Journal of Scientific Computing (2026) 106:18
https://doi.org/10.1007/s10915-025-03130-6

TECHNICAL NOTE

Relaxation Runge-Kutta Methods with First-Same-as-Last Structure

Sebastian Bleecke[1] · **Hendrik Ranocha**[1]

Received: 20 August 2025 / Revised: 20 August 2025 / Accepted: 4 November 2025 /
Published online: 4 December 2025

Abstract
Preserving invariants of differential equations under discretization can be important, e.g., to improve the qualitative and quantitative properties of numerical solutions. Recently, relaxation methods have been proposed as small modifications of standard time integration schemes guaranteeing the correct evolution of functionals of the solution. We investigate how to combine relaxation with error-based step size control for explicit Runge-Kutta methods using the first-same-as-last (FSAL) technique and demonstrate the improved efficiency of our new methods.

Keywords Explicit Runge-Kutta methods · Step size control · PID controller · First-same-as-last (FSAL) technique · Invariant conservation · Relaxation

Mathematics Subject Classification 65L06 · 65M20 · 65M70

1 Introduction

Consider an ordinary differential equation (ODE)

$$u'(t) = f(u(t)), \quad u(t^0) = u^0. \tag{1}$$

We are interested in such initial value problems equipped with an invariant η called *entropy* satisfying $\forall t, u\colon \eta'(u) f(t,u) = 0$, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t}\eta(u(t)) = 0. \tag{2}$$

It is often desirable to preserve the conservation of invariants when solving the ODE numerically since this can lead both qualitative and quantitative improvements. We concentrate on *relaxation Runge-Kutta methods* [1–3] that can be summarized as follows. First, the baseline Runge-Kutta method is used to compute the preliminary update u^{n+1} at time $t^{n+1} = t^n + \Delta t_n$.

✉ Sebastian Bleecke
sbleecke@uni-mainz.de

Hendrik Ranocha
hendrik.ranocha@uni-mainz.de

[1] Institute of Mathematics, Johannes Gutenberg University Mainz, Staudingerweg 9, 55128 Mainz, Germany

Then, the relaxed solution $u_\gamma^{n+1} = u^n + \gamma(u^{n+1} - u^n)$ is computed, where the scalar relaxation parameter γ is determined in each time step as the solution of the nonlinear scalar equation $\eta(u_\gamma^{n+1}) = \eta(u^n)$. From here on, we continue the time integration with u_γ^{n+1} as approximation at the relaxed time $t_\gamma^{n+1} = t^n + \gamma\Delta t_n$ instead of u^{n+1} at time $t^{n+1} = t^n + \Delta t_n$. The available theory [3] guarantees that the nonlinear scalar equation has a unique solution

$$\gamma = 1 + \mathcal{O}(\Delta t_n^{p-1}) \tag{3}$$

under some technical assumptions, where p is the order of the baseline method; the resulting relaxation Runge-Kutta method has at least the same order of accuracy p as the baseline method and preserves the invariant η.

Typically, explicit Runge-Kutta methods are used together with error-based step size control and the first-same-as-last (FSAL) technique [4]. However, combining relaxation with FSAL methods leads to efficiency problems [5, 6]. To demonstrate this, we follow take an approach that is common in software packages like OrdinaryDiffEq.jl [7] or SUNDIALS [8, 9][1]. Given a Runge-Kutta method with Butcher coefficients a_{ij}, b_i, c_i, we first compute the stages

$$y^i = u_\gamma^n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} f(y^j), \tag{4}$$

followed by the update of the main solution

$$u^{n+1} = u_\gamma^n + \Delta t_n \sum_{i=1}^{s} b_i f(y^i). \tag{5}$$

Next, we compute the embedded solution

$$\widehat{u}^{n+1} = u_\gamma^n + \Delta t_n \sum_{i=1}^{s} \widehat{b}_i f(y^i) + \Delta t_n \widehat{b}_{s+1} f(u^{n+1}) \tag{6}$$

and use a PID controller [11–13] to update the time step size Δt and decide whether the step should be accepted or rejected. When the step is accepted, relaxation is performed by computing the relaxation parameter γ as root of the scalar equation $\eta(u_\gamma^{n+1}) = \eta(u_\gamma^n)$, where

$$u_\gamma^{n+1} = u_\gamma^n + \gamma(u^{n+1} - u_\gamma^n). \tag{7}$$

However, we have to compute $f(u_\gamma^{n+1})$ at the beginning of the next step since we cannot reuse the value $f(u^{n+1})$ from the FSAL stage as would be possible without relaxation. The goal of this article is to develop new methods that combine relaxation efficiently with FSAL methods without additional right-hand side (RHS) evaluations. We present the new approaches in the following Section 2, demonstrate their performance in numerical experiments in Section 3, and summarize our findings in Section 4.

2 New Methods

We present two new approaches to avoid additional RHS evaluations.

[1] Relaxation methods are available in recent versions of ARKODE [10] in SUNDIALS.

2.1 FSAL-R

Following the structure of the naive combination of FSAL and relaxation, we compute everything as before but avoid computing the RHS $f(y^1) = f(u_\gamma^{n+1})$ at the beginning of the next step. Instead, we use the approximation

$$f(u_\gamma^{n+1}) \approx f(u_\gamma^n) + \gamma\big(f(u^{n+1}) - f(u_\gamma^n)\big). \tag{8}$$

This is justified by the following results.

Lemma 1 *A twice continuously differentiable function $g \in \mathcal{C}^2$ satisfies*

$$g(u_\gamma^{n+1}) = g(u_\gamma^n) + \gamma(g(u^{n+1}) - g(u_\gamma^n)) + \mathcal{O}(\Delta t_n^{p+1}). \tag{9}$$

Proof We use two Taylor approximations of g at u^{n+1}. First, we have

$$g(u_\gamma^{n+1}) = g(u^{n+1}) + g'(u^{n+1})(u_\gamma^{n+1} - u^{n+1}) + \mathcal{O}(\|u_\gamma^{n+1} - u^{n+1}\|^2), \tag{10}$$

where

$$\|u_\gamma^{n+1} - u^{n+1}\|^2 = \|(\gamma - 1)(u^{n+1} - u^n)\|^2 = \mathcal{O}(\Delta t_n^{2p}) \tag{11}$$

due to (3) and $u^{n+1} - u^n = \mathcal{O}(\Delta t_n)$. Second, we have

$$g(u_\gamma^n) = g(u^{n+1}) + g'(u^{n+1})(u_\gamma^n - u^{n+1}) + \mathcal{O}(\Delta t_n^2). \tag{12}$$

Thus, we obtain

$$\begin{aligned} &g(u_\gamma^{n+1}) - g(u_\gamma^n) - \gamma\big(g(u^{n+1}) - g(u_\gamma^n)\big) \\ &= (1-\gamma)g(u^{n+1}) + g'(u^{n+1})(u_\gamma^{n+1} - u^{n+1}) + \mathcal{O}(\Delta t_n^{2p}) \\ &\quad - (1-\gamma)g(u^{n+1}) - (1-\gamma)g'(u^{n+1})(u_\gamma^n - u^{n+1}) + \mathcal{O}((1-\gamma)\Delta t_n^2) \\ &= \mathcal{O}(\Delta t_n^{p+1}) \end{aligned} \tag{13}$$

due to (3) and the definition of u_γ^{n+1}. □

Theorem 1 *Using the approximation $f(u_\gamma^n) + \gamma(f(u^{n+1}) - f(u_\gamma^n))$ instead of $f(u_\gamma^{n+1})$ in the FSAL-R method ensures the local accuracy*

$$\begin{aligned} u(t^{n+1}) - u^{n+1} &= \mathcal{O}(\Delta t_n^{p+1}), \\ u(t^{n+1}) - \widehat{u}^{n+1} &= \mathcal{O}(\Delta t_n^{p}). \end{aligned} \tag{14}$$

Proof When computing the stages and the main/embedded solution, the approximated first stage derivative $f(u_\gamma^n)$ is multiplied by Δt_n. Thus, Lemma 1 shows that the difference between the approximated and the exact first stage is of order $\Delta t_{n-1}^{p+1}\Delta t_n$. Hence, the order of accuracy is not changed compared to the baseline method. □

The FSAL-R approach follows the basic structure of existing software libraries. Additionally, relaxation is only performed if a step is accepted. Finally, the right-hand side update $f(u_\gamma^{n+1})$ is computed using only a linear combination of values that are already computed in the process, thus saving one additional right-hand side in comparison to the naive approach. However, there can be issues with approximating the first stage, e.g., when using relaxation for dissipative problems. Thus, we also consider another approach.

2.2 R-FSAL

The other option to avoid additional RHS evaluations is to perform relaxation before estimating the error for step size control. In this case, we compute the stages y^i and the main solution u^{n+1} as in the naive approach. Then, we compute the relaxed solution u_γ^{n+1} at time t_γ^{n+1} before computing the embedded solution $\widehat{u}^{n+1}$. To save an additional RHS evaluation, we approximate $f(u^{n+1})$ required for $\widehat{u}^{n+1}$ by

$$f(u^{n+1}) \approx f(u_\gamma^n) + \frac{1}{\gamma}(f(u_\gamma^{n+1}) - f(u_\gamma^n)). \tag{15}$$

This is justified by the following result.

Lemma 2 *A twice continuously differentiable function $g \in \mathcal{C}^2$ satisfies*

$$g(u^{n+1}) = g(u_\gamma^n) + \frac{1}{\gamma}\left(g(u_\gamma^{n+1}) - g(u_\gamma^n)\right) + \mathcal{O}(\Delta t_n^{p+1}). \tag{16}$$

Proof Rearrange the interpolation formula of Lemma 1 and use $\gamma = 1 + \mathcal{O}(\Delta t_n^{p-1})$ (cf. (3)). □

Next, we need to decide how to estimate the error. We choose to use the relaxed solution u_γ^{n+1} and the embedded solution

$$\widehat{u}_\gamma^{n+1} = u_\gamma^n + \gamma \Delta t_n \left(\sum_{i=1}^{s} \widehat{b}_i f(y^i) + \widehat{b}_{s+1} \left(f(u_\gamma^n) + \frac{1}{\gamma}(f(u_\gamma^{n+1}) - f(u_\gamma^n)) \right) \right) \tag{17}$$

as approximations at time t_γ^{n+1}.

Theorem 2 *The R-FSAL method using the embedded solution (17) satisfies*

$$u(t_\gamma^{n+1}) - \widehat{u}_\gamma^{n+1} = \mathcal{O}(\Delta t_n^p). \tag{18}$$

Proof Similar to the proof of Theorem 1, the result follows directly from Lemma 2 and the order of accuracy of the embedded baseline method. □

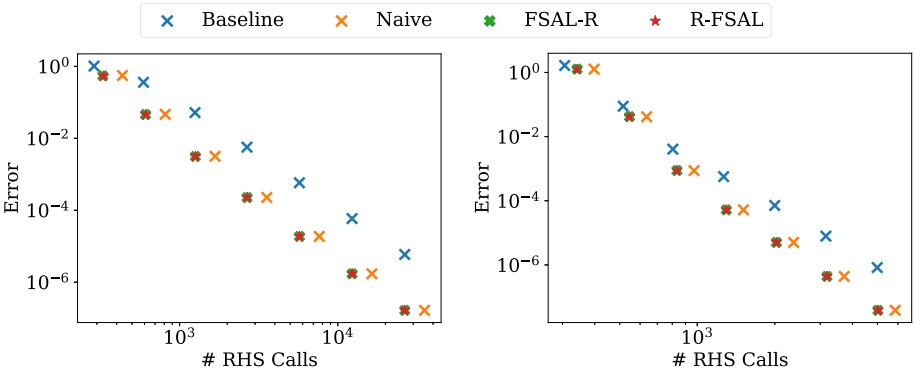
3 Numerical Experiments

Next, we briefly demonstrate the improved efficiency of the new methods. A more detailed study considering a broader range of problems and properties like step size control stability is available in the accompanying technical report [14].

We have implemented all methods in Julia [15]. To verify our implementation, we compared it to the schemes available in OrdinaryDiffEq.jl [7]. We applied the algorithm of [16] implemented in Roots.jl [17] to compute the relaxation parameter γ. We used Matplotlib [18] (wrapped in the Julia package PyPlot.jl) to create the figures. The source code to reproduce all numerical experiments is available online [19].

We consider the Benjamin-Bona-Mahony (BBM) equation [20]

$$(\mathrm{I} - \partial_x^2)\partial_t u(t,x) + \partial_x \frac{u(t,x)^2}{2} + \partial_x u(t,x) = 0 \tag{19}$$



(a) Bogacki-Shampine BS3. (b) Dormand-Prince DP5.

Fig. 1 Errors at the final time for a given number of RHS evaluations

with periodic boundary conditions. We consider the traveling wave solution

$$u(t,x) = A\cosh(K(x - c\,t))^{-2} \tag{20}$$

with parameters

$$x_{\min} = 0, \quad x_{\max} = 2, \quad c = 1.2, \quad A = 3(c-1), \quad K = 0.5\sqrt{1 - 1/c}. \tag{21}$$

We use a Fourier collocation method conserving a discrete equivalent of the quadratic invariant

$$\frac{1}{2}\int_{x_{min}}^{x_{max}} \left(u^2 + (\partial_x u)^2\right), \tag{22}$$

see [21] for details. The spatial semidiscretization uses SummationByPartsOperators.jl [22] wrapping FFTW.jl [23].

We apply the third-order scheme BS3 of Bogacki and Shampine [24] and the fifth-order Dormand-Prince method DP5 [4] in four variants: the baseline method without relaxation, the naive combination of FSAL and relaxation described in the introduction, the FSAL-R method, and the R-FSAL method. The results of the numerical experiments are visualized in the work-precision diagrams shown in Figure 1.

The naive method yields more accurate results than the baseline method for a given tolerance but uses more RHS evaluations. Both new methods (R-FSAL and FSAL-R) have the same accuracy as the naive implementation without increasing the number of RHS evaluations visibly. Thus, both new methods increase the accuracy by roughly 25% compared to the baseline method without increasing the computational work significantly.

4 Summary and Conclusions

We have developed two new approaches to combine relaxation with adaptive time stepping based on the first-same-as-last (FSAL) technique for explicit Runge-Kutta methods. The FSAL-R method performs relaxation after step size control while the R-FSAL method performs relaxation before step size control. Both methods keep the same accuracy as the naive combination of FSAL and relaxation. We have demonstrated the increased efficiency of the

new methods in numerical experiments for the Benjamin-Bona-Mahony equation, improving the accuracy by roughly 25% without increasing the computational work significantly compared to the methods without relaxation.

For entropy-conservative problems, we have not observed clear evidence that one method is superior to the other. For a general-purpose implementation, the FSAL-R appears to be preferable since it does not require changes to the core structure of many software libraries. However, the FSAL-R method may be problematic for dissipative problems since the entropy estimates required for relaxation [2, 3] cannot be used directly. In this case, the R-FSAL method can be advantageous.

Author Contributions Formal analysis and investigation: Sebastian Bleecke, Hendrik Ranocha; Funding acquisition: Hendrik Ranocha; Visualization: Sebastian Bleecke, Hendrik Ranocha; Writing - original draft preparation: Sebastian Bleecke, Hendrik Ranocha; Writing - review and editing: Sebastian Bleecke, Hendrik Ranocha.

Funding Open Access funding enabled and organized by Projekt DEAL. We acknowledge support by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation, project number 513301895) and the Daimler und Benz Stiftung (Daimler and Benz foundation, project number 32-10/22).

Data Availability The source code and datasets generated and analyzed for this study are available in our reproducibility repository [19].

Declarations

Conflicts of Interest The authors declare that they have no conflict of interest.

References

1. Ketcheson, D.I.: Relaxation Runge-Kutta methods: Conservation and stability for inner-product norms. SIAM Journal on Numerical Analysis **57**(6), 2850–2870 (2019). https://doi.org/10.1137/19M1263662. arXiv:1905.09847
2. Ranocha, H., Sayyari, M., Dalcin, L., Parsani, M., Ketcheson, D.I.: Relaxation Runge-Kutta methods: Fully-discrete explicit entropy-stable schemes for the compressible Euler and Navier-Stokes equations. SIAM Journal on Scientific Computing **42**(2), 612–638 (2020). https://doi.org/10.1137/19M1263480. arXiv:1905.09129
3. Ranocha, H., Lóczi, L., Ketcheson, D.I.: General relaxation methods for initial-value problems with application to multistep schemes. Numerische Mathematik **146**, 875–906 (2020). https://doi.org/10.1007/s00211-020-01158-4. arXiv:2003.03012
4. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980). https://doi.org/10.1016/0771-050X(80)90013-3
5. Rogowski, M., Dalcin, L., Parsani, M., Keyes, D.E.: Performance analysis of relaxation Runge-Kutta methods. The International Journal of High Performance Computing Applications (2022). https://doi.org/10.1177/10943420221085947
6. Al Jahdali, R., Dalcin, L., Parsani, M.: On the performance of relaxation and adaptive explicit Runge-Kutta schemes for high-order compressible flow simulations. J. Comput. Phys. (2022). https://doi.org/10.1016/j.jcp.2022.111333

7. Rackauckas, C., Nie, Q.: DifferentialEquations.jl – A performant and feature-rich ecosystem for solving differential equations in Julia. Journal of Open Research Software 5(1), 15 (2017) https://doi.org/10.5334/jors.151
8. Gardner, D.J., Reynolds, D.R., Woodward, C.S., Balos, C.J.: Enabling new flexibility in the SUNDIALS suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software (TOMS) (2022). https://doi.org/10.1145/3539801
9. Hindmarsh, A.C., Brown, P.N., Grant, K.E., Lee, S.L., Serban, R., Shumaker, D.E., Woodward, C.S.: SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. ACM Transactions on Mathematical Software (TOMS) **31**(3), 363–396 (2005). https://doi.org/10.1145/1089014.1089020
10. Reynolds, D.R., Gardner, D.J., Woodward, C.S., Chinomona, R.: ARKODE: A flexible IVP solver infrastructure for one-step methods. ACM Transactions on Mathematical Software **49**(2), 1–26 (2023). https://doi.org/10.1145/3594632
11. Söderlind, G.: Time-step selection algorithms: Adaptivity, control, and signal processing. Appl. Numer. Math. **56**(3–4), 488–502 (2006). https://doi.org/10.1016/j.apnum.2005.04.026
12. Söderlind, G., Wang, L.: Adaptive time-stepping and computational stability. J. Comput. Appl. Math. **185**(2), 225–243 (2006). https://doi.org/10.1016/j.cam.2005.03.008
13. Kennedy, C.A., Carpenter, M.H.: Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Appl. Numer. Math. **44**(1–2), 139–181 (2003). https://doi.org/10.1016/S0168-9274(02)00138-1
14. Bleecke, S., Ranocha, H.: Step Size Control for Explicit Relaxation Runge-Kutta Methods Preserving Invariants. https://doi.org/10.48550/arXiv.2311.14050
15. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671. arXiv:1411.1607
16. Alefeld, G., Potra, F.A., Shi, Y.: Algorithm 748: Enclosing zeros of continuous functions. ACM Transactions on Mathematical Software (TOMS) **21**(3), 327–344 (1995). https://doi.org/10.1145/210089.210111
17. Verzani, J.: Roots.jl: Root finding functions for Julia. https://github.com/JuliaMath/Roots.jl (2020)
18. Hunter, J.D.: Matplotlib: A 2D graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
19. Bleecke, S., Ranocha, H.: Reproducibility repository for "Step size control for explicit relaxation Runge-Kutta methods preserving invariants". https://github.com/ranocha/2023_FSAL_relaxation (2023). https://doi.org/10.5281/zenodo.10201246
20. Benjamin, T.B., Bona, J.L., Mahony, J.J.: Model equations for long waves in nonlinear dispersive systems. Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences 272(1220), 47–78 (1972) https://doi.org/10.1098/rsta.1972.0032
21. Ranocha, H., Mitsotakis, D., Ketcheson, D.I.: A broad class of conservative numerical methods for dispersive wave equations. Communications in Computational Physics **29**(4), 979–1029 (2021). https://doi.org/10.4208/cicp.OA-2020-0119. arXiv:2006.14802
22. Ranocha, H.: SummationByPartsOperators.jl: A Julia library of provably stable semidiscretization techniques with mimetic properties. Journal of Open Source Software 6(64), 3454 (2021) https://doi.org/10.21105/joss.03454
23. Frigo, M., Johnson, S.G.: The design and implementation of FFTW3. Proc. IEEE **93**(2), 216–231 (2005). https://doi.org/10.1109/JPROC.2004.840301
24. Bogacki, P., Shampine, L.F.: A 3(2) pair of Runge-Kutta formulas. Appl. Math. Lett. **2**(4), 321–325 (1989). https://doi.org/10.1016/0893-9659(89)90079-7