Preprint, 20 June 2023

Functional-preserving predictor-corrector multiderivative schemes

Hendrik Ranocha[1,*], **Jochen Schütz**[2,**], and **Eleni Theodosiou**[2,***]

[1] Applied Mathematics, University of Hamburg, Bundesstr. 55, 20146 Hamburg, Germany
[2] Faculty of Sciences & Data Science Institute, Hasselt University, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium

In this work, we develop a class of high-order multiderivative time integration methods that is able to preserve certain functionals discretely. Important ingredients are the recently developed Hermite-Birkhoff-Predictor-Corrector methods and the technique of relaxation for numerical methods of ODEs. We explain the algorithm in detail and show numerical results for two- and three-derivative methods, comparing relaxed and unrelaxed methods. The numerical results demonstrate that, at the slight cost of the relaxation, an improved scheme is obtained.

1 Introduction

The efficient and accurate numerical solution of time-dependent differential equations is ubiquitous in the computational sciences; examples of practical interest stem from meteorology, aerospace engineering, porous media flow and many more. There are several challenges associated to high-order temporal integration, such as efficiency and stability, which are obviously intertwined. In classical numerical schemes, high-order has been reached through an increase in either stages or steps, or both, see, e.g., [1]. By now, Runge-Kutta schemes and linear multistep schemes are a de-facto standard in, e.g., the computational fluid dynamics community (CFD), see [2] for an overview on the use of implicit methods in CFD. Although also a rather classical approach, see [3], the multiderivative paradigm has only been rediscovered rather recently; for some examples see [4–9] and the references therein.

To illustrate the approach, let us assume that the underlying differential equation is given by

$$w'(t) = \Phi(w(t)), \quad t \in [0, T_{end}], \qquad w(0) = w_0, \tag{1}$$

for some unknown function $w : [0, T_{end}] \to \mathbb{R}^{\dim}$ and a given smooth function $\Phi : \mathbb{R}^{\dim} \to \mathbb{R}^{\dim}$. Obviously, the second derivative of w can be computed from Φ and its Jacobian through

$$w''(t) = \Phi'(w(t))\Phi(w(t)) =: \dot{\Phi}(w(t)). \tag{2}$$

Obviously, also the third temporal derivative $\ddot{\Phi}(w)$ and higher derivatives of w can be computed. Multiderivative time integrators explicitly take the quantities $\Phi, \dot{\Phi}, \ldots$ into account, which results for, e.g., a given number of stages, in a higher order than in a classical approach. In this work, we consider a peculiar predictor-corrector form of the implicit multiderivative method, inspired by spectrally deferred correction methods [10]. This HBPC (Hermite-Birkhoff-Predictor-Corrector) method was initially developed and motivated as an IMEX scheme in [11] and then subsequently extended to higher orders in [12,13]. HBPC has shown favorable behavior for the solution of compressible flow equations [9,14,15].

While linear stability of HBPC has been tackled in [13], the behaviour of the method for large values of T_{end} has not been investigated yet. As for most schemes, it is to be expected that the numerical error grows tremendously with growing T_{end}. In this work, we consider the case of a functional $\eta : \mathbb{R}^{\dim} \to \mathbb{R}$ that is preserved under the solution, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}t}\eta(w(t)) \equiv 0. \tag{3}$$

For Hamiltonian problems, η could simply be the Hamiltonian function; for smooth flow problems, it could be entropy and so on. First, we show how the classical HBPC method behaves in terms of η and in terms of the numerical error growth over time. Second, we extend the HBPC method with a relaxation procedure, originally developed in [16–18]; based on an older idea from [19]. This relaxation procedure, outlined below, enforces the preservation of η through an additional projection step. This projection step necessitates the solution of a scalar equation, typically through Newton's method or more efficient variants of the bisection method. While for explicit low-order schemes, this might constitute a significant overhead [20], it is negligible in our setting of implicit schemes. We show that with this very simple addendum to the algorithm, both error growth in time is reduced and the functional η is preserved for several testcases.

[*] mail@ranocha.de
[**] Corresponding author: jochen.schuetz@uhasselt.be
[***] eleni.theodosiou@uhasselt.be

2 Numerical tools

In this chapter, we describe the underlying time integration algorithm as well as its combination with relaxation.

2.1 Hermite-Birkhoff predictor-corrector time integration

The algorithm to be explained in the following is of the predictor-corrector type, iterating towards a background, fully implicit multiderivative Runge-Kutta scheme using $m \in \mathbb{N}$ temporal derivatives of w. For the ease of presentation, we first define this background scheme. Please note that this scheme is not actually used in our computations, only through the use of the corresponding quadrature rule. The scheme is of the classical multiderivative Runge-Kutta type, with s stages $w^{n,l}$, $1 \le l \le s$, and update w^{n+1}_{RK} defined by:

$$w^{n,l} := w^n + \sum_{d=1}^{m} \Delta t^d \sum_{j=1}^{s} B^{(d)}_{lj} \frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \Phi(w^{n,j}), \quad w^{n+1}_{RK} := w^n + \sum_{d=1}^{m} \Delta t^d \sum_{j=1}^{s} b^{(d)}_j \frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \Phi(w^{n,j}). \tag{4}$$

The matrices $B^{(d)}$, $1 \le d \le m$, form the Butcher tableaux. It is assumed that the $l-$th stage value of time is $t^n + c_l \Delta t$, for values $c_l \equiv \sum_{j=1}^{s} B^{(1)}_{lj}$. The coefficients for the Runge-Kutta update are denoted by $b^{(d)}_l$, $1 \le l \le s$. Please note that we have defined

$$\frac{\mathrm{d}^0}{\mathrm{d}t^0}\Phi(w^{n,j}) = \Phi(w^{n,j}), \quad \frac{\mathrm{d}^1}{\mathrm{d}t^1}\Phi(w^{n,j}) = \dot{\Phi}(w^{n,j}), \quad \frac{\mathrm{d}^2}{\mathrm{d}t^2}\Phi(w^{n,j}) = \ddot{\Phi}(w^{n,j}),$$

and so on. In this work, we rely on three Runge-Kutta schemes, two with two-derivatives, see [12, Eq. (2) and Eq. (3), respectively, for the Butcher tableaux], and a two-point three-derivative scheme with Butcher tableau

$$c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad B^{(1)} = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad B^{(2)} = \begin{pmatrix} 0 & 0 \\ \frac{1}{10} & -\frac{1}{10} \end{pmatrix}, \quad B^{(3)} = \begin{pmatrix} 0 & 0 \\ \frac{1}{120} & \frac{1}{120} \end{pmatrix}.$$

The final HBPC scheme to be presented here relies on a predictor ($k = 0$) and correction steps ($1 \le k \le k_{\max}$) for the quantities $w^{n,l}$. For short, the notation here is $w^{n,[k],l}$. The predictor is a straightforward implicit Taylor scheme making use of m temporal derivatives of w, the corrector is very similar in structure plus it additionally relies on the quadrature formula $\mathcal{I}_l$ defined through the Runge-Kutta scheme (4) by

$$\mathcal{I}_l := \sum_{d=1}^{m} \Delta t^d \sum_{j=1}^{s} B^{(d)}_{lj} \frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \Phi^{n,[k],j}.$$

Note the shorthand notation $\Phi^{n,[k],j} := \Phi(w^{n,[k],j})$. Finally, we obtain

Algorithm 1 (HBPC(m, q, $k_{\max}$) [11,12]) The algorithm consists of the following three steps:

1. **Predict.** Solve the following expression for $w^{n,[0],l}$ and $1 \le l \le s$:

$$w^{n,[0],l} := w^n + \sum_{d=1}^{m} \frac{(-1)^{d-1}(c_l \Delta t)^d}{d!} \frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \Phi^{n,[0],l}. \tag{5}$$

Subsequently:

2. **Correct.** Solve the following for $w^{n,[k+1],l}$, for each $1 \le l \le s$ and each $0 \le k < k_{\max}$:

$$w^{n,[k+1],l} := w^n + \sum_{d=1}^{m} \frac{(-1)^{d-1}\Delta t^d}{d!} \left(\frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \Phi^{n,[k+1],l} - \frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \Phi^{n,[k],l} \right) + \mathcal{I}_l \tag{6}$$

3. **Update.** Set

$$w^{n+1} := w^n + \sum_{d=1}^{m} \frac{(-1)^{d-1}\Delta t^d}{d!} \frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \left(\Phi^{n,[k_{\max}],l} - \Phi^{n,[k_{\max}-1],l} \right) + \sum_{d=1}^{m} \Delta t^d \sum_{j=1}^{s} b^{(d)}_j \frac{\mathrm{d}^{d-1}}{\mathrm{d}t^{d-1}} \Phi^{n,[k_{\max}-1],j}. \tag{7}$$

Remark 2.1 Please note that whenever the background Runge-Kutta scheme is globally stiffly accurate, i.e., there holds

$$b^{(d)}_j = B^{(d)}_{sj}, \quad 1 \le j \le s, 1 \le d \le m,$$

then the update step reduces to $w^{n+1} := w^{n,[k_{\max}],s}$. It is hence a slight generalization of [11,12], where only schemes with $c_s = 1$ are treated. In any case, the update step is explicit.

Remark 2.2 The order of convergence of this scheme is the minimum of $k_{\max} + m$ and the order q of the underlying Runge-Kutta scheme.

2.2 Relaxation procedure

The idea of a relaxation procedure as introduced in [16–18] is to consider a *scalar* parameter $\gamma \in \mathbb{R}$ and form a linear combination of w^n and w^{n+1} to obtain the quantity $w_\gamma^{n+1} = w^n + \gamma(w^{n+1} - w^n)$. The relaxation parameter γ gives the flexibility to enforce the preservation of the functional η, just as for the continuous case, see Eq. (3), through the equation (in γ!)

$$\eta\left(w^n + \gamma(w^{n+1} - w^n)\right) = \eta(w^n). \tag{8}$$

After having found a suitable γ – typically through a scalar Newton algorithm –, the relaxed update $w_\gamma^{n+1} = w^n + \gamma(w^{n+1} - w^n)$ is considered the new update step at time level $t^n + \gamma\Delta t$. Computation with Alg. 1 then continues from this adapted point in time and the corresponding linear combination of w^n and w^{n+1} as usual. Note that Δt is a constant throughout the computation (although adaptive timesteps are a possibility as well), however, the resulting time instances are not necessarily spaced equidistantly.

Obviously, $\gamma = 0$ is a (meaningless) solution to (8). It has been shown in [18] that under rather mild conditions on the timestep Δt and the functional η, there is also a unique solution γ which is close to one, in fact, it is $\mathcal{O}(\Delta t^{p+1})$ away from one. Here, p denotes the order of the method. With this solution, the relaxation approach keeps at least the order of the baseline methods. The relaxation approach is not restricted to invariants and has also been extended to general functionals η in [16–18], resulting for example in efficient, fully-discrete, and locally entropy-stable numerical methods for computational fluid dynamics [21] and nonlinear dispersive wave equations [22–24].

3 Numerical experiments

In this section, we present numerical findings of the HBPC method for a couple of test problems. As we are dealing with implicit time integration, both linear and nonlinear solvers are important ingredients. In all the numerical results to follow, we use a damped Newton procedure for the nonlinear equations, together with the standard backslash operator in Matlab to solve the linear systems. The Newton tolerance is always set to a very fine tolerance 10^{-14}, and a maximum of 1000 iterations is allowed. Obviously, we did not go for the most efficient solution here. For considerations regarding Newton efficiency, we refer the reader to [14]. In all the numerical results, 'error' is defined as the Euclidean error of the discrete solution at the final time T_{end}.

3.1 Nonlinear oscillator

As a first numerical example, we consider the nonlinear oscillator of [25, 26], given by

$$\Phi(w) := \frac{1}{\|w\|_2^2}\begin{pmatrix} -w_2 \\ w_1 \end{pmatrix}, \qquad w(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

The standard squared Euclidean norm is a conservative functional for this problem, i.e., $\eta(w) := \|w\|_2^2$ is a constant along the solution for all times $t \in \mathbb{R}^+$.

Error growth In a first step, we consider the error growth for the HBPC scheme in dependency of time with and without relaxation. As final time, the rather large $T_{end} = 100$ is chosen in combination with the large timesteps $\Delta t = 0.5$ and $\Delta t = 0.2$, respectively. As a time integrator, the HBPC(2,6,4) method is used, i.e., order six is to be expected. Please note that the behavior of this method is representative. Time against error can be seen in the top of Fig. 1 for the algorithm with and without relaxation. It can be clearly seen that the numerical error for the relaxed HBPC method behaves linearly in both cases. At least for smaller t, the error of the unrelaxed method behaves quadratically. For $\Delta t = 0.5$, it starts to oscillate at some point. This is also reflected in the fact that Newton's algorithm did not converge for the unrelaxed method and $\Delta t = 0.5$. In this sense, the relaxation improved the algorithm tremendously, even if one is not interested in an accurate representation of η. The bottom of Fig. 1 shows the evolution of $\eta - \eta_0$ ($\eta_0 = \eta(w(0))$) for the two values of Δt. As expected, the relaxed version preserves η, even if the error level, at least for $\Delta t = 0.5$, is also rather high for the relaxed method. All these results are very much in line with the results from literature as presented in [27–29].

Convergence properties In a subsequent step, we analyze the convergence properties of the method. Fig. 2 shows convergence results for two two-derivative and one three-derivative scheme, each with and without relaxation. From Rem. 2.2, the order of convergence is supposed to be the minimum of $k_{\max} + m$ and the ultimate order q of the background Runge-Kutta scheme. It can be seen for the unrelaxed case, that this order is indeed met. For the relaxed version, we see an odd-even decoupling of the order, i.e., for an odd value of $k_{\max}$, the order is one order better than expected. The maximum order of consistency, however, remains q. This has been proved in [20] for general B-series methods and the special situation of Euclidean Hamiltonian problems as in this case. In any case, the error constants seem to be tremendously lower for the relaxed version which is obviously also backed up through the findings from Fig. 1.

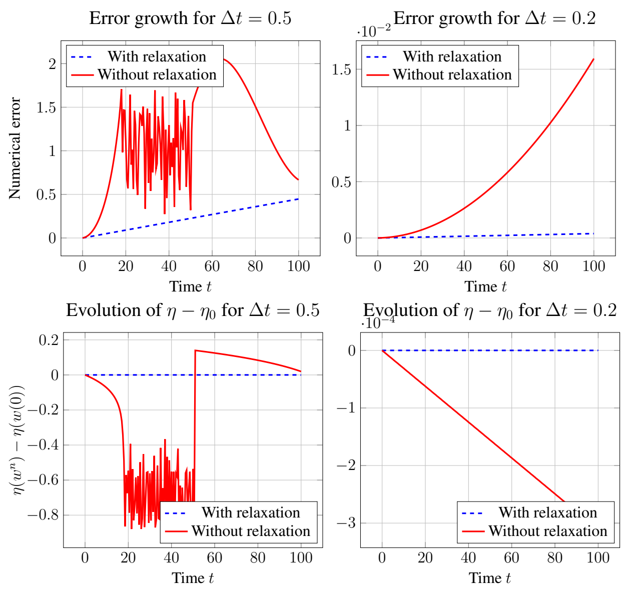

Fig. 1 The (serial) HBPC(2, 6, 4) predictor-corrector scheme of [13], see also [12], applied to the entropy-conserving nonlinear oscillator with $T_{end} = 100$. Left are numerical results for $\Delta t = 0.5$, right are results for $\Delta t = 0.2$. Top: numerical error as a function of time; bottom: the deviation in the functional η evaluated for the discrete solution. It is clearly visible that in all cases, the relaxed method behaves significantly better than its unrelaxed counterpart.

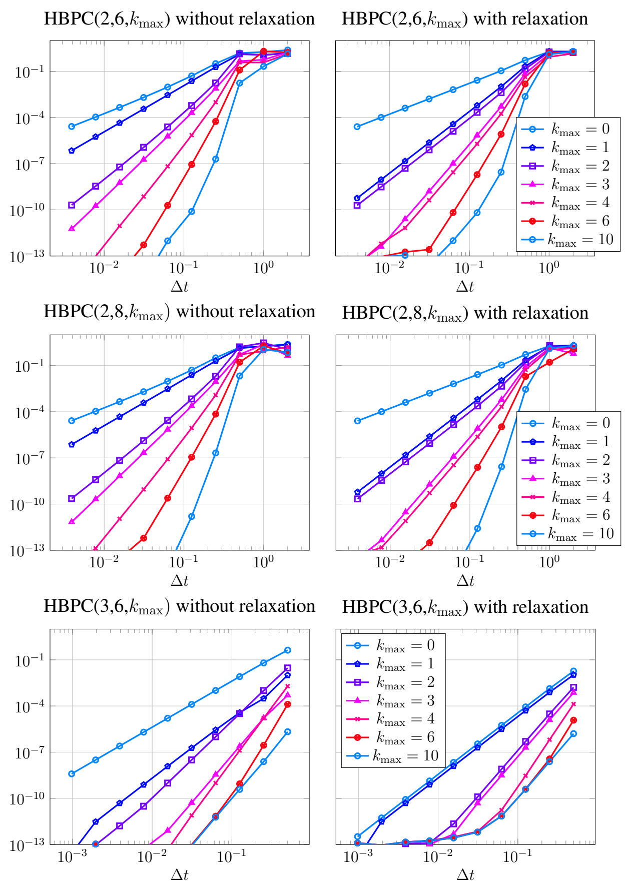

Fig. 2 The (serial) HBPC(2, 6, $k_{\max}$) (top), HBPC(2, 8, $k_{\max}$) (middle) and HBPC(3, 6, $k_{\max}$) predictor-corrector scheme of [13], see also [12], applied to the entropy-conserving nonlinear oscillator at $T_{end} = 10$ for various values of $k_{\max}$. Left: without a relaxation procedure. Right: with relaxation procedure. The order of convergence to be expected, see Rem. 2.2, is $\min\{6, k_{\max} + 2\}$ for the HBPC(2, 6, $k_{\max}$) scheme, $\min\{8, k_{\max} + 2\}$ for the HBPC(2, 8, $k_{\max}$) scheme and $\min\{6, k_{\max} + 3\}$ for the HBPC(3, 6, $k_{\max}$) scheme. This expected order is met for the unrelaxed version. The relaxed version shows an odd-even decoupling, so for odd $k_{\max}$, the order is increased by one.

3.2 Kepler's problem

To confirm some of the results from the previous section, and to show that the odd-even decoupling is not so much a feature of the method, but more of the underlying problem, we consider here Kepler's problem as in [20]. The problem is given by

$$\Phi(w) := \begin{pmatrix} w_3 \\ w_4 \\ -\frac{w_1}{(w_1^2+w_2^2)^{\frac{3}{2}}} \\ -\frac{w_2}{(w_1^2+w_2^2)^{\frac{3}{2}}} \end{pmatrix}, \qquad w(0) = \begin{pmatrix} 1/2 \\ 0 \\ 0 \\ \sqrt{1/3.} \end{pmatrix}.$$

The angular momentum

$$\eta(w) := w_1 w_4 - w_2 w_3$$

is a conserved quantity. For this example, $\Delta t = 0.5$ is way too coarse, and the relaxed version was not able to run due to the fact that at some point, the relaxation parameter γ from (8) could not be computed anymore. In this way, the relaxed algorithm also gives some extra information on the quality of the solution. Hence, we use smaller Δt here. As in the example before, we start with error growth as a function of t for two values of Δt, in this case $\Delta t = 0.2$ and $\Delta t = 0.05$, see Fig. 3. Again, we can see that the error growth for the relaxed method is slower than for the unrelaxed version. It is not a clear linear / quadratic relation as before due to periodic effects, but the overall growth seems in fact to be dominated by linear (relaxed) and quadratic (unrelaxed) terms. Fig. 3, bottom, shows the deviation of the functional η from the value $\eta_0 = \eta(w(0))$. As expected, for the relaxed version, it is preserved, while deviations for the unrelaxed algorithm are visible.

Fig. 4 shows convergence plots for the three different methods used here, two two-derivative and one three-derivative method. In contrast to the results before, there is no odd-even decoupling anymore, and the order of convergence of $\min\{k_{\max}+m, q\}$ is clearly met. This clearly indicates that this odd-even decoupling of the order for the relaxed version cannot be expected for all testcases, and is really a feature of the previous problem. Also the reduction of the error constant is only visible for $k_{\max} = 1$ (here it is the most prominent) and for $k_{\max} = 2$ (slightly). For the higher $k_{\max}$, this effect is not really significant.

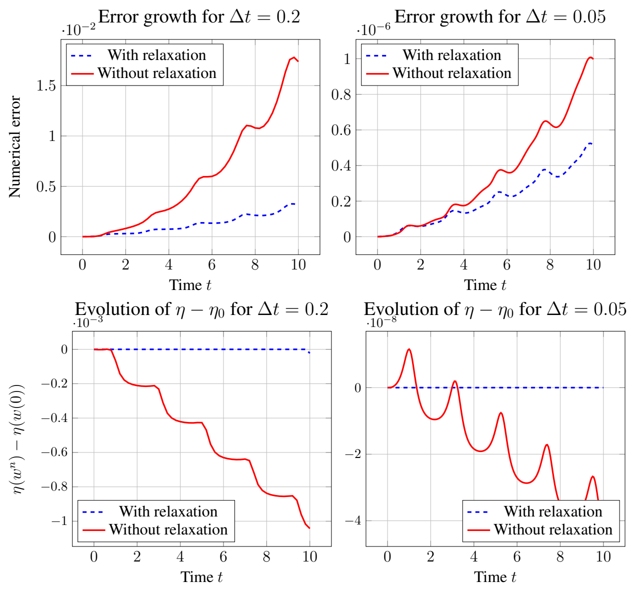

Fig. 3 The (serial) HBPC(2, 6, 4) predictor-corrector scheme of [13], see also [12], applied to Kepler's problem with $T_{end} = 10$. Left are numerical results for $\Delta t = 0.2$, right are results for $\Delta t = 0.05$. Top: numerical error as a function of time; bottom: the deviation in the functional η evaluated for the discrete solution. It is clearly visible that in all cases, the relaxed method behaves significantly better than its unrelaxed counterpart.

4 Conclusion and outlook

In this paper, we have combined recently developed relaxation techniques with also rather recently developed predictor-corrector time integration schemes. It has been shown that this can reduce error constants, and preserve functionals even if the general error level is high.

Obviously, many things are left to do. Currently, we are analyzing, both numerically and analytically, the combination of very general multiderivative methods and relaxation

- with respect to convergence properties for many different test problems, including suitably discretized PDEs,
- with respect to stability, in particular whether relaxation can change A- and L-stability properties of given methods,
- with respect to existence of γ and order considerations.

Also, dissipative problems, i.e., problems where, contrary to Eq. (3), the functional η is not preserved, but decreases over time, i.e., where there holds $\frac{\mathrm{d}}{\mathrm{d}t}\eta(y(t)) \leq 0$, are subject to investigation.

Acknowledgements HR was supported 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). ET was funded by the Fonds voor Wetenschappelijk Onderzoek (FWO, Belgium) - project no. G052419N.

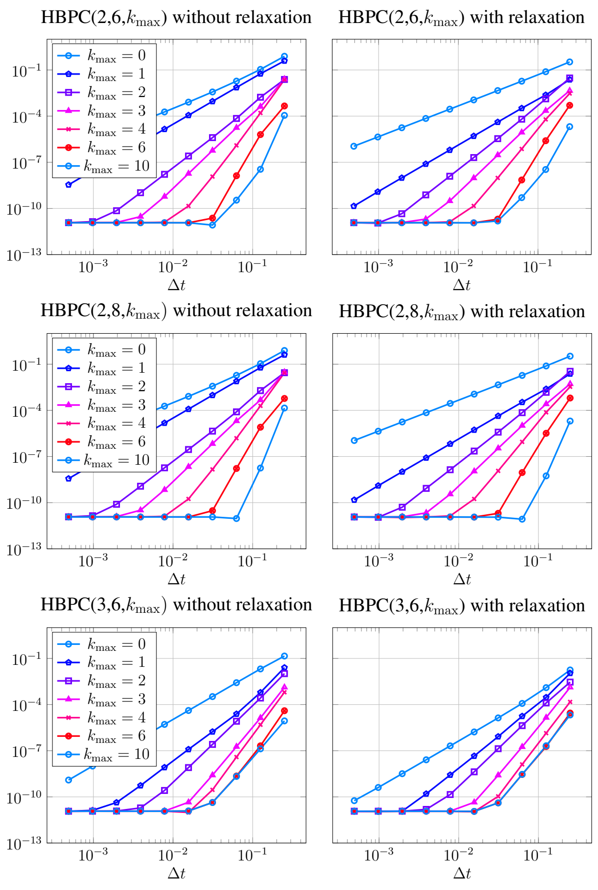

Fig. 4 The (serial) HBPC(2, 6, $k_{\max}$) (top), HBPC(2, 8, $k_{\max}$) (middle) and HBPC(3, 6, $k_{\max}$) predictor-corrector scheme of [13], see also [12], applied to Kepler's problem at $T_{end} = 5$ for various values of $k_{\max}$. Left: without a relaxation procedure. Right: with relaxation procedure. The order of convergence to be expected, see Rem. 2.2, is $\min\{6, k_{\max} + 2\}$ for the HBPC(2, 6, $k_{\max}$) scheme, $\min\{8, k_{\max} + 2\}$ for the HBPC(2, 8, $k_{\max}$) scheme and $\min\{6, k_{\max} + 3\}$ for the HBPC(3, 6, $k_{\max}$) scheme. This expected order is met for both the relaxed and the unrelaxed version. In contrast to the oscillator problem, see Fig. 2, only for $k_{\max} = 1$ and $k_{\max} = 2$, one can see significant differences in the error. Please note that the reference solution against which we compute the numerical error is also computed numerically (with another scheme) on a fine resolution. This explains why at about an error level of 10^{-11}, convergence stalls for all combinations.

References

[1] J. C. Butcher, Acta Numerica **15**, 157–256 (2006).
[2] R. Hartmann, F. Bassi, I. Bosnyakov, L. Botti, A. Colombo, A. Crivellini, M. Franciolini, T. Leicht, E. Martin, F. Massa et al., Implicit methods, in: TILDA: Towards Industrial LES/DNS in Aeronautics, (Springer, 2021), pp. 11–59.
[3] E. Hairer and G. Wanner, Computing (Arch. Elektron. Rechnen) **11**(3), 287–303 (1973).
[4] R. Chan and A. Tsai, Numerical Algorithms **53**, 171–194 (2010).
[5] D. C. Seal, Y. Güçlü, and A. Christlieb, Journal of Scientific Computing **60**, 101–140 (2014).
[6] J. Schütz, D. C. Seal, and A. Jaust, Journal of Scientific Computing **73**, 1145–1163 (2017).
[7] S. Gottlieb, Z. J. Grant, J. Hu, and R. Shu, SIAM Journal on Numerical Analysis **60**(1), 423–449 (2022).
[8] A. Moradi, A. Abdi, and G. Hojjati, Applied Numerical Mathematics **181**, 23–45 (2022).
[9] J. Zeifang and J. Schütz, Journal of Computational Physics **464**, 111353 (2022).
[10] B. W. Ong and R. J. Spiteri, Journal of Scientific Computing **83**(3), Paper No. 60, 29 (2020).
[11] J. Schütz and D. Seal, Applied Numerical Mathematics **160**, 84–101 (2021).
[12] J. Schütz, D. C. Seal, and J. Zeifang, Journal of Scientific Computing **90**(54) (2022).
[13] J. Zeifang, J. Schütz, and D. Seal, BIT Numerical Mathematics (2022).
[14] J. Zeifang, A. Thenery Manikantan, and J. Schütz, CMAT Preprint UP-22-01 (2022).
[15] A. Thenery Manikantan, J. Zeifang, and J. Schütz, CMAT Preprint UP-23-02 (2023).
[16] D. I. Ketcheson, SIAM Journal on Numerical Analysis **57**(6), 2850–2870 (2019).
[17] H. Ranocha, M. Sayyari, L. Dalcin, M. Parsani, and D. I. Ketcheson, SIAM Journal on Scientific Computing **42**(2), A612–A638 (2020).
[18] H. Ranocha, L. Lóczi, and D. I. Ketcheson, Numerische Mathematik **146**(10), 875–906 (2020).
[19] J. M. Sanz-Serna, Journal of Computational Physics **47**(2), 199–210 (1982).
[20] H. Ranocha and D. I. Ketcheson, Journal of Scientific Computing **84**(1) (2020).
[21] H. Ranocha, L. Dalcin, and M. Parsani, Computers and Mathematics with Applications **80**(5), 1343–1359 (2020).
[22] H. Ranocha, D. Mitsotakis, and D. I. Ketcheson, Communications in Computational Physics **29**(4), 979–1029 (2021).
[23] D. Mitsotakis, H. Ranocha, D. I. Ketcheson, and E. Süli, SIAM Journal on Scientific Computing **42**(04) (2021).
[24] H. Ranocha, M. Quezada de Luna, and D. I. Ketcheson, Partial Differential Equations and Applications **2**(6), 76 (2021).
[25] H. Ranocha, IMA Journal of Numerical Analysis **41**(1), 654–682 (2021).
[26] H. Ranocha and D. I. Ketcheson, SIAM Journal on Numerical Analysis **58**(6), 3382–3405 (2020).
[27] B. Cano and J. M. Sanz-Serna, SIAM Journal on Numerical Analysis **34**(4), 1391–1417 (1997).
[28] A. Durán and J. M. Sanz-Serna, Nonlinearity **11**(6), 1547 (1998).
[29] M. Calvo, M. Laburta, J. I. Montijano, and L. Rández, Mathematics and Computers in Simulation **81**(12), 2646–2661 (2011).