_Prace_Integrator	
ATOL(:)	1.0d-2	
RTOL(:)	1.0d-2	
ICNTRL(2)	1	
ICNTRL(3)	3	
ICNTRL(5)	1	2
ICNTRL(1, 4, 6:20)	0	
RCNTRL(:)	0.0	

Table 4: Parameters for tests [posdef0] and [posdef1] 1.

	[QSSA<1:6>_$<\tau^{max}_{QSSA}>$]
KPP version	2.2.1_with_Prace_Integrator
ATOL(:)	1.0d-2
RTOL(:)	1.0d-2
ICNTRL(2)	1
ICNTRL(3)	3
ICNTRL(1, 4, 5, 7:20)	0
ICNTRL(6)	1, 2, 3, 4, 5, 6
RCNTRL(1:9, 11:20)	0.0
RCNTRL(10)	10^{-9}, 10^{-6}, 10^{-3}, 10^{0}

Table 5: Parameters for tests [QSSA<1:6>_$<\tau_{ind}>$].

	[opt_ros4_na]	[opt_ros4_a]	[opt_rodas3_na]	[opt_rodas3_a]
KPP version	2.2.1_with_Prace_Integrator			
ATOL(:)	1.0d-2			
RTOL(:)	1.0d-2			
ICNTRL(1)	0	1	0	1
ICNTRL(2)	1			
ICNTRL(3)	3		4	
ICNTRL(6)	2			
ICNTRL(4, 5, 6:20)	0			
RCNTRL(1:7, 11:20)	0.0			
RCNTRL(8)	1.0			
RCNTRL(9)	2.0			
RCNTRL(10)	1.0			

Table 6: Parameters for tests [opt_<int>_<mode>].

	err_{rel}	#tsteps	CPU time [s]	#tsteps	CPU time [s]	#tsteps	CPU time [s]
		a) first call only		b) in average		c) in total	
[ref0]	0.1906E-06	59	66.554	8	11.045	266	331.341
[ref1]	0.4712E-07	21	26.440	11	14.510	346	435.300
[ref2]	0.4712E-07	21	8.376	11	4.603	346	138.098
[timestep_1_2]	0.5346E-06	47	17.299	6	2.419	183	72.561
[timestep_1_4]	0.8385E-07	22	8.642	8	3.454	258	103.635
[linint1]	0.4712E-07	21	25.059	11	13.753	346	412.575
[linint2]	NaN	NaN	NaN	NaN	NaN	NaN	NaN
[posdef1]	0.4712E-07	21	8.410	11	4.621	346	138.618
[posdef2]	0.4712E-07	21	8.453	11	4.638	346	139.132
[QSSA2_1E-9]	0.5352E-07	21	8.551	11	4.698	346	140.946
[QSSA3_1E-9]	0.5352E-07	21	8.498	11	4.692	346	140.762
[QSSA6_1E-9]	NaN	NaN	NaN	NaN	NaN	NaN	NaN
[QSSA2_1E-6]	0.5352E-07	21	8.493	11	4.684	346	140.516
[QSSA3_1E-6]	0.5352E-07	21	8.532	11	4.701	346	141.042
[QSSA6_1E-6]	NaN	NaN	NaN	NaN	NaN	NaN	NaN
[QSSA2_1E-3]	0.5352E-07	21	8.555	11	4.705	346	141.156
[QSSA3_1E-3]	0.5352E-07	21	8.579	11	4.721	346	141.635
[QSSA6_1E-3]	NaN	NaN	NaN	NaN	NaN	NaN	NaN
[QSSA2_1E0]	0.2264E-03	21	8.579	4	2.049	146	61.472
[QSSA3_1E0]	0.2264E-03	21	8.530	4	2.060	146	61.790
[QSSA6_1E0]	NaN	NaN	NaN	NaN	NaN	NaN	NaN
[opt_ros4_a]	0.2287E-03	47	16.959	4	1.820	138	54.595
[opt_rodas3_a]	0.2256E-03	18	6.859	3	1.458	107	43.735
[opt_ros4_na]	0.2287E-03	47	17.446	4	1.892	138	56.749
[opt_rodas3_na]	0.2256E-03	18	7.197	3	1.536	107	46.075

Table 7: Results for all tests on Piz Daint.

4 Conclusions

All results are only discussed with regard to option a) first call only. This decision has been guided by the comment in footnote 1.

Discussion measurements on Piz Daint at CSCS Table 7 shows the results for measurements carried out on one node of Piz Daint at CSCS. All tests led to very small errors $< 10^{-4}$. The integrator [ref2] led to results, which were approx. 3x faster than [ref0] and [ref1]. Again, both tests of the new time stepping algorithm are significantly faster than the reference [ref0], also faster than [ref1]. Best performance was achieved with $b = 1$ and $k = 2$, which was approximately 5x faster than the reference [ref0]. Further, all QSSA tests with ICNTRL(6) = 2 or 3 were faster than the reference [ref0]. All other choices for ICNTRL(6) led to very (>> reference*10) long computing times or errors. These tests have therefore only been executed partly. The fastest results could be achieved with the parameters chosen in accordance to the poster by Taraborelli et al. [1], including a QSSA assumption (ICNTRL(6) = 2) and $\tau_{QSSA}^{max} = 1$ sec and the new time stepping control h211b with $b = 1$ and $k = 2$ in autonomous mode. Rodas3 shows better results (3.5x faster) than Ros4 in terms of performance.

References

[1] G. Rädel G. S. Fanourgakis A. Pozzer B. Steil B. Stevens D. Taraborrelli and J. Lelieveld. Linking composition and circulation on intermediate spatio - temporal scales temporal - licos (861 and 869).

[2] A. Sandu. Time-stepping methods that favor positivity for atmospheric chemistry modeling. In D.P. Chock and G.R. Carmichael, editors, *IMA Volume on Atmospheric Modeling*, pages 1–21. Springer-Verlag, 2001.

[3] Gustaf Söderlind. Automatic control and adaptive time-stepping. In *Numerical Algorithms*, volume 31, pages 281–310, 2002.

[4] Bruno Sportisse and Rafik Djouad. Reduction of Chemical Kinetics in Air Pollution Modeling. *Journal of Computational Physics*, 164(2):354–376, 2000.

[5] William R. Stockwell, Paulette Middleton, Julius S. Chang, and Xiaoyan Tang. The second generation regional acid deposition model chemical mechanism for regional air quality modeling, 1990.

[6] T Turanyi, A S Tomlin, and M J Pilling. On the error of the quasi-steady-state approximation. *The Journal of Physical Chemistry*, 97(1):163–172, 1993.