KU LEUVEN

Project Bateman: Status Update

Daan Camps, Raf Vandebril, Gert Van den Eynde

December 6, 2016

Bateman Problem

Quick recapitulation

Concentration of nuclides y_i obey a system of first-order linear differential equations[1]:

$$\mathbf{y}' = A\,\mathbf{y} \quad \longrightarrow \quad \mathbf{y}(t) = e^{At}\mathbf{y}_0$$

Properties of the system

Transmutation matrix $A \in \mathbb{R}^{3771\times3771}$

Sparse matrix ($\sim 0.5\%$), Numerical range: $10^{-40} \cdots 10^{22}$, Singular values: $10^{-22} \cdots 10^{22}$

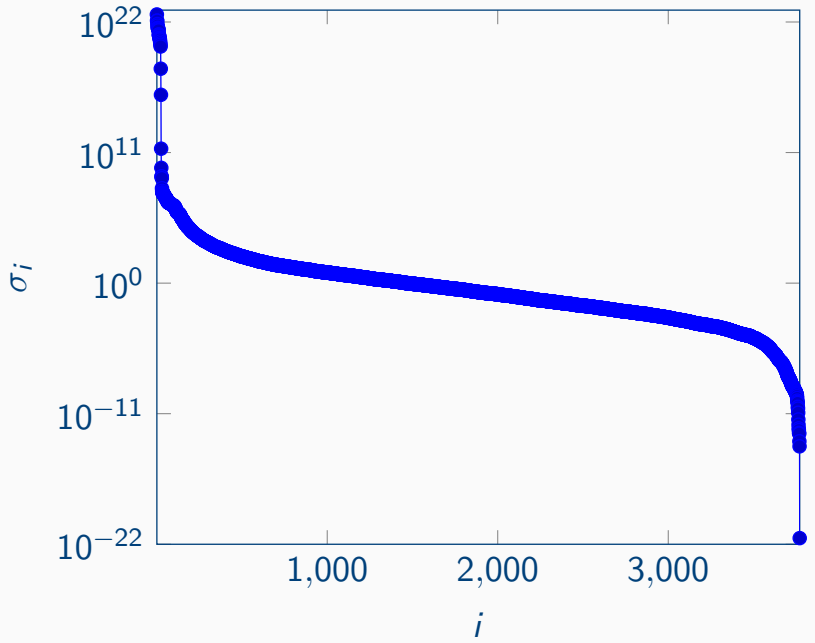
[Stankovskiy, A. & Van den Eynde, G., 2012]

[1] under certain assumptions: constant particle fluxes and spectra, both spatially and temporally

Solving the Bateman problem numerically

A bit more on problem specific numerics[2]

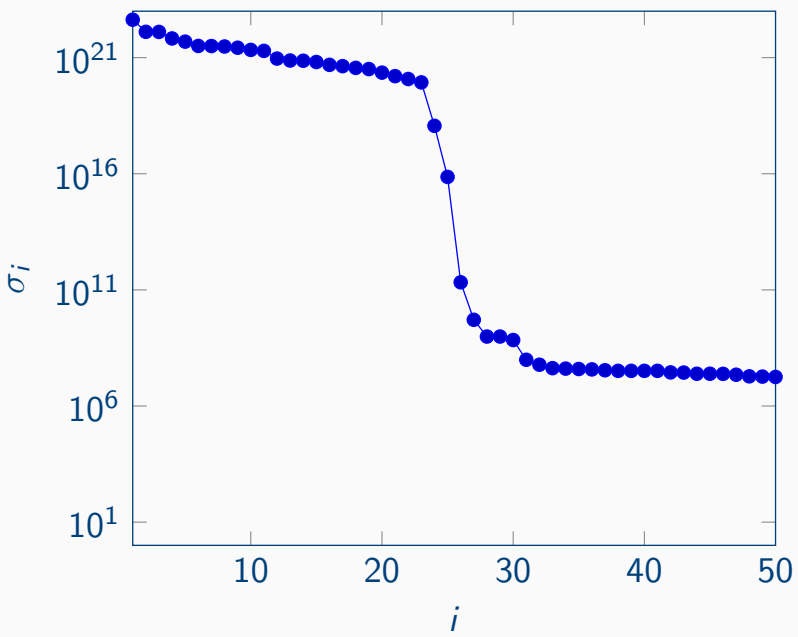
System is not of full rank:

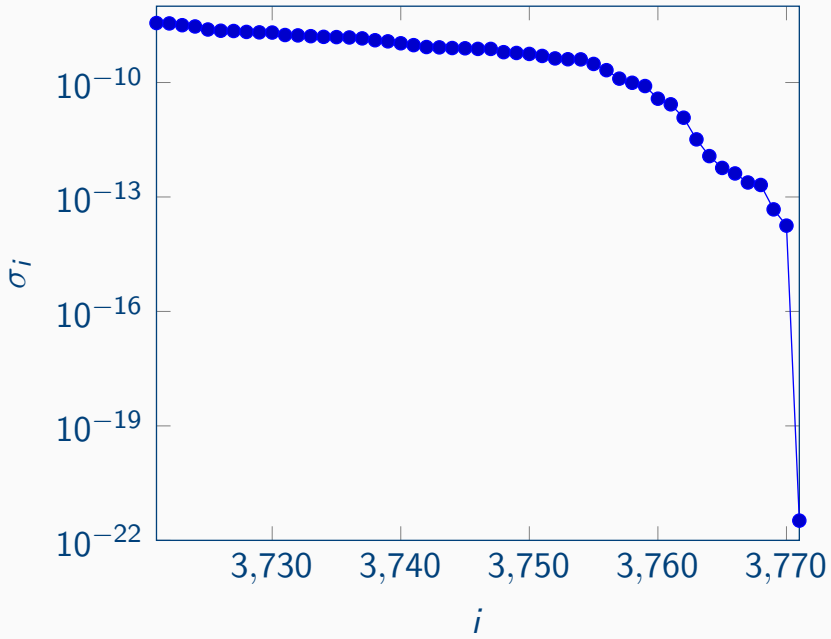


[2] based on "Problem 2"

A bit more on problem specific numerics

System is not of full rank:





A bit more on problem specific numerics

The problem is very *stiff*:

$$\max\left(|\Re(\lambda_i)|\right) \gg \min\left(|\Re(\lambda_i)|\right)$$
$$3 \cdot 10^{22} \gg 0$$

RADAU IIA

- 3-stage Implicit Runge-Kutta scheme of order $p = 5$
- L-stable $\rightarrow$ well-suited for stiff problems

RADAU IIA

- 3-stage Implicit Runge-Kutta scheme of order $p = 5$
- L-stable $\rightarrow$ well-suited for stiff problems
- ... but expensive!

RADAU IIA

- 3-stage Implicit Runge-Kutta scheme of order $p = 5$
- L-stable $\rightarrow$ well-suited for stiff problems
- ... but expensive!
- Our goal: make it as cheap and efficient as possible

[Hairer, E. & Wanner, G., 1996]

Linear RADAU IIA

General setting for nonlinear problems

General s-stage implicit Runge-Kutta method for nonlinear $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$:

$$\begin{cases} \mathbf{g}_i & = \mathbf{y}_0 + h \sum_{j=1}^{s} \tilde{a}_{ij}\, \mathbf{f}(t_0 + \tilde{c}_j h, \mathbf{g}_j) \quad i = 1, \ldots, s \\ \mathbf{y}_1 & = \mathbf{y}_0 + h \sum_{j=1}^{s} \tilde{b}_j\, \mathbf{f}(t_0 + \tilde{c}_j h, \mathbf{g}_j) \end{cases}$$

Linear RADAU IIA

General setting for linear problems

General s-stage implicit Runge-Kutta method for linear $\mathbf{y}' = A\,\mathbf{y}$:

$$\begin{cases} \mathbf{g}_i &= \mathbf{y}_0 + h \sum_{j=1}^{s} \tilde{a}_{ij}\, A\mathbf{g}_j \quad i = 1, \ldots, s \\ \mathbf{y}_1 &= \mathbf{y}_0 + h \sum_{j=1}^{s} \tilde{b}_j\, A\mathbf{g}_j \end{cases} \tag{1}$$

Linear RADAU IIA

Change of variables

Reformulate (1) in terms of $\mathbf{z}_i = \mathbf{g}_i - \mathbf{y}_0$:

$$\mathbf{z}_i = h \sum_{j=1}^{3} \tilde{a}_{ij} \, A(\mathbf{y}_0 + \mathbf{z}_j) \quad i = 1, \ldots, 3 \tag{2}$$

Linear RADAU IIA

Change of variables

Reformulate (1) in terms of $\mathbf{z}_i = \mathbf{g}_i - \mathbf{y}_0$:

$$\mathbf{z}_i = h \sum_{j=1}^{3} \tilde{a}_{ij} A(\mathbf{y}_0 + \mathbf{z}_j) \quad i = 1, \ldots, 3 \tag{2}$$

The step update formula from (1) becomes:

$$\mathbf{y}_1 = \mathbf{y}_0 + \sum_{i=1}^{3} \tilde{d}_i \mathbf{z}_i \quad \text{with } \tilde{d} = \tilde{b}\tilde{A}^{-1} = (0, 0, 1)$$

Linear RADAU IIA

Matrix notation

Equation (2) is in matrix notation:

$$\begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_3 \end{bmatrix} = h\tilde{A} \begin{bmatrix} A(\mathbf{y}_0 + \mathbf{z}_1) \\ A(\mathbf{y}_0 + \mathbf{z}_2) \\ A(\mathbf{y}_0 + \mathbf{z}_3) \end{bmatrix}$$

Linear RADAU IIA

Kronecker product notation

Which admits a compact notation with Kronecker product structure:

$$\mathbf{Z} = (h\tilde{A} \otimes A)(\mathbf{Y}_0 + \mathbf{Z})$$

with $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \mathbf{z}_3]^T$, $\mathbf{Y}_0 = [\mathbf{y}_0\ \mathbf{y}_0\ \mathbf{y}_0]^T$ vectors of length $3N$.

Linear RADAU IIA

Kronecker product notation

Which admits a compact notation with Kronecker product structure:

$$\mathbf{Z} = (h\tilde{A} \otimes A)(\mathbf{Y}_0 + \mathbf{Z})$$

with $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \mathbf{z}_3]^T$, $\mathbf{Y}_0 = [\mathbf{y}_0\ \mathbf{y}_0\ \mathbf{y}_0]^T$ vectors of length $3N$.

This can be rewritten as:

$$\left((I_3 \otimes A) - \left(\left(h\,\tilde{A}\right)^{-1} \otimes I_N\right)\right)\mathbf{Z} = -\left(I_3 \otimes A\right)\mathbf{Y}_0 \qquad (3)$$

Linear RADAU IIA

A useful similarity transformation

Next, make use of the similarity transformation:

$$T^{-1}\tilde{A}T = \Lambda,$$

that brings the Butcher tableau to block diagonal form:

$$\Lambda = \left[\begin{array}{ccc} \times & & \\ & \times & \times \\ & \times & \times \end{array}\right].$$

$\tilde{A}$ has 1 real eigenvalue λ_1 and 1 pair of complex conjugate eigenvalues $\lambda_{2,3} = \alpha \pm i\beta$.

Linear RADAU IIA

Decoupling the system

Introducing the similarity transformation in (3), decouples the linear system into an $N \times N$ and $2N \times 2N$ system:

$$\left(I_3 \otimes A - h^{-1}\Lambda \otimes I_N\right) \mathbf{W} = -\left(I_3 \otimes A\right)\left(T^{-1} \otimes I_N\right) \mathbf{Y}_0,$$

where $\mathbf{W} = \left(T^{-1} \otimes I_N\right) \mathbf{Z}$.

Linear RADAU IIA

$N \times N$ SYSTEM

The $N \times N$ system is given by:

$$(A - h^{-1}\lambda_1 I_N)\,\mathbf{w}_1 = -t_{1x}^{inv}\,A\mathbf{y}_0, \tag{4}$$

with $\mathbf{w}_1$ the first N entries of $\mathbf{W}$ and t_{1x}^{inv} the sum of the entries in the first row of T^{-1}.

Linear RADAU IIA

$2N \times 2N$ SYSTEM

The $2N \times 2N$ system is given by:

$$\begin{bmatrix} A - h^{-1}\alpha I_N & h^{-1}\beta I_N \\ -h^{-1}\beta I_N & A - h^{-1}\alpha I_N \end{bmatrix} \begin{bmatrix} \mathbf{w}_2 \\ \mathbf{w}_3 \end{bmatrix} = \begin{bmatrix} -t_{2x}^{inv}\, A\mathbf{y}_0 \\ -t_{3x}^{inv}\, A\mathbf{y}_0 \end{bmatrix},$$

which can be reduced to a complex system of size $N \times N$:

$$\left(A - h^{-1}(\alpha + i\,\beta) I_N\right)(\mathbf{w}_2 + i\,\mathbf{w}_3) = -(t_{2x}^{inv} + i\,t_{3x}^{inv})\, A\mathbf{y}_0 \qquad (5)$$

Linear RADAU IIA

Step update

The step update procedure becomes:

$$\mathbf{y}_1 = \mathbf{y}_0 + \mathbf{z}_3 \tag{6}$$

$$= \mathbf{y}_0 + t_{31}\,\mathbf{w}_1 + t_{32}\,\mathbf{w}_2 + t_{33}\,\mathbf{w}_3 \tag{7}$$

$$= \mathbf{y}_0 + t_{31}\mathbf{w}_1 + \mathbf{w}_2 \tag{8}$$

Implementation details

From this analysis we can identify some important aspects for the implementation:

- The original matrix A was singular, the matrices in Eqs. (4) and (5) are not (but still ill-conditioned)

Implementation details

From this analysis we can identify some important aspects for the implementation:

- The original matrix A was singular, the matrices in Eqs. (4) and (5) are not (but still ill-conditioned)
- As long as the step is not changed, the LU factorization ($\mathcal{O}(n^3)$) can be reused $\Rightarrow$ $\mathcal{O}(n^2)$ cost per step

Implementation details

From this analysis we can identify some important aspects for the implementation:

- The original matrix A was singular, the matrices in Eqs. (4) and (5) are not (but still ill-conditioned)
- As long as the step is not changed, the LU factorization ($\mathcal{O}(n^3)$) can be reused $\Rightarrow$ $\mathcal{O}(n^2)$ cost per step
- Option for high-level parallelism

Implementation details

From this analysis we can identify some important aspects for the implementation:

- The original matrix A was singular, the matrices in Eqs. (4) and (5) are not (but still ill-conditioned)
- As long as the step is not changed, the LU factorization ($\mathcal{O}(n^3)$) can be reused $\Rightarrow$ $\mathcal{O}(n^2)$ cost per step
- Option for high-level parallelism (but overhead is too big for given size of problems)

Implementation details

From this analysis we can identify some important aspects for the implementation:

- The original matrix A was singular, the matrices in Eqs. (4) and (5) are not (but still ill-conditioned)
- As long as the step is not changed, the LU factorization ($\mathcal{O}(n^3)$) can be reused $\Rightarrow$ $\mathcal{O}(n^2)$ cost per step
- Option for high-level parallelism (but overhead is too big for given size of problems)
- The same error estimator as for the original RADAU IIA can be used (cost: $\mathcal{O}(n^2)$)

Implementation details

Code overview

- `matrixrw`: I/O routines from file
- `units`: Specifies working precision kind (wp), parameters of RADAU IIA integration scheme and error estimation
- `utilities`: Some subroutines for printing status, warnings, errors

Implementation details

Code overview

- `matrixrw`: I/O routines from file
- `units`: Specifies working precision kind (wp), parameters of RADAU IIA integration scheme and error estimation
- `utilities`: Some subroutines for printing status, warnings, errors
- `radau_common`: Shared subroutines for constructing matrices and RHS from (4) and (5), also error estimation
- `radau_fx_seq`: Fixed step w/o error estimation. First version computes YFINAL, second also intermediate results.
- `radau_sa_seq`: Semi adaptive version - computes the error HEVALS times on logspaced points.
- `radau_ad_seq`: Adaptive version - computes error after each step

Implementation details

MUMPS Specifics

The best performance with the MUMPS solver was obtained with:

- Solving the systems (4) and (5) one after the other in a single process (sequential MUMPS)

Implementation details

MUMPS Specifics

The best performance with the MUMPS solver was obtained with:

- Solving the systems (4) and (5) one after the other in a single process (sequential MUMPS) ... if the problems would become significantly larger, a parallel implementation will become faster

Implementation details

MUMPS Specifics

The best performance with the MUMPS solver was obtained with:

- Solving the systems (4) and (5) one after the other in a single process (sequential MUMPS) ... if the problems would become significantly larger, a parallel implementation will become faster
- The MUMPS internal parameter for *amalgamation* `%KEEP(1)` has a large influence on performance of fwd/bwd substitution step. Optimal settings (on my PC for problem 1) are always between 40 and 50.

Numerical results

Execution times

t_f	Problem 1 (s)			Problem 2 (s)		
	fx	sa	ad	fx	sa	ad
1h	0.47e0	0.11e1	0.25e1	0.16e1	0.27e1	0.41e1
24h	0.95e1	0.18e1	0.45e1	0.31e2	0.70e1	0.13e2
720h	0.27e3	0.21e1	0.52e1	?	?	?

t_f	Problem 3 (s)			Problem 4 (s)		
	fx	sa	ad	fx	sa	ad
1h	0.11e1	0.85e0	0.98e0	0.17e1	0.14e1	0.16e1
24h	0.22e2	0.14e1	0.74e1	0.36e2	0.24e1	0.31e1
720h	0.66e3	0.15e2	0.14e3	0.11e4	0.31e1	0.60e1

Parameters: $h_{fx} = 20s$, $h_{init} = 0.25s$, $\#h$ evaluations $= 12$

Conclusion

Conclusion

- Highlighted the problem specific numerical difficulties
- Custom version of IRK scheme for linear first-order systems
- Implemented in Fortran90 making use of a state-of-the-art solver MUMPS

Available on: `https://github.com/campsd/linear-radau`