

A NEW CLASS OF HIGH-ORDER METHODS FOR MULTIRATE DIFFERENTIAL EQUATIONS*

VU THAI LUAN[†], RUJEKO CHINOMONA[‡], AND DANIEL R. REYNOLDS[‡]

Abstract. This work focuses on the development of a new class of high-order accurate methods for multirate time integration of systems of ordinary differential equations. The proposed methods are based on a specific subset of explicit one-step exponential integrators. More precisely, starting from an explicit exponential Runge–Kutta method of the appropriate form, we derive a multirate algorithm to approximate the action of the matrix exponential through the definition of modified “fast” initial-value problems. These fast problems may be solved using any viable solver, enabling multirate simulations through use of a subcycled method. Due to this structure, we name these *multirate exponential Runge–Kutta* (MERK) methods. In addition to showing how MERK methods may be derived, we provide rigorous convergence analysis, showing that for an overall method of order p , the fast problems corresponding to internal stages may be solved using a method of order $p - 1$, while the final fast problem corresponding to the time-evolved solution must use a method of order p . Numerical simulations are then provided to demonstrate the convergence and efficiency of MERK methods with orders three through five on a series of multirate test problems.

Key words. multirate time integration, multiple time stepping, exponential integrators, exponential Runge–Kutta methods

AMS subject classifications. 65L06, 65M20, 65L20

DOI. 10.1137/19M125621X

1. Introduction. In this paper, we focus on the construction, analysis, and implementation of efficient, highly accurate, multirate time stepping algorithms, based on explicit exponential Runge–Kutta methods. These algorithms may be applied to initial value problems (IVPs) of the form

$$(1.1) \quad u'(t) = F(t, u(t)) = \mathcal{L}u(t) + \mathcal{N}(t, u(t)), \quad u(t_0) = u_0,$$

on the interval $t_0 < t \leq T$, where the vector field $F(t, u(t))$ can be decomposed into a linear part $\mathcal{L}u(t)$ comprising the “fast” time scale, and a nonlinear part $\mathcal{N}(t, u(t))$ comprising the “slow” time scale. Such systems frequently result from so-called “multi-physics” simulations that couple separate physical processes together, or from the spatial semidiscretization of time-dependent partial differential equations (PDEs). Our primary interest in this paper lies in the case where the fast component is much less costly to compute than the slow component, thereby opening the door for methods that evolve each component with different time step sizes—so-called multirate (or multiple time stepping) methods. This case is common in practice when using a non-uniform grid for the spatial semidiscretization of PDEs, or in parallel computations

*Submitted to the journal’s Methods and Algorithms for Scientific Computing section April 15, 2019; accepted for publication (in revised form) January 13, 2020; published electronically April 27, 2020.

<https://doi.org/10.1137/19M125621X>

Funding: This work was partially supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program through the FASTMath Institute under Lawrence Livermore National Laboratory Subcontract B626484.

[†]Department of Mathematics and Statistics, Mississippi State University, Mississippi State, MS 39762 and Institute of Information Technology, Vietnam Academy of Science Technology (VAST), 18 Hoang Quoc Viet, Hanoi, Vietnam (luan@math.msstate.edu).

[‡]Department of Mathematics, Southern Methodist University, Dallas, TX 75275-0156 (rchinomona@smu.edu, reynolds@smu.edu).

where the fast component is comprised of spatially localized processes but the slow component requires communication across the parallel network.

In recent years, there has been renewed interest in the construction of multirate time integration methods for systems of ODEs. Generally, these efforts have focused on techniques to achieve orders of accuracy greater than two, since second-order methods may be obtained through simple interpolation between time scales. These recent approaches broadly fit into two categories: methods that attain higher-order through deferred correction or extrapolation of low-order methods [1, 4, 5], and methods that directly satisfy order conditions for partitioned and/or additive Runge–Kutta methods [9, 14, 15, 24, 25, 27, 39, 41, 43, 44, 45, 46, 47]. Of these, the latter category promises increased efficiency due to the need to traverse the time interval only once. However, only very recently have methods of this type been constructed that can achieve full fourth-order accuracy [39, 46, 38], and we know of no previous methods having order five or higher.

Among numerical methods that use the same time step for all components of (1.1), exponential integrators have shown great promise in recent years [6, 18, 19, 20, 21, 22, 30, 31, 32, 34, 28, 35]. Most such methods require the approximation of products of matrix functions with vectors, i.e., $\phi(\mathcal{L})v$, for $\mathcal{L} \in \mathbb{R}^{d \times d}$ and $v \in \mathbb{R}^d$.

Inspired by recent results on local time stepping methods for problems related to (1.1) [10, 11, 12, 13], and motivated by the idea in [23, sect. 5.3] that establishes a multirate procedure for exponential *multistep* methods of Adams type; here we derive multirate procedures for exponential *one-step* methods. Starting from an s -stage explicit exponential Runge–Kutta (ExpRK) method applied to (1.1), we employ the idea of backward error analysis to define $s - 1$ modified differential equations whose exact solutions coincide with the ExpRK internal stages. These modified differential equations may then be evolved using standard ODE solvers at the fast time scale. We name the resulting methods *multirate exponential Runge–Kutta* (MERK) methods.

The ability to construct modified ODEs for each slow ExpRK stage is dependent on the form of the ExpRK method itself, and we identify these restrictions within this manuscript. Using this approach, we derive a general multirate algorithm (Algorithm 3.1) that can be interpreted as a particular implementation (without matrix functions) of explicit ExpRK methods. With this algorithm in hand, we perform a rigorous convergence analysis for the proposed MERK methods. We additionally construct MERK schemes with orders of accuracy two through five, based on some well-known ExpRK methods from the literature.

We note that the resulting methods show strong similarities to the multirate infinitesimal step (MIS) methods in [24, 25, 43, 44, 45, 47] and the follow-on relaxed multirate infinitesimal step (RMIS) methods [46] and multirate infinitesimal general-structure additive Runge–Kutta (MRI-GARK) methods [39, 38], in that the MERK algorithm requires the construction of a set of modified fast IVPs that must be solved to proceed between slow stages, and where these modifications take the form of polynomials based on slow function data. While these approaches indeed result in similar algorithmic structure, (R)MIS and MRI-GARK methods are based on partitioned and generalized-structure additive Runge–Kutta theory [40], and as such their derivation requires satisfaction of many more order conditions than MERK methods, particularly as the desired method order increases, to the end that no MIS method of order greater than three, and no RMIS or MRI-GARK methods of order greater than four, have ever been proposed. Additionally, to obtain an overall order p method, all fast IVPs for (R)MIS and MRI-GARK methods must be solved to order p , whereas the

internal stages in MERK methods may use an order $p-1$ solver. Finally, both (R)MIS and the MRI-GARK methods from [39] require sorted abscissas $c_1 \leq c_2 \leq \dots \leq c_s$, a requirement that is not present for MERK methods or for the step predictor corrector (SPC-MRI-GARK) method in [38].

The outline of this paper is as follows: in section 2, we derive the general class of Exprk methods in a way that facilitates construction of MERK procedures. In section 3, we then derive the general MERK algorithm for exponential Runge–Kutta methods, and provide a rigorous convergence analysis for these schemes. In section 4, we derive specific MERK methods based on existing exponential Runge–Kutta methods. We present a variety of numerical examples in section 5 to illustrate the efficiency of the new MERK schemes with order of accuracy up to five. The main contributions of this paper are Algorithm 3.1, convergence analysis for MERK methods (Theorem 3.4), and the proposed MERK schemes with order of accuracy up to five.

2. Motivation. We begin with a general derivation of exponential Runge–Kutta methods [20, 22], which motivates a multirate procedure for solving (1.1).

2.1. Exponential Runge–Kutta methods. When deriving Exprk methods, it is crucial to represent the exact solution of (1.1) at time $t_{n+1} = t_n + H$ using the variation-of-constants formula,

$$(2.1) \quad u(t_{n+1}) = u(t_n + H) = e^{H\mathcal{L}}u(t_n) + \int_0^H e^{(H-\tau)\mathcal{L}}\mathcal{N}(t_n + \tau, u(t_n + \tau))d\tau.$$

The integral in (2.1) is then approximated using a quadrature rule having nodes c_i and weights $b_i(H\mathcal{L})$ ($i = 1, \dots, s$), which yields

$$(2.2) \quad u(t_{n+1}) \approx e^{H\mathcal{L}}u(t_n) + H \sum_{i=1}^s b_i(H\mathcal{L})\mathcal{N}(t_n + c_iH, u(t_n + c_iH)).$$

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

$$(2.3) \quad u(t_n + c_iH) = e^{c_iH\mathcal{L}}u(t_n) + \int_0^{c_iH} e^{(c_iH-\tau)\mathcal{L}}\mathcal{N}(t_n + \tau, u(t_n + \tau))d\tau.$$

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 $a_{ij}(H\mathcal{L})$ to approximate the integrals in (2.3). This gives

$$(2.4) \quad u(t_n + c_iH) \approx e^{c_iH\mathcal{L}}u(t_n) + H \sum_{j=1}^s a_{ij}(H\mathcal{L})\mathcal{N}(t_n + c_jH, u(t_n + c_jH)).$$

Now, denoting the approximations $u_n \approx u(t_n)$ and $U_{n,i} \approx u(t_n + c_iH)$, then from (2.2) and (2.4) one may obtain the so-called Exprk methods

$$(2.5a) \quad U_{n,i} = e^{c_iH\mathcal{L}}u_n + H \sum_{j=1}^s a_{ij}(H\mathcal{L})\mathcal{N}(t_n + c_jH, U_{n,j}), \quad i = 1, \dots, s,$$

$$(2.5b) \quad u_{n+1} = e^{H\mathcal{L}}u_n + H \sum_{i=1}^s b_i(H\mathcal{L})\mathcal{N}(t_n + c_iH, U_{n,i}).$$

The formula (2.5) is considered *explicit* when $a_{ij}(H\mathcal{L}) = 0$ for all $i \leq j$ (thus $c_1 = 0$ and consequentially $U_{n,1} = u_n$). Throughout this paper we restrict our attention to explicit exponential Runge–Kutta methods, which can be reformulated as (see [31, 33])

$$(2.6a) \quad U_{n,i} = u_n + c_i H \varphi_1(c_i H \mathcal{L}) F(t_n, u_n) + H \sum_{j=2}^{i-1} a_{ij}(H\mathcal{L}) D_{n,j}, \quad i = 2, \dots, s,$$

$$(2.6b) \quad u_{n+1} = u_n + H \varphi_1(H\mathcal{L}) F(t_n, u_n) + H \sum_{i=2}^s b_i(H\mathcal{L}) D_{n,i},$$

where

$$(2.7) \quad D_{n,i} = \mathcal{N}(t_n + c_i H, U_{n,i}) - \mathcal{N}(t_n, u_n), \quad i = 2, \dots, s.$$

Here, the coefficients $a_{ij}(H\mathcal{L})$ and $b_i(H\mathcal{L})$ are often linear combinations of the functions $\varphi_k(c_i H \mathcal{L})$ and $\varphi_k(H\mathcal{L})$, respectively, wherein $\varphi_k(z)$ are given by

$$(2.8) \quad \varphi_k(z) = \int_0^1 e^{(1-\theta)z} \frac{\theta^{k-1}}{(k-1)!} d\theta, \quad k \geq 1,$$

and satisfy the recurrence relations

$$(2.9) \quad \varphi_k(z) = \frac{\varphi_{k-1}(z) - \varphi_{k-1}(0)}{z}, \quad \varphi_0(z) = e^z.$$

2.2. Adopting the idea of backward error analysis. Motivated by the idea of [23, sect. 5.3], and recalling (2.1) and (2.3), we note that $u(t_{n+1})$ and $u(t_n + c_i H)$ are the exact solutions of the differential equation

$$(2.10) \quad v'(\tau) = \mathcal{L}v(\tau) + \mathcal{N}(t_n + \tau, u(t_n + \tau)), \quad v(0) = u(t_n),$$

evaluated at $\tau = H$ and $\tau = c_i H$, respectively. In other words, solving (2.10) exactly (by means of the variation-of-constants formula) on the time intervals $[0, H]$ and $[0, c_i H]$ shows that $v(H) = u(t_{n+1})$ and $v(c_i H) = u(t_n + c_i H)$. Unfortunately, explicit representations of these analytical solutions are generally impossible to find, since $u(t_n)$ and $u(t_n + \tau)$ are unknown values. This observation, however, suggests the use of backward error analysis (see, for instance, [16, Chap. IX]).

Given an exponential Runge–Kutta method (2.5), we therefore search for modified differential equations of the form (2.10), such that their exact solutions at $\tau = c_i H$ and $\tau = H$ coincide with the ExpRK approximations $U_{n,i}$ ($i = 2, \dots, s$) and u_{n+1} , respectively. We may then approximate solutions to these modified equations to compute our overall approximation of (1.1).

3. MERK methods. In this section, we construct a new multirate procedure based on approximation of ExpRK schemes; we call the resulting algorithms MERK methods. Following this derivation, we present a rigorous convergence analysis.

3.1. Construction of modified differential equations. We begin with the construction of MERK methods through definition of modified differential equations corresponding with the ExpRK stages $U_{n,i}$ ($i = 2, \dots, s$) and solution u_{n+1} .

THEOREM 3.1. *Assuming that the coefficients $a_{ij}(H\mathcal{L})$ and $b_i(H\mathcal{L})$ of an explicit exponential Runge–Kutta method (2.6) may be written as linear combinations*

$$(3.1) \quad a_{ij}(H\mathcal{L}) = \sum_{k=1}^{\ell_{ij}} \alpha_{ij}^{(k)} \varphi_k(c_i H \mathcal{L}), \quad b_i(H\mathcal{L}) = \sum_{k=1}^{m_i} \beta_i^{(k)} \varphi_k(H\mathcal{L})$$

for some positive integers ℓ_{ij} and m_i , and where the functions $\varphi_k(c_i H \mathcal{L})$ and $\varphi_k(H \mathcal{L})$ are given in (2.8), then $U_{n,i}$ and u_{n+1} are the exact solutions of the linear differential equations

$$(3.2a) \quad v'_{n,i}(\tau) = \mathcal{L}v_{n,i}(\tau) + p_{n,i}(\tau), \quad v_n(0) = u_n, \quad i = 2, \dots, s,$$

$$(3.2b) \quad v'_n(\tau) = \mathcal{L}v_n(\tau) + q_n(\tau), \quad v_n(0) = u_n$$

at the times $\tau = c_i H$ and $\tau = H$, respectively. Here $p_{n,i}(\tau)$ and $q_n(\tau)$ are polynomials in τ given by

$$(3.3a) \quad p_{n,i}(\tau) = \mathcal{N}(t_n, u_n) + \sum_{j=2}^{i-1} \left(\sum_{k=1}^{\ell_{ij}} \frac{\alpha_{ij}^{(k)}}{c_i^k H^{k-1} (k-1)!} \tau^{k-1} \right) D_{n,j},$$

$$(3.3b) \quad q_n(\tau) = \mathcal{N}(t_n, u_n) + \sum_{i=2}^s \left(\sum_{k=1}^{m_i} \frac{\beta_i^{(k)}}{H^{k-1} (k-1)!} \tau^{k-1} \right) D_{n,i}.$$

Proof. By changing the integration variable to $\tau = H\theta$ in (2.8), we obtain

$$(3.4) \quad \varphi_k(z) = \frac{1}{H^k} \int_0^H e^{(H-\tau)\frac{z}{H}} \frac{\tau^{k-1}}{(k-1)!} d\tau, \quad k \geq 1.$$

Substituting $z = c_i H \mathcal{L}$ and $z = H \mathcal{L}$ into (3.4) and inserting the obtained results for $\varphi_k(c_i H \mathcal{L})$ and $\varphi_k(H \mathcal{L})$ into (3.1) shows that

$$(3.5a) \quad a_{ij}(H \mathcal{L}) = \int_0^{c_i H} e^{(c_i H - \tau)\mathcal{L}} \sum_{k=1}^{\ell_{ij}} \frac{\alpha_{ij}^{(k)}}{(c_i H)^k (k-1)!} \tau^{k-1} d\tau,$$

$$(3.5b) \quad b_i(H \mathcal{L}) = \int_0^H e^{(H - \tau)\mathcal{L}} \sum_{k=1}^{m_i} \frac{\beta_i^{(k)}}{H^k (k-1)!} \tau^{k-1} d\tau.$$

Using the fact that $F(t_n, u_n) = \mathcal{L}u_n + \mathcal{N}(t_n, u_n)$ and

$$(3.6) \quad \varphi_1(Z) = \frac{1}{H} \int_0^H e^{(H-\tau)\frac{Z}{H}} d\tau = (e^Z - I)Z^{-1},$$

we can write (2.6) in an equivalent form:

$$(3.7a) \quad U_{n,i} = e^{c_i H \mathcal{L}} u_n + c_i H \varphi_1(c_i H \mathcal{L}) \mathcal{N}(t_n, u_n) + H \sum_{j=2}^{i-1} a_{ij}(H \mathcal{L}) D_{n,j},$$

$$(3.7b) \quad u_{n+1} = e^{H \mathcal{L}} u_n + H \varphi_1(H \mathcal{L}) \mathcal{N}(t_n, u_n) + H \sum_{i=2}^s b_i(H \mathcal{L}) D_{n,i}$$

for $i = 2, \dots, s$. We now insert the integral form of $\varphi_1(Z)$ into (3.6) (with $Z = c_i H \mathcal{L}$ and $Z = H \mathcal{L}$) and (3.5) into (3.7) to get

$$(3.8a) \quad U_{n,i} = e^{c_i H \mathcal{L}} u_n + \int_0^{c_i H} e^{(c_i H - \tau)\mathcal{L}} p_{n,i}(\tau) d\tau, \quad i = 2, \dots, s,$$

$$(3.8b) \quad u_{n+1} = e^{H \mathcal{L}} u_n + \int_0^H e^{(H - \tau)\mathcal{L}} q_n(\tau) d\tau$$

with $p_{n,i}(\tau)$ and $q_n(\tau)$ as shown in (3.3). Clearly, these representations (variation-of-constant formulas) show the conclusion of Theorem 3.1. In particular, $U_{n,i} = v_{n,i}(c_i H)$ and $u_{n+1} = v_n(H)$. Thus one can consider (3.2) as modified differential equations with identical solutions as are the ExpRK approximations to (2.10). \square

We note that the idea of using an ODE to represent a linear combination of matrix vector $\varphi_k(A)v_k$ was also used in [36].

3.2. MERK methods and a multirate algorithm. We now present an algorithm to solve the modified ODEs (3.2a) and (3.2b) numerically. For later use, we first denote the numerical solutions of (3.2a) on $[0, c_i H]$ and (3.2b) on $[0, H]$ by $\hat{U}_{n,i}$ and \hat{u}_{n+1} , respectively. This means that $\hat{U}_{n,i} \approx v_{n,i}(c_i H) = U_{n,i}$ and $\hat{u}_{n+1} \approx v_n(H) = u_{n+1}$.

Clearly, the polynomials $p_{n,i}(\tau)$ and $q_n(\tau)$ in (3.3) are not given analytically since the $D_{n,i}$ are unknowns; however, these polynomials can be numerically determined as follows. For simplicity, we illustrate our procedure by starting with $n = 0$ and $i = s = 2$. In this case we know $u_0 = u(t_0)$ and $p_{0,2}(\tau) = \mathcal{N}(t_0, u_0)$, so one can solve the ODE (3.2a) on $[0, c_2 H]$ to get an approximation to $U_{0,2}$, $\hat{U}_{0,2} \approx U_{0,2} = v_{0,2}(c_2 H)$. Then replacing the unknown $U_{0,2}$ in (3.3b) by $\hat{U}_{0,2}$, we have

$$\hat{q}_0(\tau) = \mathcal{N}(t_0, u_0) + \sum_{k=1}^{m_i} \frac{\beta_2^{(k)}}{H^{k-1}(k-1)!} \tau^{k-1} \hat{D}_{0,2},$$

where $\hat{D}_{0,2} = \mathcal{N}(t_0 + c_2 H, \hat{U}_{0,2}) - \mathcal{N}(t_0, u_0)$. Since $\hat{q}_0(\tau) \approx q_0(\tau)$, we may then solve the ODE (3.2b) on $[0, H]$ with $\hat{q}_0(\tau)$ in place of $q_0(\tau)$ to obtain an approximation $\hat{u}_1 \approx u_1 = v_0(H)$.

This general process may be extended to larger numbers of stages $s \geq 2$ and for subsequent time steps $n \geq 0$. Approximating $\hat{u}_n \approx u_n$ (with $\hat{u}_0 = u_0$), then for $i = 2, \dots, s$, we define the following perturbed linear ODEs over $\tau \in [0, c_i H]$:

$$(3.9) \quad y'_{n,i}(\tau) = \mathcal{L}y_{n,i}(\tau) + \hat{p}_{n,i}(\tau), \quad y_{n,i}(0) = \hat{u}_n,$$

with

$$(3.10) \quad \hat{p}_{n,i}(\tau) = \mathcal{N}(t_n, \hat{u}_n) + \sum_{j=2}^{i-1} \left(\sum_{k=1}^{\ell_{ij}} \frac{\alpha_{ij}^{(k)}}{c_i^k H^{k-1}(k-1)!} \tau^{k-1} \right) \hat{D}_{n,j},$$

$$(3.11) \quad \hat{D}_{n,i} = \mathcal{N}(t_n + c_i H, \hat{U}_{n,i}) - \mathcal{N}(t_n, \hat{u}_n),$$

that provide the approximations

$$\hat{U}_{n,i} \approx y_{n,i}(c_i H) \approx v_{n,i}(c_i H) = U_{n,i}.$$

With these in place, we then solve the linear ODE

$$(3.12) \quad y'_n(\tau) = \mathcal{L}y_n(\tau) + \hat{q}_n(\tau), \quad y_n(0) = \hat{u}_n$$

over $\tau \in [0, H]$ with

$$(3.13) \quad \hat{q}_n(\tau) = \mathcal{N}(t_n, \hat{u}_n) + \sum_{i=2}^s \left(\sum_{k=1}^{m_i} \frac{\beta_i^{(k)}}{H^{k-1}(k-1)!} \tau^{k-1} \right) \hat{D}_{n,i},$$

to obtain the approximate time step solutions,

$$\hat{u}_{n+1} \approx y_n(H) \approx v_n(H) = u_{n+1}.$$

Since the above procedure uses a “macro” time step H to integrate the slow process, and a “micro” time step h to integrate the fast process (via solving the ODEs (3.9) and (3.12)), we call the resulting methods (3.9)–(3.11) MERK methods. By construction, MERK methods offer several interesting features. They reduce the solution of nonlinear problems (1.1) to the solution of a sequence of linear differential equations (3.9) and (3.12), using very few evaluations of the nonlinear operator \mathcal{N} . Thus they can be more efficient for problems where the linear part is much less costly to compute than the nonlinear part. Additionally, they do not require the computation of matrix functions, as is the case with Exprk methods. Moreover, these methods do not require a starting value procedure as in multirate algorithms for exponential multistep methods [7, 23].

We provide the following Algorithm 3.1 to give a succinct overview of the implementation of our MERK methods.

Algorithm 3.1 MERK method.

- **Input:** \mathcal{L} ; $\mathcal{N}(t, u)$; t_0 ; u_0 ; s ; c_i ($i = 1, \dots, s$); H
 - **Initialization:** Set $n = 0$; $\hat{u}_n = u_0$.
 While $t_n < T$
 1. Set $\hat{U}_{n,1} = \hat{u}_n$.
 2. For $i = 2, \dots, s$ do
 - (a) Find $\hat{p}_{n,i}(\tau)$ as in (3.10).
 - (b) Solve (3.9) on $[0, c_i H]$ to obtain $\hat{U}_{n,i} \approx y_{n,i}(c_i H)$.
 3. Find $\hat{q}_{n,s}(\tau)$ as in (3.13)
 4. Solve (3.12) on $[0, H]$ to get $\hat{u}_{n+1} \approx y_n(H)$.
 5. Update $t_{n+1} := t_n + H$, $n := n + 1$.
 - **Output:** Approximate values $\hat{u}_n \approx u_n$, $n = 1, 2, \dots$ (where u_n is the numerical solution at time t_n obtained by an Exprk method).
-

3.3. Convergence analysis. Since MERK methods are constructed to approximate Exprk methods, we perform their error analysis in the framework of analytic semigroups on a Banach space X , under the following assumptions (see, e.g., [21, 31]).

Assumption 1. The linear operator \mathcal{L} is the infinitesimal generator of an analytic semigroup $e^{t\mathcal{L}}$ on X . This implies that

$$(3.14) \quad \|e^{t\mathcal{L}}\|_{X \leftarrow X} \leq C, \quad t \geq 0,$$

and consequently $\varphi_k(H\mathcal{L})$, $a_{ij}(H\mathcal{L})$ and $b_i(H\mathcal{L})$ are bounded operators.

Assumption 2 (for high-order methods). The solution $u : [t_0, T] \rightarrow X$ of (1.1) is sufficiently smooth with derivatives in X , and $\mathcal{N} : [t_0, T] \times X \rightarrow X$ is sufficiently Fréchet differentiable in a strip along the exact solution. All derivatives occurring in the remainder of this section are therefore assumed to be uniformly bounded.

We analyze the error in MERK methods starting with the local error of Exprk methods. Therefore, we first consider (2.5) (in its explicit form) with exact initial

value $u(t_n)$:

$$(3.15a) \quad \check{U}_{n,i} = e^{c_i H \mathcal{L}} u(t_n) + H \sum_{j=1}^{i-1} a_{ij}(H \mathcal{L}) \mathcal{N}(t_n + c_j H, \check{U}_{n,j}), \quad i = 2, \dots, s,$$

$$(3.15b) \quad \check{u}_{n+1} = e^{H \mathcal{L}} u(t_n) + H \sum_{i=1}^s b_i(H \mathcal{L}) \mathcal{N}(t_n + c_i H, \check{U}_{n,i}),$$

and thus the MERK methods (3.2)–(3.3) are considered with polynomials

$$(3.16a) \quad \check{p}_{n,i}(\tau) = \mathcal{N}(t_n, u(t_n)) + \sum_{j=2}^{i-1} \left(\sum_{k=1}^{\ell_{ij}} \frac{\alpha_{ij}^{(k)}}{c_i^k H^{k-1} (k-1)!} \tau^{k-1} \right) \check{D}_{n,j},$$

$$(3.16b) \quad \check{q}_n(\tau) = \mathcal{N}(t_n, u(t_n)) + \sum_{i=2}^s \left(\sum_{k=1}^{m_i} \frac{\beta_i^{(k)}}{H^{k-1} (k-1)!} \tau^{k-1} \right) \check{D}_{n,i},$$

where $\check{D}_{n,i} = \mathcal{N}(t_n + c_i H, \check{U}_{n,i}) - \mathcal{N}(t_n, u(t_n))$.

Error notation. Since MERK methods consist of approximations to approximations, we must clearly isolate the errors induced at each approximation level. To this end, we let $\hat{e}_{n+1} = \hat{u}_{n+1} - u(t_{n+1})$ denote the global error at time t_{n+1} of a MERK method (3.9)–(3.11). Let $\check{e}_{n+1} = \check{u}_{n+1} - u(t_{n+1})$ denote the local error at t_{n+1} of the base ExpRK method. Let $\hat{\varepsilon}_{n,i} = \hat{U}_{n,i} - y_{n,i}(c_i H)$ and $\hat{e}_{n+1} = \hat{u}_{n+1} - y_n(H)$ denote the (global) errors of the ODE solvers when integrating (3.9) on $[0, c_i H]$ and (3.12) on $[0, H]$ (note that $\hat{\varepsilon}_{n,1} = \hat{u}_n - y_{n,i}(0) = 0$ since $c_1 = 0$).

First, we may write total error as the sum of the errors in each approximation:

$$(3.17) \quad \hat{e}_{n+1} = \hat{e}_{n+1} + (y_n(H) - \check{u}_{n+1}) + \check{e}_{n+1}.$$

Applying the variation-of-constants formula to (3.13) and using (3.5b), we then write

$$(3.18) \quad y_n(H) = e^{H \mathcal{L}} \hat{u}_n + H \sum_{i=1}^s b_i(H \mathcal{L}) \mathcal{N}(t_n + c_i H, \hat{U}_{n,i}).$$

Inserting $y_n(H)$ and \check{u}_{n+1} from (3.18) and (3.15b) into (3.17) gives

$$(3.19) \quad \hat{e}_{n+1} = e^{H \mathcal{L}} \hat{e}_n + \hat{e}_{n+1} + H \mathcal{S}_{n,s} + \check{e}_{n+1},$$

where

$$(3.20) \quad \mathcal{S}_{n,s} = \sum_{i=1}^s b_i(H \mathcal{L}) (\mathcal{N}(t_n + c_i H, \hat{U}_{n,i}) - \mathcal{N}(t_n + c_i H, \check{U}_{n,i})).$$

Next, we prove some preliminary results.

LEMMA 3.2. Denoting $\widehat{E}_{n,i} = \widehat{U}_{n,i} - \check{U}_{n,i}$ and $\check{N}_{n,i} = \frac{\partial \mathcal{N}}{\partial u}(t_n + c_i H, \check{U}_{n,i})$, we have

$$(3.21) \quad \widehat{E}_{n,i} = \widehat{\varepsilon}_{n,i} + e^{c_i H \mathcal{L}} \widehat{e}_n + H \sum_{j=1}^{i-1} a_{ij}(H \mathcal{L})(\check{N}_{n,j} \widehat{E}_{n,j} + \widehat{R}_{n,j})$$

with

$$(3.22) \quad \widehat{R}_{n,j} = \int_0^1 (1-\theta) \frac{\partial^2 \mathcal{N}}{\partial u^2}(t_n + c_j H, \check{U}_{n,j} + \theta \widehat{E}_{n,j})(\widehat{E}_{n,j}, \widehat{E}_{n,j}) d\theta.$$

Furthermore, under Assumption 2, the bound

$$(3.23) \quad \|\widehat{R}_{n,j}\| \leq C \|\widehat{E}_{n,j}\|^2, \text{ i.e., } \widehat{R}_{n,j} = \mathcal{O}(\|\widehat{E}_{n,j}\|^2)$$

is held as long as $\widehat{E}_{n,j}$ remains in a sufficiently small neighborhood of 0.

Proof. We first rewrite

$$(3.24) \quad \widehat{E}_{n,i} = \widehat{U}_{n,i} - y_{n,i}(c_i H) + (y_{n,i}(c_i H) - \check{U}_{n,i}) = \widehat{\varepsilon}_{n,i} + (y_{n,i}(c_i H) - \check{U}_{n,i}).$$

Here $y_{n,i}(c_i H)$ is the exact solution of (3.9), which can be represented by the variation-of-constants formula and then rewritten by using (3.5a) and (3.10) as

$$(3.25) \quad y_{n,i}(c_i H) = e^{c_i H \mathcal{L}} \widehat{u}_n + H \sum_{j=1}^{i-1} a_{ij}(H \mathcal{L}) \mathcal{N}(t_n + c_j H, \widehat{U}_{n,j}).$$

Subtracting (3.15a) from (3.25) and inserting the obtained result into (3.24) gives

$$(3.26) \quad \widehat{E}_{n,i} = \widehat{\varepsilon}_{n,i} + e^{c_i H \mathcal{L}} \widehat{e}_n + H \sum_{j=1}^{i-1} a_{ij}(H \mathcal{L})(\mathcal{N}(t_n + c_j H, \widehat{U}_{n,j}) - \mathcal{N}(t_n + c_j H, \check{U}_{n,j})).$$

Using the Taylor series expansion of $\mathcal{N}(t, u)$ at $(t_n + c_j H, \check{U}_{n,j})$, we get

$$(3.27) \quad \mathcal{N}(t_n + c_j H, \widehat{U}_{n,j}) - \mathcal{N}(t_n + c_j H, \check{U}_{n,j}) = \check{N}_{n,j} \widehat{E}_{n,j} + \widehat{R}_{n,j}$$

with the remainder $\widehat{R}_{n,j}$ given in (3.22), which clearly satisfies (3.23) due to Assumption 2. Inserting (3.27) into (3.26) shows (3.21). \square

LEMMA 3.3. Under Assumptions 1 and 2, there exist bounded operators $\mathcal{T}_{n,i}(\widehat{\varepsilon}_{n,i})$ and $\mathcal{B}_n(\widehat{e}_n)$ on X such that

$$(3.28) \quad \mathcal{S}_{n,s} = \sum_{i=2}^s (b_i(H \mathcal{L}) \check{N}_{n,i} + H \mathcal{T}_{n,i}(\widehat{\varepsilon}_{n,i})) \widehat{\varepsilon}_{n,i} + \mathcal{B}_n(\widehat{e}_n) \widehat{e}_n.$$

Note that $\mathcal{T}_{n,i}$ also depends on H , $\widehat{\varepsilon}_{n,j}$, $a_{ij}(H \mathcal{L})$, $\check{N}_{n,j}$ ($j = 2, \dots, i-1$), and \widehat{e}_n ; and \mathcal{B}_n also depends on H , $b_i(H \mathcal{L})$, $a_{ij}(H \mathcal{L})$, c_i , and $\check{N}_{n,i}$.

Proof. Inserting (3.27) (with i in place of j) into (3.20) gives

$$(3.29) \quad \mathcal{S}_{n,s} = \sum_{i=1}^s b_i(H \mathcal{L})(\check{N}_{n,i} \widehat{E}_{n,i} + \widehat{R}_{n,i}).$$

Using the recursion (3.21) from Lemma 3.2, we further expand $\widehat{E}_{n,i}$ as
(3.30)

$$\begin{aligned}\widehat{E}_{n,i} &= \widehat{e}_{n,i} + H \sum_{j=1}^{i-1} a_{ij}(H\mathcal{L})\check{N}_{n,j}\widehat{e}_{n,j} + H^2 \sum_{j=1}^{i-1} a_{ij}(H\mathcal{L})\check{N}_{n,j} \sum_{k=1}^{j-1} a_{jk}(H\mathcal{L})\check{N}_{n,k}\widehat{E}_{n,k} \\ &\quad + H \sum_{j=1}^{i-1} a_{ij}(H\mathcal{L})\widehat{R}_{n,j} + H^2 \sum_{j=1}^{i-1} a_{ij}(H\mathcal{L})\check{N}_{n,j} \sum_{k=1}^{j-1} a_{jk}(H\mathcal{L})\widehat{R}_{n,k} \\ &\quad + \left(e^{c_i H\mathcal{L}} + H \sum_{j=1}^{i-1} a_{ij}(H\mathcal{L})\check{N}_{n,j} e^{c_j H\mathcal{L}} \right) \widehat{e}_n.\end{aligned}$$

Using (3.21) and (3.23) ($\widehat{R}_{n,i} = \mathcal{O}(\|\widehat{E}_{n,i}\|^2)$) and proceeding by induction, one can complete the recursion (3.30) for $\widehat{E}_{n,i}$. Inserting this recursion into (3.29) (and noting that $\widehat{e}_{n,1} = 0$) yields (3.28). Based on the structure of (3.30) and (3.29), under the given assumptions it is clear that the boundedness of $\mathcal{T}_{n,i}(\widehat{e}_{n,i})$ and $\mathcal{B}_n(\widehat{e}_n)$ follow from the boundedness of $a_{ij}(H\mathcal{L})$, $b_i(H\mathcal{L})$, $\check{N}_{n,i}$, and $\widehat{R}_{n,i}$. \square

We now present the main convergence result for MERK methods.

THEOREM 3.4. *Let the IVP (1.1) satisfy Assumptions 1–2. Consider for its numerical solution a MERK method (3.9)–(3.11) that is constructed from an ExpRK method of global order p . We further assume that the fast ODEs (3.9) and (3.12) associated with the MERK method are integrated with micro time step $h = H/m$ by using ODE solvers that have global order of convergence q and r , respectively, and where m is the number of fast steps per slow step. Then, the MERK method is convergent, and has error bound*

$$(3.31) \quad \|u_n - u(t_n)\| \leq C_1 H^p + C_2 H h^q + C_3 h^r$$

on compact time intervals $t_0 \leq t_n = t_0 + nH \leq T$. Here, the constant C_1 depends on $T - t_0$, but is independent of n and H ; and the constants C_2 and C_3 also depend on the error constants of the chosen ODE solvers.

Proof. We first note that since we only employ the fast ODE solvers on time intervals $[0, c_i H]$ and $[0, H]$, then our assumption regarding their accuracies of order q and r is typically equivalent to [17, Thm. 3.6]

$$(3.32a) \quad \widehat{e}_{n,i} = \frac{\tilde{c}_2}{\Lambda_i} h^q (e^{\Lambda_i c_i H} - 1) = \tilde{c}_2 c_i \varphi_1(\Lambda_i c_i H) H h^q = c_2 H h^q,$$

$$(3.32b) \quad \widehat{e}_n = \frac{\tilde{c}_3}{\Lambda} h^r (e^{\Lambda H} - 1) = \tilde{c}_3 \varphi_1(\Lambda H) H h^r = c_3 H h^r$$

(due to (3.6)), where Λ_i, Λ are the Lipschitz constants for the increment functions of the ODE solvers applied to the problems (3.9) and (3.12), respectively.

For simplicity of notation, we denote $B_{n,i} = b_i(H\mathcal{L})\check{N}_{n,i} + H\mathcal{T}_{n,i}(\widehat{e}_{n,i})$. Clearly, $B_{n,i}$ is a bounded operator and thus (3.28) becomes

$$(3.33) \quad \mathcal{S}_{n,s} = \sum_{i=2}^s B_{n,i} \widehat{e}_{n,i} + \mathcal{B}_n(\widehat{e}_n) \widehat{e}_n.$$

Inserting this into (3.19) gives

$$(3.34) \quad \widehat{e}_{n+1} = e^{H\mathcal{L}} \widehat{e}_n + H \mathcal{B}_n(\widehat{e}_n) \widehat{e}_n + \check{e}_{n+1} + H \left(\sum_{i=2}^s B_{n,i} \widehat{e}_{n,i} + \widehat{e}_{n+1} \right).$$

Solving recursion (3.34) and using $\hat{e}_0 = 0$ (since $\hat{u}_0 = u_0 = u(t_0)$) finally yields

$$(3.35) \quad \hat{e}_n = H \sum_{j=0}^{n-1} e^{(n-1-j)H\mathcal{L}} \mathcal{B}_j(\hat{e}_j) \hat{e}_j + \sum_{j=0}^{n-1} e^{jH\mathcal{L}} \left(\check{e}_{n-j} + H \sum_{i=2}^s B_{n-1-j,i} \hat{e}_{n-1-j,i} + \hat{e}_{n-j} \right).$$

Since the ExpRK method has global order p , we have the local error $\check{e}_{n-j} = \mathcal{O}(H^{p+1})$, and from (3.32) we have $\hat{e}_{n-1-j,i} = \mathcal{O}(Hh^q)$, $\hat{e}_{n-j} = \mathcal{O}(Hh^r)$. Using (3.14) we derive from (3.35) that

$$(3.36) \quad \|\hat{e}_n\| \leq H \sum_{j=0}^{n-1} C \|\hat{e}_j\| + \sum_{j=0}^{n-1} C (c_1 H^{p+1} + c_2 H^2 h^q + c_3 H h^r).$$

An application of a discrete Gronwall lemma to (3.36) results in the bound (3.31). \square

Remark 3.5. Under the assumption of a fixed time-scale separation factor, $m = H/h$, Theorem 3.4 implies that for a MERK method (3.9)–(3.11) to converge with order p , the inner ODE solvers for (3.9) and (3.12) must have orders $q \geq p-1$ and $r \geq p$, respectively.

Remark 3.6. Alternately, Theorem 3.4 shows that MERK methods may in fact use a method of *any* order to integrate the modified fast ODEs, as long as m is adjusted accordingly. Specifically, if methods of order q and r are used to solve the modified fast ODEs (3.9) and (3.12), then an overall order p MERK method may be retained by use of $m \geq C_1 H^{(q-p+1)/q}$ for (3.9) and $m \geq C_2 H^{(r-p)/r}$ for (3.12) for appropriate constants C_1 and C_2 .

4. Derivation of MERK methods. Based on the theory presented in section 3, we now derive MERK schemes up to order 5, relying heavily on ExpRK schemes that fit the assumption of Theorem 3.1. As we are interested in problems with significant time-scale separation $H \gg h$, we primarily focus on stiffly accurate ExpRK schemes. Since MERK methods involve linear ODEs (3.9) and (3.12) with a fixed coefficient matrix \mathcal{L} for the fast portion, they are characterized by the polynomials defined in (3.10) and (3.13). Therefore, when deriving MERK schemes we display only their corresponding polynomials $\hat{p}_{n,i}(\tau)$ and $\hat{q}_n(\tau)$.

4.1. Second-order methods. When searching for stiffly accurate second-order ExpRK methods, we find the following scheme that uses $s = 2$ stages (see [20, sect. 5.1]) and satisfies Theorem 3.1:

$$(4.1) \quad \begin{aligned} U_{n,2} &= u_n + c_2 H \varphi_1(c_2 H \mathcal{L}) F(t_n, u_n), \\ u_{n+1} &= u_n + H \varphi_1(H \mathcal{L}) F(t_n, u_n) + h \frac{1}{c_2} \varphi_2(H \mathcal{L}) D_{n,2}. \end{aligned}$$

From this, using the conclusion of Theorem 3.1, we derive the corresponding family of second-order MERK methods, which we call MERK2:

$$(4.2) \quad \begin{aligned} \hat{p}_{n,2}(\tau) &= \mathcal{N}(t_n, \hat{u}_n), & \tau \in [0, c_2 H], \\ \hat{q}_n(\tau) &= \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{c_2 H} \hat{D}_{n,2}, & \tau \in [0, H]. \end{aligned}$$

Since we do not use this scheme in our numerical experiments, we do not specify a

value for c_2 . We note that for these methods, the fast time scale must be evolved to a duration of $(1 + c_2)H$ for each slow time step.

4.2. Third-order methods. Also from [20, sect. 5.2] we consider the following family of third-order, 3-stage, ExpRK methods that satisfy Theorem 3.1:

$$(4.3) \quad \begin{aligned} U_{n,2} &= u_n + c_2 H \varphi_1(c_2 H \mathcal{L}) F(t_n, u_n), \\ U_{n,3} &= u_n + \frac{2}{3} H \varphi_1(\frac{2}{3} H \mathcal{L}) F(t_n, u_n) + \frac{4}{9c_2} \varphi_2(\frac{2}{3} H \mathcal{L}) D_{n,2}, \\ u_{n+1} &= u_n + H \varphi_1(H \mathcal{L}) F(t_n, u_n) + h \frac{3}{2} \varphi_2(H \mathcal{L}) D_{n,3}. \end{aligned}$$

From these, we construct the following third-order MERK3 scheme:

$$(4.4) \quad \begin{aligned} \hat{p}_{n2}(\tau) &= \mathcal{N}(t_n, \hat{u}_n), & \tau \in [0, c_2 H], \\ \hat{p}_{n3}(\tau) &= \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{c_2 H} \hat{D}_{n,2}, & \tau \in [0, \frac{2}{3} H], \\ \hat{q}_n(\tau) &= \mathcal{N}(t_n, \hat{u}_n) + \frac{3\tau}{2H} \hat{D}_{n,3}, & \tau \in [0, H]. \end{aligned}$$

In our numerical experiments with this scheme, we choose $c_2 = \frac{1}{2}$. Hence, the fast time scale must be evolved to a duration of $\frac{13}{6}H$ for each slow time step.

4.3. Fourth-order methods. To the best of our knowledge, the only 5-stage, stiffly accurate ExpRK method of order four was given in [20, sect. 5.3]. However, this scheme does not satisfy Theorem 3.1 due to the coefficient

$$a_{52}(H\mathcal{L}) = \frac{1}{2} \varphi_2(c_5 H \mathcal{L}) - \varphi_3(c_4 H \mathcal{L}) + \frac{1}{4} \varphi_2(c_4 H \mathcal{L}) - \frac{1}{2} \varphi_3(c_5 H \mathcal{L}),$$

which is not a linear combination of $\{\varphi_k(c_5 H \mathcal{L})\}_{k=1}^5$. Therefore, we cannot use it to derive a fourth-order MERK scheme. However, in a very recently submitted paper [29], we have derived a family of fourth-order, 6-stage, stiffly accurate ExpRK methods (named **expRK4s6**), that additionally fulfill Theorem 3.1:

$$(4.5) \quad \begin{aligned} U_{n,2} &= u_n + \varphi_1(c_2 H \mathcal{L}) c_2 H F(t_n, u_n), \\ U_{n,k} &= u_n + \varphi_1(c_k H \mathcal{L}) c_k H F(t_n, u_n) + \varphi_2(c_k H \mathcal{L}) \frac{c_k^2}{c_2} H D_{n,2}, & k = 3, 4, \\ U_{n,j} &= u_n + \varphi_1(c_j H \mathcal{L}) c_j H F(t_n, u_n) + \varphi_2(c_j H \mathcal{L}) \frac{c_j^2}{c_3 - c_4} H \left(\frac{-c_4}{c_3} D_{n,3} + \frac{c_3}{c_4} D_{n,4} \right) \\ &\quad + \varphi_3(c_j H \mathcal{L}) \frac{2c_j^3}{c_3 - c_4} H \left(\frac{1}{c_3} D_{n,3} - \frac{1}{c_4} D_{n,4} \right), & j = 5, 6, \\ u_{n+1} &= u_n + \varphi_1(H \mathcal{L}) h F(t_n, u_n) + \varphi_2(H \mathcal{L}) \frac{1}{c_5 - c_6} H \left(\frac{-c_6}{c_5} D_{n,5} + \frac{c_5}{c_6} D_{n,6} \right) \\ &\quad + \varphi_3(H \mathcal{L}) \frac{2}{c_5 - c_6} H \left(\frac{1}{c_5} D_{n,5} - \frac{1}{c_6} D_{n,6} \right). \end{aligned}$$

Since the pairs of internal stages $\{U_{n,3}, U_{n,4}\}$ and $\{U_{n,5}, U_{n,6}\}$ are independent of one another (they can be computed simultaneously) and have the same format, this scheme behaves like a 4-stage method. Hence, instead of using 6 polynomials we need only 4 to derive the following family of fourth-order MERK schemes, which we call **MERK4**:

$$(4.6) \quad \begin{aligned} \hat{p}_{n,2}(\tau) &= \mathcal{N}(t_n, \hat{u}_n), & \tau \in [0, c_2 H], \\ \hat{p}_{n,3}(\tau) &= \hat{p}_{n,4}(\tau) = \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{c_2 H} \hat{D}_{n,2}, & \tau \in [0, c_3 H], \\ \hat{p}_{n,5}(\tau) &= \hat{p}_{n,6}(\tau) = \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{H} \left(\frac{-c_4}{c_3(c_3 - c_4)} \hat{D}_{n,3} + \frac{c_3}{c_4(c_3 - c_4)} \hat{D}_{n,4} \right) \\ &\quad + \frac{\tau^2}{H^2} \left(\frac{1}{c_3(c_3 - c_4)} \hat{D}_{n,3} - \frac{1}{c_4(c_3 - c_4)} \hat{D}_{n,4} \right), & \tau \in [0, c_5 H], \\ \hat{q}_n(\tau) &= \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{H} \left(\frac{-c_6}{c_5(c_5 - c_6)} \hat{D}_{n,5} + \frac{c_5}{c_6(c_5 - c_6)} \hat{D}_{n,6} \right) \\ &\quad + \frac{\tau^2}{H^2} \left(\frac{1}{c_5(c_5 - c_6)} \hat{D}_{n,5} - \frac{1}{c_6(c_5 - c_6)} \hat{D}_{n,6} \right), & \tau \in [0, H]. \end{aligned}$$

For our numerical experiments, we choose the coefficients $c_2 = c_3 = \frac{1}{2}$, $c_4 = c_6 = \frac{1}{3}$, and $c_5 = \frac{5}{6}$. With this choice, we may then solve the linear ODE (3.9) using the polynomial $\hat{p}_{n,3}(\tau)$ on $[0, c_3H]$ to get both $\hat{U}_{n,3} \approx U_{n,3} = v_{n,3}(c_3H)$ and $\hat{U}_{n,4} \approx U_{n,4}$ (since $c_4 < c_3$) without solving an additional fast differential equation on $[0, c_4H]$. Similarly, we may solve the linear ODE (3.9) with the polynomial $\hat{p}_{n,5}(\tau)$ on $[0, c_5H]$ to obtain both $\hat{U}_{n,5} \approx U_{n,5}$ and $\hat{U}_{n,6} \approx U_{n,6}$. As a result, the fast time scale must only be evolved for a total duration of $\frac{17}{6}H$ for each slow time step.

4.4. Fifth-order methods. Similarly to fourth-order ExpRK methods, there are no stiffly accurate fifth-order methods available in the literature that fulfill Theorem 3.1. In particular, the only existing fifth-order scheme (**expRK5s8**, that requires 8 stages) was constructed in [31]. However, its coefficients $a_{75}(H\mathcal{L})$, $a_{76}(H\mathcal{L})$, $a_{85}(H\mathcal{L})$, $a_{86}(H\mathcal{L})$, and $a_{87}(H\mathcal{L})$ involve several different linear combinations of $\varphi_k(c_iH\mathcal{L})$ with different scalings c_6, c_7, c_8 , and may not be used to create a MERK method. Again, in [29], we have constructed a new family of efficient, fifth-order, 10-stage, stiffly accurate ExpRK methods (called **expRK5s10**) that fulfills Theorem 3.1:

$$\begin{aligned}
 (4.7a) \quad U_{n,2} &= u_n + \varphi_1(c_2H\mathcal{L})c_2HF(t_n, u_n), \\
 U_{n,k} &= u_n + \varphi_1(c_kH\mathcal{L})c_kHF(t_n, u_n) + \varphi_2(c_kH\mathcal{L})\frac{c_k^2}{c_2}HD_{n,2}, \quad k = 3, 4, \\
 U_{n,j} &= u_n + \varphi_1(c_jH\mathcal{L})c_jHF(t_n, u_n) + \varphi_2(c_jH\mathcal{L})c_j^2H(\alpha_3D_{n,3} + \alpha_4D_{n,4}) \\
 &\quad + \varphi_3(c_jH\mathcal{L})c_j^3H(\beta_3D_{n,3} - \beta_4D_{n,4}), \quad j = 5, 6, 7, \\
 (4.7b) \quad U_{n,m} &= u_n + \varphi_1(c_mH\mathcal{L})c_mHF(t_n, u_n) \\
 &\quad + \varphi_2(c_mH\mathcal{L})c_m^2H(\alpha_5D_{n,5} + \alpha_6D_{n,6} + \alpha_7D_{n,7}) \\
 &\quad + \varphi_3(c_mH\mathcal{L})c_m^3H(\beta_5D_{n,5} - \beta_6D_{n,6} - \beta_7D_{n,7}) \\
 &\quad + \varphi_4(c_mH\mathcal{L})c_m^4H(\gamma_5D_{n,5} + \gamma_6D_{n,6} + \gamma_7D_{n,7}), \quad m = 8, 9, 10, \\
 u_{n+1} &= u_n + \varphi_1(H\mathcal{L})HF(t_n, u_n) + \varphi_2(H\mathcal{L})H(\alpha_8D_{n,8} + \alpha_9D_{n,9} + \alpha_{10}D_{n,10}) \\
 &\quad - \varphi_3(H\mathcal{L})H(\beta_8D_{n,8} + \beta_9D_{n,9} + \beta_{10}D_{n,10}) \\
 &\quad + \varphi_4(H\mathcal{L})H(\gamma_8D_{n,8} + \gamma_9D_{n,9} + \gamma_{10}D_{n,10})
 \end{aligned}$$

with coefficients given by

$$\begin{aligned}
 (4.7c) \quad \alpha_3 &= \frac{c_4}{c_3(c_4 - c_3)}, \quad \alpha_4 = \frac{c_3}{c_4(c_3 - c_4)}, \\
 \alpha_5 &= \frac{c_6c_7}{c_5(c_5 - c_6)(c_5 - c_7)}, \quad \alpha_6 = \frac{c_5c_7}{c_6(c_6 - c_5)(c_6 - c_7)}, \quad \alpha_7 = \frac{c_5c_6}{c_7(c_7 - c_5)(c_7 - c_6)}, \\
 \alpha_8 &= \frac{c_9c_{10}}{c_8(c_8 - c_9)(c_8 - c_{10})}, \quad \alpha_9 = \frac{c_8c_{10}}{c_9(c_9 - c_8)(c_9 - c_{10})}, \quad \alpha_{10} = \frac{c_8c_9}{c_{10}(c_{10} - c_8)(c_{10} - c_9)}, \\
 \beta_3 &= \frac{2}{c_3(c_3 - c_4)}, \quad \beta_4 = \frac{2}{c_4(c_3 - c_4)}, \\
 \beta_5 &= \frac{2(c_6 + c_7)}{c_5(c_5 - c_6)(c_5 - c_7)}, \quad \beta_6 = \frac{2(c_5 + c_7)}{c_6(c_6 - c_5)(c_6 - c_7)}, \quad \beta_7 = \frac{2(c_5 + c_6)}{c_7(c_7 - c_5)(c_7 - c_6)}, \\
 \beta_8 &= \frac{2(c_9 + c_{10})}{c_8(c_8 - c_9)(c_8 - c_{10})}, \quad \beta_9 = \frac{2(c_8 + c_{10})}{c_9(c_9 - c_8)(c_9 - c_{10})}, \quad \beta_{10} = \frac{2(c_8 + c_9)}{c_{10}(c_{10} - c_8)(c_{10} - c_9)}, \\
 \gamma_5 &= \frac{6}{c_5(c_5 - c_6)(c_5 - c_7)}, \quad \gamma_6 = \frac{6}{c_6(c_6 - c_5)(c_6 - c_7)}, \quad \gamma_7 = \frac{6}{c_7(c_7 - c_5)(c_7 - c_6)}, \\
 \gamma_8 &= \frac{6}{c_8(c_8 - c_9)(c_8 - c_{10})}, \quad \gamma_9 = \frac{6}{c_9(c_9 - c_8)(c_9 - c_{10})}, \quad \gamma_{10} = \frac{6}{c_{10}(c_{10} - c_8)(c_{10} - c_9)}.
 \end{aligned}$$

Although this scheme has 10 stages, again its structure facilitates an efficient implementation. Specifically, we note that there are multiple stages $U_{n,i}$ which share

the same format (same matrix functions with different inputs c_i), and are independent of one another (namely, $\{U_{n,3}, U_{n,4}\}$, $\{U_{n,5}, U_{n,6}, U_{n,7}\}$, and $\{U_{n,8}, U_{n,9}, U_{n,10}\}$). These groups of stages can again be computed simultaneously, allowing the scheme to behave like a 5-stage method. We therefore propose the corresponding fifth-order MERK methods that use only 5 polynomials, which we name **MERK5**:

$$\begin{aligned}
 \hat{p}_{n,2}(\tau) &= \mathcal{N}(t_n, \hat{u}_n), & \tau \in [0, c_2 H], \\
 \hat{p}_{n,3}(\tau) &= \hat{p}_{n,4}(\tau) = \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{c_2 H} \hat{D}_{n,2}, & \tau \in [0, c_3 H], \\
 \hat{p}_{n,5}(\tau) &= \hat{p}_{n,6}(\tau) = \hat{p}_{n,7}(\tau) = \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{H} (\alpha_3 \hat{D}_{n,3} + \alpha_4 \hat{D}_{n,4}) \\
 &\quad + \frac{\tau^2}{2H^2} (\beta_3 \hat{D}_{n,3} - \beta_3 \hat{D}_{n,4}), & \tau \in [0, c_5 H], \\
 \hat{p}_{n,8}(\tau) &= \hat{p}_{n,9}(\tau) = \hat{p}_{n,10}(\tau) = \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{H} (\alpha_5 \hat{D}_{n,5} + \alpha_6 \hat{D}_{n,6} + \alpha_7 \hat{D}_{n,7}) \\
 &\quad - \frac{\tau^2}{2H^2} (\beta_5 \hat{D}_{n,5} + \beta_6 \hat{D}_{n,6} + \beta_7 \hat{D}_{n,7}) \\
 &\quad + \frac{\tau^3}{6H^3} (\gamma_5 \hat{D}_{n,5} + \gamma_6 \hat{D}_{n,6} + \gamma_7 \hat{D}_{n,7}), & \tau \in [0, c_8 H], \\
 \hat{q}_n(\tau) &= \mathcal{N}(t_n, \hat{u}_n) + \frac{\tau}{H} (\alpha_8 \hat{D}_{n,8} + \alpha_9 \hat{D}_{n,9} + \alpha_{10} \hat{D}_{n,10}) \\
 &\quad - \frac{\tau^2}{2H^2} (\beta_8 \hat{D}_{n,8} + \beta_9 \hat{D}_{n,9} + \beta_{10} \hat{D}_{n,10}) \\
 &\quad + \frac{\tau^3}{6H^3} (\gamma_8 \hat{D}_{n,8} + \gamma_9 \hat{D}_{n,9} + \gamma_{10} \hat{D}_{n,10}), & \tau \in [0, H].
 \end{aligned}
 \tag{4.8}$$

For our numerical experiments, we choose $c_2 = c_3 = c_5 = c_9 = \frac{1}{2}$, $c_4 = c_6 = \frac{1}{3}$, $c_7 = \frac{1}{4}$, $c_8 = \frac{7}{10}$, and $c_{10} = \frac{2}{3}$. Again, since $c_4 < c_3$, when solving the fast time-scale problem (3.9) with polynomial $\hat{p}_{n,3}(\tau)$ on $[0, c_3 H]$ gives $\hat{U}_{n,3} \approx U_{n,3} = v_{n,3}(c_3 h)$ and $\hat{U}_{n,4} \approx U_{n,4} = v_{n,3}(c_4 h)$. Similarly, since $c_7 < c_6 < c_5$, $\hat{U}_{n,5}$, $\hat{U}_{n,6}$, and $\hat{U}_{n,7}$ can be obtained by solving a single fast time-scale problem with polynomial $\hat{p}_{n,5}(\tau)$ on $[0, c_5 H]$. Finally since $c_9 < c_{10} < c_8$, one can compute $\hat{U}_{n,8}$, $\hat{U}_{n,9}$, and $\hat{U}_{n,10}$ by solving a single fast time-scale problem with polynomial $\hat{p}_{n,8}(\tau)$ on $[0, c_8 H]$. The sum total of these solves corresponds to evolving the fast time scale for an overall duration of $\frac{16}{5}H$ for each slow time step.

5. Numerical experiments. In this section we present results from a variety of numerical tests to examine the performance of the proposed **MERK3**, **MERK4**, and **MERK5** methods. These tests are designed to confirm the theoretical convergence rates from section 3.3, and compare efficiency against the MIS method **MIS-KW3**, which uses a similar approach of evolving the fast component using modified systems of differential equations [26, 47, 43, 45]. Unless otherwise noted, we run these methods with inner explicit Runge–Kutta ODE solvers of the same order of convergence as the MERK method, p :

- Third-order **MIS-KW3** uses the **Knoth–Wolke–ERK** inner method [26].
- Third-order **MERK3** uses the **ERK-3-3** inner method,

$$\begin{array}{c|ccc}
 0 & & & \\
 1/2 & 1/2 & & \\
 1 & -1 & 2 & \\
 \hline
 & 1/6 & 2/3 & 1/6
 \end{array}
 .$$

- Fourth-order **MERK4** uses the **ERK-4-4** inner method [17, Table 1.2, left].
- Fifth-order **MERK5** uses the **Cash–Karp–ERK** inner method [2].

We note that although Theorem 3.4 guarantees that when using a MERK method of order p , the internal stage solutions (3.9) can be computed with a solver of order $q = p - 1$ and the step solution (3.12) can use a solver of order $r = p$, for simplicity we

have used $r = q = p$ in the majority of our tests. However, we more closely investigate these inner solver order requirements in section 5.3 below.

Not all of our test problems have convenient analytical solutions; for these tests, we compute a reference solution using an eighth-order explicit or a twelfth-order implicit Runge–Kutta method with a time step smaller than the smallest micro time step h . When computing solution error, we report the maximum absolute error over all time steps and solution components. From these, we compute convergence rates using a linear least-squares fit of the log-error versus log-macro time step H . For each test we present three types of plots: one convergence plot (error versus H) and two efficiency plots. Generally, efficiency plots present error versus the computational cost. However in the multirate context, fast and slow function costs can differ dramatically. As such, we separately consider efficiency using total function calls and slow function calls. Since the dominant number of total calls are from the fast function, the “total” plots represent the method efficiency for simulations with comparable fast/slow function cost, whereas the “slow-only” plots represent the method efficiency for simulations in which the slow function calls are significantly more expensive (as explained in section 1 as our original motivation for multirate methods). Individual applications will obviously lie somewhere between these extremes, but we assume that they are typically closer to the slow-only results.

Applications scientists traditionally use multirate solvers for one of two reasons. The first category is concerned with simulations of stiff systems, but where they choose to use a subcycled explicit method instead of an implicit one for the stiff portion of the problem. Generally, these applications are primarily concerned with selecting h to satisfy stability of the fast time scale (instead of accuracy). The second category considers simulations wherein it is essential to capture the coupling between the slow and fast times scales accurately, since temporal errors at the fast time scale can significantly deteriorate the slow time scale solution; here h is chosen based on accuracy considerations. We therefore separately explore test problems in both of these categories in sections 5.1 and 5.2 below.

To facilitate reproducibility of the results in this section, we have provided an open-source MATLAB implementation of the MERK3, MERK4, MERK5, and MIS-KW3 methods, along with scripts to perform all tests from this section [3].

5.1. Category I. As this category of problems is concerned with stability at the fast time scale, we choose a fixed, linearly stable micro time step h , and vary the macro time step H (and similarly, $m = H/h$). To this end, we focus on two stiff applications: a reaction diffusion problem and a stiff version of the brusselator problem.

5.1.1. Reaction diffusion. We consider a reaction diffusion problem with a traveling wave solution similar to the one considered by Savcenko, Hundsdorfer, and Verwer [42]:

$$\begin{aligned} u_t &= \frac{1}{100} u_{xx} + u^2(1 - u), & 0 < x < 5, \quad 0 < t \leq 3, \\ u_x(0, t) &= u_x(5, t) = 0, & u(x, 0) &= (1 + e^{\lambda(x-1)})^{-1}, \end{aligned}$$

where $\lambda = 5\sqrt{2}$. We discretize in space using a second-order accurate central finite difference scheme using 1000 spatial points. This gives us a system for which we take \mathcal{L} and $\mathcal{N}(t, u(t))$ to be the discretized versions of $\frac{1}{100}u_{xx}$ and $u^2(1 - u)$, respectively. The micro time step is chosen to satisfy the Courant–Friedrichs–Lewy (CFL) linear stability condition, $h = 10^{-3}$.

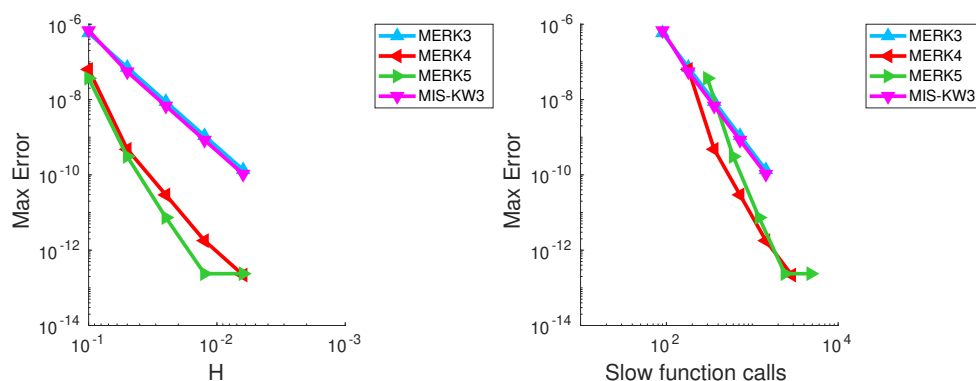


FIG. 1. Reaction diffusion convergence (left) and slow-only efficiency (right). The best fit convergence rates are 3.03, 4.93, 5.71, 3.20 (MERK3, MERK4, MERK5, and MIS-KW3, respectively). The most “efficient” methods at a given error are to the left of their less efficient counterparts.

In the left of Figure 1 we plot the method convergence as H is varied, which shows convergence rates that are slightly better than predicted for all methods tested. As this behavior is not consistently observed for the remaining test problems, we believe that this is an artifact of this particular test problem. Here, we compute the best-fit rates using only the error values larger than $\sim 10^{-13}$, where the error stagnates due to the accuracy of the reference solution.

The efficiency plots for both test problems in this category are very similar, so we present the slow-only efficiency plot for this problem in the right of Figure 1, saving the total efficiency plot for the next test. Here, we note that for tolerances larger than 10^{-7} , MERK3 and MIS-KW3 are the most efficient, but for tighter tolerances MERK4 is the best. Although MERK5 has a higher rate of convergence, the increased cost per step causes it to lag behind until it reaches the reference solution accuracy, where it begins to overtake MERK4.

5.1.2. Brusselator. The brusselator is an oscillating chemical reaction problem for which one of the reaction products acts as a catalyst. A two-component version of this problem is widely used as a test for ODE solvers [17, (16.12) of sect. I.16]. We examine a stiff version of this problem that has been used to test IMEX and multirate methods [46, 37]:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix}' = \begin{bmatrix} a - (w+1)u + u^2v \\ wu - u^2v \\ \frac{b-w}{\epsilon} - uw \end{bmatrix}, \quad \mathbf{u}(0) = \begin{bmatrix} 1.2 \\ 3.1 \\ 3 \end{bmatrix}$$

over the interval $t \in (0, 2]$, with parameters $a = 1$, $b = 3.5$, and $\frac{1}{\epsilon} = 100$. We convert this to have the multirate form (1.1) by defining

$$\mathcal{L} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\epsilon} \end{bmatrix}, \quad \mathcal{N}(t, \mathbf{u}(t)) = \begin{bmatrix} a - (w+1)u + u^2v \\ wu - u^2v \\ \frac{b}{\epsilon} - uw \end{bmatrix}.$$

In the left of Figure 2 we plot the error versus H , and list the corresponding best-fit convergence rates. We observe that all the tested methods perform slightly worse than their predicted convergence rates, which we attribute to order reduction

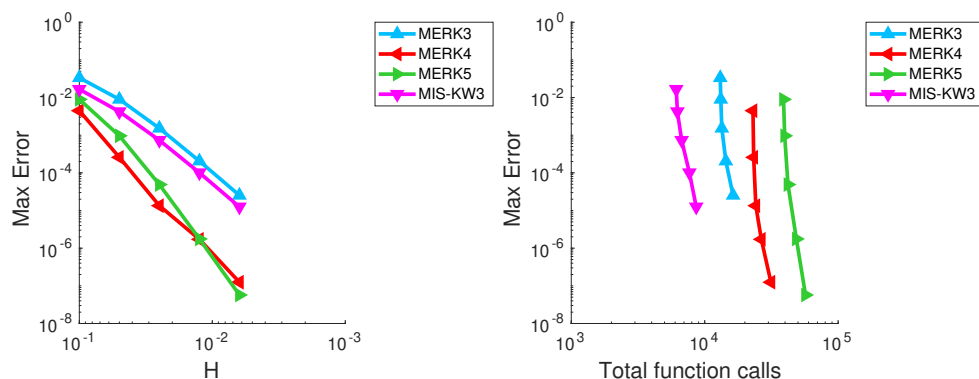


FIG. 2. *Brusselator convergence (left) and total efficiency (right). The best-fit convergence rates are 2.62, 3.75, 4.36, and 2.61 (MERK3, MERK4, MERK5, and MIS-KW3, respectively). Note the near-vertical lines in the efficiency plots, indicating the dominance of fast function calls in the estimate of total cost.*

due to the stiffness of the problem; however, the relative convergence rates of each method compare as expected against one another.

For this test problem, we plot the efficiency based on total function calls in the right of Figure 2. We note that each curve is almost vertical since the micro time step h is held constant for these tests, and is significantly smaller than H . Here, MIS-KW3 takes the least amount of total function calls since its structure ensures that it only traverses the time step interval $[t_n, t_n + H]$ once when evaluating the modified ODEs, whereas MERK3, MERK4, and MERK5 require approximately 2, 3, and 3 traversals, respectively. We note that although these additional traversals of the time step interval $[t_n, t_n + H]$ result in significant increases in the number of fast function calls, the number of potentially more costly slow function calls for all methods is equal to the number of slow stages.

5.2. Category II. Recalling that our second category of multirate applications focuses on accurately coupling the fast and slow processes, for these test problems we choose a fixed time-scale separation factor m for each method/test, and vary H (and, proportionally, $h = H/m$). For this group of tests we consider a linear multirate problem from Estep, Ginting, and Tavener [8] for which the fast variables are coupled into the slow equation (one-directional coupling) and a linear multirate problem of our own design where both the fast and slow variables are coupled (bidirectional coupling). Since the “optimal” value of m for each multirate algorithm is problem dependent, we describe our approach for determining this m value in section 5.2.1 below.

5.2.1. One-directional coupling. We consider a linear system of ODEs [8]:

$$(5.1) \quad \begin{bmatrix} u \\ v \\ w \end{bmatrix}' = \begin{bmatrix} 0 & -50 & 0 \\ 50 & 0 & 0 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad \mathbf{u}(0) = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$$

over the interval $t \in (0, 1]$. This has analytical solution $u(t) = \cos(50t)$, $v(t) = \sin(50t)$, and $w(t) = \frac{5051}{2501}e^{-t} - \frac{49}{2501}\cos(50t) + \frac{51}{2501}\sin(50t)$. We convert this problem

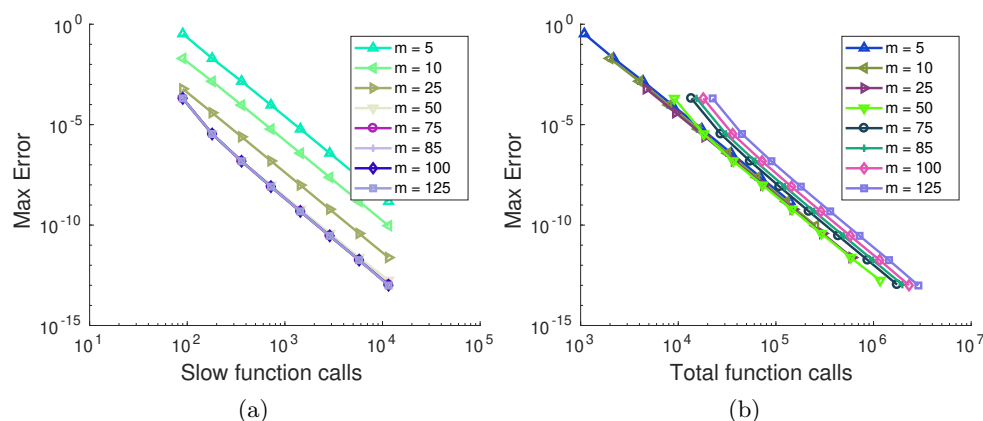


FIG. 3. Efficiency plots for MERK4 applied to the one-directional coupling test, resulting from various m factors.

to multirate form (1.1) by decomposing it as

$$\mathcal{L} = \begin{bmatrix} 0 & -50 & 0 \\ 50 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}, \quad \mathcal{N}(t, \mathbf{u}(t)) = \begin{bmatrix} 0 \\ 0 \\ -w \end{bmatrix}.$$

We first discuss our approach in determining the optimal time-scale separation factor m . For illustration, we consider MERK4 on this problem; however, we apply this approach to all methods for both this test and the following bidirectional coupling test in section 5.2.2. We begin by repeatedly solving the problem (5.1) using the multirate method with different factors $m = \{5, 10, 25, 50, 75, 85, 100, 125\}$. For each value of m , we vary H (and hence $h = H/m$). We then analyze the resulting total and slow-only efficiency plots for each fixed m value, as shown in Figure 3.

We first note that both plots show a group of m values with identical efficiency, along with other less efficient results. In Figure 3(a), the more efficient group is comprised of *larger* m values, whereas in Figure 3(b) the more efficient group has *smaller* m values. This is unsurprising, since increases in m for a fixed H correspond to decreases in h , leading to accuracy improvements at the fast time scale alone. While this will result in increased total function calls, the number of slow function calls will remain fixed. We therefore define the optimal m as the value where the fast and slow solution errors are balanced. Hence, in Figure 3(b) this corresponds to the largest m that remains in the more efficient group, and in Figure 3(a) this corresponds to the smallest m that remains in the more efficient group. Inspecting both plots in Figure 3, the optimal value for MERK4 on this problem is $m = 50$. Carrying out a similar process for the other methods on this problem, MERK3 has an optimal value of $m = 75$, MERK5 $m = 25$, and MIS-KW3 has an optimal value of $m = 75$.

Using these m values, in Figure 4 we plot the convergence results for the four methods on this problem, confirming the analytical orders of convergence, with errors stagnating around 10^{-13} due to accumulation of floating-point roundoff. While we find slightly better-than-expected convergence rates for the MERK methods, and only the expected rate for MIS-KW3, we do not draw conclusions regarding this behavior.

Similarly, in Figure 5 we plot both the total and slow-only efficiencies of each method on the one-directional test problem. When measuring only slow function

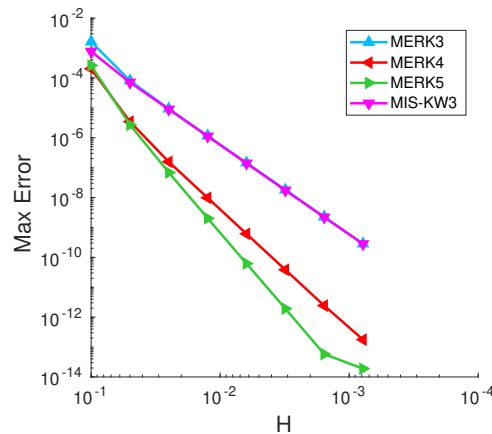


FIG. 4. One-directional coupling convergence. Best-fit convergence rates are 3.16, 4.28, 5.26, and 3.04 (MERK3, MERK4, MERK5, and MIS-KW3, respectively).

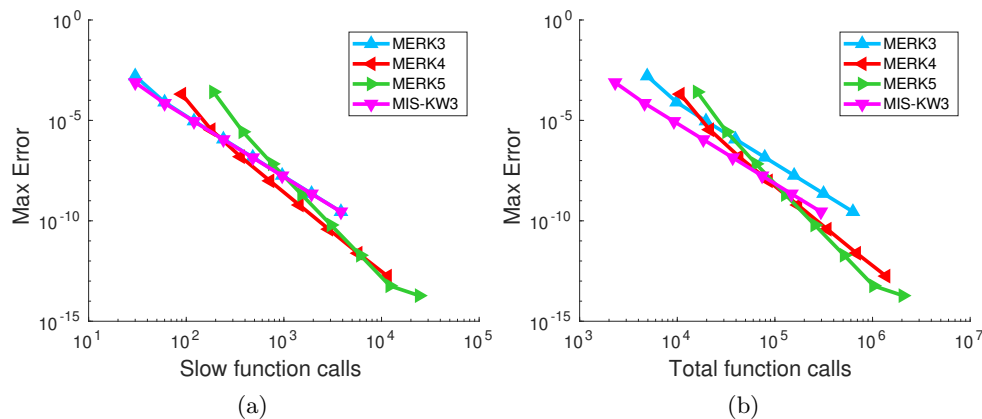


FIG. 5. One-directional coupling efficiency. The most efficient method depends on how “cost” is measured, as well as on the desired accuracy.

calls, both MIS-KW3 and MERK3 tie for errors larger than 10^{-6} , MERK4 is the most efficient for errors between 10^{-6} and 10^{-12} , and MERK5 is the most efficient at the tightest error values. When the fast function calls are given equal weight as the slow, however, MIS-KW3 is the most efficient at errors larger than 10^{-8} , while MERK5 is the most efficient at tighter error thresholds.

5.2.2. Bidirectional coupling. Taking inspiration from the preceding one-directional test, we designed a problem with coupling between both the fast and slow components to further demonstrate the flexibility and robustness of MERK methods. To this end, we consider the following test problem:

$$(5.2) \quad \begin{bmatrix} u \\ v \\ w \end{bmatrix}' = \begin{bmatrix} 0 & 100 & 1 \\ -100 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad \mathbf{u}(0) = \begin{bmatrix} 9001/10001 \\ 100000/10001 \\ 1000 \end{bmatrix}$$

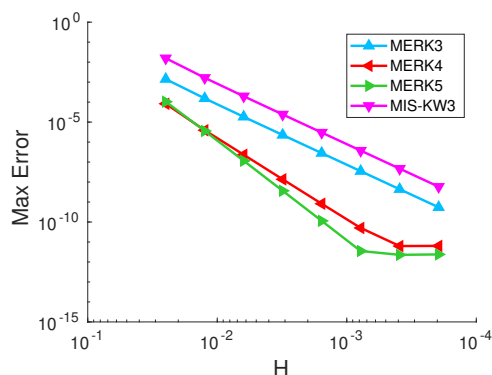


FIG. 6. *Bidirectional coupling convergence. Best-fit convergence rates are 3.03, 3.99, 4.97, and 3.06 (MERK3, MERK4, MERK5, MIS-KW3, respectively).*

over $t \in (0, 2]$. Converting to multirate form (1.1), we set \mathcal{L} and $\mathcal{N}(t, \mathbf{u}(t))$ as

$$\mathcal{L} = \begin{bmatrix} 0 & 100 & 0 \\ -100 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \mathcal{N}(t, \mathbf{u}(t)) = \begin{bmatrix} w \\ 0 \\ -w \end{bmatrix}.$$

While this is a linear test problem that may be solved using the matrix exponential, this solution is difficult to represent in closed form, and so we use a reference solution for convenience. Using the previously described approach for determining the optimal time-scale separation factor m for each method on this problem, we have $m = 50$ for MERK3 and MERK4, $m = 10$ for MERK5, and $m = 25$ for MIS-KW3.

In Figure 6 we plot the convergence rates of each method on this test problem, again confirming the analytical orders of convergence, with errors stagnating around 10^{-12} due to the reference solution accuracy.

Similarly, in Figure 7 we plot both the slow-only and total efficiency plots for this problem. Here, when measuring only the slow function calls, the most efficient method is MERK3 at error thresholds above 10^{-5} , and MERK5 for smaller error values. Strikingly, when considering the total number of function calls, MERK5 is the most efficient at nearly all error thresholds. We note, however, that the optimal time-scale-separation factor for MERK5 is $m = 10$ for this problem, which results in reduced fast function calls per slow step, and hence an overall reduction in total function calls.

5.3. Variations in the fast method. We finish by demonstrating the effects of using inner methods with differing orders of accuracy. Here, we consider only the MERK methods, applied to the bidirectional coupling problem (5.2). Here, we vary the order of the method applied for computing both the internal stage solutions (3.9), q , and the step solution (3.12), r . Recalling the convergence theory presented in Theorem 3.4, a MERK method of order p should use inner methods of orders $q \geq p - 1$ and $r \geq p$. However, in these tests we apply other variations on orders to ascertain whether (a) the inner methods could have even lower order and still obtain overall order p , or (b) use of higher-order inner methods can result in overall convergence higher than p . We present the best-fit convergence rates for this ensemble of tests in Table 1.

These numerical results show that in fact the inner method order requirements presented in Theorem 3.4 are both necessary and sufficient, i.e., the least-expensive combination for attaining a MERK method of order p is to compute stage solutions

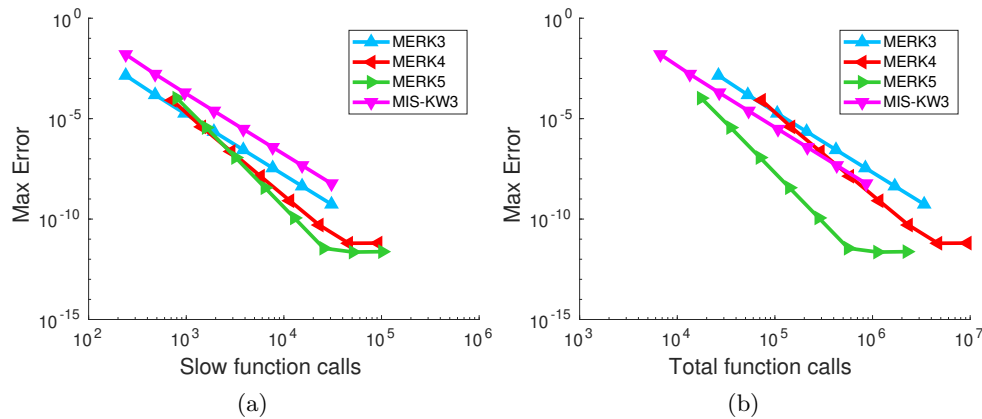


FIG. 7. Bidirectional coupling efficiency. Again, the most efficient method depends on how cost is measured, as well as on the desired accuracy; however, MERK5 demonstrates the best overall performance.

TABLE 1
Convergence rate dependence on inner ODE solvers.

MERK3($p = 3$)			MERK4($p = 4$)			MERK5($p = 5$)		
q	r	Observed order	q	r	Observed order	q	r	Observed order
2	2	2.00	3	3	3.01	4	4	4.00
3	2	2.00	4	3	3.01	5	4	4.00
2	3	3.03	3	4	3.99	4	5	4.97
3	3	3.03	4	4	3.99	5	5	4.97
4	4	3.03	5	5	3.99	6	6	4.96

(3.9) using an inner method of order $p - 1$, and the time step solution (3.12) using an inner method of order p . Furthermore, use of higher-order inner methods with orders $q = r > p$ does not result in overall order higher than p , due to the first term $C_1 H^p$ in Theorem 3.4, that corresponds to the coupling between the fast and slow processes.

6. Conclusion. We propose a novel class of multirate methods constructed from explicit exponential Runge–Kutta methods, wherein the action of the matrix exponential is approximated via solution of fast IVPs for each ExprK stage. Algorithmically, these methods offer a number of desirable properties. Since these are created through defining a set of modified IVPs (like (R)MIS and MRI-GARK methods), MERK implementations have near complete freedom in evolving the problem at the fast time scale; however, unlike (R)MIS and MRI-GARK, MERK methods may utilize inner solvers of reduced accuracy for the internal stages. Additionally, since the MERK structure follows directly from ExprK methods satisfying Theorem 3.1, derivation of high-order MERK methods, including versions supporting embeddings for temporal adaptivity, is much simpler than for alternate multirate frameworks. As a result, MERK methods constitute the first multirate algorithms of order five, without requiring deferred correction or extrapolation techniques. Furthermore, the proposed approach may be similarly applied to exponential Rosenbrock methods, allowing for problems where the fast time scale is nonlinear, although such methods are not considered in this work.

In addition to proposing the MERK class of multirate methods and providing rigorous analysis of their convergence, we provide numerical comparisons of the per-

formance of multiple MERK and MIS methods on a variety of multirate test problems. Based on these experiments, we find that the MERK methods indeed exhibit their theoretical orders of convergence, including tests that clearly demonstrate our primary convergence result in Theorem 3.4. Furthermore, the proposed methods compare favorably against standard MIS multirate methods, particularly when increased accuracy is desired and for problems wherein the slow right-hand side function is significantly more costly than the fast.

This work may be extended in numerous ways. As alluded to above, extensions of these approaches to explicit exponential Rosenbrock methods are straightforward, and are already under investigation. Additionally, extensions to higher order will follow from related developments of higher-order exponential methods. Finally, we plan to investigate the use of embeddings at both the fast and slow time scales to perform temporal adaptivity in both H and h for efficient, tolerance-based calculations.

Acknowledgments. The first author would like to thank Prof. Hochbruck and Dr. Demirel for their fruitful discussions during his visit at the Karlsruhe Institute of Technology (KIT) in 2013 under the support of the DFG Research Training Group 1294 “Analysis, Simulation and Design of Nanotechnological Processes.”

REFERENCES

- [1] E. L. BOUZARTH AND M. L. MINION, *A multirate time integrator for regularized Stokeslets*, J. Comput. Phys., 229 (2010), pp. 4208–4224, <https://doi.org/10.1016/j.jcp.2010.02.006>.
- [2] 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), pp. 201–222, <https://doi.org/10.1145/79505.79507>.
- [3] R. CHINOMONA, D. R. REYNOLDS, AND V. T. LUAN, *Multirate Exponential Runge–Kutta Methods (MERK)*, <https://github.com/rujekoc/merk.v1>, 2019.
- [4] E. CONSTANTINESCU AND A. SANDU, *Extrapolated implicit-explicit time stepping*, SIAM J. Sci. Comput., 31 (2010), pp. 4452–4477.
- [5] E. M. CONSTANTINESCU AND A. SANDU, *Extrapolated multirate methods for differential equations with multiple time scales*, J. Sci. Comput., 56 (2013), pp. 28–44.
- [6] S. M. COX AND P. C. MATTHEWS, *Exponential time differencing for stiff systems*, J. Comput. Phys., 176 (2002), pp. 430–455.
- [7] A. DEMIREL, J. NIEGEMANN, K. BUSCH, AND M. HOCHBRUCK, *Efficient multiple time-stepping algorithms of higher order*, J. Comput. Phys., 285 (2015), pp. 133–148.
- [8] D. ESTEP, V. GINTING, AND S. TAVENER, *A posteriori analysis of a multirate numerical method for ordinary differential equations*, Comput. Methods Appl. Mech. Engrg., 223–224 (2012), pp. 10–27.
- [9] P. W. FOK, *A linearly fourth order multirate Runge–Kutta method with error control*, J. Sci. Comput., 66 (2016), pp. 177–195, <https://doi.org/10.1007/s10915-015-0017-4>.
- [10] 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, Berlin, 2013, pp. 377–385.
- [11] 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, I. Graham, U. Langer, J.-M. Melenk, and M. Sini, eds., de Gruyter, Berlin, 2013, pp. 187–218.
- [12] M. J. GROTE AND T. MITKOVA, *Explicit local time-stepping methods for Maxwell’s equations*, J. Comput. Appl. Math., 234 (2010), pp. 3283–3302.
- [13] M. J. GROTE AND T. MITKOVA, *High-order explicit local time-stepping methods for damped wave equations*, J. Comput. Appl. Math., 239 (2013), pp. 270–289.
- [14] M. GÜNTHER, A. KVÆRNØ, AND P. RENTROP, *Multirate partitioned Runge–Kutta methods*, BIT, 41 (2001), pp. 504–514.
- [15] M. GÜNTHER AND A. SANDU, *Multirate generalized additive Runge–Kutta methods*, Numer. Math., 133 (2016), pp. 497–524, <https://doi.org/10.1007/s00211-015-0756-z>.
- [16] E. HAIRER, C. LUBICH, AND G. WANNER, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Ser. Comput. Math. 31, Springer, Berlin, 2006.

- [17] E. HAIRER, S. P. NØRSETT, AND G. WANNER, *Solving Ordinary Differential Equations, I, 2nd rev. ed., Nonstiff Problems*, Springer, Berlin, 1993.
- [18] M. HOCHBRUCK AND C. LUBICH, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.
- [19] M. HOCHBRUCK, C. LUBICH, AND H. SELHOFER, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.
- [20] M. HOCHBRUCK AND A. OSTERMANN, *Explicit exponential Runge–Kutta methods for semilinear parabolic problems*, SIAM J. Numer. Anal., 43 (2005), pp. 1069–1090.
- [21] M. HOCHBRUCK AND A. OSTERMANN, *Exponential Runge–Kutta methods for parabolic problems*, Appl. Numer. Math., 53 (2005), pp. 323–339.
- [22] M. HOCHBRUCK AND A. OSTERMANN, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.
- [23] M. HOCHBRUCK AND A. OSTERMANN, *Exponential multistep methods of Adams-type*, BIT, 51 (2011), pp. 889–908.
- [24] O. KNOTH AND M. ARNOLD, *Numerical solution of multiscale problems in atmospheric modeling*, Applied Numer. Math., 62 (2012), pp. 1531–1543.
- [25] O. KNOTH AND J. WENSCH, *Generalized split-explicit Runge–Kutta methods for the compressible Euler equations*, Mon. Weather Rev., 142 (2014), pp. 2067–2081.
- [26] O. KNOTH AND R. WOLKE, *Implicit-explicit Runge–Kutta methods for computing atmospheric reactive flows*, Appl. Numer. Math., 28 (1998), pp. 327–341, [https://doi.org/10.1016/S0168-9274\(98\)00051-8](https://doi.org/10.1016/S0168-9274(98)00051-8).
- [27] A. KVÆRNØ, *Stability of multirate Runge–Kutta schemes*, International J. Differential Equations Appl., 1 (2000), pp. 97–105.
- [28] V. T. LUAN, *Fourth-order two-stage explicit exponential integrators for time-dependent PDEs*, Appl. Numer. Math., 112 (2017), pp. 91–103.
- [29] V. T. LUAN, *Derivation of new efficient exponential Runge–Kutta methods*, BIT, submitted, (2019).
- [30] V. T. LUAN AND A. OSTERMANN, *Exponential B-series: The stiff case*, SIAM J. Numer. Anal., 51 (2013), pp. 3431–3445.
- [31] 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.
- [32] 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.
- [33] V. T. LUAN AND A. OSTERMANN, *Stiff order conditions for exponential Runge–Kutta methods of order five*, in Modeling, Simulation and Optimization of Complex Processes - HPSC 2012, H. B. et al., ed., Springer, 2014, pp. 133–143.
- [34] V. T. LUAN AND A. OSTERMANN, *Parallel exponential Rosenbrock methods*, Comput. Math. Appl., 71 (2016), pp. 1137–1150.
- [35] V. T. LUAN, J. A. PUDYKIEWICZ, AND D. R. REYNOLDS, *Further development of efficient and accurate time integration schemes for meteorological models*, J. Comput. Phys., 376 (2019), pp. 817–837.
- [36] J. NIESEN AND W. M. WRIGHT, *Algorithm 919: A Krylov subspace algorithm for evaluating the φ -functions appearing in exponential integrators*, ACM Trans. Math. Software (TOMS), 38 (2012), p. 22.
- [37] D. R. REYNOLDS, *Example programs for ARKode*, Tech. Report LLNL-SM-668082, Lawrence Livermore National Laboratory, 2016.
- [38] S. ROBERTS, A. SARSHAR, AND A. SANDU, *Coupled Multirate Infinitesimal GARK Schemes for Stiff Systems with Multiple Time Scales*, <https://arxiv.org/abs/1812.00808>, 2018.
- [39] A. SANDU, *A class of multirate infinitesimal GARK methods*, SIAM J. Numer. Anal., 57 (2019), pp. 2300–2327, <https://doi.org/10.1137/18M1205492>.
- [40] A. SANDU AND M. GÜNTHER, *A generalized-structure approach to additive Runge–Kutta methods*, SIAM J. Numer. Anal., 53 (2015), pp. 17–42.
- [41] A. SARSHAR, S. ROBERTS, AND A. SANDU, *Design of high-order decoupled multirate GARK schemes*, SIAM J. Sci. Comput., 41 (2019), pp. A816–A847.
- [42] V. SAVCENCO, W. HUNSDORFER, AND J. G. VERWER, *A multirate time stepping strategy for stiff ordinary differential equations*, BIT, 47 (2007), pp. 137–155.
- [43] M. SCHLEGEL, O. KNOTH, M. ARNOLD, AND R. WOLKE, *Multirate Runge–Kutta schemes for advection equations*, J. Comput. Appl. Math., 226 (2009), pp. 345–357.
- [44] M. SCHLEGEL, O. KNOTH, M. ARNOLD, AND R. WOLKE, *Implementation of multirate time integration methods for air pollution modelling*, Geoscientific Model Development, 5 (2012), pp. 1395–1405, <https://doi.org/10.5194/gmd-5-1395-2012>.

- [45] M. SCHLEGEL, O. KNOTH, M. ARNOLD, AND R. WOLKE, *Numerical solution of multiscale problems in atmospheric modeling*, Appl. Numer. Math., 62 (2012), pp. 1531–1543, <https://doi.org/10.1016/j.apnum.2012.06.023>.
- [46] J. M. SEXTON AND D. R. REYNOLDS, *Relaxed Multirate Infinitesimal Step Methods*, <https://arxiv.org/abs/1808.03718> (2018).
- [47] J. WENSCH, O. KNOTH, AND A. GALANT, *Multirate infinitesimal step methods for atmospheric flow simulation*, BIT, 49 (2009), pp. 449–473.