NEW MULTIRATE METHODS FOR STIFF DIFFERENTIAL EQUATIONS *

VU THAI LUAN, DANIEL R. REYNOLDS, RUJEKO CHINOMONA †

Abstract. The aim of this work is to construct efficient multiple time stepping (MTS) algorithms based on various important classes of explicit one-step exponential integrators. More precisely, starting from explicit exponential Runge–Kutta methods, we derive such algorithms which can be interpreted as particular implementations of these integrators. The proposed approach can be applied to the exponential Rosenbrock methods. In contrast to the well-known MTS techniques in the literature which are inspired by multistep methods, our new algorithms do not require a computation of starting values and easily perform an adaptive time step control. Another great advantage of these algorithms compared to the standard implementations of exponential integrators is that they can avoid matrix functions. The efficiency of the new algorithms and the resulting MTS schemes with order of accuracy up to four is illustrated on some numerical examples.

Key words. multiple time stepping, exponential integrators, exponential Runge–Kutta methods, exponential quadrature rules, exponential time differencing (ETD) methods

AMS subject classifications.

1. Introduction.

In this paper, we are concerned with the construction and implementation of efficient MTS algorithms based on various classes of explicit one-step exponential integrators. The resulting algorithms will be applied for solving stiff ordinary differential equations (ODEs) of the form

$$u'(t) = F(t, u(t)) = Au(t) + g(t, u(t)), \quad u(t_0) = u_0, \tag{1.1}$$

on the interval $t_0 \leq t \leq T$, where the vector field $F(t, u(t))$ can be decomposed into a linear stiff part $Au(t)$ and a nonlinear nonstiff part $g(t, u(t))$. Such systems belong to the class of stiff-nonstiff problems (couple systems of different time scales). They are usually resulted from the spatial discretization of time-dependent partial differential equations (PDEs) by means of a finite-difference, finite-element or some spectral method. Our main interest lies in the case where the stiff part is often cheap to compute while the nonstiff part is expensive to evaluate. This case is common in practice when using a non-uniform grid for the spatial discretization of PDEs.

Among numerical methods for solving (1.1), exponential integrators have shown to be very competitive in recent years, see for instances, [10, 11, 2, 13, 12, 17, 16]. So far, most methods for the implementation of exponential integrators require the approximation of products of matrix functions with vectors, i.e., $\phi(A)v$, $A \in \mathbb{R}^{d\times d}$, $v \in \mathbb{R}^d$. Depending on the structure of A, there are available a number of efficient methods, for instances, diagonalization, Padé approximation (if A is not too large), Chebyshev methods (if A is Hermitian or skew-Hermitian), (rational) Krylov subspace methods and Leja interpolation (if A is large). For further details, we refer the reader to an overview on exponential integrators and their implementation, see [14].

Inspired by very recent results [4, 6, 7, 5] on local-time stepping methods for problems related to (1.1) and motivated by the idea in [15, Sect. 5.3] in establishing a MTS procedure for exponential multistep methods of Adams-type, we will show how to derive MTS procedures for various classes of explicit one-step exponential integrators

*Submitted to the editors DATE.

Funding: This work was funded by the...

†... (vluan@smu.edu).

as well. Starting from a s-stage explicit exponential Runge–Kutta method applied to (1.1), our approach is to employ the idea of backward errors analysis. In particular, in each integration step, we will search for $s-1$ modified differential equations such that their exact solutions coincide with the corresponding internal stages; We then show how to compute approximately such exact solutions (i.e., internal stages). From this, we construct an additional modified differential equation for computing approximately the numerical approximation to the exact solution. The construction of such mentioned differential equations is heavily based on the forms of the method's coefficients.

We derive general MTS algorithms (Algorithms 3.1-**??**) for which can be interpreted as particular implementations (without matrix functions) of explicit exponential Runge-Kutta (including exponential quadrature rules, exponential time differencing (ETD) methods) and exponential Rosenbrock methods. With such MTS procedures at hand, we are going to construct various examples of MTS schemes with order of accuracy up to four for some well-known exponential integrators in the literature. They can be implemented by solving modified differential equations with the help of ODE solvers, which have order of at least the same as the order of the considered exponential integrator.

For our considered problems, the new algorithms turn out to be very competitive compared to the standard methods. Instead of solving nonlinear problems (1.1), at each integration step, they reduces to solve full linear ones with some polynomial in t in place of the nonlinearity $g(t, u(t))$ (The stability of ODE solvers for solving such linear problems is ensured by using smaller (micro) time steps). Morever, they do not require a starting values procedure as in MTS algorithms for exponential multistep methods and they can easily perform with an adaptive time step control.

The outline of the paper is the following: In Section 2, the derivation of the general class of exponential Runge-Kutta methods is represented in a way that leads us to a motivation for constructing MTS procedures based on such schemes. Section 3 is devoted to the derivation of general MTS algorithms for exponential quadrature rules, ETD methods and exponential Rosenbrock methods. In Section 4, we discuss about the stability of such MTS algorithms that leads to an appropriate choice of ODEs solvers for solving the resulting differential equations. Numerical examples are given in Section 5 to illustrate the efficiency of the new MTS schemes with order of accuracy up to four. The main results of the paper are Algorithm 3.1, Algorithm 3.2, Algorithm **??** and various MTS schemes with order of accuracy up to four based on several well-known exponential integrators in the literature.

2. Exponential Runge–Kutta methods and motivation.

In this section, we recall the idea for deriving the general class of exponential Runge-Kutta methods for solving (1.1). Our presentation below will be in a way that motivates a MTS procedure for such schemes.

2.1. Exponential Runge–Kutta methods.

For the derivation of exponential Runge-Kutta methods, it is crucial to use the following representation of the exact solution of (1.1) at time $t_{n+1} = t_n + h$ by the variation-of-constants formula

$$u(t_{n+1}) = u(t_n + h) = \mathrm{e}^{hA} u(t_n) + \int_0^h \mathrm{e}^{(h-\tau)A} g(t_n + \tau, u(t_n + \tau)) \mathrm{d}\tau. \tag{2.1}$$

Similarly to the construction of classical Runge–Kutta methods, the idea is to approximate the integral in (2.1) by a quadrature rule with nodes c_i and weights $b_i(hA)$

$(1 \le i \le s)$. This yields

$$u(t_{n+1}) \approx \mathrm{e}^{hA}u(t_n) + h\sum_{i=1}^{s} b_i(hA)g(t_n + c_ih, u(t_n + c_ih)). \tag{2.2}$$

By applying (2.1) (with c_ih in place of h), the unknown intermediate values $u(t_n+c_ih)$ in (2.2) can be represented as

$$u(t_n + c_ih) = \mathrm{e}^{c_ihA}u(t_n) + \int_0^{c_ih} \mathrm{e}^{(c_ih-\tau)A}g(t_n + \tau, u(t_n + \tau))\mathrm{d}\tau. \tag{2.3}$$

Again, one can use another quadrature rule with the same nodes c_i as before (to avoid the generation of new unknowns) and new weights $a_{ij}(hA)$ to approximate the integral in (2.3). This gives

$$u(t_n + c_ih) \approx \mathrm{e}^{c_ihA}u(t_n) + h\sum_{j=1}^{s} a_{ij}(hA)g(t_n + c_jh, u(t_n + c_jh)). \tag{2.4}$$

Now, assuming that approximations $u_n \approx u(t_n)$ and $U_{n,i} \approx u(t_n + c_ih)$ are given. From (2.4) and (2.2) we obtain the following general class of one-step methods, so-called exponential Runge–Kutta methods

$$U_{n,i} = \mathrm{e}^{c_ihA}u_n + h\sum_{j=1}^{s} a_{ij}(hA)g(t_n + c_jh, U_{n,j}), \quad 1 \le i \le s, \tag{2.5a}$$

$$u_{n+1} = \mathrm{e}^{hA}u_n + h\sum_{i=1}^{s} b_i(hA)g(t_n + c_ih, U_{n,i}). \tag{2.5b}$$

It turns out that the equilibria of (1.1) are preserved if the coefficients $a_{ij}(z)$ and $b_i(z)$ of the method fulfill the following simplifying assumptions (see [12])

$$\sum_{i=1}^{s} b_i(z) = \varphi_1(z), \qquad \sum_{j=1}^{s} a_{ij}(z) = c_i\varphi_1(c_iz), \quad 1 \le i \le s, \tag{2.6}$$

where $\varphi_1(z) = (\mathrm{e}^z - 1)/z$.
Throughout the paper we will consider methods of the general form (2.5) that satisfy (2.6).

Note that an *explicit* method of the form (2.5) is the case when $a_{ij}(hA) = 0$ for all $i \le j$ (implying $c_1 = 0$ due to the second condition in (2.6) and consequentially $U_{n,1} = u_n$). The sum in (2.5a) is then considered over index j from 1 to $i-1$ only. Thus, the internal stages $U_{n,i}$ $(2 \le i \le s)$ can be computed explicitly one after the other, which will be finally inserted into (2.5b) to find u_{n+1}.

Clearly, an efficient algorithm for computing the products of matrix functions with vectors of the form $\phi(hA)v$, $A \in \mathbb{R}^{d\times d}$, $v \in \mathbb{R}^d$ plays an important role in implementing (2.5). As we have mentioned in the Introduction, there are several options for performing this task, depending on the structure of A. In contrast to these approaches, however, we will introduce an alternative way to implement the class of exponential one-step methods (2.5) without matrix functions. This method will be later considered as a MTS algorithm. Inspired by the idea presented in [15, Sect. 5.3], first, we start with an observation which will be described as below.

2.2. Motivation. In view of (2.1) and (2.4), one can see that $u(t_{n+1})$ and $u(t_n + c_i h)$ are the exact solutions of the following differential equation

$$v'(\tau) = Av(\tau) + g(t_n + \tau, u(t_n + \tau)), \quad v(0) = u(t_n), \tag{2.7}$$

evaluated at $\tau = h$ and $\tau = c_i h$, respectively. In other words, solving (2.7) exactly (by means of the variation-of-constants formula) on the time intervals $[0, h]$ and $[0, c_i h]$ shows that $v(h) = u(t_{n+1})$, $v(c_i h) = u(t_n + c_i h)$. Unfortunately, one could not find such analytical solutions explicitly, since $u(t_n)$ and $u(t_n + \tau)$ are unknown values. This observation, however, suggests us to employ the idea of backward error analysis (see, for instance [8, Chap. IX]).

Given an exponential Runge-Kutta method (2.5), we are going to search for modified differential equations of (2.7) in which their exact solutions at $\tau = c_i h$ and $\tau = h$ will be $U_{n,i}$ $(2 \le i \le s)$ and u_{n+1}, respectively. If one could find such desired equations, then by solving them numerically we can obtain the corresponding approximations for $U_{n,i}$ and u_{n+1}. In the following sections, we will show how this can be done with most popular subclasses of explicit method of the form (2.5).

3. Explicit one-step exponential integrators and MTS algorithms. In this section, we restrict our attention to explicit methods of the general form (2.5). Guided by the observation in the previous section, in this section we will derive general MTS algorithms for exponential quadrature rules, ETD methods and exponential Rosenbrock methods.

3.1. Derivation of a MTS algorithm for exponential quadrature rules. As a special subclass of exponential Runge–Kutta methods, we mention exponential quadrature rules of collocation type (see [13, 14]). This integrator can be used for solving (1.1) in the form of linear parabolic problems, that is,

$$u'(t) = Au(t) + g(t), \quad u(t_0) = u_0. \tag{3.1}$$

It takes the form

$$u_{n+1} = \mathrm{e}^{hA} u_n + h \sum_{i=1}^{s} b_i(hA) g(t_n + c_i h) \tag{3.2a}$$

with

$$b_i(hA) = \int_0^1 \mathrm{e}^{(1-\theta)hA} \ell_i(\theta) \mathrm{d}\tau. \tag{3.2b}$$

Here, $\ell_i(\tau)$ are the Lagrange basis polynomials

$$\ell_i(\theta) = \prod_{\substack{m=1 \\ m \ne i}}^{s} \frac{\theta - c_m}{c_i - c_m}, \quad i = 1, \ldots, s. \tag{3.3}$$

It should be noted that (3.2) is resulted from a concrete quadrature rule for approximating the integral in (2.1), that is based on replacing g by its Lagrange interpolation polynomial using non-confluent collocation nodes $c_1, \ldots, c_s$).
Inserting the form of $b_i(hA)$ in (3.2b) into (3.2a) and changing the integration variable to $\tau = h\theta$, we get

$$u_{n+1} = \mathrm{e}^{hA} u_n + \int_0^h \mathrm{e}^{(h-\tau)A} \varrho_{n,s}(\tau) \mathrm{d}\tau, \tag{3.4}$$

where $\varrho_{n,s}(\tau)$ is a polynomial in τ and is given by

$$\varrho_{n,s}(\tau) = \sum_{i=1}^{s} \ell_i(\tau/h) g(t_n + c_i h). \tag{3.5}$$

This polynomial satisfies the collocation conditions $\varrho_{n,s}(c_i h) = g(t_n + c_i h)$. Taking a closer look at (3.4), we see that $u_{n+1} = v_n(h)$, where $v_n(h)$ is the exact solutions of the following differential equation,

$$v_n'(\tau) = A v_n(\tau) + \varrho_{n,s}(\tau), \quad v_n(0) = u_n \tag{3.6}$$

over the time interval $[0, h]$. From this point of view, one can consider (3.6) as a modified differential equation of (2.7). By setting $\hat{u}_0 = u_0$, in each step an approximation $\hat{u}_{n+1}$ of u_{n+1} (written as $\hat{u}_{n+1} \approx u_{n+1}$) can be computed by solving the differential equation

$$y_n'(\tau) = A y_n(\tau) + \varrho_{n,s}(\tau), \quad y_n(0) = \hat{u}_n \tag{3.7}$$

numerically on the interval $[0, h]$. This task can be carried out by an ODE solver (see Section 4) using micro time steps. One obtains $\hat{u}_{n+1} \approx y_n(h) \approx v_n(h) = u_{n+1}$. Therefore, such method can be interpreted as a MTS procedure for exponential quadrature rules (since a macro time step h is also used for computing the polynomials in (3.5)). By using this method, in each integration step, we have to solve the linear problem (3.7) but only with polynomial $\varrho_{n,s}$ instead of using function g as in the original problem (3.1). We will give some examples of $\varrho_{n,s}$ as below.

EXAMPLE 3.1.1. By denoting $g_{n,i} = g(t_n + c_i h)$ and using formula (3.5) for $s = 1, 2, 3$, we obtain the following polynomials:

$$\varrho_{n,1}(\tau) = g_{n,1}, \tag{3.8a}$$

$$\varrho_{n,2}(\tau) = \frac{1}{h(c_2 - c_1)} \big[(g_{n,1} + g_{n,2})\tau - h(c_1 g_{n,1} - c_2 g_{n,2}) \big], \tag{3.8b}$$

$$\begin{aligned} \varrho_{n,3}(\tau) = \frac{1}{h^2} \Big[&\frac{(\tau - c_2 h)(\tau - c_3 h)}{(c_1 - c_2)(c_1 - c_3)} g_{n,1} + \frac{(\tau - c_1 h)(\tau - c_3 h)}{(c_2 - c_1)(c_2 - c_3)} g_{n,2} \\ &+ \frac{(\tau - c_1 h)(\tau - c_2 h)}{(c_3 - c_1)(c_3 - c_2)} g_{n,3} \Big]. \end{aligned} \tag{3.8c}$$

Step size control. It is known that the exponential quadrature rule (3.2) converges with order s (see [14, Sect. 2.2]). For an implementation of this scheme using variable step sizes, one can consider together with (3.2) (or its equivalent form (3.4)) a method of order $s - 1$

$$\bar{u}_{n+1} = \mathrm{e}^{hA} u_n + \int_0^h \mathrm{e}^{(h-\tau)A} \varrho_{n,s-1}(\tau) \mathrm{d}\tau. \tag{3.9}$$

The step size selection is based on the error estimate $\mathtt{err}_{n+1} = u_{n+1} - \bar{u}_{n+1}$ (see [9, Chapter IV.8]). By subtracting (3.9) from (3.4), it is easy to see that $\mathtt{err}_{n+1} = w_n(h)$, where $w_n(h)$ is the exact solution of the following differential equation

$$w_n'(\tau) = A w_n(\tau) + \varrho_{n,s}(\tau) - \varrho_{n,s-1}(\tau), \quad w_n(0) = 0 \tag{3.10}$$

over the time interval $[0, h]$. Therefore, in order to avoid computing matrix functions, we can solve (3.10) numerically using micro time steps to obtain an approximate value $\widehat{\mathtt{err}}_{n+1}$ for $\mathtt{err}_{n+1}$. This can be done by using the same chosen ODE solver for integrating (3.6).

As a summarization for such procedure, we state a general MTS algorithm for exponential quadrature rules.

Algorithm 3.1 A general MTS algorithm for exponential quadrature rules

- **Input:** A; $g(t)$; $[t_0, T]$; u_0; s; non-confluent nodes c_i $(i = 1, \ldots, s)$; initial step size h (for the case of constant step size, $h = (T - t_0)/N$, where N is the number of sub-intervals).
- **Initialization:** Set $n = 0$; $\hat{u}_0 = u_0$.
  While $t_0 < T$
  1. Solve (3.7) on $[0, h]$ to get $\hat{u}_1 \approx y_0(h)$.
  2. [Step size control] Solve (3.10) on $[0, h]$ to get $\widehat{\mathtt{err}}_1$ and perform $h := h_{\text{new}}$.
  3. Update $t_0 := t_0 + h$, $\hat{u}_0 := \hat{u}_1$, $n := n + 1$.
- **Output:** Approximate values $\hat{u}_n \approx u_n, n = 1, 2, \ldots$ (where u_n is the numerical solution at time t_n obtained by an exponential quadrature rule).

3.2. Derivation of a general MTS algorithm for ETD methods. The class of ETD methods (see for example [2, 3]) is a particular and important subclass of exponential Runge-Kutta methods in the explicit form of (2.5). For these integrators, the coefficients $a_{ij}(hA)$ and $b_i(hA)$ are linear combinations of the entire functions $\varphi_k(c_i hA)$ and $\varphi_k(hA)$, respectively. Hence, we can write

$$a_{ij}(hA) = \sum_{k=1}^{\ell_{ij}} \alpha_{ij}^{(k)} \varphi_k(c_i hA), \ b_i(hA) = \sum_{k=1}^{m_i} \beta_i^{(k)} \varphi_k(hA), \tag{3.11}$$

where ℓ_{ij} and m_i are positive integers and φ_k are given by

$$\varphi_k(z) = \int_0^1 \mathrm{e}^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} \mathrm{d}\theta, \quad k \geq 1. \tag{3.12}$$

They satisfy the recurrence relations

$$\varphi_k(z) = \frac{\varphi_{k-1}(z) - \varphi_{k-1}(0)}{z}, \quad \varphi_0(z) = \mathrm{e}^z. \tag{3.13}$$

By changing the integration variable to $\tau = h\theta$ in (3.12), we obtain

$$\varphi_k(z) = \frac{1}{h^k} \int_0^h \mathrm{e}^{(h-\tau)\frac{z}{h}} \frac{\tau^{k-1}}{(k-1)!} \mathrm{d}\tau, \quad k \geq 1. \tag{3.14}$$

Substituting $z = hA$ and $z = c_i hA$ into (3.14) and inserting the obtained results for $\varphi_k(c_i hA)$ and $\varphi_k(hA)$ into (3.11) finally shows that

$$a_{ij}(hA) = \int_0^{c_i h} \mathrm{e}^{(c_i h - \tau)A} \sum_{k=1}^{\ell_{ij}} \frac{\alpha_{ij}^{(k)}}{(c_i h)^k (k-1)!} \tau^{k-1} \mathrm{d}\tau, \tag{3.15a}$$

$$b_i(hA) = \int_0^h \mathrm{e}^{(h-\tau)A} \sum_{k=1}^{m_i} \frac{\beta_i^{(k)}}{h^k (k-1)!} \tau^{k-1} \mathrm{d}\tau. \tag{3.15b}$$

We now insert (3.15) into (2.5) (in explicit form) to get

$$U_{n,i} = \mathrm{e}^{c_i hA} u_n + \int_0^{c_i h} \mathrm{e}^{(c_i h-\tau)A} p_{n,i}(\tau)\mathrm{d}\tau, \quad 2 \le i \le s, \tag{3.16a}$$

$$u_{n+1} = \mathrm{e}^{hA} u_n + \int_0^{h} \mathrm{e}^{(h-\tau)A} q_{n,s}(\tau)\mathrm{d}\tau \tag{3.16b}$$

with

$$p_{n,i}(\tau) = \sum_{j=1}^{i-1} \Big(\sum_{k=1}^{\ell_{ij}} \frac{\alpha_{ij}^{(k)}}{c_i^k h^{k-1}(k-1)!} \tau^{k-1} \Big) g(t_n + c_j h, U_{n,j}), \tag{3.17a}$$

$$q_{n,s}(\tau) = \sum_{i=1}^{s} \Big(\sum_{k=1}^{m_i} \frac{\beta_{i}^{(k)}}{h^{k-1}(k-1)!} \tau^{k-1} \Big) g(t_n + c_i h, U_{n,i}) \tag{3.17b}$$

are polynomials in τ.
From (3.16), we realize that $U_{n,i} = v_{n,i}(c_i h),\ u_{n+1} = v_n(h)$, where $v_{n,i}(c_i h)$ and $v_n(h)$ are the exact solutions of the following differential equations,

$$v'_{n,i}(\tau) = Av_{n,i}(\tau) + p_{n,i}(\tau), \quad v_{n,i}(0) = u_n \quad (2 \le i \le s), \tag{3.18a}$$

$$v'_{n}(\tau) = Av_{n}(\tau) + q_{n,s}(\tau), \quad v_{n}(0) = u_n \tag{3.18b}$$

over the time intervals $[0, c_i h]$ and $[0, h]$, respectively. These equations can be thus considered as modified differential equations of (2.7).

Clearly, one could not solve (3.18) analytically on the considered intervals for finding $U_{n,i}$ and u_{n+1}. Given $\hat{u}_0 = u_0$, however, we can compute $\widehat{U}_{n,i}$ and $\hat{u}_{n+1}$ which denote the approximations of $U_{n,i}$ and u_{n+1}, written as $\widehat{U}_{n,i} \approx U_{n,i},\ \hat{u}_{n+1} \approx u_{n+1}$ $(n \ge 0)$, respectively. First, the idea is to replace the polynomials in (3.17) by

$$\hat{p}_{n,i}(\tau) = \sum_{j=1}^{i-1} \Big(\sum_{k=1}^{\ell_{ij}} \frac{\alpha_{ij}^{(k)}}{c_i^k h^{k-1}(k-1)!} \tau^{k-1} \Big) g(t_n + c_j h, \widehat{U}_{n,j}), \tag{3.19a}$$

$$\hat{q}_{n,s}(\tau) = \sum_{i=1}^{s} \Big(\sum_{k=1}^{m_i} \frac{\beta_{i}^{(k)}}{h^{k-1}(k-1)!} \tau^{k-1} \Big) g(t_n + c_i h, \widehat{U}_{n,i}), \tag{3.19b}$$

respectively. Then, $\widehat{U}_{n,i}$ can be computed as numerical solutions (by means of an ODE solver) of the differential equations

$$y'_{n,i}(\tau) = Ay_{n,i}(\tau) + \hat{p}_{n,i}(\tau), \quad y_{n,i}(0) = \hat{u}_n \quad (2 \le i \le s) \tag{3.20}$$

considered on $[0, c_i h]$. By doing so, we have $\widehat{U}_{n,i} \approx y_{n,i}(c_i h) \approx v_{n,i}(c_i h) = U_{n,i}$.
The strategy for computing $\widehat{U}_{n,i}$ is listed as an iteration below:

1. Setting $\widehat{U}_{n,1} = \hat{u}_n$.
2. Knowing $\widehat{U}_{n,1}$, we get $\hat{p}_{n,2}(\tau)$ from (3.19a) and solve (3.20) with $i = 2$ to obtain $\widehat{U}_{n,2} \approx y_{n,2}(c_2 h)$.
3. Knowing $\widehat{U}_{n,1}, \widehat{U}_{n,2}$, we get $\hat{p}_{n,3}(\tau)$ from (3.19a) and solve (3.20) with $i = 3$ to obtain $\widehat{U}_{n,3} \approx y_{n,3}(c_3 h)$.

   $\vdots$

4. Knowing $\widehat{U}_{n,1},\dots,\widehat{U}_{n,s-1}$, we get $\hat{p}_{n,s}(\tau)$ from (3.19a) and solve (3.20) with $i=s$ to obtain $\widehat{U}_{n,s}\approx y_{n,s}(c_s h)$.

With $\widehat{U}_{n,i}$ at hand, the next step is to find $\hat{q}_{n,s}(\tau)$ from (3.19b). Finally, $\hat{u}_{n+1}$ can be computed by an appropriate ODE solver using micro time steps to solve the differential equation

$$y_n'(\tau)=Ay_n(\tau)+\hat{q}_{n,s}(\tau),\quad y_n(0)=\hat{u}_n \tag{3.21}$$

on the interval $[0,h]$. Indeed, we have $\hat{u}_{n+1}\approx y_n(h)\approx v_n(h)=u_{n+1}$.

Further details on choosing such ODE solvers will be discussed in Section 4. Since we have to use a macro time step h for computing the polynomials in (3.19), the method can be interpreted as a MTS procedure based on ETD methods. One can consider this method as a particular implementation of ETD methods. It offers several interesting features. Instead of solving nonlinear problems (1.1), the method reduces to solve full linear differential equations (3.20) and (3.21) with the polynomials in t in place of the nonlinearity $g(t,u(t))$. It can easily perform adaptive time steps selection (see below) and it does not require a starting values procedure as in MTS algorithms for exponential multistep methods. The stability is ensured by using micro time steps. Finally, it is cheap if a few stages are used.

Step size control. It is possible to establish a variable step sizes implementation for such MTS procedure without computing matrix functions. First, we consider together with (2.5b) embedded methods of lower orders

$$\bar{u}_{n+1}=\mathrm{e}^{hA}u_n+h\sum_{i=1}^{s}\bar{b}_i(hA)g(t_n+c_ih,U_{n,i}) \tag{3.22}$$

that use the same stages $U_{n,i}$, and with weights $\bar{b}_i(hA)$. Again, we can represent $\bar{b}_i(hA)=\sum_{k=1}^{q_i}\bar{\beta}_i^{(k)}\varphi_k(hA)$ and show that

$$\bar{u}_{n+1}=\mathrm{e}^{hA}u_n+\int_0^h \mathrm{e}^{(h-\tau)A}\bar{q}_{n,s}(\tau)\mathrm{d}\tau \tag{3.23}$$

with

$$\bar{q}_{n,s}(\tau)=\sum_{i=1}^{s}\Big(\sum_{k=1}^{q_i}\frac{\bar{\beta}_i^{(k)}}{h^{k-1}(k-1)!}\tau^{k-1}\Big)g(t_n+c_ih,U_{n,i}). \tag{3.24}$$

To perform a step size selection, it is crucial to compute the error estimate $\mathtt{err}_{n+1}=u_{n+1}-\bar{u}_{n+1}$. In view of (3.16b) and (3.23), we deduce that $\mathtt{err}_{n+1}=w_n(h)$, where $w_n(h)$ is the exact solution of the following differential equation

$$w_n'(\tau)=Aw_n(\tau)+q_{n,s}(\tau)-\bar{q}_{n,s}(\tau),\quad w_n(0)=0 \tag{3.25}$$

over the interval $[0,h]$. Again, one could not solve (3.25) analytically. We therefore use the same idea as for solving (3.18), that is to replace $q_{n,s}(\tau)$ by $\hat{q}_{n,s}(\tau)$ given in (3.19b), $\bar{q}_{n,s}(\tau)$ by $\widehat{\bar{q}}_{n,s}(\tau)$ which is given by

$$\widehat{\bar{q}}_{n,s}(\tau)=\sum_{i=1}^{s}\Big(\sum_{k=1}^{q_i}\frac{\bar{\beta}_i^{(k)}}{h^{k-1}(k-1)!}\tau^{k-1}\Big)g(t_n+c_ih,\widehat{U}_{n,i}). \tag{3.26}$$

Instead of working with (3.25), we now consider the differential equation

$$z_n'(\tau)=Az_n(\tau)+\hat{q}_{n,s}(\tau)-\widehat{\bar{q}}_{n,s}(\tau),\quad z_n(0)=0 \tag{3.27}$$

on $[0,h]$. By applying the same chosen ODE solver (for (3.21)) to (3.27), one can compute an approximate value $\widehat{\mathtt{err}}_{n+1} \approx z_n(h) \approx w_n(h) = \mathtt{err}_{n+1}$.

We are now ready to summary such MTS procedure by Algorithm 3.2 below.

Algorithm 3.2 A general MTS algorithm for ETD methods

- **Input:** A; $g(u)$; $[t_0, T]$; u_0; s; c_i $(i = 1, \ldots, s)$; initial step size h (for the case of constant step size, $h = (T - t_0)/N$, where N is the number of sub-intervals).
- **Initialization:** Set $n = 0$; $\hat{u}_0 = u_0$.
  While $t_0 < T$
  1. Set $\widehat{U}_{0,1} = \hat{u}_0$.
  2. For $i = 2, \ldots, s$ do
     (a) Find $\hat{p}_{0,i}(\tau)$ as in (3.19a).
     (b) Solve (3.20) on $[0, c_i h]$ to obtain $\widehat{U}_{0,i} \approx y_{0,i}(c_i h)$.
  3. Find $\hat{q}_{0,s}(\tau)$ as in (3.19b) (for a step size control, find also $\widehat{\hat{q}}_{0,s}(\tau)$ as in (3.26)).
  4. Solve (3.21) on $[0,h]$ to get $\hat{u}_1 \approx y_0(h)$.
  5. [Step size control] Solve (3.27) on $[0,h]$ to get $\widehat{\mathtt{err}}_1$) and perform $h := h_{\text{new}}$.
  6. Update $t_0 := t_0 + h$, $\hat{u}_0 := \hat{u}_1$, $n := n + 1$.
- **Output:** Approximate values $\hat{u}_n \approx u_n, n = 1, 2, \ldots$ (where u_n is the numerical solution at time t_n obtained by an ETD method).

In order to perform Algorithm 3.2, we consider some well-known ETD methods in the literature and give explicit forms of the polynomials in (3.19) and (3.26) for a given value of s. For simplicity, we denote $\varphi_k = \varphi_k(z), \varphi_{k,i} = \varphi_k(c_i z)$.

EXAMPLE 3.2.1. For $s = 2$, we consider the second-order (stiff order two) ETD methods (see [12, Sect. 5.1]) with a first-order error estimate (the exponential Euler method) which will be called `exprk21`. Its coefficients are displayed in the following Butcher tableau

$$
\begin{array}{c|cc}
0 & & \\
c_2 & c_2\varphi_{1,2} & \\
\hline
 & \varphi_1 - \frac{1}{c_2}\varphi_2 & \frac{1}{c_2}\varphi_2 \\
 & \varphi_1 & 0
\end{array}.
$$

In this case, we have $\ell_{21} = 1, m_1 = m_2 = 2, q_1 = 1, \alpha_{21}^{(1)} = c_2, \beta_1^{(1)} = \bar{\beta}_1^{(1)} = 1, \beta_1^{(2)} = -\frac{1}{c_2}, \beta_2^{(1)} = 0, \beta_2^{(2)} = \frac{1}{c_2}$ $(c_2 > 0)$, $\bar{\beta}_2^{(k)} = 0$ $\forall k = 1, \ldots, q_2$.
Using this, one obtains from (3.19) and (3.26) that

$$\hat{p}_{n,2}(\tau) = g(t_n, \hat{u}_n), \tag{3.28a}$$

$$\hat{q}_{n,2}(\tau) = \tfrac{1}{c_2 h} g(t_n + c_2 h, \widehat{U}_{n,2})\tau + \big(1 - \tfrac{\tau}{c_2 h}\big)\hat{p}_{n,2}(\tau), \tag{3.28b}$$

$$\widehat{\hat{q}}_{n,2}(\tau) = \hat{p}_{n,2}(\tau). \tag{3.28c}$$

EXAMPLE 3.2.2. A class of three-stage $(s = 3)$ ETD methods of stiff order three was constructed in [12, Sect. 5.2]. We consider this method with a second-order error estimate and name it as `exprk32`. It is given by

$$
\begin{array}{c|ccc}
0 & & & \\
c_2 & c_2\varphi_{1,2} & & \\
\frac{2}{3} & \frac{2}{3}\varphi_{1,3} - \frac{4}{9c_2}\varphi_{2,3} & \frac{4}{9c_2}\varphi_{2,3} & \\
\hline
 & \varphi_1 - \frac{3}{2}\varphi_2 & 0 & \frac{3}{2}\varphi_2 \\
 & \varphi_1 & \frac{3}{3c_2-2}\varphi_2 & \frac{3}{2-3c_2}\varphi_2
\end{array}.
$$

From this Butcher tableau, we have $\ell_{21} = q_1 = 1, \ell_{31} = \ell_{32} = m_1 = m_3 = q_2 = q_3 = 2$; $\alpha_{21}^{(1)} = c_2, \alpha_{31}^{(1)} = \frac{2}{3}, \alpha_{31}^{(2)} = -\frac{4}{9c_2}, \alpha_{32}^{(1)} = 0, \alpha_{32}^{(2)} = \frac{4}{9c_2}$; $\beta_1^{(1)} = 1, \beta_1^{(2)} = -\frac{3}{2}, \beta_2^{(k)} = 0\ \forall k = 1, \dots, m_2, \beta_3^{(1)} = 0, \beta_3^{(1)} = \frac{3}{2}$; $\bar{\beta}_2^{(1)} = \bar{\beta}_3^{(1)} = 0, \bar{\beta}_2^{(2)} = -\bar{\beta}_3^{(2)} = \frac{3}{3c_2-2}(c_2 \neq \frac{2}{3})$. It now follows from (3.19) and (3.26) that

$$\hat{p}_{n,2}(\tau) = g(t_n, \hat{u}_n), \tag{3.29a}$$

$$\hat{p}_{n,3}(\tau) = \tfrac{1}{c_2 h} g(t_n + c_2 h, \widehat{U}_{n,2})\tau + \big(1 - \tfrac{\tau}{c_2 h}\big)\hat{p}_{n,2}(\tau), \tag{3.29b}$$

$$\hat{q}_{n,3}(\tau) = \tfrac{3}{2h} g(t_n + \tfrac{2}{3}h, \widehat{U}_{n,3})\tau + \big(1 - \tfrac{3\tau}{2h}\big)\hat{p}_{n,2}(\tau), \tag{3.29c}$$

$$\widehat{\bar{q}}_{n,3}(\tau) = \tfrac{3}{(3c_2-2)h}\big(g(t_n + c_2 h, \widehat{U}_{n,2}) - g(t_n + \tfrac{2}{3}h, \widehat{U}_{n,3})\big)\tau + \hat{p}_{n,2}(\tau). \tag{3.29d}$$

Other three-stage ETD methods can be found in the literature. For instances, a method called ETD3RK was constructed in [2] or another one called ETD2CF3 was given in [1]. For these methods, we can use the same way to compute polynomials in (3.19) and (3.26). We omit details.

4. Error analysis and on choosing ODE solvers.

In this section, we will give error bounds and discuss how to choose ODE solvers for solving modified differential equations arising in the above mentioned MTS algorithms. The main aim is to obtain resulted approximations which should have the same order of accuracy as the considered exponential integrators.

For the local error analysis of scheme (2.5), we consider one step with initial value $\tilde{u}_n = u(t_n)$ on the exact solution, i.e.

$$U_{n,i} = \mathrm{e}^{c_i hA} u_n + h \sum_{j=1}^{s} a_{ij}(hA) g(t_n + c_j h, U_{n,j}), \quad 1 \leq i \leq s, \tag{4.1a}$$

$$u_{n+1} = \mathrm{e}^{hA} u_n + h \sum_{i=1}^{s} b_i(hA) g(t_n + c_i h, U_{n,i}). \tag{4.1b}$$

Let $\tilde{e}_{n+1} = \hat{u}_{n+1} - \tilde{u}_{n+1}$ denote the local error, i.e., the difference between the numerical solution $\hat{u}_{n+1}$ after one step starting from $\tilde{u}_n$ and the corresponding exact solution of (1.1) at t_{n+1}.

5. Numerical experiments.

6. Conclusion.

Acknowledgments. The first author was supported by the Deutsche Forschungsgemeinschaft (DFG) through the DFG Research Training Group 1294 "Analysis, Simulation and Design of Nanotechnological Processes" at the Karlsruhe Institute of Technology (KIT).

REFERENCES

[1] E. Celledoni, A. Marthinsen, and B. Owren, *Commutator-free lie group methods*, Future Generation Computer Systems, 19 (2003), pp. 341–352.

[2] S. M. Cox and P. C. Matthews, *Exponential time differencing for stiff systems*, Journal of Computational Physics, 176 (2002), pp. 430–455.

[3] A. Friedli, *Verallgemeinerte runge-kutta verfahren zur lösung steifer differentialgleichungssysteme*, in Numerical treatment of differential equations, Springer, 1978, pp. 35–50.

[4] M. J. Gander and L. Halpern, *Techniques for locally adaptive time stepping developed over the last two decades*, in Domain decomposition methods in science and engineering XX, Springer, 2013, pp. 377–385.

[5] M. Grote and T. Mitkova, *Explicit local time-stepping methods for time-dependent wave propagation*, in Direct and Inverse Problems in Wave Propagation and Applications, J.-M. M. M. S. I. Graham, U. Langer, ed., de Gruyter, 2013, pp. 187–218.

[6] M. J. Grote and T. Mitkova, *Explicit local time-stepping methods for maxwells equations*, Journal of Computational and Applied Mathematics, 234 (2010), pp. 3283–3302.

[7] M. J. Grote and T. Mitkova, *High-order explicit local time-stepping methods for damped wave equations*, Journal of Computational and Applied Mathematics, 239 (2013), pp. 270–289.

[8] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, vol. 31, Springer Science & Business Media, 2006.

[9] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer, New York, 1996.

[10] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.

[11] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.

[12] M. Hochbruck and A. Ostermann, *Explicit exponential Runge–Kutta methods for semilinear parabolic problems*, SIAM J. Numer. Anal., 43 (2005), pp. 1069–1090.

[13] M. Hochbruck and A. Ostermann, *Exponential runge–kutta methods for parabolic problems*, Applied Numerical Mathematics, 53 (2005), pp. 323–339.

[14] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 19 (2010), pp. 209–286.

[15] M. Hochbruck and A. Ostermann, *Exponential multistep methods of adams-type*, BIT Numerical Mathematics, 51 (2011), pp. 889–908.

[16] V. T. Luan and A. Ostermann, *Explicit exponential Runge–Kutta methods of high order for parabolic problems*, J. Comput. Appl. Math., 256 (2014), pp. 168–179.

[17] V. T. Luan and A. Ostermann, *Exponential Rosenbrock methods of order five–construction, analysis and numerical comparisons*, J. Comput. Appl. Math., 255 (2014), pp. 417–431.