COMM. APP. MATH. AND COMP. SCI.
Vol. , No. , 2021
https://doi.org/10.2140/camcos.2021..101

POSITIVITY-PRESERVING ADAPTIVE RUNGE–KUTTA METHODS

STEPHAN NÜSSLEIN, HENDRIK RANOCHA AND DAVID I. KETCHESON

Many important differential equations model quantities whose value must remain positive or stay in some bounded interval. These bounds may not be preserved when the model is solved numerically. We propose to ensure positivity or other bounds by applying Runge–Kutta integration in which the method weights are adapted in order to enforce the bounds. The weights are chosen at each step after calculating the stage derivatives, in a way that also preserves (when possible) the order of accuracy of the method. The choice of weights is given by the solution of a linear program. We investigate different approaches to choosing the weights by considering adding further constraints. We also provide some analysis of the properties of Runge–Kutta methods with perturbed weights. Numerical examples demonstrate the effectiveness of the approach, including application to both stiff and non-stiff problems.

1. Introduction

Many physical processes can be described with differential equations. The physical quantities that are involved in these processes often only make sense if they remain within certain bounds. For instance, concentrations must be non-negative (we will often say simply *positive* for short), while probabilities or mass fractions must remain in $[0,1]$. The ordinary differential equations (ODEs) or partial differential equations (PDEs) that model these quantities are often too complex to be solved analytically and therefore require numerical approximation. Numerical methods generally may not satisfy these bound constraints. In the present work, we develop an approach to ensuring positivity or other bound constraints using Runge–Kutta methods (RKMs) for the solution of ODEs or semi-discretized PDEs.

Color: 1 3 4 6 7 8 9 10 11 12

We say an initial value problem

$$u'(t) = f(t,u), \tag{1}$$

$$u(0) = u_0, \tag{2}$$

ORCID: Nüßlein 0000-0002-2455-4222; Ranocha 000-0002-3456-2277; Ketcheson 0000-0002-1212-126X.
MSC2020: 65L06, 65L20, 65M12.
Keywords: positivity preserving, bound preserving, Runge–Kutta methods, linear programming.

where $u : [0, T] \to \mathbb{R}^m$ is positive if

$$u(0) \geq 0 \implies u(t) \geq 0 \text{ for all } t \in [0, T]. \tag{3}$$

Here and in the following, inequalities like $u \geq 0$ are meant componentwise. A sufficient condition for positivity of (1) is

$$u_i = 0 \implies f_i(t, [u_1, \ldots, u_i, \ldots, u_n]^T) \geq 0 \quad \forall u \geq 0, \quad \forall t \in [0, T]. \tag{4}$$

For such ODEs, the backward Euler method is guaranteed to preserve positivity under any step size, while the forward Euler will preserve positivity for small enough Δt [12]. Any RKM (or in fact any general linear method) that is unconditionally positivity preserving for all positive ODEs must have order ≤ 1 [3]. For any higher order method, we expect positivity only under some restriction on the time step size.

Several approaches to ensuring numerical positivity exist in the literature. The most basic approach is orthogonal projection onto the positive orthant, which means simply setting negative values to zero [29]. This approach is often problematic; for instance, it will violate linear invariants such as mass conservation. As another approach, one may use event finding methods in order to stop when any solution component reaches zero, and then proceed in some special way [30]. This approach is implemented in the MATLAB ODE Suite along with the idea of redefining the ODE outside the positive orthant (usually by evaluating at the nearest point on the boundary of the positive orthant). If positivity is preserved under a forward Euler step (with some step size restriction $\Delta t \leq \Delta t_{\text{FE}}$), then any strong stability preserving Runge–Kutta (SSPRK) method will also preserve positivity (with a modified step size restriction) [8]. Specifically, the positivity of the method is ensured for time steps $\Delta t \leq \mathscr{C} \Delta t_{\text{FE}}$, where $\mathscr{C}$ depends on the SSPRK method. Modified Patankar–Runge–Kutta (MPRK) methods represent another approach to ensuring positivity for specific classes of ODEs. MPRK methods introduce multiplicative factors within the Runge–Kutta stages to ensure positivity, but require the solution of a linear algebraic system; see e.g. [19] and references therein. Finally, we mention diagonally split Runge–Kutta (DSRK) methods, which can be unconditionally positive and have order higher than one. Like MPRK schemes, DSRK methods avoid the restriction mentioned above because they are not general linear methods [11]. However, in practice unconditionally positive DSRK methods are less accurate than backward Euler for large step sizes [21].

The rather discouraging theoretical result of [3] shows that one should not hope to preserve positivity with a single method for every problem and every initial condition. In the present work we take an approach based on the idea that for a particular problem and initial condition, there often exists a method of high order that is positivity preserving, at least for a single step. The main idea is to adaptively choose the weights b of the RKM, after the stage values are known, in a way that

ensures positivity. The selection of the weights requires the solution of a linear program (LP) at every step for which the numerical solution would otherwise be non-positive. This is a significant cost, but may in some cases be an economical alternative to rejecting a step or using excessively small step sizes.

The idea of using different weights within an RKM is not new; for instance it is the basis of error approximation using embedded RK pairs [9]. The idea of adapting the weights after calculating the stage values has also been used, for instance in [16]. In this case it is used to adapt the properties of the time integrator for a method of lines solution of a PDE. Another class of methods that adapt the weights at the end of an RK step are the relaxation Runge–Kutta (RRK) methods. In these, the weights are scaled by a scalar relaxation parameter to guarantee conservation or monotonicity of a desired functional; e.g. to conserve or dissipate energy or entropy [15; 27; 26].

Our means to ensure positivity can be interpreted as a projection approach, where the numerical solution is adapted to satisfy the positivity constraint at the end of each step. In contrast to simple orthogonal projection, which has also been proposed to deal with positivity constraints [29], our approach preserves all linear invariants of the given ODE. These invariants can be very important, e.g. the total mass for a transport problem or in reaction systems. Preservation of linear invariants has been shown to be an important advantage of RRK methods over orthogonal projection methods [25]. Of course, it is also possible to enforce the preservation of linear invariants in projection methods, but the invariants have to be known explicitly [28].

The paper unfolds as follows. In Section 2 the main idea is explained. Section 3 contains the formulation of the linear program for selection of the weights at each step. Section 4.1 describes how the new approach can be used with different RKMs, how it can be combined with adaptive error control, and how the region of absolute stability can be approximated. In Section 5 numerical results are given for multiple test problems. A conclusion is given in Section 6.

2. Bound-preserving adaptive Runge–Kutta methods

When computing the solution of an ODE $u' = f(t, u)$ using an RKM with s stages and the Butcher tableau

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} \tag{5}$$

the stage values are computed according to

$$y_j = u^n + \Delta t \sum_{k=1}^{s} a_{jk} f(t^n + \Delta t c_k, y_k), \quad j = 1, \ldots, s. \tag{6}$$

Based on these values, the next solution u^{n+1} is computed as

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} f(t^n + \Delta t c_j, y_j) b_j. \tag{7}$$

Let $f_j = f(t^n + \Delta t c_j, y_j)$; then we can write (7) as

$$u^{n+1} = u^n + \Delta t F b, \tag{8}$$

where the jth column of F is f_j. We wish to impose the discrete analog of (3); i.e.

$$u^n \geq 0 \implies u^{n+1} \geq 0, \tag{9}$$

or more general bound constraints

$$\alpha \leq u^n \leq \beta \implies \alpha \leq u^{n+1} \leq \beta. \tag{10}$$

We will focus on the case of positivity while keeping in mind that the methodology extends to general bounds. The main idea of the present work is that if the new solution u^{n+1} contains negative entries, we can replace the weights in (8) with a set of modified weights $\tilde{b}$ such that the resulting solution is positive:

$$\tilde{u}^{n+1} = u^n + \Delta t F \tilde{b}^n \geq 0. \tag{11}$$

Indeed, we can view (11) as a linear constraint on the choice of modified weights $\tilde{b}^n$. Since we have already computed the intermediate stages, F is a known, fixed matrix. In order to ensure that the modified solution $\tilde{u}^{n+1}$ is accurate, we can also constrain $\tilde{b}$ to satisfy the Runge–Kutta order conditions up to some order (ideally, the same order as the original method). Observe that all of the order conditions are linear in the weights, so that these additional constraints take the form

$$Q\tilde{b} = r$$

for some fixed matrix Q and vector r. By applying this technique at each step, we integrate (1) with a sequence of Runge–Kutta methods with coefficients $(A, \tilde{b}^n)$. At any step for which the solution u^{n+1} produced by method (A, b) is positive, we do not need to modify the weights and can simply accept this unmodified solution. Note that linear invariants (such as mass conservation) of the solution are automatically preserved in this approach, since at each step we use a Runge–Kutta method.

Example I. The main goal is to choose a method (A, b) such that u^{n+1} approximates the solution of the ODE $u(t_{n+1})$. An obvious objective while modifying the Runge–Kutta coefficients is to retain a high order of accuracy, but this does not fully determine the choice of weights in general. To get a better understanding for the method we consider the behavior for a simple problem.

We take the linear, positivity preserving ODE [18]

$$u'(t) = Lu(t), \quad u(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad L = \begin{pmatrix} -5 & 1 \\ 5 & -1 \end{pmatrix}, \tag{12}$$

and use the three stage, third order SSP method SSP(3,3)

$$\begin{array}{c|ccc} 0 & & & \\ 1 & 1 & & \\ 1/2 & 1/4 & 1/4 & \\ \hline & 1/6 & 1/6 & 2/3 \end{array} \tag{13}$$

of [31]. The matrix L has the eigenvalues zero and -6 and its operator norm is $2\sqrt{13}$. The real-axis stability interval of SSP33 includes the interval $[-2.5, 0]$. We take $\Delta t = 1/3$, which satisfies the spectral condition and guarantees boundedness (though not monotonicity) of the solution. The corresponding stage derivatives are

$$f(y_1) = \begin{pmatrix} -5 \\ 5 \end{pmatrix}, \quad f(y_2) = \begin{pmatrix} 5 \\ -5 \end{pmatrix}, \quad f(y_3) = \begin{pmatrix} -5 \\ 5 \end{pmatrix}. \tag{14}$$

The value of the next step using the standard weights is

$$u^1 = \begin{pmatrix} -1/9 \\ 10/9 \end{pmatrix}. \tag{15}$$

Since the first component of the new solution is negative, we want to adapt the weights to ensure positivity. All weights that comply with the constraints for first and second order of accuracy can be expressed as

$$\tilde{b} = \begin{pmatrix} 1/6 \\ 1/6 \\ 2/3 \end{pmatrix} + \alpha \begin{pmatrix} 1/2 \\ 1/2 \\ -1 \end{pmatrix}, \qquad \alpha \in \mathbb{R}. \tag{16}$$

We have one degree of freedom for the choice of the weights, parameterized by α. If the general expression for the weights is inserted in (8) the general solution is

$$u^1 = u^0 + \Delta t\,(f_1, f_2, f_3)\,\tilde{b} = \begin{pmatrix} -1/9 \\ 10/9 \end{pmatrix} + \alpha \begin{pmatrix} 5 \\ -5 \end{pmatrix}. \tag{17}$$

By changing the parameter α, the weights and the new solution are altered. With a suitable choice of $\alpha \in [1/45, 2/9]$, any u that complies with mass conservation and positivity can be reached. By adding additional constraints on the weights, the choice of α can be narrowed down. An objective function is also needed to make the choice unique. This should be designed in a way to prefer weights that are close to the original weights.

We see that the choice of $\tilde{b}$ is subject to linear equality and inequality constraints. If we choose a linear objective function, the resulting problem for finding the modified weights is a linear program, which can be efficiently solved by standard algorithms. A natural choice of objective function is

$$\text{minimize } \|\tilde{b} - b\|_1.$$

The resulting problem can be phrased as an LP by using slack variables. In general, this LP may not have a solution; we can relax the constraints by requiring a lower order of consistency than the design order of the method. These choices and alternatives will be considered in Section 3.

In contrast to other projection methods [29; 28], minimizing the deviation of the weights instead of the deviation of the projected solution is computationally much more efficient for large systems, arising for example in the discretization of PDEs.

Example II. To illustrate the usage of the method we consider the following reaction system [19]:

$$u_1' = 0.01u_2 + 0.01u_3 + 0.003u_4 - \frac{u_1u_2}{0.01+u_1}, \tag{18a}$$

$$u_2' = \frac{u_1u_2}{0.01+u_1} - 0.01u_2 - 0.5(1-\exp(-1.21u_2^2))u_3 - 0.05u_2, \tag{18b}$$

$$u_3' = 0.5(1-\exp(-1.21u_2^2))u_3 - 0.01u_3 - 0.02u_3, \tag{18c}$$

$$u_4' = 0.05u_2 + 0.02u_3 + 0.003u_4, \tag{18d}$$

with initial conditions

$$u(0) = (8, 2, 1, 4)^T. \tag{19}$$

We wrote (18) as in [19], sometimes using multiple terms containing the same variables but with different constants, e.g. $-0.01u_2 - 0.05u_2$ in the time derivative of u_2. This notation is useful to see the structure of a production-destruction system which is exploited for positivity-preserving (modified) Patankar–Runge–Kutta methods as in [19]. We will use the same notation also later in this article.

Using the Cash–Karp RK5 method [4] and $\Delta t = 0.005$ to solve (18), the approximated solution contains negative values. This causes qualitatively wrong solutions to the problem. In Figure 1 the results obtained are plotted with dashed lines. At $t = 1.905$ the value of u_1 gets negative. This leads to a diverging solution.

Now the weights are adapted. The adapted weights are of fourth order. The results are also plotted in Figure 1, with solid lines. The positivity constraint is now fulfilled. A qualitatively correct solution is obtained.

The difference $\|\tilde{b} - b\|_1$ is also plotted in Figure 1. No modification of the weights is required for $t < 1.905$. At $t = 1.905$ the weights are first adapted to ensure the positivity of the solution. For $t > 2.63$ the original set of weights again lead to a positive solution, and no further modification is necessary.

3. Selection of modified weights

In this section, we consider further the formulation of the LP to choose the modified weights $\tilde{b}$. In particular, we focus on the choice of objective function and how to relax the constraints to ensure that a feasible solution exists.

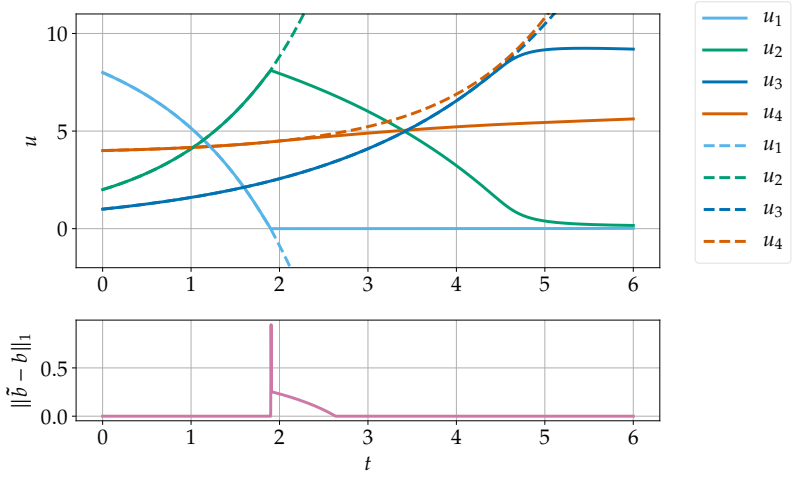

Figure 1. Numerical approximation of the reaction problem (18) computed with Cash–Karp RK5 and $\Delta t = 0.005$. The dashed lines are the approximations obtained without the adaption of the weights. The lower plot shows the adaptation of the weights.

3.1. ***Order conditions.*** The order conditions for an s-stage, order p RKM are a set of equations depending on A, b, and c. As mentioned already, if A and c are given, the order conditions are linear in b and can be written as $Q_p b = r_p$, where $Q_p \in \mathbb{R}^{v \times s}$, $r_p \in \mathbb{R}^v$ represent the set of all conditions up to and including order p. Here v is the number of order conditions. It may not be possible to find modified weights that also satisfy the conditions of order p and yield positivity, so in general the modified weights will be a solution of

$$Q_{\tilde{p}} \tilde{b} = r_{\tilde{p}}$$

for some $\tilde{p} \leq p$. Since we have s degrees of freedom $\tilde{b}_j$, we need at a minimum to choose $\tilde{p}$ so that $\operatorname{rank}(Q_{\tilde{p}}) < s$. Because the quadrature conditions are linearly independent, we have $\operatorname{rank}(Q_p) \geq p$, so we must take $\tilde{p} < s$. In general we may need to take $\tilde{p}$ even smaller in order to achieve positivity.

3.2. ***Choice of objective function and additional constraints.*** In the design of Runge–Kutta methods, weights are carefully chosen not only to satisfy the order conditions but also to give desirable properties such as a good region of absolute stability, small error coefficients, and so forth. Replacing these carefully chosen weights b with arbitrary weights $\tilde{b}$ could lead to the loss of these desirable properties. In order to preserve as much as possible the good properties of the method, we use as objective function $\|\tilde{b} - b\|_1$. This has the additional benefit of penalizing weights with large magnitude in general, avoiding large truncation or cancellation errors. This also ensures that if no negative solution values appear, the solution of

the LP is simply the original method weights. Thus we have the following LP (see also Figure 2, left):

LP

(Free adaptation) Given F, $\tilde{p}$, and b, find $\tilde{b}$ that minimizes $\|\tilde{b}-b\|_1$ subject to

$$u^{n+1} = u^n + \Delta t F \tilde{b} \geq 0, \tag{20a}$$

$$Q_{\tilde{p}} \tilde{b} = r_{\tilde{p}}. \tag{20b}$$

Of course, there is still no guarantee that the modified weights will be close to the original method weights. In some examples we have observed that large modifications of the weights can lead to inaccurate solutions even though the order conditions are satisfied. In order to avoid issues that might be caused by poor weights, we can additionally use either or both of the following ideas:

- Convex adaptation: Select in advance a set of desirable weight vectors $b^1, b^2, \ldots, b^K$ corresponding to known good methods, and restrict the choice of $\tilde{b}$ to convex combinations of this set. (See Figure 2, right.)
- Stepsize control: Require that the perturbation $\|\tilde{u} - u\|$ is small and reject the step if it is not.

We discuss the first idea here; the second is deferred to Section 4.2. Ideally every element of the set of potential weight vectors would correspond to a method of the same order as the original method. Due to linearity of the order conditions, any linear combination of such weights would also yield a method of the same order. On the other hand, it is natural to include a weight vector corresponding to the forward Euler method (for explicit methods) or backward Euler method (for implicit methods), since these two methods guarantee positivity (unconditionally for backward Euler and conditionally for forward Euler). We can formulate an LP using the approach of convex adaptation as follows. Let B denote the matrix with

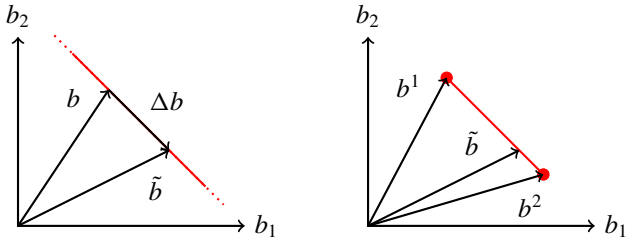

Figure 2. Graphical representation of the two different approaches to adapt the weights for a two-stage method. Left: free adaptation. Right: convex adaptation.

columns $b^1, b^2, \ldots, b^K$ and let $g \in \mathbb{R}^K$. The LP is then as follows:

LP

(Convex adaptation) Given F, B, and b, find g that minimizes $\|\tilde{b} - b\|_1$ subject to

$$\tilde{b} = Bg, \tag{21a}$$

$$0 \le g_k \le 1, \tag{21b}$$

$$\sum_{k=1}^{K} g_k = 1, \tag{21c}$$

$$u^{n+1} = u^n + \Delta t F \tilde{b} \ge 0. \tag{21d}$$

We do not need to impose the order conditions here, since they will be satisfied by each of the methods and thus (by linearity) by the modified method. The order of the modified method will in general be equal to the lowest order among the component methods.

3.3. ***Reduction of number of positivity constraints.*** The number of positivity constraints implied by (11) is equal to m, the number of ODEs being solved. This number may be very large, for instance if the system is a semi-discretization of a PDE. This makes solution of the LP very costly. But in most cases, positivity is violated only for a very small subset $h \subseteq \{1, \ldots, m\}$ of the solution components. We can solve a much less expensive LP by replacing (11) with

$$u_i^n + \Delta t \sum_{j=0}^{s} F_{i,j} b_j \ge 0 \quad \forall i \in h \subseteq \{1, \ldots, m\}. \tag{22}$$

Of course, it must be checked that the solution of the resulting LP still satisfies the full set of constraints (11). In practice, we have found the following approach to be effective. First, set

$$h_0 = \{i \in \{1, \ldots, m\} \mid u_i^{n+1} < 0\}.$$

Solve the LP and let $\tilde{u}^{n+1}$ denote the new solution. If $\tilde{u}^{n+1}$ satisfies (11), accept this as the new solution; otherwise, repeatedly take

$$h_{a+1} = \{i \in \{1, \ldots, m\} | \tilde{u}_i^{n+1} < 0\} \cup h_a$$

until $\tilde{u}^{n+1}$ is found to satisfy (11). In the examples we have studied, this approach was found to always converge in at most 2 iterations.

When enforcing a maximum value, the number of constraints can be reduced using the same technique. When enforcing both maximum and minimum values

two separate sets of active constraints are used. In this case it is important to update these sets simultaneously.

3.4. ***Summary of the algorithm.*** Our proposed method to solve a positive initial value problem (1) is summarized in Algorithm 1.

1: Initialize $n \leftarrow 0$, $u^n \leftarrow u_0$, $t \leftarrow 0$
2: **while** $t < t_{\text{end}}$ **do**
3: Choose Δt (fixed or via an adaptive stepsize control)
4: Calculate $F = (f_1, \ldots, f_s)$ according to (6)
5: $u^{n+1} \leftarrow u^n + \Delta t F b$, according to (8)
6: **if** $u^{n+1} \geq 0$ **then**
7: **GOTO** line 24
8: **else**
9: $\tilde{p} \leftarrow p_{start}$
10: **while** $\tilde{p} \geq p_{min}$ **do**
11: Solve LP (20)
12: **if** LP is feasible **then**
13: $\delta \leftarrow \|\Delta t F(\tilde{b} - b)\|$
14: **if** $\delta < tol_\delta$ **then**
15: $u^{n+1} \leftarrow u^n + \Delta t F \tilde{b}$
16: **GOTO** line 24
17: **end if**
18: **end if**
19: $\tilde{p} \leftarrow \tilde{p} - 1$
20: **end while**
21: Reduce Δt
22: **GOTO** line 4
23: **end if**
24: Estimate *error* according to (24)
25: **if** $error \leq tol_{error}$ **then**
26: $t \leftarrow t + \Delta t$, $n \leftarrow n + 1$
27: **else**
28: Reduce Δt
29: **GOTO** line 4
30: **end if**
31: **end while**

Algorithm 1. Pseudocode for the algorithm using a free adaption of weights.

4. Properties of adaptive RKMs

In the previous sections an algorithm for choosing positivity preserving weights $\tilde{b}$ has been presented. In the next section properties of the adaptive RKMs are discussed.

4.1. ***Choice of baseline method.*** An important property of the baseline method is the existence of embedded methods and the degrees of freedom for the weights $\tilde{b}$. As noted in Section 3.1 the number of stages has to be higher than the order. It is natural to use explicit and diagonally implicit methods, both for their efficiency and because the order need not be reduced as much in order to satisfy the condition $\tilde{p} < s$. For a given method and reduced order $\tilde{p}$, the number of degrees of freedom for the choice of the new weights is given by $s - \operatorname{rank}(Q_{\tilde{p}})$. The resulting number of degrees of freedom is shown in Table 1 for some explicit methods and in Table 2 for several implicit methods. The backward Euler extrapolation methods use the harmonic sequence as described in [9, Section II.9] and [10, Section IV.9].

For explicit methods with the number of stages equal to the order of the method, the order must be reduced in order to allow any freedom in the weights. If the classical RK4 method is used the order has to be reduced more because the RK4

Method	s	Order $\tilde{p}$					
		1	2	3	4	5	6
Classical RK4 [20]	4	3	2	0	0	—	—
SSPRK(10,4) [14]	10	9	8	6	4	—	—
Cash–Karp RK5(4)6 [4]	6	5	4	2	1	0	—
Dormand–Prince RK5(4)7 [24]	7	6	5	3	1	0	—

Table 1. Degrees of freedom for the choice of the weights for some explicit methods.

Method	s	Order $\tilde{p}$					
		1	2	3	4	5	6
Backward Euler	1	0	—	—	—	—	—
Lobatto IIIC4 [5]	4	3	2	1	0	0	0
Radau IIA3 [7]	3	2	1	0	0	0	—
SDIRK(5,4) [10, (6.18)]	5	4	3	1	0	—	—
TR-BDF2 [2]	3	2	1	—	—	—	—
Extrapolation BE 2 [9, Sec. II.9]	3	2	1	—	—	—	—
Extrapolation BE 3 [9, Sec. II.9]	6	5	4	2	—	—	—
Extrapolation BE 4 [9, Sec. II.9]	10	9	8	6	3	—	—

Table 2. Degrees of freedom for the choice of the weights for some implicit methods.

method does not have embedded methods of order 3. In contrast to this, some methods with $s > p$ admit changes to the weights without reducing the order. An example of this is SSPRK(10,4), that has 4 degrees of freedom for $\tilde{p} = p$. For Cash–Karp RK5 and Dormand–Prince RK5, even though the number of stages is higher than the order, the order must be reduced in order to allow any modification of the weights.

Regarding implicit methods, we can see that the fully implicit methods Lobatto IIIC4 and Radau IIA3 require a drastic reduction of the order, as expected. The diagonally implicit SDIRK(5,4) method only requires an order reduction of one to get one degree of freedom for the weights. The TR-BDF2 method even allows adaptations without reducing the order. The backward Euler extrapolation methods also exhibit degrees of freedom without a reduction of the order.

It is also desirable that the baseline method have a large stability region.

Note that for many diagonally implicit methods, the first stage is a scaled backward Euler step. For such methods, by allowing the order to be reduced to one we can guarantee the existence of a solution to the LP, since the backward Euler method is unconditionally positive. For explicit methods, reducing the order to one is guaranteed to yield a solution of the LP only if the step size is small enough.

4.2. ***Error detection and approximation.*** Stability analysis for the proposed approach is very challenging, since in principle a different method may be used at every step. At the same time, as long as the exact solution is positive, we expect that as the step size goes to zero, eventually no modification of the weights will be required and the convergence of the unmodified method will be observed. This holds true in the examples shown in Section 5. We are thus more concerned with the behavior of the modified method outside the asymptotic convergence regime.

To approximate the error of a new step we propose the following approximation of the local error:

$$err = \|u(t^{n+1}) - \tilde{u}^{n+1}\| = \|u(t^{n+1}) - (u^{n+1} + \Delta t F(\tilde{b} - b))\| \tag{23}$$

$$\leq \underbrace{\|u(t^{n+1}) - u^{n+1}\|}_{\approx err_T} + \underbrace{\|\Delta t F(\tilde{b} - b)\|}_{= \delta}. \tag{24}$$

The total error is split up in the truncation error and the perturbation δ using the triangle inequality. The truncation error can be estimated using the standard error estimators $err_T = \|u_b^n - u_{\hat{b}}^n\|$. After adapting the weights, the perturbation is calculated. If the perturbation is larger than the tolerance, the weights are rejected. The two values are added to get an approximation of the total error $err = err_T + \delta$. This type of error estimation is easy to implement because it can be easily incorporated in an existing step size control and takes advantage of the standard error approximation.

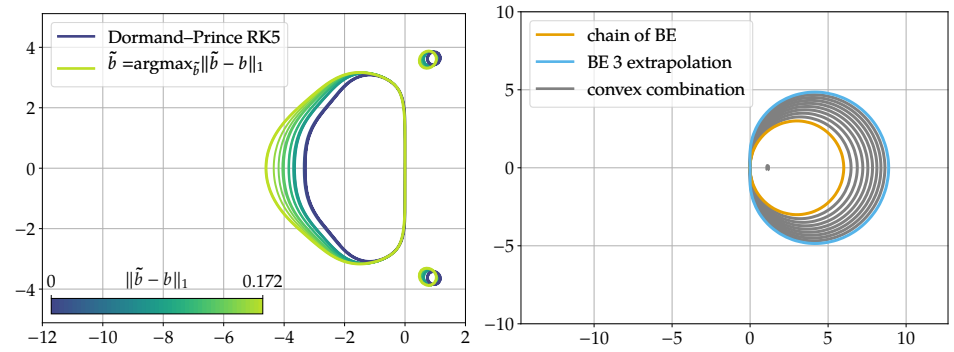

Figure 3. Change of stability region for the free adaptation and the convex adaptation of the weights. Left: Dormand–Prince RK5 with free adaptation of the weights $\tilde{b}$ as in the example in Figure 5. Right: BE 3 extrapolation method and embedded chain of three BE steps with convex combinations of both weights. Regions with $|R(z)| \leq 1$ are hatched for the original methods.

4.3. ***Stability region.*** Adapting the weights b changes the RK method. Hence, the stability function is altered and the region of absolute stability varies.

As an example, the stability regions of adapted RKMs are visualized in the next two figures. In Figure 3, left, the Dormand–Prince RK5 method is freely adapted. The weights are taken from the example in Section 5.1. In Figure 3, right, the stability regions of the BE 3 extrapolation method and the embedded chain of three BE steps with time step $\Delta t/3$ are plotted. Additionally the stability regions of convex combinations of these two methods are shown.

Let the stability function be denoted by $R(z) : \mathbb{C} \to \mathbb{C}$. Since we intend to vary the weights, we view $R(z)$ as a function parameterized by the weight vector b:

$$R_b(z) = 1 + zb^T(I - zA)^{-1}e, \tag{25}$$

where $e = (1, \ldots, 1)^T \in \mathbb{R}^s$. The stability function is an affine function of the weights.

4.3.1. *Stability of convex adaptation.* If the new weights are chosen by convex adaptation of given weights, it is easy to prove some properties of the stability region.

Theorem 4.1. *The stability region of a Runge–Kutta method $(A, \tilde{b})$ where $\tilde{b} = \sum_i g_i b^i$ is a convex combination of $b^1, \ldots, b^m \in \mathbb{R}^s$ (i.e. $g_i \in [0,1]$, $\sum_i g_i = 1$), contains the intersection of the stability regions of the methods $(A, b^1), \ldots, (A, b^m)$.*

Proof. Since the stability function is an affine-linear function of the weights, $R_{\sum_i g_i b^i}(z) = \sum_i g_i R_{b^i}(z)$. Hence, if z is in the stability region of all methods

$(A, b^1), \ldots, (A, b^m)$,

$$|R_{\sum_i g_i b^i}(z)| \leq \sum_i g_i |R_{b^i}(z)| \leq 1. \qquad \square$$

This result is particularly important for implicit methods. If all the embedded methods used to construct the new weights are A-stable, the resulting method is also A-stable.

4.3.2. *Stability of free adaptation.* If the weights are adapted freely, in general we have no result like Theorem 4.1. Still, if the change in the weights is small then the resulting stability function is by some measure similar to the stability function of the baseline method.

Lemma 4.2. *The stability function $R_{\tilde{b}}$ of an adapted RK method satisfies*

$$|R_{\tilde{b}}(z)| \leq |R_b(z)| + \|\tilde{b} - b\|_1 \|z(\mathrm{I} - zA)^{-1} e\|_\infty. \tag{26}$$

Proof. Compute

$$\begin{aligned} |R_{\tilde{b}}(z)| &\leq |R_b(z)| + |R_{\tilde{b}}(z) - R_b(z)| = |R_b(z)| + |z(\tilde{b} - b)^T (\mathrm{I} - zA)^{-1} e| \\ &\leq |R_b(z)| + \|\tilde{b} - b\|_1 \|z(\mathrm{I} - zA)^{-1} e\|_\infty. \end{aligned} \qquad \square$$

This result suggests that the objective $\min \|\tilde{b} - b\|_1$ is an appropriate choice to control the change of the region of absolute stability, in particular for explicit methods for which $\|z(\mathrm{I} - zA)^{-1} e\|_\infty$ can be bounded by a polynomial in $|z|$.

5. Results of numerical experiments

The implementation of the algorithms described above and code to reproduce the numerical examples reported here can be found in [23]. The methods are implemented in Python using NumPy/SciPy [32], NodePy [17], and Matplotlib [13] for the visualizations. We have used MOSEK [22] via CVXPY to solve the LPs [6; 1].

The adaptive RKM can be used with ODEs that satisfy (4). For problems where the exact solution is positive for certain u_0 but do not satisfy (4) tests did not show promising results. Additionally, it is not certain whether the computed solutions would be reasonable.

5.1. ***Non-stiff problem with fixed stepsize.*** First, adaptive RKMs based on explicit methods are tested on non-stiff problems with a fixed step size. When used with explicit methods the cost of solving the LP is significant because the computation of the stage derivatives only requires s evaluations of the right-hand side (RHS). For most of the linear test problems tried, the explicit methods yield to positive results. When increasing the step size, issues with stability occur before getting negative

values. An example for this is the ODE in Example I. Some nonlinear RHS may require very small time steps to preserve positivity. For these, adapting the weights could be a possible way to solve them. An example is the reaction equation solved in Example II. Since the stage values are not guaranteed to be positive, the RHS must be defined also for negative values. If there is not a natural definition for negative values, one can instead extend the function in a smooth way or simply replace negative stage values by zero, e.g. replace `sqrt(u)` by `sqrt(max(u, 0))`.

A test problem similar to [30] is the PDE

$$\begin{aligned} \frac{\partial u(t,x)}{\partial t} &= -a\frac{\partial u(t,x)}{\partial x} - Ku(t,x), && x \in (0,1), t \in (0,1), \\ u(t,0) &= 1, && t \in (0,1], \\ u(0,x) &= 0, && x \in [0,1], \end{aligned} \tag{27}$$

which consists of an advection part and an exponential decay. The numerical approximation uses the method of lines and a first order upwind finite difference semidiscretization with $N = 100$ points. This leads to the positivity preserving ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i = \frac{a}{\Delta x}(u_{i-1} - u_i) - Ku_i. \tag{28}$$

The parameters are set to $a = 1$ and $K = 1$. We use the Dormand–Prince RK5 method and adapt the weights using the free adaptation. In Figure 4 the results for $\Delta t = 0.015$ are plotted for different values of time t. We can see that the solution approaches an exponential function with $t \to \infty$. Figure 5 plots the weights used. For $t \leq 0.375$ the weights are altered. For $t > 0.375$ the original weights lead to a positive solution.

Next, different time steps are used. For $\Delta t \leq 0.0082$ the unaltered method leads to positive solutions. For a larger Δt the original method leads to negative values and the weights are altered. For $\Delta t > 0.016$ the baseline method is no longer stable.

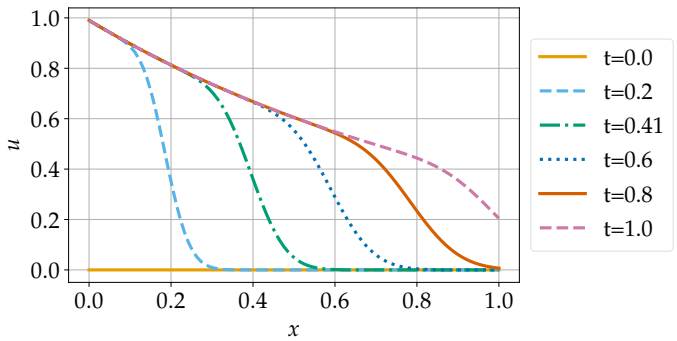

Figure 4. Advection decay problem (27): numerical solution at different times.

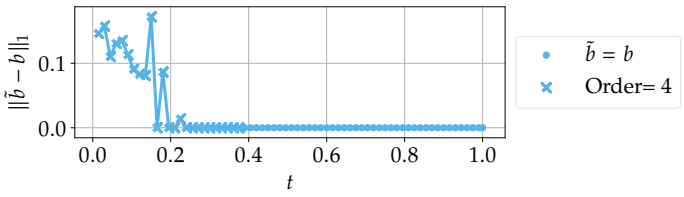

Figure 5. Numerical results for the advection decay problem (27): adaptation of the weights.

For the ODE, the reference solution can be computed using the matrix exponential. In Figure 6 the convergence for $t = 0.5$ is plotted for the altered and unaltered method.

The unaltered method has the order $p = 5$. The values marked with a cross denote the numerical experiments that required an adaption of the weights. Even though the order is reduced, most errors are still close to the error of the unaltered method.

It is natural to ask whether the adaptation of the weights through the algorithm proposed here is more efficient than simply using a smaller step size. For this problem that can be discretized explicitly with a right-hand-side that is relatively cheap to evaluate, using a smaller step size is generally more efficient, at least with the current un-optimized implementation of the LP solution. For the problem considered in the next section, where an implicit integrator is used, it is more efficient to maintain positivity with our proposed approach instead of reducing the step size. Adaptation of the weights could be made even more efficient with an optimized implementation of the LP setup and solve; this is the subject of future work.

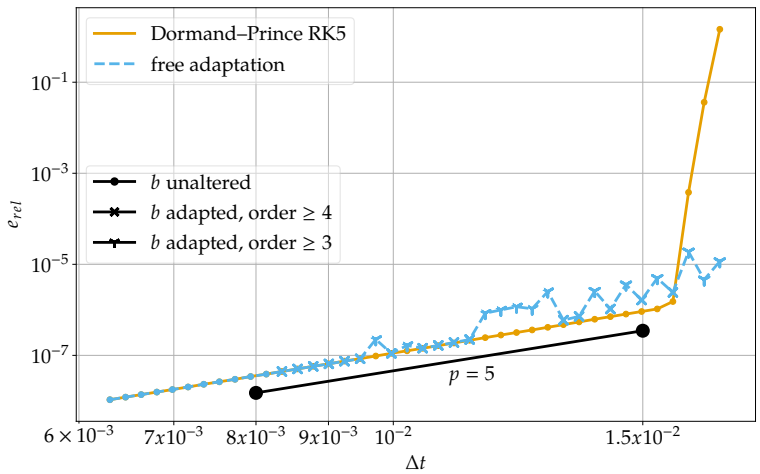

Figure 6. Convergence results of the adapted Dormand–Prince RK5 for the advection decay problem (27).

5.2. ***Stiff problem with fixed step size.*** Next, adaptive RKMs based on implicit methods are tested on stiff problems. Implicit methods are an advantageous choice for a couple of reasons. Firstly, the cost of solving the LP is relatively small compared to the cost of solving the stage equations. Secondly, the time step is not limited by the stability of the method. Therefore, it is possible to use larger time steps that are more likely to lead to negative values.

A very interesting class of methods are the implicit extrapolation methods. These allow changes of the weights without a reduction of the order, as discussed in Section 4.1. Moreover, all stage values are computed using the BE method. Hence, all intermediate stages are positive. Furthermore, an embedded BE step is included. This ensures that a positive solution always exists, even if it is of first order.

We test the proposed adaptation algorithm on the diffusion equation

$$\frac{\partial}{\partial t}u = D\frac{\partial^2}{\partial x^2}u \tag{29}$$

with homogeneous Dirichlet boundary conditions on the domain $x = [-0.5, 0.5]$ with $N = 100$ points. The equation is semidiscretized using the 3-point-scheme

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i = \frac{d}{\Delta x^2}\left(u_{i-1} - 2u_i + u_{i+1}\right). \tag{30}$$

As initial condition $u^0 = (0, \ldots, 0, 1, 0, \ldots, 0)^T$ is used. The diffusion coefficient is $D = 1$.

The ODE is solved using the BE 3 extrapolation method. For large Δt the method computes negative values for u^1. These can be corrected by adapting the weights.

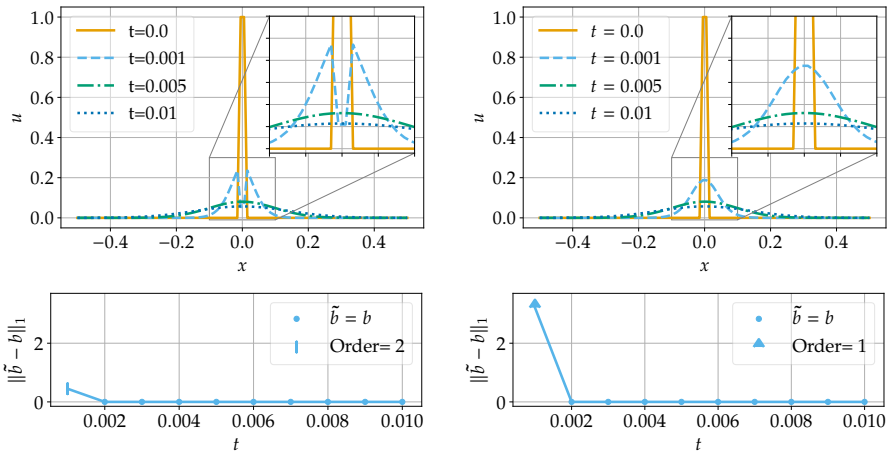

Figure 7. Numerical results for the diffusion problem (29) and the adapted BE 3 extrapolation method. Top: Solution using free adaptation (left) and convex adaptation (right). Bottom: change of weights for free adaptation (left) and convex adaptation (right).

The solutions are computed using the free adaptation and convex adaptation for $\Delta t = 1 \cdot 10^{-3}$. The results for the free adaptation are plotted in Figure 7, top left, and the corresponding change of the weights is shown in the bottom left pane of the figure. The original solution for the first step is negative. Therefore, the weights have to be changed. If we take a look at the solution after the first time step at $t = 0.001$ we can see that at $x = 0$ the solution is smaller than the solution at the surrounding points. This is not physical. The next time steps lead to physical solutions again. To prevent this glitch from happening we choose the weights based on a convex adaptation. A first order embedded method is added. The solution is shown in the top right pane and the weights are visualized on the bottom right. The weights for the first step are altered again. The weights obtained by the convex adaptation are different from the weights obtained by taking the free adaptation. The solution for $t = 0.001$ computed with the convex adaptation is physical. For both approaches, the remaining steps can be computed with the standard weights.

In Figure 8, the convergence is shown for the unaltered BE 3 extrapolation method (potentially resulting in negative values), the adaptive method with free adaptation, and the adapted method using convex adaptation. Additionally, results for the BE method are plotted. It is only of first order but preserves positivity for all Δt. For $\Delta t < 3 \cdot 10^{-5}$ the standard weights yield to a positive result. For larger Δt the weights have to be adapted to ensure positivity. The free adaptation results in similar convergence properties as the original method. This can be expected, because the adapted method is still of third order. The convex adaptation yields larger errors than the free adaptation but leads to physical solutions for all time-steps. This is no surprise because the adapted RKM used for the first step is only of first order. But the adaptive method still outperforms the BE, even when accounting for the higher cost per step.

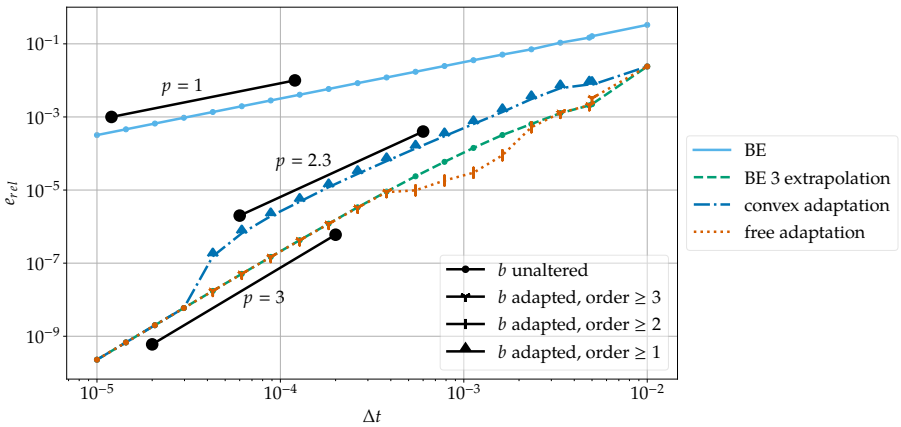

Figure 8. Convergence result of baseline and adapted BE 3 extrapolation methods for the diffusion problem (29).

5.3. ***Stiff problem with adaptive step size.*** Next we test the adaptive RKM on a more complex problem. For this, we consider the following advection-diffusion-production-destruction system [19]:

$$\frac{\partial u_1}{\partial t} = -a\frac{\partial u_1}{\partial x} + d\frac{\partial^2 u_1}{\partial x^2} + 0.01u_2 + 0.01u_3 + 0.003u_4 - \frac{u_1u_2}{0.01+u_1}, \tag{31a}$$

$$\frac{\partial u_2}{\partial t} = -a\frac{\partial u_2}{\partial x} + d\frac{\partial^2 u_2}{\partial x^2} + \frac{u_1u_2}{0.01+u_1} - 0.01u_2 - 0.5(1-\exp(-1.21u_2^2))u_3 - 0.05u_2, \tag{31b}$$

$$\frac{\partial u_3}{\partial t} = -a\frac{\partial u_3}{\partial x} + d\frac{\partial^2 u_3}{\partial x^2} + 0.5(1-\exp(-1.21u_2^2))u_3 - 0.01u_3 - 0.02u_3, \tag{31c}$$

$$\frac{\partial u_4}{\partial t} = -a\frac{\partial u_4}{\partial x} + d\frac{\partial^2 u_4}{\partial x^2} + 0.05u_2 + 0.02u_3 - 0.003u_4, \tag{31d}$$

with parameters $a = 1\cdot 10^{-2}$ and $d = 1\cdot 10^{-6}$. The PDE is simulated on the domain $x = [0,1]$ with $N = 100$ points and periodic boundary conditions. The advection part is semidiscretized using a first order upwind scheme and the diffusion part is semidiscretized using a central 3-point-scheme. This leads to a positivity preserving system of ODEs which conserves the total mass $\sum u$. The computation is done using the BE 3 extrapolation method with free adaptation. As step size control a PI-control from [10] is used. The error was estimated using (24). The tolerance was set to $tol = 0.01$. The final time is $t_{\text{end}} = 50$.

The simulation required 264 steps. Of these, 72 required an adaptation of the weights. All adapted weights are still of third order. The solutions for $t = 9$, $t = 18$, $t = 27$ and $t = 50$ are plotted in Figure 9. At $t = 18$ it can be seen that the reaction occurs in a small interfaces. Outside of this regions quantities are close to zero. Therefore, it is very likely that negative values occur in the numerical approximation. At $t = 27$ it can be seen that the two reaction interfaces merged. Later the reaction stops and the behavior is mainly controlled by the advection and diffusion part.

In Figure 10 different values are plotted. In the first subplot the step size is plotted. For $t < 25$ the time steps are small. After $t = 30$ the step size increases, because the solution only evolves slowly afterwards. In the second subplot the minimum of u_1, u_2, u_3, u_4 is plotted for all time steps that initially lead to negative values. This value is computed before and after adapting the weights. We can see that relatively large negative values occurred at some time steps. After the adaption of the weights, all values are close to 0. Therefore, the adaption of weights successfully preserved positivity. In the third subplot the approximated truncation error err_T and the perturbation *perturb* are plotted. We can see that *perturb* is of a similar magnitude as the truncation error. Therefore, the total error of the method is not increased drastically. In the next subplots objective function is plotted. We can see that the changes to the weights are only very small. The adapted RKM is

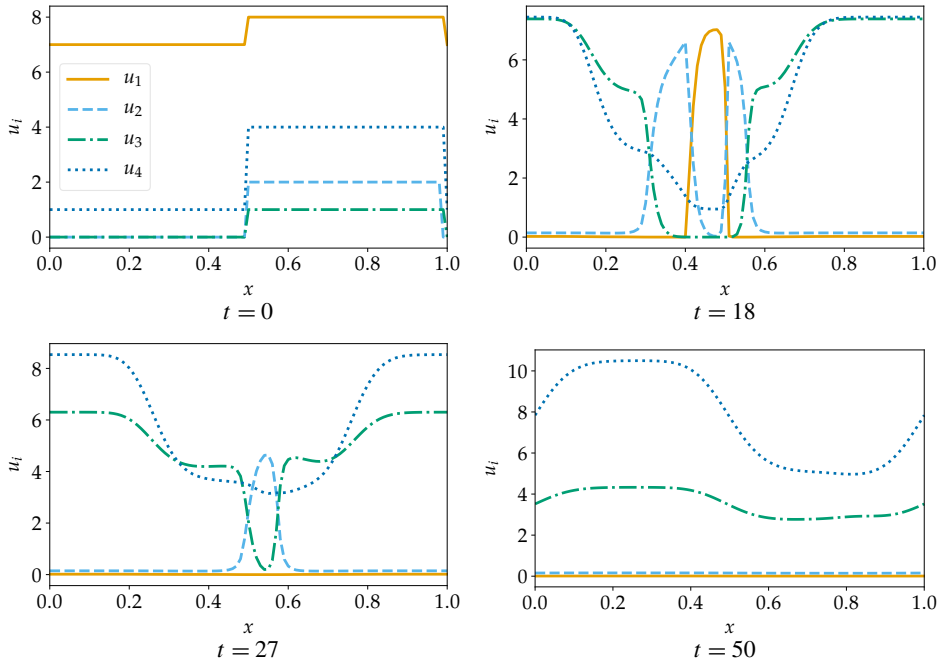

Figure 9. Numerical solution of the advection-diffusion-reaction problem (31) at different times.

still very close to the original RKM. In the last subplot the deviation of the sum over u_1, u_2, u_3, u_4 from the initial sum is plotted. The mass is conserved within roundoff error.

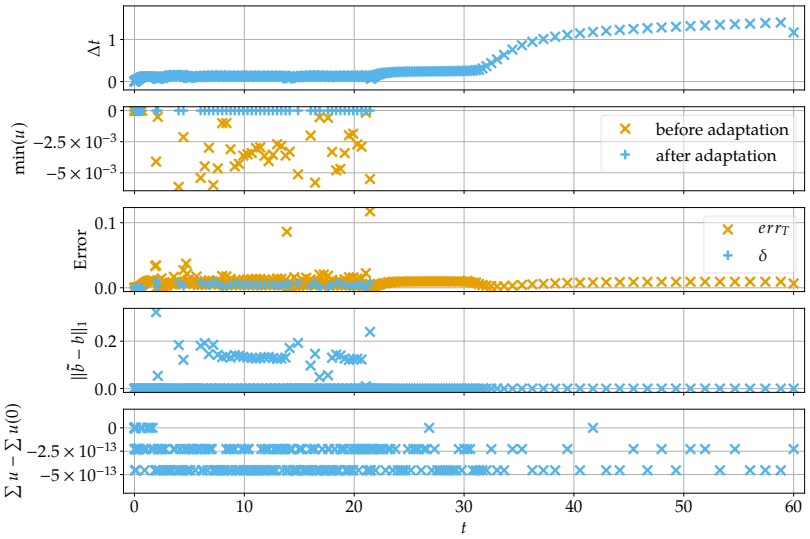

Figure 10. Statistics of computation of the advection-diffusion-reaction problem (31).

5.4. ***Stiff problem with adaptive step size II.*** Here, we consider the stratospheric reaction problem of [28], which models the reactions of the substances in the concentration vector $u = [O^{1D}, O, O_3, O_2, NO, NO_2]$. This ODE has two linear invariants

$$m_O^T u = \text{const}, \qquad m_O = [1, 1, 3, 2, 1, 2]^T, \tag{32}$$

$$m_N^T u = \text{const}, \qquad m_N = [0, 0, 0, 0, 1, 1]^T, \tag{33}$$

which describe the conservation of the total mass of oxygen and nitrogen.

The reaction system

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t} O^{1D} = r_5 - r_6 - r_7, && \frac{\mathrm{d}}{\mathrm{d}t} O = 2r_1 - r_2 + r_3 - r_4 + r_6 - r_9 + r_{10} - r_{11},\\
&\frac{\mathrm{d}}{\mathrm{d}t} O_3 = r_2 - r_3 - r_4 - r_5 - r_7 - r_8, && \frac{\mathrm{d}}{\mathrm{d}t} O_2 = -r_1 - r_2 + r_3 + 2r_4 + r_5 + 2r_7 + r_8 + r_9,\\
&\frac{\mathrm{d}}{\mathrm{d}t} NO = -r_8 + r_9 + r_{10} - r_{11}, && \frac{\mathrm{d}}{\mathrm{d}t} NO_2 = r_8 - r_9 - r_{10} + r_{11},
\end{aligned} \tag{34}$$

with time t in seconds, is given by the reaction rates

$$\begin{aligned}
&r_1 = k_1\, O_2, && k_1 = 2.643 \cdot 10^{-10} \sigma^3, && r_2 = k_2\, O\, O_2, && k_2 = 8.018 \cdot 10^{-17},\\
&r_3 = k_3\, O_3, && k_3 = 6.120 \cdot 10^{-4} \sigma, && r_4 = k_4\, O_3\, O, && k_4 = 1.567 \cdot 10^{-15},\\
&r_5 = k_5\, O_3, && k_5 = 1.070 \cdot 10^{-3} \sigma^2, && r_6 = k_6\, M\, O^{1D}, && k_6 = 7.110 \cdot 10^{-11},\\
&r_7 = k_7\, O^{1D}\, O_3, && k_7 = 1.200 \cdot 10^{-10}, && r_8 = k_8\, O_3\, NO, && k_8 = 6.062 \cdot 10^{-15},\\
&r_9 = k_9\, NO_2\, O, && k_9 = 1.069 \cdot 10^{-11}, && r_{10} = k_{10}\, NO_2, && k_{10} = 1.289 \cdot 10^{-2} \sigma,\\
&r_{11} = k_{11}\, NO\, O, && k_{11} = 1.0 \cdot 10^{-8},
\end{aligned} \tag{35}$$

where $M = 8.120 \cdot 10^6$ and

$$T = (t/3600) \bmod 24, \quad T_r = 4.5, \quad T_s = 19.5 \tag{36}$$

$$\sigma(T) = \begin{cases} 0.5 + 0.5 \cos\left(\pi \left| \frac{(2T - T_r - T_s)}{(T_s - T_r)} \right| \frac{(2T - T_r - T_s)}{(T_s - T_r)}\right) & \text{if } T_r \le T \le T_s, \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

The initial conditions are

$$u(t_0) = [9.906 \cdot 10^1, 6.624 \cdot 10^8, 5.326 \cdot 10^{11}, 1.697 \cdot 10^{16}, 4.000 \cdot 10^6, 1.093 \cdot 10^9]^T. \tag{38}$$

The system was normalized internally such that $\forall n : u_n(t_0) = 1$ for the computation to achieve a suitable error estimation. The system is solved in the time from $t_0 = 12\,\mathrm{h}$ to $t_{\mathrm{end}} = 84\,\mathrm{h}$ using the BE 3 extrapolation method with free adaptation and step size control.

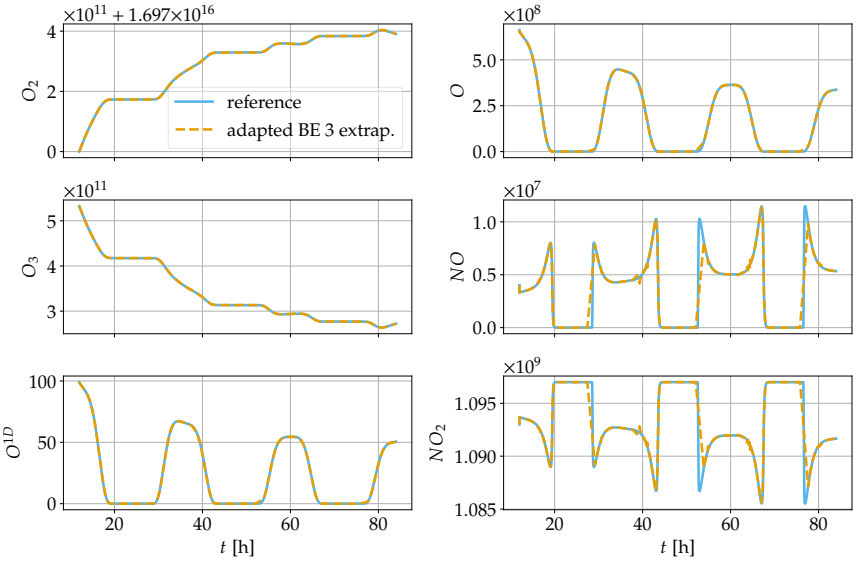

Figure 11. Numerical approximation of stratospheric reaction system (34).

The results are shown in Figure 11. The adapted solution is close to the reference solution obtained with the unadapted BE 3 extrapolation method and a higher accuracy. For this solution, 249 steps were computed; two of these were rejected due to a violation of the error bound. More details are shown in Figure 12. The rejected steps are drawn with thick crosses. The step size Δt undergoes multiple sudden changes due to the explicit dependence on time of the problem. The minimum values before and after the adaptation are also shown for all steps where the initial values were negative. This was only the case for some time intervals. 25 steps exhibited negative values. Almost all of them were very close to zero and the adaptation did only show a small improvement. The smallest value of the solution is $\min(u) = -1.59 \cdot 10^{-11}$. The used weights are also very close to the original weights, except of the steps that were rejected anyway due to a violation of the tolerance. The change of the two linear invariants are shwon in the last two subplots. Both are preserved within roundoff error.

6. Conclusion

It is possible to adapt the weights to enforce positivity for RKMs that are not positivity preserving. One main limitation is that the resulting order has to be lower than the number of stages. An error approximation for this method was given. The region of absolute stability is altered by changing the weights. This effect can be predicted or controlled. Used with explicit methods the positivity for some test problems could be recovered. Because the time step size is limited by the stability

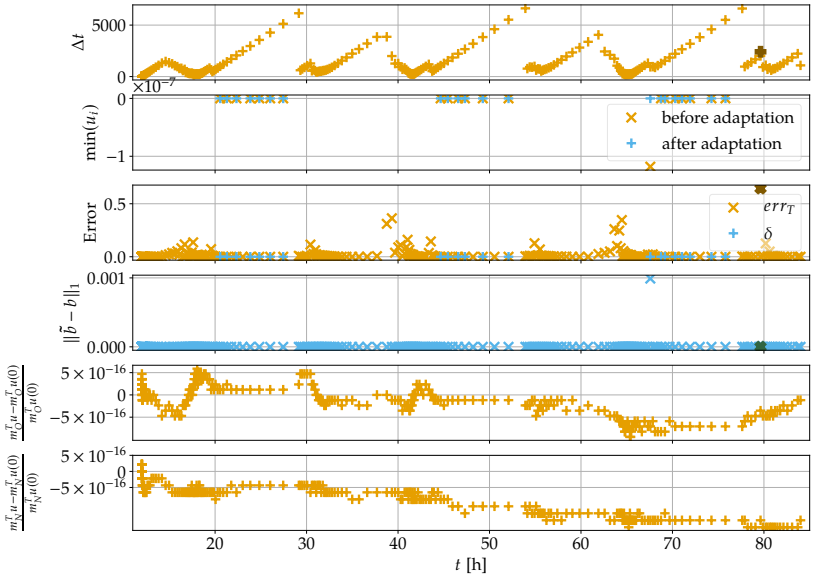

Figure 12. Statistics of computation of the stratospheric reaction system (34).

it is only useful for a small interval of time steps. The adaptive method is mainly interesting for diagonally implicit methods. The times step size is not limited by stability. Also, the cost of solving the LP is not a crucial factor. If the negative values occurring are not too large, which can be expected for most computations, adapting the weights is a potential way to ensure positivity.

Acknowledgments

Research reported in this publication was supported by the King Abdullah University of Science and Technology (KAUST). Stephan Nüßlein conducted this work while he was a visiting student at KAUST.

References

[1] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, *A rewriting system for convex optimization problems*, Journal of Control and Decision **5** (2018), no. 1, 42–60.

[2] R. E. Bank, W. M. Coughran, W. Fichtner, E. H. Grosse, D. J. Rose, and R. K. Smith, *Transient simulation of silicon devices and circuits*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems **4** (1985), no. 4, 436–451.

[3] C. Bolley and M. Crouzeix, *Conservation de la positivité lors de la discrétisation des problèmes d'évolution paraboliques*, RAIRO Anal. Num. **12** (1978), no. 3, 237–245.

[4] J. R. Cash and A. H. Karp, *A variable order Runge–Kutta method for initial value problems with rapidly varying right-hand sides*, ACM Trans. Math. Software **16** (1990), no. 3, 201–222.

[5] F. H. Chipman, *A-stable Runge–Kutta processes*, BIT **11** (1971), no. 4, 384–388.

[6] S. Diamond and S. Boyd, *CVXPY: A Python-embedded modeling language for convex optimization*, J. Machine Learning Res. **17** (2016), no. 83, 1–5.

[7] B. L. Ehle, *On Padé approximations to the exponential function and A-stable methods for the numerical solution of initial value problems*, Ph.D. thesis, University of Waterloo Waterloo, Ontario, 1969.

[8] S. Gottlieb, D. I. Ketcheson, and C.-W. Shu, *Strong stability preserving Runge–Kutta and multistep time discretizations*, World Scientific, Singapore, 2011.

[9] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations I: Nonstiff problems*, Springer Series in Computational Mathematics, no. 8, Springer, 2008.

[10] E. Hairer and G. Wanner, *Solving ordinary differential equations II: Stiff and differential-algebraic problems*, Springer Series in Computational Mathematics, no. 14, Springer, Berlin Heidelberg, 2010.

[11] Z. Horváth, *Positivity of Runge–Kutta and diagonally split Runge–Kutta methods*, Appl. Numer. Math. **28** (1998), no. 2-4, 309–326.

[12] W. Hundsdorfer and J. G. Verwer, *Numerical solution of time-dependent advection-diffusion-reaction equations*, Springer, 2003.

[13] J. D. Hunter, *Matplotlib: a 2D graphics environment*, Comput. Sci. Eng. **9** (2007), no. 3, 90–95.

[14] D. I. Ketcheson, *Highly efficient strong stability-preserving Runge–Kutta methods with low-storage implementations*, SIAM J. Sci. Comput. **30** (2008), no. 4, 2113–2136.

[15] ———, *Relaxation Runge–Kutta methods: conservation and stability for inner-product norms*, SIAM J. Numer. Anal. **57** (2019), no. 6, 2850–2870.

[16] D. I. Ketcheson, C. B. MacDonald, and S. J. Ruuth, *Spatially partitioned embedded Runge–Kutta methods*, SIAM J. Numer. Anal. **51** (2013), no. 5, 2887–2910.

[17] D. I. Ketcheson, H. Ranocha, M. Parsani, U. bin Waheed, and Y. Hadjimichael, *NodePy: a package for the analysis of numerical ODE solvers*, Journal of Open Source Software **5** (2020), no. 55, 2515.

[18] S. Kopecz and A. Meister, *Unconditionally positive and conservative third order modified Patankar–Runge–Kutta discretizations of production–destruction systems*, BIT **58** (2018), no. 3, 691–728.

[19] ———, *A comparison of numerical methods for conservative and positive advection–diffusion–production–destruction systems*, PAMM **19** (2019), no. 1.

[20] W. Kutta, *Beitrag zur näherungsweisen Integration totaler Differentialgleichungen*, Z. Math. Phys. **46** (1901), 435–453.

[21] C. B. Macdonald, S. Gottlieb, and S. J. Ruuth, *A numerical study of diagonally split Runge–Kutta methods for PDEs with discontinuities*, J. Sci. Comput. **35** (2008), 89–112.

[22] MOSEK ApS, *Introducing the MOSEK Optimization Suite 9.2.3*, Tech. report, 2020.

[23] S. Nüßlein, H. Ranocha, and D. I. Ketcheson, *Positive_RK_Reproducibility: positivity-preserving adaptive Runge–Kutta methods*, Tech. report, 05 2020.

[24] P. J. Prince and J. R. Dormand, *High order embedded Runge–Kutta formulae*, J. Comput. Appl. Math. **7** (1981), no. 1, 67–75.

[25] H. Ranocha and D. I. Ketcheson, *Relaxation Runge–Kutta methods for Hamiltonian problems*, J. Sci. Comput. **84** (2020), no. 1.

[26] H. Ranocha, L. Lóczi, and D. I. Ketcheson, *General relaxation methods for initial-value problems with application to multistep schemes*, Numer. Math. **146** (2020), 875–906.

[27] H. Ranocha, M. Sayyari, L. Dalcin, M. Parsani, and D. I. Ketcheson, *Relaxation Runge–Kutta methods: fully-discrete explicit entropy-stable schemes for the compressible Euler and Navier–Stokes equations*, SIAM J. Sci. Comput. **42** (2020), no. 2, A612–A638.

[28] A. Sandu, *Positive numerical integration methods for chemical kinetic systems*, J. Comput. Phys. **170** (2001), no. 2, 589–602.

[29] L. F. Shampine, *Conservation laws and the numerical solution of ODEs*, Comp. Math. Appl. **12** (1986), no. 5-6, 1287–1296.

[30] L. F. Shampine, S. Thompson, J. Kierzenka, and G. Byrne, *Non-negative solutions of ODEs*, Appl. Math. Comput. **170** (2005), no. 1, 556–569.

[31] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys. **77** (1988), no. 2, 439–471.

[32] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, I. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, *SciPy 1.0: fundamental algorithms for scientific computing in Python*, Nature Methods **17** (2020), 261–272.

Received May 13, 2020. Revised March 4, 2021.

STEPHAN NÜSSLEIN: stephan.nuesslein@tum.de
Department of Electrical and Computer Engineering, Technical University of Munich, 80333 München, Germany

HENDRIK RANOCHA: mail@ranocha.de
Computer Electrical and Mathematical Science and Engineering (CEMSE) Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia
Current address: *Applied Mathematics, University of Münster, 48149 Münster, Germany*

DAVID I. KETCHESON: david.ketcheson@kaust.edu.sa
Division of Computer, Electrical, and Mathematical Sciences and Engineering, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia