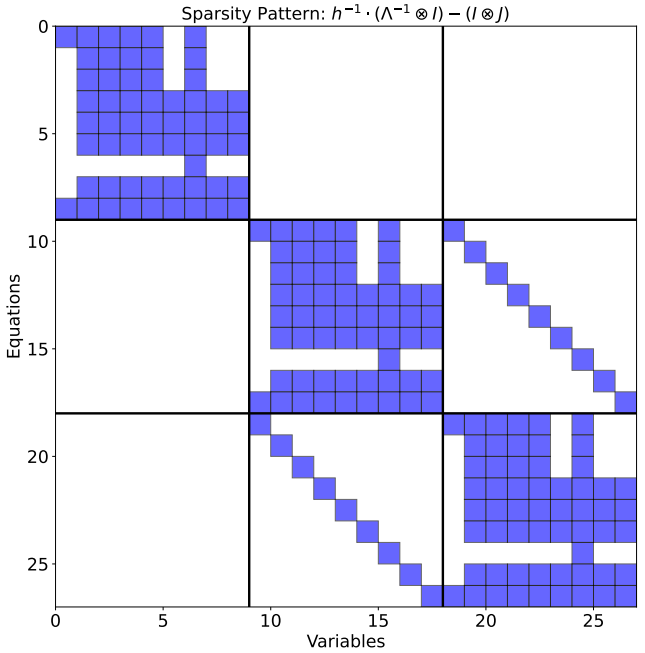
Hochschule Bielefeld

University of Applied Sciences and Arts



Advancements in Numerical Methods for the GBODE Solver

Linus Langenkamp, Bernhard Bachmann

02.02.2026

Hochschule Bielefeld University of Applied Sciences and Arts

HS'BI Hochschule Bielefeld University of Applied Sciences and Arts

TABLE OF CONTENTS

1. Implicit Runge-Kutta Methods
2. Block-Diagonal Transformations
3. Termination of Newtons Method
4. Step Size Selection after Events
5. Performance

GBODE

- Generic Bi-Rate Ordinary Differential Equation Solver - $\dot{\boldsymbol{y}}(t) = \boldsymbol{f}(t, \boldsymbol{y})$
- Implementation supports a wide range of Runge-Kutta methods
  - → Implicit: ESDIRK, SDIRK, Lobatto, Radau, Gauss, ...
  - → Explicit: Fehlberg, Merson, DOPRI45, ...
- Arising nonlinear systems of implicit methods are solved with KINSOL
- Special structure and context of the NLS is not used to full extend
  - → Convergence and speed can be improved by exploiting the structure of the NLS
  - → Strategies are implemented in the new flag `-gbnls=internal`

RUNGE-KUTTA METHODS

- Runge-Kutta methods compute several stage slopes

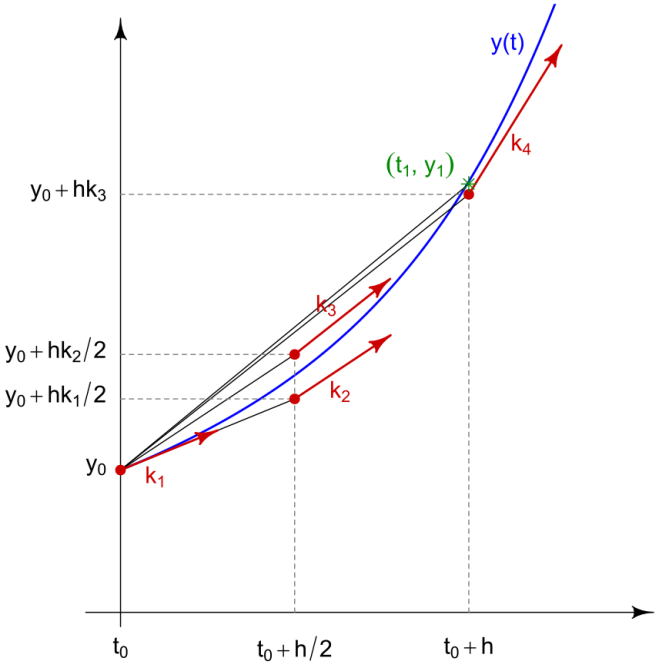
$$\boldsymbol{k}_i = \boldsymbol{f}\left(t_k + c_i h,\ \boldsymbol{y}_k + h \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j \right), \qquad i = 1, \dots, s$$

- These stage slopes are combined to update the solution

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + h \sum_{i=1}^{s} b_i \boldsymbol{k}_i$$

- The method coefficients are summarized in the Butcher tableau:

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}$$



CLASSES OF IMPLICIT RUNGE-KUTTA METHODS

Singly Diagonal Implicit Runge-Kutta (ESDIRK / SDIRK)

$$\begin{array}{c|cccc} \gamma & \gamma & 0 & \cdots & 0 \\ c_2 & a_{21} & \gamma & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ c_s & a_{s1} & \cdots & a_{s,s-1} & \gamma \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

- Stages solved sequentially ($n \times n$ systems)
- Single diagonal parameter γ
- Uses a single LU factorization - $\mathcal{O}(n^3)$
- Efficient Newton iterations
- A- / L-stable schemes available

Fully Implicit Runge-Kutta (FIRK)

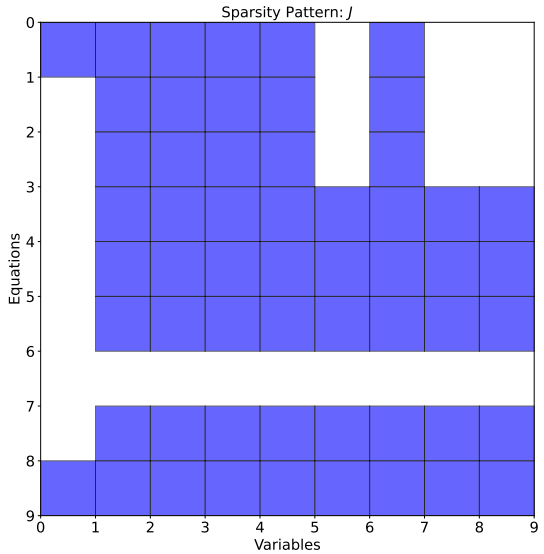
$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

- Fully coupled stages
- Solve one $sn \times sn$ system - $\mathcal{O}\left((sn)^3\right)$
- Higher achievable order
- Excellent stability properties
- Seems to be very expensive compared to SDIRK methods

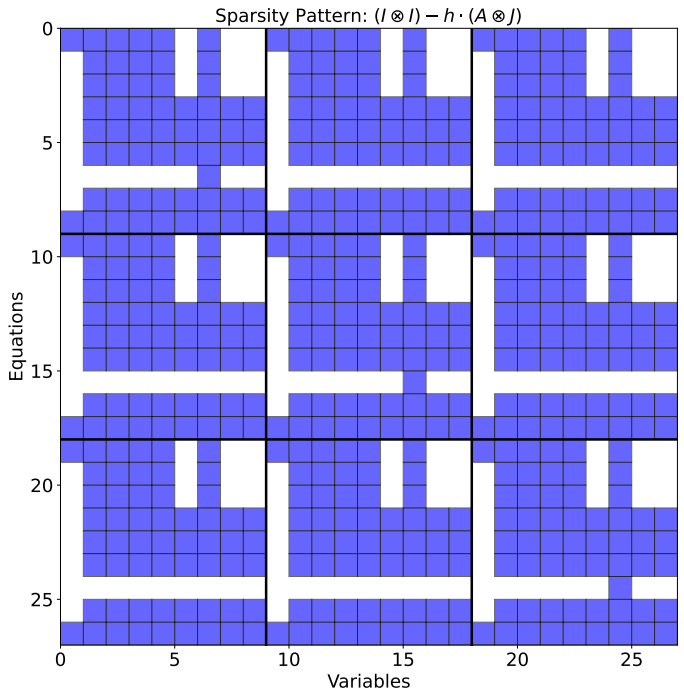
→ How to make this efficient?

EFFICIENT NLS FOR FIRK

- In the next few slides we will apply a series of transformations to the naive nonlinear system
- The visualizations and example matrices are for the specific 3-step Radau IIA method
- The theory is implemented for all FIRK methods in GBODE (incl. singular Butcher matrices)
- Consider the ODE sparsity pattern:



EFFICIENT NLS FOR FIRK: NAIVE / K-SPACE FORMULATION



Nonlinear System

$$\boldsymbol{g}_i(\boldsymbol{K}) := \boldsymbol{k}_i - \boldsymbol{f}\left(t_k + c_i h,\ \boldsymbol{y}_k + h \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j\right) = \boldsymbol{0}$$

Block-Jacobian

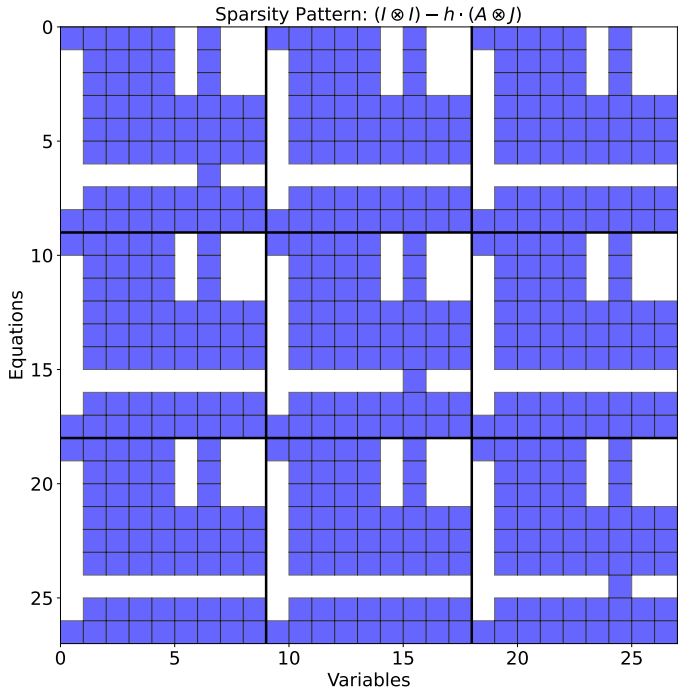
$$\frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{k}_j} = \delta_{ij} I_n - h\, a_{ij}\, J, \qquad J := \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t_0, \boldsymbol{y}_0)$$

Nonlinear System Jacobian

$$\frac{\partial \boldsymbol{G}_K}{\partial \boldsymbol{K}} = I_s \otimes I_n - h\, A \otimes J$$

→ LU factorization is $\mathcal{O}((sn)^3)$, here $\mathcal{O}(27n^3)$

EFFICIENT NLS FOR FIRK: Z-SPACE FORMULATION



Variable Transformation (assuming $\det A \neq 0$)

$$\boldsymbol{Z} := h(A \otimes I_n)\boldsymbol{K}$$

Nonlinear System

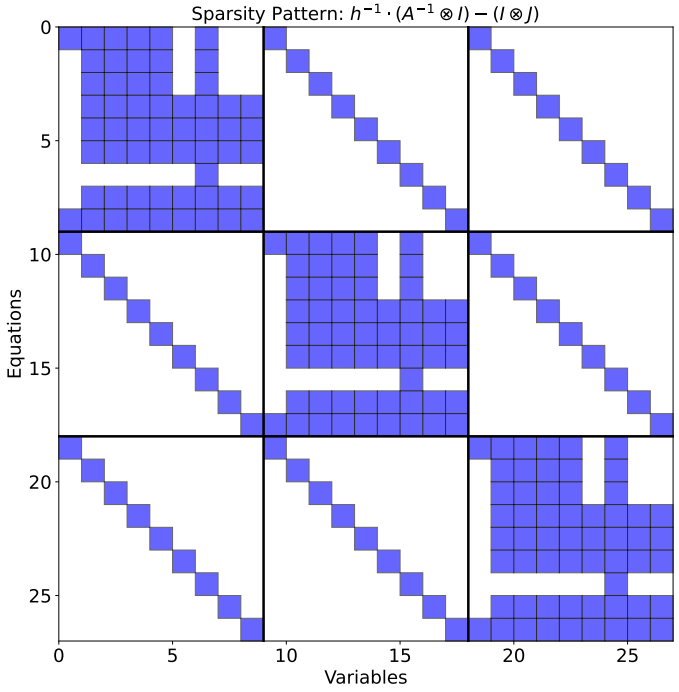
$$\boldsymbol{0} = \boldsymbol{Z} - h(A \otimes I_n)\boldsymbol{F}(\boldsymbol{Z})$$

Jacobian

$$\frac{\partial \boldsymbol{G}_Z}{\partial \boldsymbol{Z}} = I_s \otimes I_n - h\, A \otimes J$$

→ Reduces absorption effects by a bit

EFFICIENT NLS FOR FIRK: DIFFERENTIATION FORMULATION



Nonlinear System

$$\mathbf{0} = \boldsymbol{Z} - h(A \otimes I_n)\boldsymbol{F}(\boldsymbol{Z})$$
$$\iff \mathbf{0} = h^{-1}(A^{-1} \otimes I_n)\boldsymbol{Z} - \boldsymbol{F}(\boldsymbol{Z})$$

Jacobian

$$\frac{\partial \boldsymbol{G}_D}{\partial \boldsymbol{Z}} = h^{-1}A^{-1} \otimes I_n - I_s \otimes J$$

- Still a $3n \times 3n$ system
- Sparsity is way better already
- This formulation is used in Direct Collocation Dynamic Optimization

THE BLOCK-DIAGONAL TRANSFORMATION

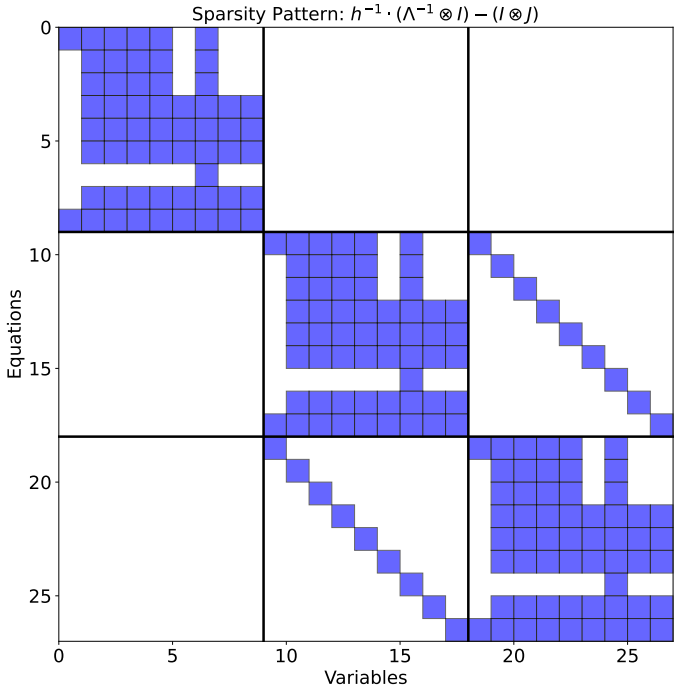
- Find a block-diagonal Transformation $A^{-1} = T\Lambda T^{-1}$
- Example: 3-step Radau IIA with $\gamma \approx 3.63783, \alpha \approx 2.68108, \beta \approx 3.05043$

$$A^{-1} = T\Lambda T^{-1} \text{ with } \Lambda = \begin{pmatrix} \gamma_1 & 0 & 0 \\ 0 & \alpha_1 & -\beta_1 \\ 0 & \beta_1 & \alpha_1 \end{pmatrix}, \; T = \begin{pmatrix} 0.094438 & -0.141255 & 0.030029 \\ 0.250213 & 0.204129 & -0.382942 \\ 1 & 1 & 0 \end{pmatrix}$$

- Set $\boldsymbol{W} := (T^{-1} \otimes I_n)\boldsymbol{Z}$ and it follows that

$$\begin{aligned} \mathbf{0} &= h^{-1} \left(A^{-1} \otimes I_n\right) \boldsymbol{Z} - \boldsymbol{F}(\boldsymbol{Z}) \\ &= h^{-1} \left(T\Lambda T^{-1} \otimes I_n\right) \boldsymbol{Z} - \boldsymbol{F}(\boldsymbol{Z}) \\ &= h^{-1} \left(T\Lambda \otimes I_n\right) \boldsymbol{W} - \boldsymbol{F}((T \otimes I_n)\boldsymbol{W}) \\ &= h^{-1} \left(\Lambda \otimes I_n\right) \boldsymbol{W} - (T^{-1} \otimes I_n)\boldsymbol{F}((T \otimes I_n)\boldsymbol{W}) \end{aligned}$$

EFFICIENT NLS FOR FIRK: W-SPACE FORMULATION



Nonlinear System

$$\mathbf{0} = h^{-1}(\Lambda \otimes I_n)\boldsymbol{W} - (T^{-1} \otimes I_n)\boldsymbol{F}((T \otimes I_n)\boldsymbol{W})$$

Jacobian

$$\frac{\partial \boldsymbol{G}_W}{\partial \boldsymbol{W}} = h^{-1}(\Lambda \otimes I_n) - (I_s \otimes J)$$

- System is split into one $n \times n$ and one $2n \times 2n$ system
- Runtime is reduced to $\mathcal{O}(n^3 + 8n^3) = \mathcal{O}(9n^3)$
- Overhead of matrix-vector products is just $\mathcal{O}(s^2 n)$

$\rightarrow$ 3 times faster than the naive implementation

EFFICIENT NLS FOR FIRK: COMPLEX ARITHMETIC

- Each complex conjugate eigenpair creates $2n \times 2n$ real systems

$$\begin{pmatrix} A & -B \\ B & A \end{pmatrix} \begin{pmatrix} \Delta \boldsymbol{w}_1 \\ \Delta \boldsymbol{w}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{pmatrix}$$

- Solving the linear system is equivalent to solving the $n \times n$ complex system

$$(A + \mathrm{i}B)(\Delta \boldsymbol{w}_1 + \mathrm{i}\Delta \boldsymbol{w}_2) = \boldsymbol{r}_1 + \mathrm{i}\boldsymbol{r}_2$$

- Just $\mathcal{O}(n^3 + 4n^3) = \mathcal{O}(5n^3)$ flops for Radau IIA 3-step - more than 5 times faster!!

CONVERGENCE, DIVERGENCE AND JACOBIAN CALLBACKS

- New convergence condition for the Newton loop, which is closely tied to the integrators acceptance / rejection norm
- Main idea: if the estimated residual obeys a stronger condition than the states for acceptance, then the NLS is converged

→ Avoids solving the NLS to unnecessary precision - in some sense to the precision that is exactly required!

- Detect divergence of the Newton loop early
- Keep the Jacobian constant, if the convergence rate is sufficiently fast

→ Jacobians of linear ODEs are evaluated just once

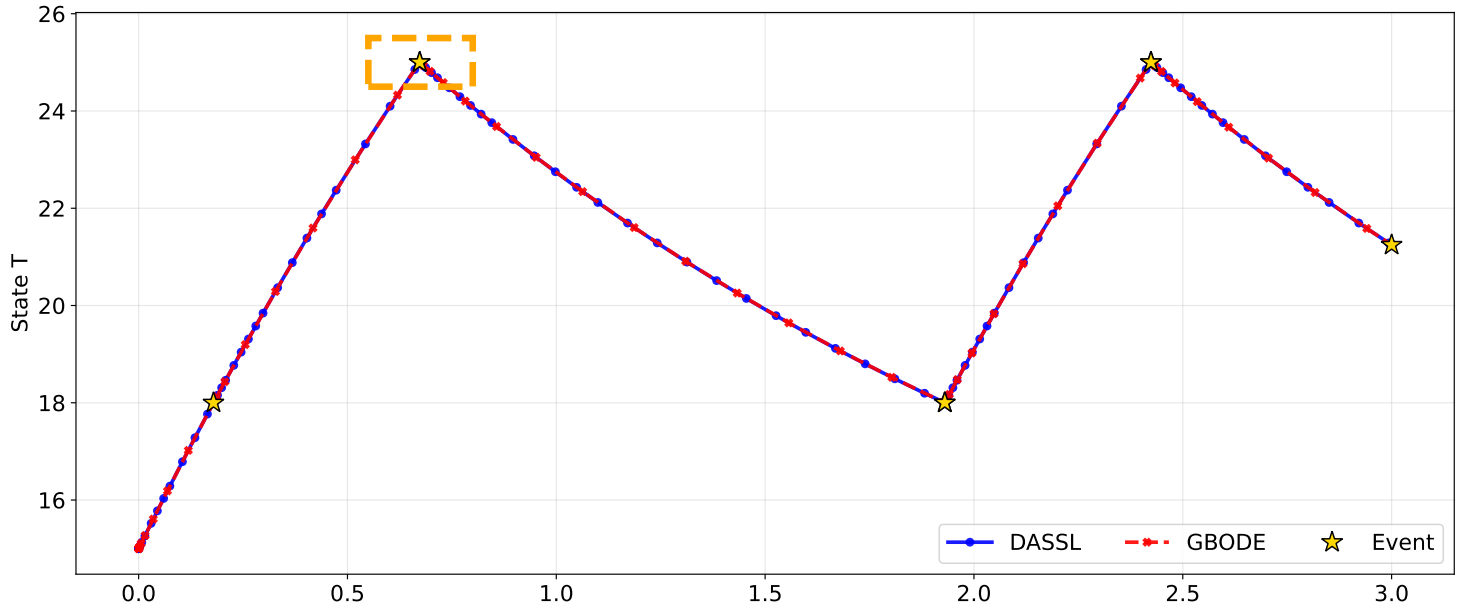
- Update start values for all collocation methods

A TAKE ON EVENTS

- After an event, BDF methods like DASSL or IDA have to redo bootstrapping and effectively fall back to an implicit Euler method

→ Step size must be completely reset to a tiny value

- This is often unnecessary and very expensive if you have a lot events



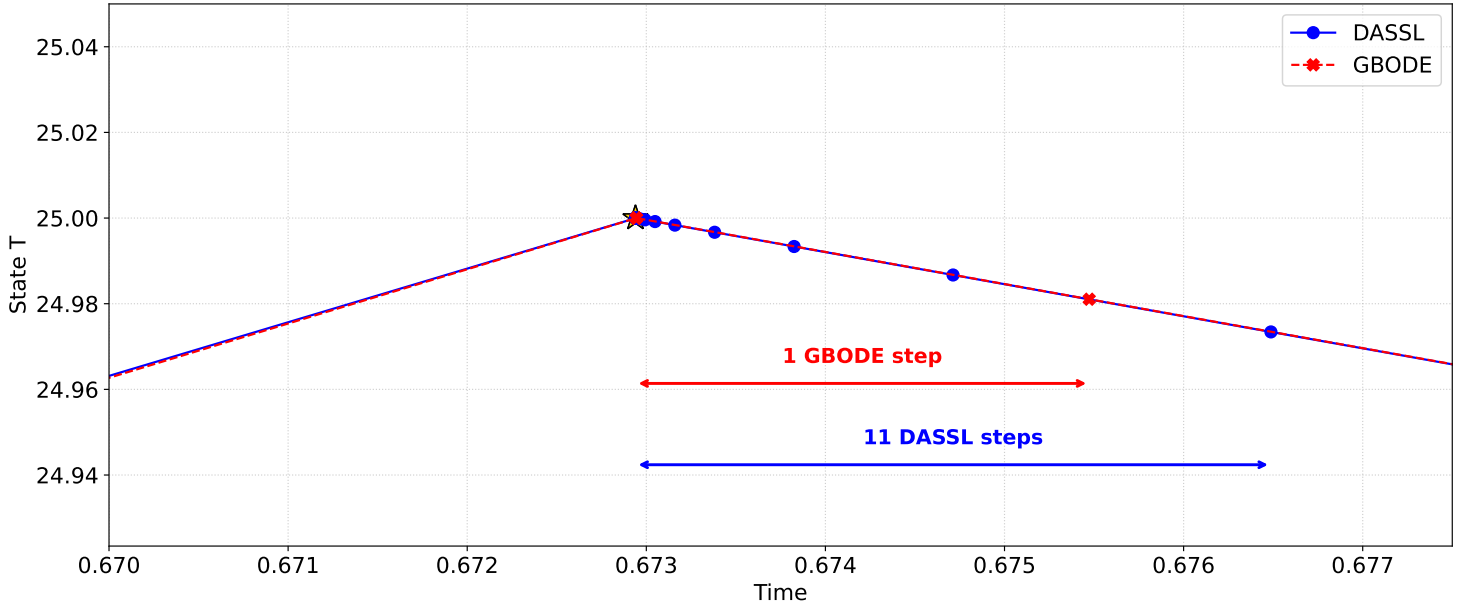
A TAKE ON EVENTS

- Runge-Kutta methods are 1-step methods and do not require bootstrapping

→ Step size must not be reset

→ Implemented a strategy that sets $h_{\text{new}} := \max\{0.1h_{\text{old}}, h_{\text{init}}\}$

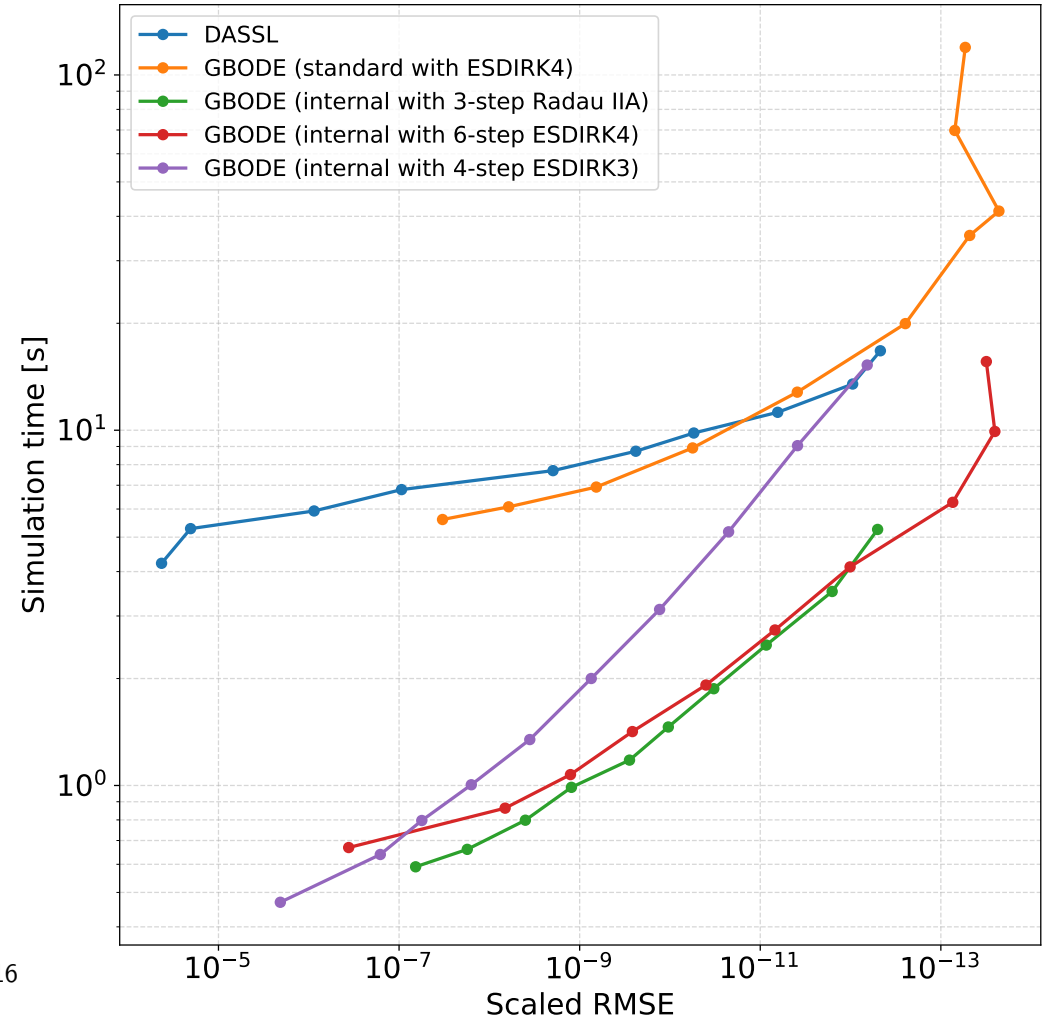
- Where to check zero-crossings?



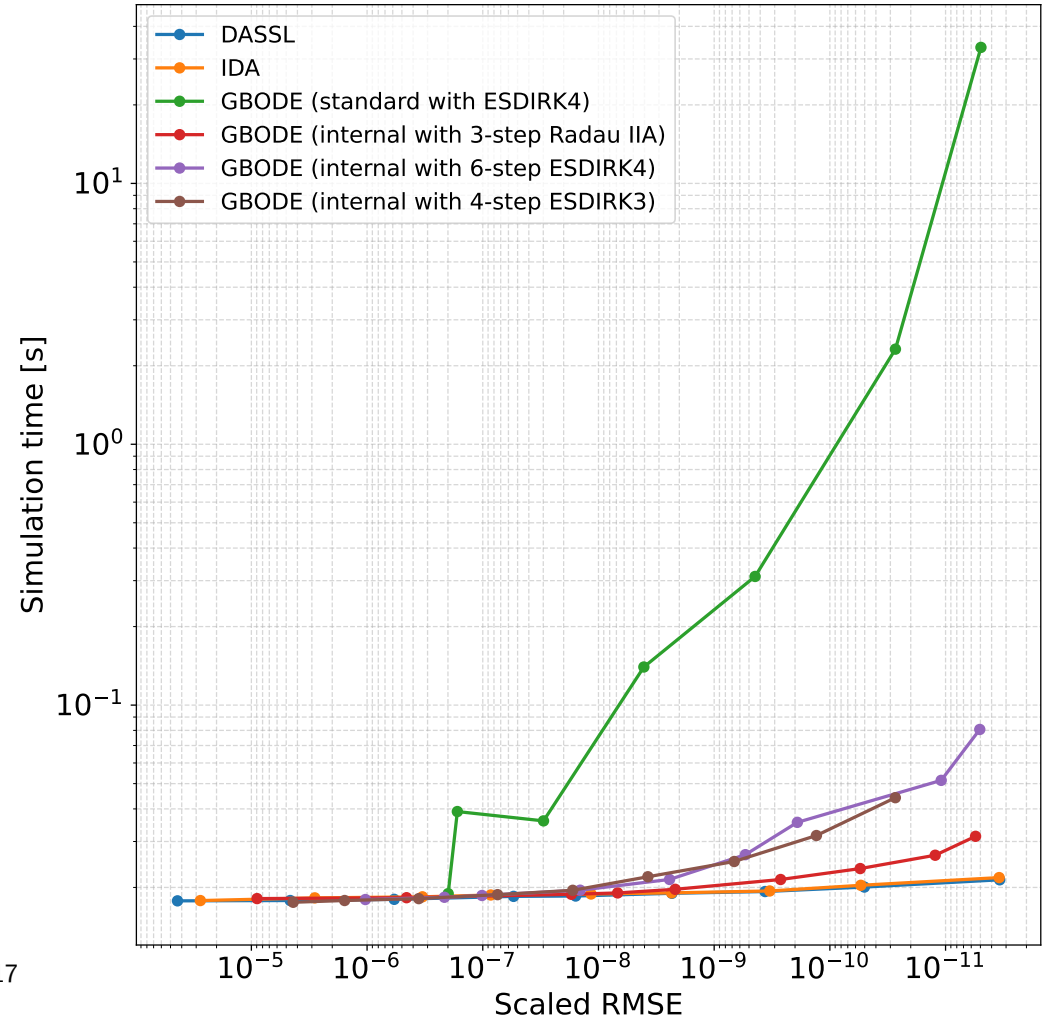
PERFORMANCE

- We test the new flag `-gbnls=internal` on several examples
- Always set `-gbnls=internal` and `-gberr=embedded`
- The tested methods are `-gbm=radauIIA3`, `-gbm=esdirk4` and `-gbm=esdirk3`
- IDA and DASSL are also included in the benchmark
- Tests are run on the latest OpenModelica master
- Timings are averaged over several runs and the error is given as the scaled RMSE with respect to the reference (IDA or DASSL) with tolerance 10^{-12}

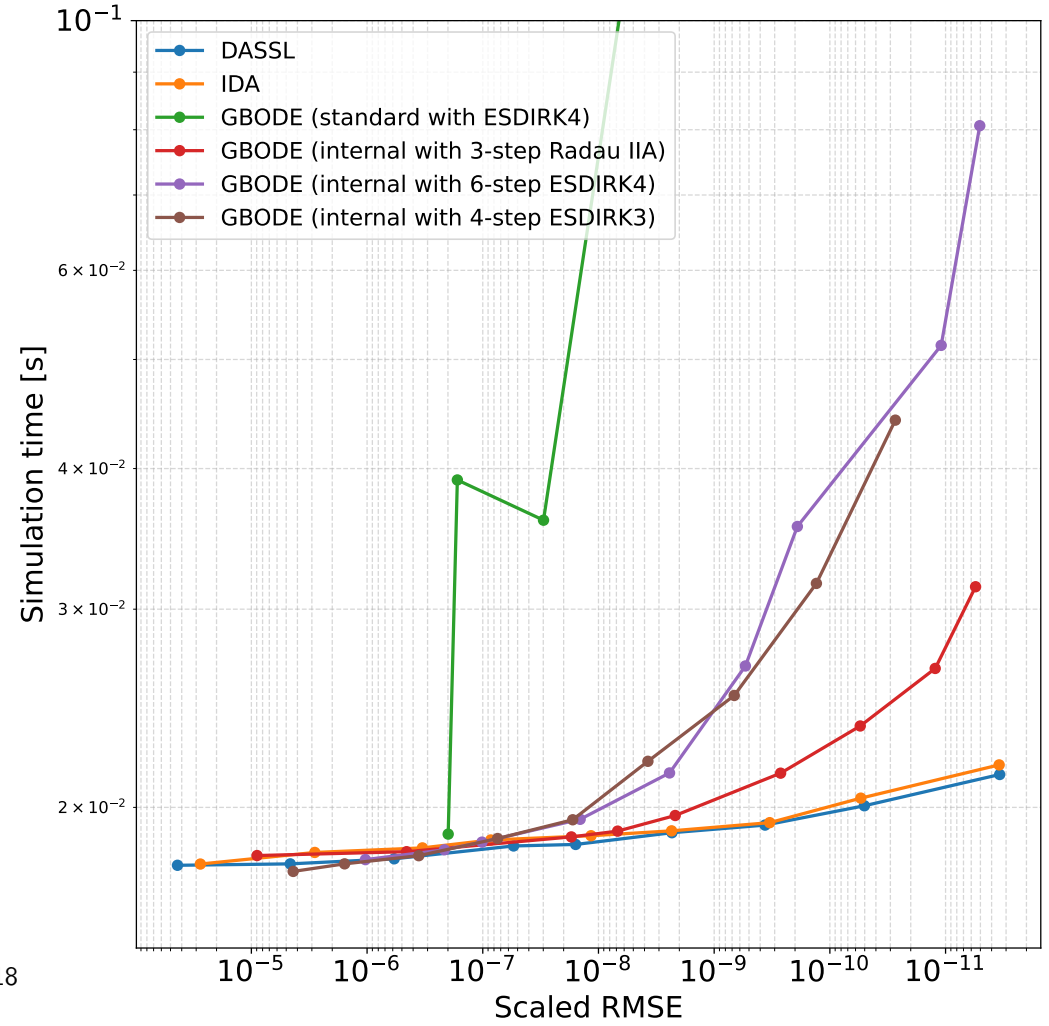
EXOGENOUS HEATING SYSTEM



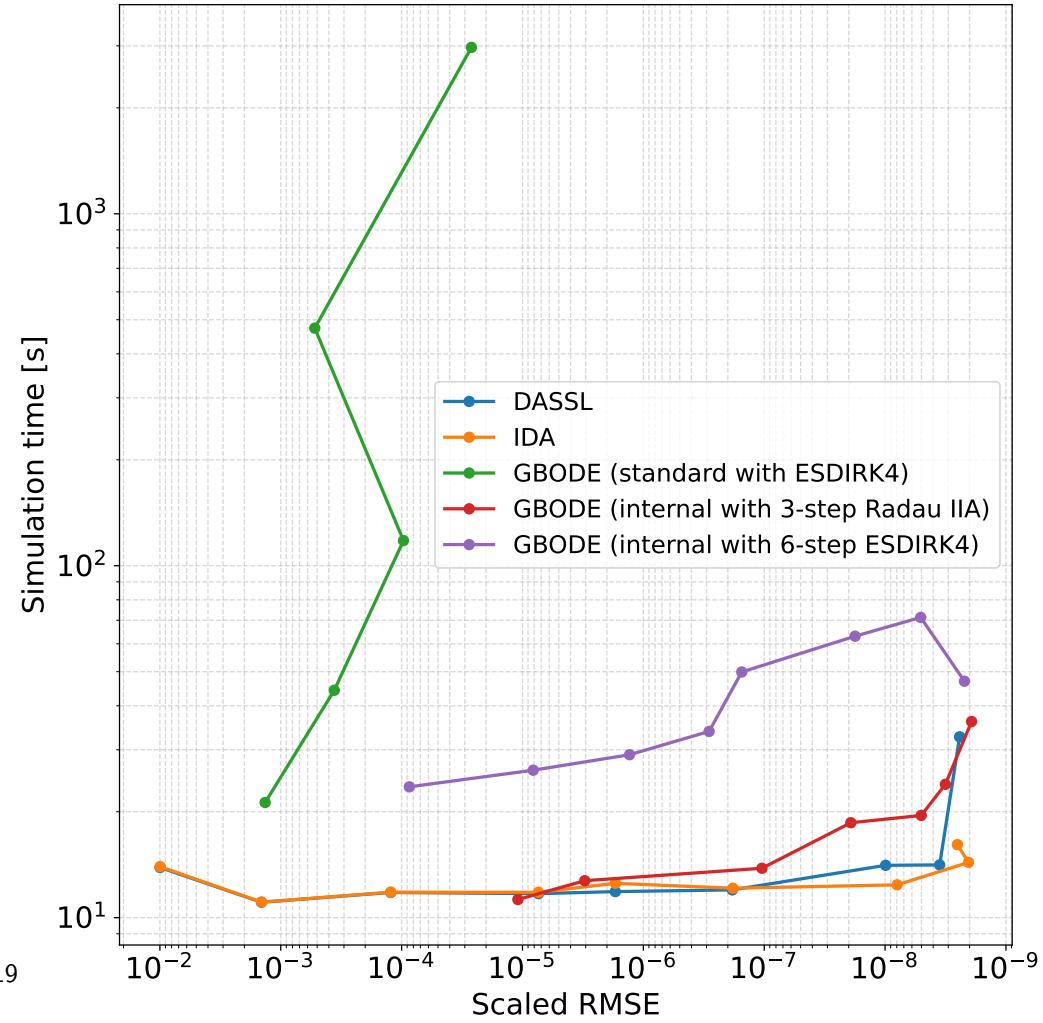
ROBERTSON REACTION I



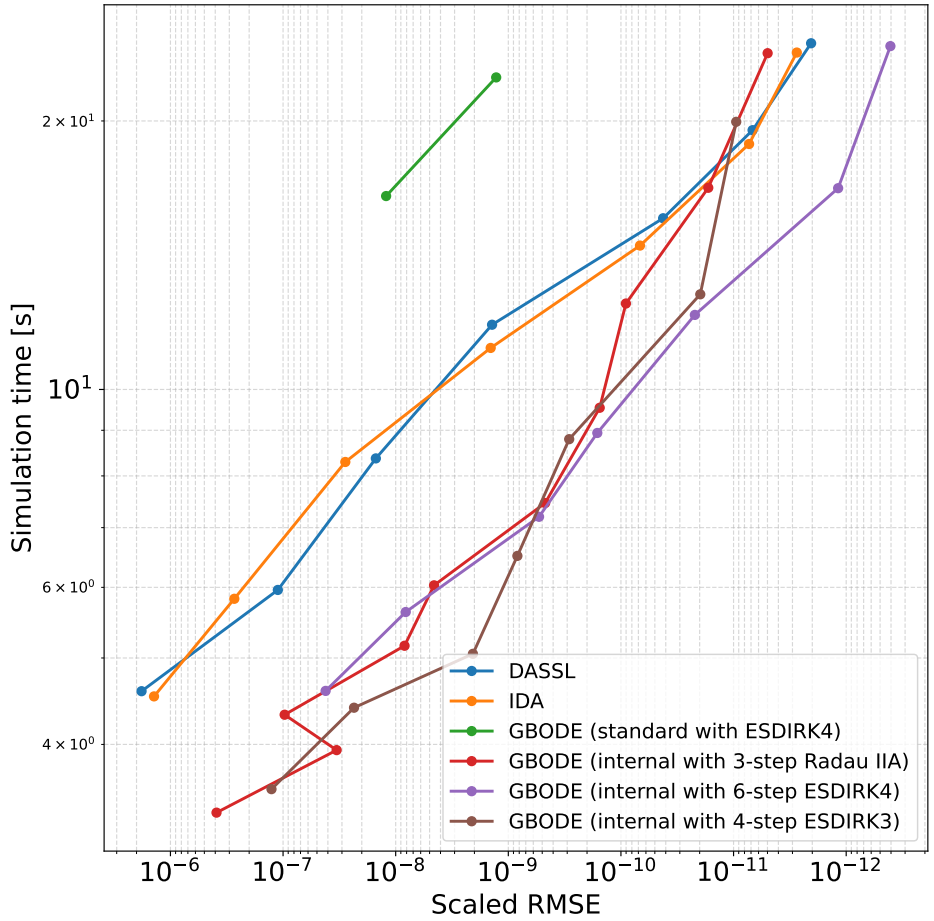
ROBERTSON REACTION II



SOFCPOLIMI.TESTS.BENCHMARKLINCONCLOSSES



BUILDINGS.*.SINGLEZONE.VAV.EXAMPLES.GUIDELINE36



SUMMARY

- What does `-gbnls=internal` include?
  - Efficient NLS for FIRK methods
  - New convergence, divergence and Jacobian recomputation strategies
  - Experimental step-size selection after events

→ Its suggested to use the latest master or nightly and `-gberr=embedded` `-gbm=radauIIA3`, if you want to try it

- If using the latest release, you must use a symbolic Jacobian
- Still to do
  - Tuning of parameters
  - Stage-Value Predictors for (E)SDIRK methods
  - Better event handling in GBODE
  - Extend `-gbnls=internal` to multi-rate simulations