# PARAEXP: A PARALLEL INTEGRATOR FOR LINEAR INITIAL-VALUE PROBLEMS[*]

MARTIN J. GANDER AND STEFAN GÜTTEL[†]

**Abstract.** A novel parallel algorithm for the integration of linear initial-value problems is proposed. This algorithm is based on the simple observation that homogeneous problems can typically be integrated much faster than inhomogeneous problems. An overlapping time-domain decomposition is utilized to obtain decoupled inhomogeneous and homogeneous subproblems, and a near-optimal Krylov method is used for the fast exponential integration of the homogeneous subproblems. We present an error analysis and discuss the parallel scaling of our algorithm. The efficiency of this approach is demonstrated with numerical examples.

**Key words.** parallelization, linear initial-value problem, rational Krylov, matrix exponential

**AMS subject classifications.** 65L05, 65Y05, 65F60

## 1. Introduction.

We are interested in the parallel integration of a linear initial-value problem

$$\mathbf{u}'(t) = A\mathbf{u}(t) + \mathbf{g}(t), \quad \mathbf{u}(0) = \mathbf{u}_0, \quad t \in [0, T], \tag{1.1}$$

where $A \in \mathbb{C}^{N \times N}$ is a square matrix, $\mathbf{u}(t) \in \mathbb{C}^N$ is the sought function and $\mathbf{g}(t) \in \mathbb{C}^N$ is a source term which is possibly difficult to integrate numerically. For example, $\mathbf{g}(t)$ could contain high-frequency oscillations, or it could be given in form of a time-consuming computer subroutine. Problems of the form (1.1) arise often from the space discretization (e.g., by finite differences or finite elements) of partial differential equations, in which case $A$ is large and sparse. While the algorithm we are going to propose here is designed for large-scale problems by the use of rational Krylov approximation, our overlapping time-domain decomposition can as well be applied for the parallel integration of scalar or medium-sized initial-value problems.

The fast solution of initial-value problems is of practical interest in scientific computing, and various parallelization techniques have been proposed in the literature. A recent iterative approach for solving (not necessarily linear) initial-value problems $\mathbf{u}'(t) = \varphi(t, \mathbf{u})$ in time-parallel fashion is the *parareal* method by Lions, Maday & Turinici [38]. It is based on a time-domain partitioning and utilizes two integrators: a *coarse integrator* for the fast propagation of information over the full time-interval $[0, T]$, and a *fine integrator* which solves initial-value problems to high accuracy on disjoint time subdomains. The main goal of parareal is to reduce the time-to-solution by making use of additional parallelism in the time direction when, for example, spatial parallelization has been saturated. Parareal has not been designed to necessarily achieve high parallel efficiency, and in fact its parallel efficiency is limited by the reciprocal number of iterations required to achieve a desired accuracy. Typically, at least two iterations are required, which is why it is often stated that the parallel efficiency of parareal is at most 50 % (see, e.g., the discussion in [40]). The convergence behavior of parareal was analyzed in [25, 23]. For linear initial-value problems, it was proposed in [21] to enhance the coarse integrator of parareal by projecting with

[*] This work was supported by the Swiss National Science Foundation grant no. 200020-131826/1.

[†]University of Geneva, Department of Mathematics, CH-1211 Geneva, Switzerland, martin.gander@unige.ch, stefan.guettel@unige.ch. Current address of the second author: The University of Manchester, School of Mathematics, M13 9PL Manchester, UK.

respect to a linear space spanned by the results of previous fine integrations, and this enhancement was found to be crucial for solving certain second-order initial-value problems and hyperbolic problems [24]. On the other hand, this modification does not overcome the inherent limitation of the parallel efficiency of parareal, and also one has to be aware of additional computational issues (such as maintaining the accuracy of the involved projections for numerical stability, see [45]).

Various *direct* (i.e., non-iterative) parallel methods exploiting the linear structure of (1.1) have been proposed as well in the literature; for other parallelization approaches we refer to the monograph by Burrage [8].

In 1989, Gallopoulos & Saad [22] considered the parallel solution of (1.1) with symmetric $A$ and autonomous source term $\mathbf{g}(t) = \mathbf{g} = \text{const}$. The authors utilize the fact that the exact solution of this problem can be written in terms of the function $\varphi_1(z) := (e^z - 1)/z$ as

$$\begin{aligned}\mathbf{u}(t) &= \exp(tA)\mathbf{u}_0 + t\varphi_1(tA)\mathbf{g} \\ &= \mathbf{u}_0 + t\varphi_1(tA)\mathbf{v}, \quad \text{with} \;\; \mathbf{v} = A\mathbf{u}_0 + \mathbf{g}.\end{aligned}$$

The action of a matrix function $\varphi_1(tA)$ onto a vector $\mathbf{v}$ can be computed in parallel either by a polynomial Krylov method, in which case matrix-vector products with $A$ are parallelized, or by a rational approximation $r(z)$ of $\varphi_1(z)$ on the spectral interval of $(tA)$ in partial fraction form

$$\varphi_1(z) \approx r(z) = \sum_{j=1}^{p} \frac{w_j}{s_j - z}. \tag{1.2}$$

The evaluation of

$$\mathbf{u}(t) \approx \mathbf{u}_0 + t \cdot r(tA)\mathbf{v} = \mathbf{u}_0 + \sum_{j=1}^{p} t \cdot w_j \cdot (s_j I - tA)^{-1}\mathbf{v}$$

for a given time parameter $t$ amounts to the solution of $p$ complex shifted linear systems $(s_j I - tA)\mathbf{x}_j = \mathbf{v}$, each of which can be assigned to another processor. Possible choices for the rational function $r(z)$ include Padé approximants, Chebyshev ($L^\infty$) approximations computed by the Remez algorithm, or near-best approximations computed by the Caratheodory–Fejér method (see [10, 49]).

Another popular approach is to consider (1.1) in Laplace-transformed form (see [51, 28])

$$s\widehat{\mathbf{u}}(s) - \mathbf{u}_0 = A\widehat{\mathbf{u}}(s) + \widehat{\mathbf{g}}(s).$$

The solution $\mathbf{u}(t)$ may be represented as a contour integral of the inverse transform

$$\mathbf{u}(t) = \frac{1}{2\pi i}\int_\Gamma e^{ts}\widehat{\mathbf{u}}(s)\,\mathrm{d}s$$

with a suitable contour $\Gamma$ enclosing all singularities of $\widehat{\mathbf{u}}(s)$ (which are the eigenvalues of $A$ and all singularities of $\widehat{\mathbf{g}}(s)$). The discretization of this integral by a quadrature rule with nodes $s_j$ and weights $w_j$ again yields a rational function

$$\mathbf{u}(t) \approx \sum_{j=1}^{p} w_j(t)\widehat{\mathbf{u}}(s_j) = \sum_{j=1}^{p} w_j(t)(s_j I - A)^{-1}(\mathbf{u}_0 + \widehat{g}(s_j)),$$

which can be evaluated in parallel (cf. Talbot [53] for a related idea). Note that the shifted systems to be solved are $(s_j I - A)\widehat{\mathbf{u}}(s_j) = \mathbf{u}_0 + \widehat{\mathbf{g}}(s_j)$, and the dependency on the parameter $t$ is solely contained in the residues $w_j(t)$. This can be advantageous when $\mathbf{u}(t)$ is to be computed for many $t$ in a parameter range $[t_{\min}, t_{\max}]$, but one has to take into account that the accuracy of the approximation degrades as one moves away from some optimal parameter $t_{\text{opt}}$ (because an optimal choice of the contour $\Gamma$ would incorporate the influence of the time parameter $t$). Related approaches for homogeneous problems (i.e., $\mathbf{g} \equiv \mathbf{0}$) based on quadrature of the inverse Laplace transform have been proposed by Sheen et al. [50], Gavrilyuk & Makarov [27], and Trefethen et al. [55]. Another Laplace inversion method with real nodes $s_j$ has been used by Davies et al. [12].

An interesting method for the fast implicit Runge–Kutta approximation of (1.1) has been presented by López-Fernández et al. [39]. It uses a discrete variation-of-constants formula and a quadrature rule to reduce the number of linear systems to be solved. Banjai & Petersheim [3] proposed parallel linear multistep discretizations of (1.1) based on the fast approximate inversion of Toeplitz matrices [7]. Although this approach does not make direct use of inverse Laplace transforms, it also relies on the parallel solution of complex shifted linear systems by a multigrid preconditioned GMRES iteration. The Toeplitz representation of linear multistep methods used in [3] seems to be possible only if equal time steps $\Delta T$ are taken, so that time adaptivity is restricted by the number of restarts of the method.

Yet another approach, close in spirit to the one proposed here, is known as exponential quadrature (see [34, 35]). It is based on the variation-of-constants formula

$$\mathbf{u}(t) = \exp(tA)\mathbf{u}_0 + \int_0^t \exp((t-\tau)A)\mathbf{g}(\tau)\,\mathrm{d}\tau,$$

and the approximation of the integrand by a quadrature rule with nodes $\tau_1, \ldots, \tau_p$. This yields $p+1$ independent problems of the form $\exp((t-\tau_j)A)\mathbf{g}(\tau_j)$ and $\exp(tA)\mathbf{u}_0$. These matrix functions could again be approximated by a Krylov method. Exponential quadrature, however, is impractical if the source term $\mathbf{g}(t)$ is "difficult enough" so that too many quadrature nodes are needed to approximate the integral to a prescribed accuracy.

To overcome these problems, we propose a decomposition of (1.1) into subproblems on overlapping time intervals. The difficult inhomogeneous problems are integrated over short time intervals, whereas exponential propagation is used to integrate homogeneous problems over long time intervals. These subproblems are completely decoupled and can be assigned to different processors. Our method, which we call *paraexp* (for *parallel exponential propagation*), requires almost no communication or synchronization between the processors, except a summation step at the end of the algorithm. Paraexp allows any available serial integrator for (1.1) to be used in black-box fashion, which can be a major advantage in its implementation. Because the efficiency of our algorithm relies on the fast integration of homogeneous linear initial-value problems, Section 3 contains a brief discussion of state-of-the-art polynomial and rational Krylov methods for computing the matrix exponential. In Sections 4 and 5 we explore some practical aspects of our method, in particular, we show how to tune the accuracy of the integrators for each of the subproblems and how to balance the work load between the processors. We show that paraexp can achieve a high parallel efficiency when the number of processors is moderate, while keeping communication and synchronization at a minimum. In Section 6 we discuss extensions to more general problems of the form

$$M(t)\mathbf{u}'(t) = K(t)\mathbf{u}(t) + \mathbf{g}(t), \qquad \mathbf{u}(0) = \mathbf{u}_0, \qquad t \in [0, T],$$

where $M(t), K(t)$ are matrices varying on a larger time scale than $\mathbf{g}(t)$ and $M(t)$ may even be singular, in which case we are solving a differential-algebraic equation. In Section 7 we present results of some numerical experiments.

## 2. The paraexp algorithm.

Our approach for the parallel solution of (1.1) relies on two basic facts. First, the exact solution of an homogeneous initial-value problem

$$\mathbf{w}'(t) = A\mathbf{w}(t), \quad \mathbf{w}(0) = \mathbf{u}_0,$$

is given in terms if the matrix exponential as

$$\mathbf{w}(t) = \exp(tA)\mathbf{u}_0 := (I + (tA) + (tA)^2/2! + \cdots)\mathbf{u}_0. \tag{2.1}$$

Second, if we integrate an inhomogeneous problem with zero initial-value,

$$\mathbf{v}'(t) = A\mathbf{v}(t) + \mathbf{g}(t), \quad \mathbf{v}(0) = \mathbf{0}, \tag{2.2}$$

then the solution of the original problem (1.1) is given by superposition as

$$\mathbf{u}(t) = \mathbf{v}(t) + \mathbf{w}(t).$$

The serial integration of (2.2) over the time interval $[0, T]$ is as difficult as solving (1.1) itself. To employ $p$ processors with this task, we introduce a partitioning of $[0, T]$ into time intervals $[T_{j-1}, T_j]$ with $j = 1, \ldots, p$ and $0 = T_0 < T_1 < \cdots < T_p = T$, and consider the following subproblems.

**Type 1 :** For $j = 1, \ldots, p$ solve

$$\mathbf{v}_j'(t) = A\mathbf{v}_j(t) + \mathbf{g}(t), \quad \mathbf{v}_j(T_{j-1}) = \mathbf{0}, \quad t \in [T_{j-1}, T_j],$$

using some serial integrator.

**Type 2 :** For $j = 1, \ldots, p$ solve

$$\mathbf{w}_j'(t) = A\mathbf{w}_j(t), \quad \mathbf{w}_j(T_{j-1}) = \mathbf{v}_{j-1}(T_{j-1}), \quad t \in [T_{j-1}, T],$$

using a near-optimal exponential propagator (we set $\mathbf{v}_0(T_0) := \mathbf{u}_0$).

What we mean by a *near-optimal exponential propagator* will be discussed in Section 3. Note that the $p$ subproblems of Type 1 for $\mathbf{v}_j$ are completely decoupled due to the zero initial values. Also the subproblems of Type 2 for $\mathbf{w}_j$ are decoupled and can be integrated once the initial value $\mathbf{v}_{j-1}(T_{j-1})$ is available:

$$\mathbf{w}_j(t) = \exp((t - T_{j-1})A)\mathbf{v}_{j-1}(T_{j-1}). \tag{2.3}$$

Therefore it is natural to assign the integrations for $\mathbf{v}_{j-1}(t)$ and $\mathbf{w}_j(t)$ to the same processor, so that there is no need for communication and synchronization between the two types of subproblems. Note also that the time intervals $[T_{j-1}, T]$ for the $\mathbf{w}_j$ are overlapping. In Figure 2.1 we illustrate this overlapping decomposition into subproblems.

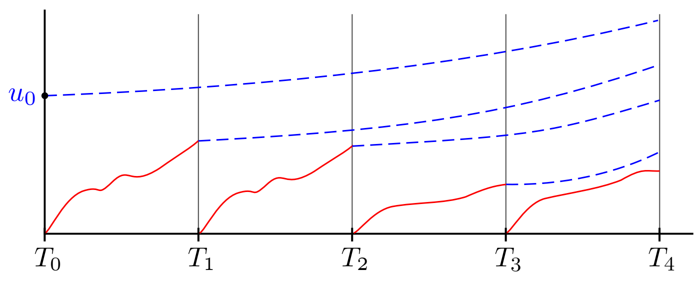

FIG. 2.1. *Overlapping time decomposition of an initial-value problem into four inhomogeneous problems with zero initial guess (Type 1, solid red curves) and four homogeneous problems (Type 2, dashed blue curves), the latter of which are exponentially propagated. The solution of the original problem is obtained by summation of all these curves.*

By superposition, the solution of (1.1) is given by

$$\mathbf{u}(t) = \mathbf{v}_k(t) + \sum_{j=1}^{k} \mathbf{w}_j(t) \quad \text{with } k \text{ such that} \quad t \in [T_{k-1}, T_k].$$

This sum is the only communication point between the processors. Our parallel algorithm, with generic name *paraexp* (*parallel exponential propagation*), is given by simultaneously integrating the subproblems of Type 1 and Type 2, and finally forming the sum for $\mathbf{u}(t)$ at the required time points $t$.

**3. Near-optimal approximation of the matrix exponential.** The overlapping propagation of the linear homogeneous subproblems of Type 2 is redundant. To obtain an efficient parallel method, we require that the computations of matrix exponentials (2.3) in our algorithm are fast compared to integrating the inhomogeneous subproblems of Type 1.

For scalar problems ($N = 1$) the computation of the exponential is a trivial task. For computing the exponential of small to medium-sized dense matrices (say, with $N \lessapprox 1000$) there are various methods available, see the review by Moler & Van Loan [41] and the monograph by Higham [32].

The computations become more challenging when the problem size $N$ gets large, in which case the matrix $A$ is hopefully sparse. In this case we can make use of the fact that not the matrix exponential $\exp(tA)$ itself is required, but the product of this matrix with some vector $\mathbf{v}$. Polynomial and rational Krylov methods have proven to be efficient for this task (see [29, 30] and the references therein). These methods can roughly be grouped into two categories: projection-based and expansion-based methods.

**3.1. Projection-based methods.** We briefly describe a variant of the *restricted-denominator Arnoldi method* by Moret & Novati [43] (see also van den Eshof & Hochbruck [20]), which extracts an approximation $\mathbf{a}_n(t) \approx \exp(tA)\mathbf{v}$ from a Krylov space built with the matrix $S := (I - A/\sigma)^{-1}A$,

$$\mathcal{K}^n(S, \mathbf{v}) = \operatorname{span}\{\mathbf{v}, S\mathbf{v}, \dots, S^{n-1}\mathbf{v}\}.$$

The parameters $n$ and $\sigma \in \overline{\mathbb{C}} \setminus (\Lambda(A) \cup \{0\})$ need to be chosen such that the search space $\mathcal{K}^n(S, \mathbf{v}_0)$ contains a good approximation to the exact solution $\exp(tA)\mathbf{v}$ for a given $t$. For $\sigma = \infty$ we obtain a standard Krylov space with the matrix $A$, i.e.,

$\mathcal{K}^n(S, \mathbf{v}) = \mathcal{K}^n(A, \mathbf{v})$. If $\mathcal{K}^n(S, \mathbf{v})$ is of full dimension $n$, as we assume in the following, we can compute an orthonormal basis $V_n \in \mathbb{C}^{N \times n}$ by the well-known Arnoldi orthogonalization process (see [25, §9.3.5]). The *nth-order Arnoldi approximation of* $\exp(tA)\mathbf{v}$ is then defined as

$$\mathbf{a}_n(t) := V_n \exp(t[S_n^{-1} + \sigma^{-1} I_n]^{-1}) V_n^* \mathbf{v}, \quad S_n := V_n^* S V_n. \tag{3.1}$$

Provided that $n$ is small, the computation of $\mathbf{a}_n(t)$ requires the evaluation of an $n \times n$ matrix function, which is small compared to the original $N \times N$ matrix exponential. The entries of the matrix $S_n$ are a by-result of the Arnoldi orthogonalization process and no explicit projection of $S$ with respect to $V_n$ is required. Remarkably, one can show that the Arnoldi approximation based on orthogonal projection is *near-optimal* in the following sense: with $\mathbb{W}(A) := \{\mathbf{x}^* A \mathbf{x} : \|\mathbf{x}\| = 1\}$ denoting the numerical range of $A$, there exists a universal constant $C \le 11.08$ such that

$$\| \exp(tA)\mathbf{v} - \mathbf{a}_n(t)\|_2 \le 2C \|\mathbf{v}\|_2 \min_{p \in \mathcal{P}_{n-1}} \left\| e^{tz} - \frac{p(z)}{(1 - z/\sigma)^{n-1}} \right\|_{\mathbb{W}(A)}, \tag{3.2}$$

where the norm on the right is the maximum norm over $\mathbb{W}(A)$ and $\mathcal{P}_{n-1}$ denotes the set of polynomials of degree $\le n-1$ (see [43, 4, 29]). In practice, the Arnoldi method performs even better than this bound suggests, and the approximations $\mathbf{a}_n(t)$ are extremely close to the least squares approximation $V_n V_n^* \exp(tA)\mathbf{v}$ (this observation can also be made in Figure 3.1 below).

The near-optimality of the Arnoldi method has an important implication: it is very unlikely that any polynomial (that is, explicit) time-stepping method for $\mathbf{u}' = A\mathbf{u}$, $\mathbf{u}(0) = \mathbf{v}$, can outperform the polynomial Arnoldi method (to which the restricted-denominator Arnoldi method reduces when $\sigma = \infty$) in the sense of requiring less matrix-vector products with $A$ to achieve the same accuracy. For example, the explicit Euler method (whose approximation of $\exp(tA)\mathbf{v}$ is given by $p(tA)\mathbf{v}$, where $p(z) = (1 + z/n)^n$ and $n$ is the number of time steps) or the classical Runge–Kutta method (with $p(z) = (1 + z/n + (z/n)^2/2 + (z/n)^3/6 + (z/n)^4/24)^n)$ typically do not produce the same quality of spectrally adapted polynomials as does the Arnoldi method, although all of these polynomials are approximations to the exponential function.

If the numerical range of $A$ extends far into the complex plane (i.e., the linear term is "stiff"), polynomials of very high degree are required for approximating the exponential function on this set. This means that polynomial Krylov spaces of very large dimension need to be computed. Although various polynomial methods have been proposed which prevent the dimension $n$ of $\mathcal{K}^n(A, \mathbf{v})$ to grow above memory limit (restarted Krylov methods [17, 1]), or which avoid the storage of a Krylov basis (e.g., methods based on explicit interpolation [36, 9] or evaluation of expansions of the exponential function [13, 5, 42]), it may be beneficial to use instead the above restricted-denominator Arnoldi method with a *finite* shift $\sigma$. Of course, this amounts to the solution of a linear system with a constant coefficient matrix $I - A/\sigma$ in every Arnoldi step. If the use of a direct solver is feasible, then a factorization of this matrix needs to be computed only once.

In Figure 3.1 we compare the discussed methods for computing $\exp(tA)\mathbf{v}$, $t = 1$, with the matrices

$$A_1 = \text{tridiag}(30, -40, 10) \in \mathbb{R}^{199 \times 199} \quad \text{and} \quad A_2 = \text{tridiag}(60, -90, 30) \in \mathbb{R}^{299 \times 299},$$

which can be interpreted as finite-difference discretizations of the same 1D advection–diffusion problem on the spatial interval $[0,1]$ with homogeneous boundary conditions. The vector $\mathbf{v}$ is chosen randomly with normal-distributed entries. We also show the error of the orthogonal projection of the exact solution $\exp(A)\mathbf{v}$ onto $\mathcal{K}^n(A,\mathbf{v})$ and $\mathcal{K}^n(S,\mathbf{v})$, respectively (in the latter case we have chosen $\sigma = 40$). Comparing the number of matrix-vector products $n$ required by each algorithm to achieve a certain accuracy, we clearly find that it is beneficial to use the Arnoldi method rather than a time-stepping method for the propagation of linear homogeneous problems. Note that the difference between the projection and time-stepping methods gets larger for higher accuracies. Moreover, we observe that the iteration number of the implicit methods (in particular, the rational Arnoldi method) is almost constant when the discretization becomes finer, i.e., the convergence is almost *mesh-independent*.

To estimate the error of rational Krylov approximations, one can make use of the fast error decay (following an initial stagnation phase that may occur, cf. Figure 3.1). It is often sufficient to use the difference of two consecutive Krylov iterates as an estimate for the propagation error $\|\exp(tA)\mathbf{v} - \mathbf{a}_n(t)\|$, that is

$$\|\exp(tA)\mathbf{v} - \mathbf{a}_n(t)\|_\infty \approx \|\mathbf{a}_{n+\ell}(t) - \mathbf{a}_n(t)\|_\infty, \tag{3.3}$$

where $\ell \geq 1$. In our numerical tests in Section 7 we have always used $\ell = 1$, yielding sufficiently good estimates for our purposes. For the derivation of rigorous error bounds we refer to the literature, e.g., [33, 4, 29] and the references therein.

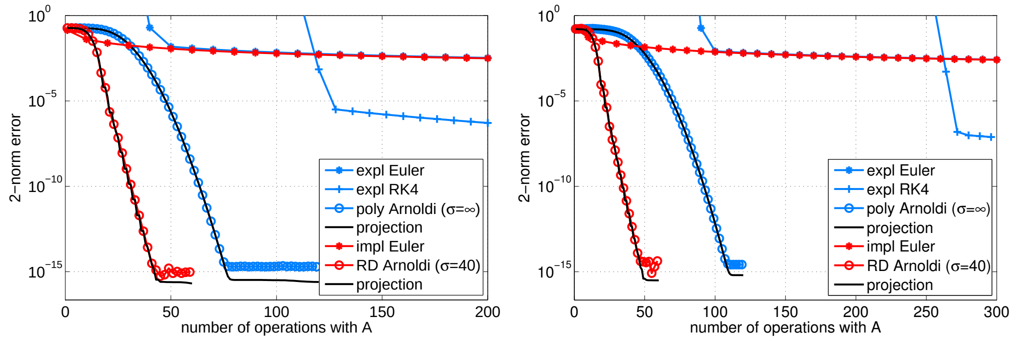

Fig. 3.1. *Comparison of various time-stepping methods and Krylov methods for computing the solution* $\mathbf{u}(1)$ *of a linear homogeneous advection–diffusion problem* $\mathbf{u}' = A\mathbf{u}$, $\mathbf{u}(0) = \text{random}$, *with 199 (top) and 299 (bottom) finite-difference discretization points.*

**3.2. Expansion-based methods.** Another class of methods for computing approximations to $\exp(tA)\mathbf{v}$ is based on expansions of the form

$$\exp(tz) \approx \sum_{j=0}^{n-1} \beta_j(t) p_j(z),$$

where the $p_j$ are polynomials or rational functions which possibly satisfy a short-term recurrence. The approximations are then obtained as

$$\exp(tA)\mathbf{v} \approx \sum_{j=0}^{n-1} \beta_j(t) p_j(A)\mathbf{v}. \tag{3.4}$$

For example, if $A$ is a normal-1 matrix (that is, normal with eigenvalues on a straight line), then the $p_j(z)$ can be taken as Chebyshev polynomials $T_j(z) = \cos(j \arccos(z))$ transformed to the spectral interval of $A$, and the $\beta_j(t)$ can be taken as the coefficients of the Chebyshev expansion of $\exp(tz)$ on this interval (see [13]). The well-known three-term recurrence for Chebyshev polynomials then translates into a recurrence for the vectors in (3.4):

$$[T_{j+1}(A)\mathbf{v}] = 2\alpha A[T_j(A)\mathbf{v}] - \beta[T_{j-1}(z)\mathbf{v}],$$

where $\alpha, \beta$ are constants depending on the spectral interval of $A$. This approach is applicable if (an estimate for) the spectral interval of $A$ is known, and if this interval is not too wide such that not too many terms are needed to achieve a desired accuracy in (3.4).

If the eigenvalues of $A$ are not contained in some interval or $A$ is not normal, then $\exp(tz)$ is typically approximated on the numerical range $\mathbb{W}(A)$. For such problems, expansions in Faber polynomials or methods based on polynomial or rational Newton interpolation have proved to be efficient (see [36, 9, 29]), although knowledge of $\mathbb{W}(A)$ may be a problematic requirement.

## 4. Error control.

Many integrators for ordinary differential equations, for example those in Matlab, use an error control criterion like

$$\|\mathbf{e}(t)\|_\infty \le \max\left\{\texttt{reltol}\|\widetilde{\mathbf{u}}(t)\|_\infty, \texttt{abstol}\right\}, \quad t \in [0, T],$$

where $\mathbf{e}(t) = \mathbf{u}(t) - \widetilde{\mathbf{u}}(t)$ is the (estimated) error of the computed solution $\widetilde{\mathbf{u}}(t)$. Because the inhomogeneous subproblems of Type 1 for $\mathbf{v}_j(t)$ are integrated with zero initial guess, it is not advisable to use an error criterion which is relative to the norm of the solution. Hence we assume that all of these subproblems are integrated with an absolute error $\|\mathbf{e}_j(t)\|_\infty = \|\mathbf{v}_j(t) - \widetilde{\mathbf{v}}_j(t)\|_\infty \le \texttt{abstol}$ over the time interval $[T_{j-1}, T_j]$. This error is then propagated exponentially over the remaining interval $[T_j, T]$, hence we have to study the transient behavior of

$$\|\exp(tA)\mathbf{e}_j(T_j)\|_\infty \le \|\exp(tA)\|_\infty \texttt{abstol} \tag{4.1}$$

for $t \in [0, T - T_j]$. Finally we will add the errors from all these exponential propagations to obtain the overall error produced by our algorithm.

It is well known that for a *stable* matrix $A$ (i.e., all eigenvalues lie in the left half of the complex plane) the limit $\lim_{t\to\infty} \|\exp(tA)\|_\infty$ is finite. Unfortunately, the norm may initially grow arbitrarily large before convergence sets in, a phenomenon usually referred to as *hump* (see [41]). However, for a diagonally dominant matrix $A = [a_{ij}]$ with $a_{ii} \le 0$ this cannot happen, as one can show as follows (see [44]): Define $\rho = \max_i\{a_{ii} + \sum_{j\ne i} |a_{ij}|\} \le 0$. By the formula $\exp(tA) = \lim_{k\to\infty}(I + tA/k)^k$ we have $\|\exp(tA)\|_\infty \le \lim_{k\to\infty} \|I + tA/k\|_\infty^k$. For $k$ sufficiently large we have

$$\|I + tA/k\|_\infty = \max_i \left\{1 + t\Big(a_{ii} + \sum_{j\ne i} |a_{ij}|\Big)/k\right\} = 1 + t\rho/k,$$

hence

$$\|\exp(tA)\|_\infty \le \lim_{k\to\infty} (1 + t\rho/k)^k = e^{t\rho} \le 1 \quad \text{for all } t \ge 0. \tag{4.2}$$

A simple example for a diagonally dominant matrix $A$ whose numerical range is not contained in the left half of the complex plane, but $\|\exp(tA)\|_\infty = 1$ for all $t \geq 0$, is

$$A = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix}.$$

An error estimate for not necessarily diagonally dominant matrices in the 2-norm can be obtained by a theorem of Crouzeix [11]: there exists a universal constant $C \leq 11.08$ such that

$$\|\exp(tA)\|_2 \leq C \max_{z\in\mathbb{W}(A)} |\exp(tz)|, \tag{4.3}$$

hence $\|\exp(tA)\|_2$ is bounded by $C$ for all $t \geq 0$ provided that $\mathbb{W}(A)$ is contained in the left half of the complex plane, which includes the important special case of sectorial operators. Clearly, if $A$ is normal we can choose $C = 1$, and it is actually conjectured that the constant "11.08" can be replaced by "2" for every square matrix $A$ (see [11]). Using the relation $\|A\|_2 \leq \sqrt{N}\|A\|_\infty$ (recall that $A \in \mathbb{C}^{N\times N}$), we may summarize our findings in the following theorem.

THEOREM 4.1. *Let the matrix $A \in \mathbb{C}^{N\times N}$ be stable. Assume that all exponential propagations $\mathbf{w}_j(t)$ are computed exactly and the absolute errors for all subproblems $\mathbf{v}_j(t)$ are bounded, that is $\|\mathbf{e}_j(t)\|_\infty \leq$ `abstol` for $t \in [T_{j-1}, T_j]$. Then in the following cases there exists a constant $C$ with*

- *$C \leq 11.08\sqrt{N}$ if $\mathbb{W}(A)$ is contained in the left half of the complex plane, or*
- *$C \leq \sqrt{N}$ if $A$ is normal, or*
- *$C \leq 1$ if $A$ is diagonally dominant with nonpositive diagonal entries,*

*such that the numerical solution $\widetilde{\mathbf{u}}(t)$ of the paraexp algorithm satisfies*

$$\|\mathbf{e}(t)\|_\infty = \|\mathbf{u}(t) - \widetilde{\mathbf{u}}(t)\|_\infty \leq (C(k-1)+1)\,\texttt{abstol}, \tag{4.4}$$

*where $k$ is chosen such that $t \in [T_{k-1}, T_k]$.*

We remark that the error bound in Theorem 4.1 is universal, but typically it is too pessimistic for practical computations because

1. it is based on (4.3) which may lead to crude overestimations in particular if $A$ is highly nonnormal,
2. the worst-case error is attainable only if all error vectors $\mathbf{e}_j(T_j)$ are collinear,
3. the error vectors $\mathbf{e}_j(T_j)$ may actually be damped when being exponentially propagated. For example, if $A$ is Hermitian with spectral interval $[\lambda_{\min}, \lambda_{\max}]$ and $\lambda_{\max} < 0$, then the propagated error in (4.1) will decay like $e^{t\lambda_{\max}} < 1$.

Note that there exist possibly sharper bounds on the norm of the matrix exponential than (4.2) or (4.3), but we do not attempt to review them here and instead refer to [56] or Chapter 14 and 15 in [54].

REMARK 1. *The exponential propagations $\mathbf{w}_j(t)$ are assumed to be exact in Theorem 4.1, which of course is impossible in a practical computation. However, in practice it is only required that the computed exponential propagations $\widetilde{\mathbf{w}}_j(t)$ are sufficiently accurate relative to the accuracy of the Type 1 integrations. For example, if all computed exponential propagations $\widetilde{\mathbf{w}}_j(t)$ are guaranteed to satisfy $\|\mathbf{w}_j(t)-\widetilde{\mathbf{w}}_j(t)\|_\infty \leq$ `abstol`$/p$ for all $t \in [T_{j-1}, T]$ and $j = 1, \ldots, p$, then $\sum_{j=1}^{p} \widetilde{\mathbf{w}}_j(t)$ will have an accuracy of `abstol` and the right-hand side of the inequality (4.4) merely changes to $(C(k-1)+2)$`abstol`.*

REMARK 2. *A more practical estimate of the error is obtained via probabilistic considerations: If the elements in all vectors $\mathbf{e}_j(T_j)$ are random variables following a* *normal distribution with zero mean and standard deviation $\sigma$, then the sum of $(k-1)$ such vectors has zero mean and standard deviation $\sigma\sqrt{k-1}$. The expected maximum error norm is therefore*

$$\mathbb{E}\left[\|\mathbf{e}(t)\|_\infty\right] = (C\sqrt{k-1}+1)\,\texttt{abstol} \lessapprox C\sqrt{p}\,\texttt{abstol},$$

*where $k \leq p$ is chosen such that $t \in [T_{k-1}, T_k]$. In our numerical experiments in Section 7 we will demonstrate how this serves as a realistic estimate for the expected propagation error.*

**5. Load balancing.** Load balancing aims at leveling the amount of computational work between $p$ processors. In our algorithm we have essentially two means for balancing the work distribution. First, we may adapt the time partitioning proportional to the computation time that the time-stepping method of Type 1 spends on a certain time slice. The adaptation of the time partitioning is recommended if an adaptive time-stepping method is used because the stiffness of the source term $\mathbf{g}(t)$ in (1.1) varies largely in time. Second, we may start from an equispaced time grid and perturb it slightly by taking into account the scheduling of the exponential propagations of Type 2. This load balancing approach works well if (1.1) can be efficiently integrated by a time-stepping method with constant step size. In the following we outline both approaches.

**5.1. Load balancing for adaptive time-stepping integrators.** Assume we have a serial time-stepping method that integrates (1.1) over $[0, T]$ to some absolute error tolerance `abstol` in $\tau_0$ units of computation time. In view of Remark 2, our parallel algorithm with $p$ processors requires the error tolerance of the time-stepping method to be refined by a factor $\sqrt{p}$. If our time-stepping method is of order $q$ and its computation time is proportional to the number of time steps, then it is reasonable to expect that the integration of (1.1) to the error tolerance `abstol`$/\sqrt{p}$ requires $\sqrt{p}^{1/q}\,\tau_0$ units of computation time. If the time partitioning $0 = T_0 < T_1 < \cdots < T_p = T$ is chosen such that on each time slice $[T_{j-1}, T_j]$ the same computation time is spent by the serial integrator (Type 1), then all processors will finish synchronously after

$$\tau_1 = \frac{\sqrt{p}^{1/q}\,\tau_0}{p} = \frac{\tau_0}{p^{1-1/(2q)}}$$

units of computation time with the Type 1 integrations. If the time-stepping method is adaptive, such a balanced time partitioning could be obtained by a serial initial integration of (1.1) with very crude error tolerance, and the partitioning of $[0, T]$ into $p$ subintervals containing an equal number of time steps.

If the exponential propagations of Type 2 require $\tau_2$ units of computation time on each processor, then the total parallel computation time of our algorithm is

$$\tau_p = \tau_1 + \tau_2 = \frac{\tau_0}{p^{1-1/(2q)}} + \tau_2. \tag{5.1}$$

Note that the parallel speedup given by

$$\text{speedup} = \frac{\tau_0}{\tau_p} = \frac{1}{p^{1/(2q)-1} + \tau_2/\tau_0}$$

gets larger with increasing order $q$ of the time-stepping method and with smaller computation time $\tau_2$ for the exponential propagations.

**5.2. Load balancing for constant time-stepping.** If a nonadaptive time-stepping method with constant step size is used, we may assume that the computation time for a Type 1 integration only depends on the length $\Delta T$ of a time slice

$$\hat{\tau}_1(\Delta T) = \frac{\Delta T}{T}\sqrt{p}^{1/q}\,\tau_0.$$

In this case we may still optimize the time partitioning if the solution vectors $\mathbf{u}(T_j)$ are required at *all* time points $T_j$ $(j = 1, \ldots, p)$, by taking into account the scheduling of the exponential propagations. By the additivity of the exponential we have

$$\mathbf{w}_j(T_{j+k}) = \exp((T_{j+k} - T_{j+k-1})A)\cdots\exp((T_j - T_{j-1})A)\mathbf{w}_{j-1}(T_{j-1}),$$

and therefore we can compute $\mathbf{w}_j(T_p)$ by $p-j+1$ sequential exponential propagations. If we denote by $\mathbf{w}_j(T_k, T_\ell)$ the exponential propagation of $\mathbf{w}_j(t)$ over the interval $[T_k, T_\ell]$, the simplest way of assigning the integrations to processors is (here illustrated for $p = 5$):

$$\begin{array}{llllll}
\text{processor } 1: & \mathbf{w}_1(T_0,T_1) & \mathbf{w}_1(T_1,T_2) & \mathbf{w}_1(T_2,T_3) & \mathbf{w}_1(T_3,T_4) & \mathbf{w}_1(T_4,T_5)\\
\text{processor } 2: & \mathbf{w}_2(T_1,T_2) & \mathbf{w}_2(T_2,T_3) & \mathbf{w}_2(T_3,T_4) & \mathbf{w}_2(T_4,T_5) & \\
\text{processor } 3: & \mathbf{w}_3(T_2,T_3) & \mathbf{w}_3(T_3,T_4) & \mathbf{w}_3(T_4,T_5) & & \\
\text{processor } 4: & \mathbf{w}_4(T_3,T_4) & \mathbf{w}_4(T_4,T_5) & & & \\
\text{processor } 5: & \mathbf{w}_5(T_4,T_5) & & & &
\end{array} \tag{5.2}$$

Under the assumption that each of these exponential propagation steps requires the same computation time $\hat{\tau}_2$, processor number $j$ requires $\hat{\tau}_2(p-j+1)$ units of time. This unbalance can lead to idle processors. Fortunately, there are two simple strategies for avoiding this problem.

1. Without processor communication (perturbed time grid) : Assume that processor number 1 computes $\mathbf{w}_p(t)$ and $\mathbf{v}_1(t)$, and every other processor with number $j > 1$ computes $\mathbf{w}_{j-1}(t)$ and $\mathbf{v}_j(t)$. Then for obtaining a balanced work load, the intervals of the time partitioning should be chosen with increasing length, except the last interval which should be the shortest among all. Each component of the following equation equals the unknown total parallel computation time $\tau_p$

$$\frac{\tau_0\sqrt{p}^{1/q}}{T}\begin{pmatrix} T_p - T_{p-1}\\ T_{p-1} - T_{p-2}\\ \vdots\\ T_1 - T_0\end{pmatrix} + \tau_2\begin{pmatrix} p\\ 1\\ \vdots\\ p-1\end{pmatrix} = \tau_p\begin{pmatrix}1\\1\\ \vdots\\ 1\end{pmatrix}. \tag{5.3}$$

   Summation of all components yields an equation for $\tau_p$ (using $T = T_p - T_0$),

$$\tau_p = \frac{\tau_0\sqrt{p}^{1/q}}{p} + \frac{\hat{\tau}_2(p+1)}{2},$$

   which corresponds to (5.1) with $\tau_2 = \hat{\tau}_2(p+1)/2$ and allows us to determine the $T_j$ recursively by (5.3).
2. With processor communication (equispaced time grid) : Assume that the integrations for all $\mathbf{v}_j(t)$ finish synchronously and that the number of processors $p$ is odd (for simplicity). In total there are $p(p+1)/2$ exponential

propagation steps for the $\mathbf{w}_j(t)$ to be computed. Instead of waiting for the last processor with $p$ propagation steps to finish, it is possible to distribute $(p+1)/2$ exponential propagation steps to each of the processors if one allows for larger time steps. For example, for $p = 5$ processors one could arrange the computation in the following way:

| | | | |
|---|---|---|---|
| processor 1 : | $\mathbf{w}_1(T_0, T_1)$ | $\mathbf{w}_1(T_1, T_2)$ | $\mathbf{w}_1(T_2, T_4)$ |
| processor 2 : | $\mathbf{w}_2(T_1, T_2)$ | $\mathbf{w}_2(T_2, T_3)$ | $\mathbf{w}_2(T_3, T_4)$ |
| processor 3 : | $\mathbf{w}_3(T_2, T_3)$ | $\mathbf{w}_3(T_3, T_4)$ | $\mathbf{w}_3(T_4, T_5)$ |
| processor 4 : | $\mathbf{w}_4(T_3, T_4)$ | $\mathbf{w}_4(T_4, T_5)$ | $\mathbf{w}_2(T_3, T_5)$ |
| processor 5 : | $\mathbf{w}_5(T_4, T_5)$ | $\mathbf{w}_1(T_1, T_3)$ | $\mathbf{w}_1(T_3, T_5)$ |

Obviously there is need for communication of initial values from processor $1 \to 5$ and $2 \to 4$. Neglecting communication, the expected total parallel computation time $\tau_p$ equals (5.1).

## 6. Incorporation of a mass matrix and problems of higher order.

In this section we briefly discuss various extensions of the paraexp algorithm.

1. Consider a problem

$$M\mathbf{u}' = K\mathbf{u}(t) + \mathbf{g}(t), \qquad \mathbf{u}(0) = \mathbf{u}_0, \tag{6.1}$$

which may arise from a finite-element discretization, in which case $M$ is called a *mass matrix.* If $M$ is invertible, we can formally define $A := M^{-1}K$ and obtain a problem of the original form (1.1). Typically, mass-lumping techniques are employed in explicit time-stepping methods to avoid the inversion of $M$. If the restricted-denominator Arnoldi method described in Section 3 is used for the exponential propagations, we have to solve linear systems anyway, hence mass-lumping yields no computational savings. Note that we have

$$S = (I - A/\sigma)^{-1}A = (M - K/\sigma)^{-1}K, \tag{6.2}$$

and if $M$ and $K$ have the same sparsity pattern, computing a factorization of $M - K/\sigma$ is just as expensive as computing a factorization of $K$ alone.

2. If the matrix $M$ is not invertible, (6.1) becomes a differential-algebraic equation and the solution $\mathbf{u}(t)$ is restricted to some manifold determined by the nullspace of $M$. In this case $M^{-1}K$ is not defined, but the restricted-denominator Arnoldi method is actually still applicable when $S$ is employed in the form $(M - K/\sigma)^{-1}K$ (and when $\sigma$ is different from all generalized eigenvalues $\lambda$ satisfying $K\mathbf{x} = \lambda M\mathbf{x}$). Hence, our parallel algorithm can be used to integrate differential-algebraic equations. In fact, the reformulation (6.2) embodies a similar trick as is used for the derivation of time-stepping methods for differential-algebraic equations $M\mathbf{u}' = \varphi(\mathbf{u})$: "assume that $M$ is regular, apply an ODE method to $\mathbf{u}' = M^{-1}\varphi(\mathbf{u})$ and multiply the resulting formulas by $M$" (see [31, page 378]).

3. The treatment of higher-order linear problems, such as

$$M\mathbf{u}'' + D\mathbf{u}'(t) + K\mathbf{u}(t) = \mathbf{g}(t), \qquad \mathbf{u}(0) = \mathbf{u}_0, \qquad \mathbf{u}'(0) = \mathbf{v}_0,$$

is trivially possible after rewriting the equation as a block-system of first order, in this case

$$\begin{bmatrix} I & O \\ O & M \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}' = \begin{bmatrix} O & I \\ -K & -D \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{g}(t) \end{bmatrix}.$$

Unfortunately, the problem size is doubled and possible symmetries in the matrices $K$, $D$, or $M$ are hard to exploit in this formulation. In the special case where $M$ is invertible and $D = O$, however, the solution of the homogeneous problem is

$$\mathbf{u}(t) = \cos(t\sqrt{M^{-1}K})\mathbf{u}_0 + t\,\mathrm{sinc}(t\sqrt{M^{-1}K})\mathbf{v}_0,$$

which is a problem of the original size and where the evaluation of the matrix exponential has been replaced by two other functions of the same matrix acting on different vectors. These could again be approximated from rational Krylov spaces by the restricted-denominator Arnoldi method. As discussed above, even if $M$ is not invertible and $D = O$, that is we have a differential-algebraic equation of order 2, the matrix $M - K/\sigma$ is still invertible for almost all $\sigma \in \mathbb{C}$.

## 7. Numerical experiments.

All computations were done in MATLAB R2009b on a WINDOWS 7 notebook with Intel Core i5-450M processor running at 2.4 GHz. For the finite-element discretization in the third example we have used COMSOL 3.5 software with its MATLAB integration ("Save as Model M-file"). All timings are averages over 50 runs of our algorithm. Recall that the subproblems of Type 1 and 2 in paraexp are completely decoupled, hence it makes no difference whether we measure their performance in a serial or parallel implementation.

### 7.1. A diffusive problem with oscillating source term.

As a simple model problem we consider the 1D heat equation

$$\begin{aligned}
\partial_t u(t,x) &= \alpha \partial_{xx} u(t,x) + g(t,x) && \text{on } x \in (0,1),\\
u(t,0) &= u(t,1) = 0,\\
u(0,x) &= u_0(x) = 4x(1-x),\\
g(t,x) &= h \max\{1 - |c - x|/w, 0\}, && \text{where } c = .5 + (.5 - w)\sin(2\pi f t).
\end{aligned}$$

The source term $g(t,x)$ is a hat function centered at $c$ with half-width $w = 0.05$ and height $h = 100\alpha^{1/2}$, oscillating with frequency $f$. The finite-difference discretization at $N = 100$ points $x_j = j/(N+1)$ $(j = 1,\dots,N)$ yields the initial-value problem of the form (1.1), where $A = \alpha(N+1)^2\,\mathrm{tridiag}(1,-2,1) \in \mathbb{R}^{N\times N}$ and $\mathbf{g}(t) \in \mathbb{R}^N$ is the spatial discretization of $g(t,x)$. This problem is integrated over the time interval $[0, T = 1]$. In Figure 7.1 we give a visualization of $u(t,x)$. For the serial integration we use the classical Runge–Kutta method of order $q = 4$ with a constant step size

$$\Delta t_0 = \min\{5\cdot 10^{-5}/\alpha, 10^{-2}/f\},$$

where the first value is chosen to avoid instability of the time-stepping method caused by the stiff linear term $A\mathbf{u}(t)$ and the second value is inverse proportional to the frequency to capture the oscillations of $\mathbf{g}(t)$. As shown in Table 7.1, the absolute error ($\infty$-norm) of the serial integration is at most `abstol` $= 5\cdot 10^{-4}$ for all diffusion coefficients $\alpha \in \{0.01, 0.1, 1\}$ and frequencies $f \in \{1, 10, 100\}$ of our test set. Note that $\alpha$ and $f$ determine the stiffness of the linear term $A\mathbf{u}(t)$ and the source term $\mathbf{g}(t)$, respectively. The parameters in this problem are chosen such that the solutions $u(t,x)$ are of comparable magnitude over the whole time range $[0,T]$ (cf. Figure 7.1), so that comparing the absolute error tolerances is indeed meaningful. We have also tabulated the serial integration times $\tau_0$ (as expected, these are roughly proportional to $1/\Delta t_0$).

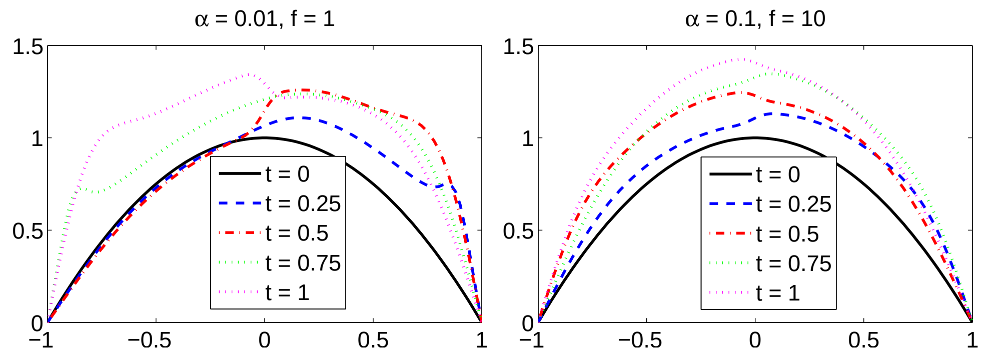

FIG. 7.1. *Solutions $u(t,x)$ of our heat equation example on the spatial domain $[-1,1]$, evaluated at the employed time grid and for two different parameter combinations $\alpha$ and $f$.*

For our paraexp algorithm we have partitioned the interval $[0,T]$ into $p = 4$ subintervals of equal length $\Delta T = 1/p$, and we have computed the solution $\mathbf{u}(t)$ at all time points $T_j = j\Delta T$ $(j = 1,\ldots,p)$. In view of Remark 2, we have chosen constant Runge–Kutta time steps of size $\Delta t_1 = \Delta t_0/p^{1/q}$ to achieve about the same absolute error `abstol` as serial integration. In Table 7.1 we show the maximal computation time $\max(\tau_1)$ for the subproblems of Type 1 among all processors.

For the subproblems of Type 2 we have used the restricted-denominator Arnoldi method with shift $\sigma = 5.3$, in combination with the $\infty$-norm error estimate (3.3) for an absolute accuracy of $10^{-4}$ (for more details on the selection of $\sigma$ we refer to van den Eshof & Hochbruck [20]) The time stepping was performed as illustrated in the scheme (5.2). In Table 7.1 we list the maximal computation time $\max(\tau_2)$ for all subproblems of Type 2 among all processors. Note that the computations of Type 2 are relatively fast compared to the integration times $\tau_1$, and the computation time is independent of the diffusion coefficient $\alpha$ (since the restricted-denominator Arnoldi method can be viewed as an *implicit* integrator). We also remark that we have *not* reused the UMFPACK LU factorization of the matrix $(I - \Delta T\, A/\sigma)$ among all restricted-denominator Arnoldi iterations, because this is only possible (without degradation of accuracy) if all time intervals have equal length $\Delta T$.

The errors of the final solutions computed with our parallel algorithm are shown in the second-last column of Table 7.1, and they are all below the errors for the serial integration. This indicates that our choice for the step size $\Delta t_1$ is reasonable. As an estimate for the parallel speedup and efficiency (neglecting the cost for the final summation of the subproblems) we have used

$$\text{speedup} = \frac{\tau_0}{\max(\tau_1 + \tau_2)} \quad \text{and} \quad \text{efficiency} = \frac{\text{speedup}}{p}.$$

Note how the parallel speedup of our algorithm increases with frequency $f$, because smaller time steps are required to integrate the inhomogeneity accurately. Because high parallel efficiencies of at least 50 % are obtained for all nine tests, we have not attempted to optimize the number of processors and the time partitioning.

| $\alpha$ | $f$ | serial | | parallel | | | effi- |
|---|---|---|---|---|---|---|---|
| | | $\tau_0$ | error | $\max(\tau_1)$ | $\max(\tau_2)$ | error | ciency |
| 0.01 | 1 | 4.97e−02 | 3.01e−04 | 1.58e−02 | 9.30e−03 | 2.17e−04 | 50 % |
| 0.01 | 10 | 2.43e−01 | 4.14e−04 | 7.27e−02 | 9.28e−03 | 1.94e−04 | 74 % |
| 0.01 | 100 | 2.43e+00 | 1.73e−04 | 7.19e−01 | 9.26e−03 | 5.68e−05 | 83 % |
| 0.1 | 1 | 4.85e−01 | 2.24e−05 | 1.45e−01 | 9.31e−03 | 5.34e−06 | 79 % |
| 0.1 | 10 | 4.86e−01 | 1.03e−04 | 1.45e−01 | 9.32e−03 | 9.68e−05 | 79 % |
| 0.1 | 100 | 2.42e+00 | 1.29e−04 | 7.21e−01 | 9.24e−03 | 7.66e−05 | 83 % |
| 1 | 1 | 4.86e+00 | 7.65e−08 | 1.45e+00 | 9.34e−03 | 1.78e−08 | 83 % |
| 1 | 10 | 4.85e+00 | 8.15e−06 | 1.45e+00 | 9.33e−03 | 5.40e−07 | 83 % |
| 1 | 100 | 4.85e+00 | 3.26e−05 | 1.44e+00 | 9.34e−03 | 2.02e−05 | 84 % |

TABLE 7.1
*Serial and parallel performance with $p = 4$ processors for the 1D heat equation with oscillating source term when the diffusion coefficient $\alpha$ and the source frequency $f$ are varied. The serial integrator is classical Runge–Kutta, and the exponential propagator is the restricted-denominator Arnoldi method.*

**7.2. A wave problem.** We consider the following wave equation

$$\begin{aligned}
\partial_{tt}u(t,x) &= \alpha^2\partial_{xx}u(t,x) + g(t,x) \qquad\qquad \text{on } x \in (0,1),\\
u(t,0) &= u(t,1) = 0,\\
u(0,x) &= 0,\\
u'(0,x) &= 0,\\
g(t,x) &= h\max\{1-|c-x|/w, 0\}, \quad \text{where } c = .5 + (.5-w)\sin(2\pi f t),
\end{aligned}$$

with the parameters $\{h, c, w\}$ chosen as in the previous example. We discretize this problem with $N = 100$ interior finite-difference nodes, resulting in a $200\times 200$ problem of the form (1.1) with

$$A = \begin{bmatrix} O & I \\ D & O \end{bmatrix} \quad \text{and} \quad \mathbf{g}(t) = \begin{bmatrix} \mathbf{0} \\ \widetilde{\mathbf{g}}(t) \end{bmatrix},$$

where $D = \alpha^2(N+1)^2\,\mathrm{tridiag}(1,-2,1) \in \mathbb{R}^{N\times N}$ and $\widetilde{\mathbf{g}}(t) \in \mathbb{R}^N$ is the spatial discretization of $g(t,x)$.

This problem is integrated over the time interval $[0, T = 1]$. For the serial integration we use the classical Runge–Kutta method of order $q = 4$ with a constant step size

$$\Delta t_0 = \min\{5\cdot 10^{-4}/\alpha, 1.5\cdot 10^{-3}/f\},$$

(note that, in contrast to the previous example, the spectral interval of $A$ is now on the imaginary axis and increases like $\alpha$). As shown in Table 7.2, the absolute error ($\infty$-norm) of the serial integration is at most `abstol` $= 5\cdot 10^{-4}$ for all squared propagation speeds $\alpha^2 \in \{0.1, 1, 10\}$ and frequencies $f \in \{1, 5, 25\}$ of our test set. Note that these parameters determine the stiffness of the linear term $A\mathbf{u}(t)$ and the source term $\mathbf{g}(t)$, respectively. We have also tabulated the serial integration times $\tau_0$ (as expected, these are roughly proportional to $1/\Delta t_0$).

For our paraexp algorithm we have partitioned the interval $[0, T]$ into $p = 8$ subintervals of equal length $\Delta T = 1/p$, and we have computed the solution $\mathbf{u}(t)$ at all time points $T_j = j\Delta T$ ($j = 1, \ldots, p$). As in the previous example, we have chosen

| $\alpha^2$ | $f$ | serial | | parallel | | | effi- |
|---|---|---|---|---|---|---|---|
| | | $\tau_0$ | error | $\max(\tau_1)$ | $\max(\tau_2)$ | error | ciency |
| 0.1 | 1 | 2.54e−01 | 3.64e−04 | 4.04e−02 | 1.48e−02 | 2.64e−04 | 58 % |
| 0.1 | 5 | 1.20e+00 | 1.31e−04 | 1.99e−01 | 1.39e−02 | 1.47e−04 | 71 % |
| 0.1 | 25 | 6.03e+00 | 4.70e−05 | 9.83e−01 | 1.38e−02 | 7.61e−05 | 76 % |
| 1 | 1 | 7.30e−01 | 1.56e−04 | 1.19e−01 | 2.70e−02 | 1.02e−04 | 63 % |
| 1 | 5 | 1.21e+00 | 4.09e−04 | 1.97e−01 | 2.70e−02 | 3.33e−04 | 68 % |
| 1 | 25 | 6.08e+00 | 1.76e−04 | 9.85e−01 | 2.68e−02 | 1.15e−04 | 75 % |
| 10 | 1 | 2.34e+00 | 6.12e−05 | 3.75e−01 | 6.31e−02 | 2.57e−05 | 67 % |
| 10 | 5 | 2.31e+00 | 4.27e−04 | 3.73e−01 | 6.29e−02 | 2.40e−04 | 66 % |
| 10 | 25 | 6.09e+00 | 4.98e−04 | 9.82e−01 | 6.22e−02 | 3.01e−04 | 73 % |

TABLE 7.2

*Serial and parallel performance with $p = 8$ processors for the 1D wave equation with oscillating source term when the propagation speed $\alpha$ and the source frequency $f$ are varied. The serial integrator is classical Runge–Kutta, and the exponential propagator is the polynomial Chebyshev method.*

constant Runge–Kutta time steps of size $\Delta t_1 = \Delta t_0/p^{1/q}$. In Table 7.2 we show the maximal computation time $\max(\tau_1)$ for the subproblems of Type 1 among all processors.

For the subproblems of Type 2 we have used a polynomial Chebyshev expansion of $\exp(tz)$ on the spectral interval of $A$, which we have bounded by

$$\Lambda(A) \subset 2\alpha(N+1)\cdot[-i,+i].$$

The coefficients $\beta_j$ of the Chebyshev expansion were computed via FFT up to order 1000 and then truncated to an index $n$ such that $\sum_{j=n}^{1000}|\beta_j| < 10^{-5}$. In Table 7.2 we list the maximal computation time $\max(\tau_2)$ for all subproblems of Type 2 among all processors. Note how the computations of Type 2 are relatively fast compared to the integration times $\tau_1$, but now the computation time $\tau_2$ is increasing with the propagation speed $\alpha$ (as the spectral interval of $A$ extends increasingly far along the imaginary axis). Still, even with as many as 8 processors, very satisfactory speedups are achieved. In particular, the parallel efficiency of paraexp is above 50 % in all test cases, and therefore exceeds the largest possible parallel efficiency of the algorithm described in [21] when more than 1 iteration is required (as is typically the case in practice).

**7.3. An advection-dominated problem.** We now consider the 2D advection–diffusion equation

$$\begin{aligned}
\partial_t u &= \frac{1}{\mathrm{Pe}}\Delta u - \mathbf{a}\cdot\nabla u && \text{in } \Omega = (-1,1)\times(0,1),\\
u &= 1-\tanh(\mathrm{Pe}) && \text{on } \Gamma_0,\\
u &= c(1+\tanh((2x+1)\mathrm{Pe})) && \text{with } c = \max\{0,\cos(2\pi f t)\}^{1/3} \text{ on } \Gamma_{\mathrm{in}},\\
\frac{\partial u}{\partial n} &= 0 && \text{on } \Gamma_{\mathrm{out}},\\
u(\mathbf{x},0) &= u_0(\mathbf{x}) && \text{in } \Omega,
\end{aligned}$$

which is a popular benchmark for discretizations of advection-dominated problems (cf. [52]). The advective field is given by

$$\mathbf{a}(x,y) = \begin{bmatrix} 2y(1-x^2) \\ -2x(1-y^2) \end{bmatrix}, \qquad (x,y) \in \Omega,$$

and the boundary $\Gamma = \partial\Omega$ is divided into the inflow boundary $\Gamma_{\text{in}} := [-1,0] \times \{0\}$ (the inflow is turned on and off periodically with frequency $f$), the outflow boundary $\Gamma_{\text{out}} := [0,1] \times \{0\}$ and the remaining portion $\Gamma_0$, see Figure 7.2. The Péclet number Pe is a non-dimensional parameter describing the strength of advection relative to diffusion and therefore also how far the discrete operators are from symmetric. The finite-element space discretization of the advection–diffusion operator with $\text{Pe} = 100$ yields the linear initial-value problem

$$M\mathbf{u}'(t) = K\mathbf{u}(t) + \mathbf{g}(t), \qquad \mathbf{u}(0) = \mathbf{u}_0,$$

with nonsymmetric matrices $K$, $M \in \mathbb{R}^{N\times N}$ of size $N = 4056$ and a source term $\mathbf{g}(t) \in \mathbb{R}^N$ resulting from the inflow boundary condition, for which we have set the frequency $f = 1.5$.

The serial integration of this problem was performed by MATLAB's adaptive integrator `ode15s` with an absolute error tolerance `abstol` $= 10^{-3}$. The computation time $\tau_0$ required for this integration of this problem over the interval $[0, T = 2]$ is shown in Table 7.3. We have provided the integrator with the exact Jacobian $K$ and the constant mass matrix $M$. In this example we test the performance of our paraexp algorithm for an equispaced and an optimized time grid consisting of $p = 8$ subintervals. The time grid optimization was done by an initial serial integration with relaxed absolute accuracy of $10^{-1}$, which required about 7 seconds.

The minimal and maximal computation times $\tau_1$ for the integrations of Type 1 in paraexp are compared in Table 7.3. Note that there is a large gap between $\min(\tau_1)$ and $\max(\tau_1)$ for the equispaced time grid, which is caused by the fact that the source term is only active on some of the intervals of the time partitioning. This causes a large variation in computation time required by the adaptive time-stepping method, and leads to idle processors.

For the exponential propagations we have used the restricted-denominator Arnoldi method with rather arbitrary shift $\xi = 50$, and we require that the error estimate (3.3) is below $10^{-4}$. In average, one of these propagations takes as few as $\text{mean}(\tau_2) = 0.3$ seconds. Although the optimized time grid obviously results in some improvement of the parallel efficiency, the gap between $\min(\tau_1)$ and $\max(\tau_1)$ could still be improved by also taking into account the scheduling of Type 2 propagations. However, a parallel efficiency above 50 % with 8 processors seems to be quite satisfactory already.

## 8. Summary.

We have described and analyzed a new method for the parallel integration of linear initial-value problems, called paraexp. Paraexp performs particularly well if the inhomogeneity is difficult enough to integrate, and it allows one to reuse any existing serial integration method. For achieving a high parallel speedup it is essential that the exponential propagations are fast compared to the serial integrations, and this can be achieved by using a near-optimal Krylov method for the matrix exponential. We have shown that the parallel speedup also depends on the order of the serial integrator. In our numerical experiments we have typically achieved parallel efficiencies above 50 % with a modest number of processors. This is above the maximal achievable parallel efficiency of the (Krylov-enhanced) parareal algorithm, in

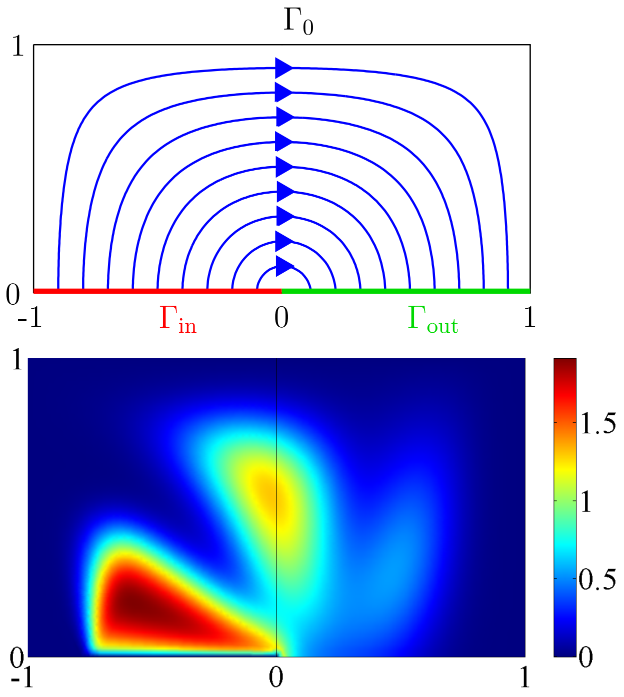

FIG. 7.2. *Advective field (above) and solution at time $t = 1.5$ (below) of the advection–diffusion problem with diffusion coefficient $\mathrm{Pe}^{-1} = 0.01$ and frequency $f = 1.5$.*

| | equispaced time | with load balancing |
|---|---|---|
| $\tau_0$ | 24.1 s | (23.7 + 7) s |
| serial error | 1.2e−03 | 8.3e−04 |
| min($\tau_1$) | 2.6 s | 2.6 s |
| max($\tau_1$) | 7.7 s | 4.9 s |
| mean($\tau_2$) | 0.3 s | 0.3 s |
| parallel err. | 4.7e−04 | 3.1e−04 |
| efficiency | 36.9 % | 58.3 % |

TABLE 7.3
*Serial and parallel performance with $p = 8$ processors for the advection–diffusion equation. The serial integrator is `ode15s`, the exponential propagator is the restricted-denominator Arnoldi method. For computing the parallel efficiency, we have added the 7 seconds of integration time required for computing an optimized time grid to the computation time of the serial integrator.*

particular the algorithm in [21] for linear initial-value problems, under the condition that these algorithms require more than 1 iteration (which is typically the case in practice). Note that paraexp is non-iterative and requires a single communication between processors at the end of the algorithm only, thereby minimizing communication.

We would like to add that it is straightforward to make use of additional parallelism that may be available within the Type 1 and 2 integrators of paraexp. Since the parallel efficiency of paraexp degrades with the number of processors (though moderately), additional parallel efficiency may be available by combining different levels of parallelism. This will be subject of future work.

An interesting class of linear problems involves slowly varying matrices, i.e.,

$$M(t)\mathbf{u}'(t) = K(t)\mathbf{u}(t) + \mathbf{g}(t), \qquad \mathbf{u}(0) = \mathbf{u}_0, \qquad t \in [0, T].$$

There exist exponential integrators for such problems (Magnus-type integrators, see [35]), and it should be possible to use these methods as Type 2 integrators in paraexp. The evaluation of the efficiency of this approach is subject of ongoing research.

**Acknowledgments.** We are grateful to the two anonymous referees for their comments and suggestions that improved this paper considerably. We would like to thank Marlis Hochbruck and Ernst Hairer for useful discussions. We are also grateful to Kurt Janssens for his comments.

## REFERENCES

[1] M. Afanasjew, M. Eiermann, O. G. Ernst and S. Güttel, *Implementation of a restarted Krylov subspace method for the evaluation of matrix functions*, Linear Algebra Appl., 429 (2008), pp. 2293–2314.

[2] G. Bal, *On the Convergence and the Stability of the Parareal Algorithm to Solve Partial Differential Equations*, Domain Decomposition Methods in Science and Engineering XI, Lecture Notes in Computational Science and Engineering 40, Springer-Verlag, 2005, pp. 425–432.

[3] L. Banjai and D. Petersheim, *Parallel multistep methods for linear evolution problems*, Technical Report 26, MPI for Mathematics in the Sciences, Leipzig, 2009.

[4] B. Beckermann and L. Reichel, *Error estimation and evaluation of matrix functions via the Faber transform*, SIAM J. Numer. Anal., 47 (2009), pp. 3849–3883.

[5] L. Bergamaschi and M. Vianello, *Efficient computation of the exponential operator for large, sparse, symmetric matrices*, Numer. Linear Algebra Appl., 7 (2000), pp. 27–45.

[6] H. Berland, B. Skaflestad and W. M. Wright *EXPINT–A MATLAB package for exponential integrators*, ACM Trans. Math. Software, 33 (2007), pp. 1–17.

[7] D. Bini, *Parallel solution of certain Toeplitz linear systems*, SIAM J. Comput., 13 (1984), pp. 268–276.

[8] K. Burrage, *Parallel and Sequential Methods for Ordinary Differential Equations*, Oxford University Press, USA, 1995.

[9] M. Caliari, M. Vianello and L. Bergamaschi, *Interpolating discrete advection–diffusion propagators at Leja sequences*, J. Comput. Appl. Math., 172 (2004), pp. 79–99.

[10] W. J. Cody, G. Meinardus and R. S. Varga, *Chebyshev rational approximations to $e^{-x}$ in $[0, +\infty)$ and applications to heat-conduction problems*, J. Approx. Theory, 2 (1969), pp. 50–65.

[11] M. Crouzeix, *Numerical range and functional calculus in Hilbert space*, J. Funct. Anal., 244 (2007), pp. 668–690.

[12] A. J. Davies, J. Mushtaq and L. E. Radford, *The numerical Laplace transform solution method on a distributed memory architecture*, Adv. High Perform. Comput., 3 (1997), pp. 245–254.

[13] V. L. Druskin and L. A. Knizhnerman, *Two polynomial methods of calculating functions of symmetric matrices*, USSR Comput. Maths. Math. Phys., 29 (1989), pp. 112–121.

[14] V. L. Druskin and L. A. Knizhnerman, *Extended Krylov subspaces: Approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 775–771.

[15] V. Druskin, L. Knizhnerman and M. Zaslavsky, *Solution of large scale evolutionary problems using rational Krylov subspaces with optimized shifts*, SIAM J. Sci. Comp., 31 (2009), pp. 3760–3780.

[16] V. Druskin, C. Lieberman and M. Zaslavsky, *On adaptive choice of shifts in rational Krylov subspace reduction of evolutionary problems*, SIAM J. Sci. Comput., 32 (2010), pp. 2485–2496.

[17] M. Eiermann and O. G. Ernst, *A restarted Krylov subspace method for the evaluation of matrix functions*, SIAM J. Numer. Anal., 44 (2006), pp. 2481–2504.

[18] H. C. Elman, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT, 1982.

[19] T. Ericsson, *Computing functions of matrices using Krylov subspace methods*, Technical Report, Chalmers University of Technology, Department of Computer Science, Göteborg, Sweden, 1990.

[20] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput., 27 (2006), pp. 1438–1457.

[21] C. Farhat, J. Cortial, C. Dastillung and H. Bavestrello, *Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses*, Int. J. Numer. Meth. Engng., 67 (2006), pp. 697–724.

[22] E. Gallopoulos and Y. Saad, *On the parallel solution of parabolic equations*, Proc. 1989 ACM Internat. Conf. on Supercomputing, Heraklion, Greece, 1989, pp. 17–28.

[23] M. J. Gander and E. Hairer, *Nonlinear convergence analysis for the parareal algorithm*, Domain Decomposition Methods in Science and Engineering XVII, Lecture Notes in Computational Science and Engineering 60, Springer-Verlag, 2007, pp. 45–56.

[24] M. J. Gander and M. Petcu, *Analysis of a Krylov subspace enhanced parareal algorithm*, ESAIM Proc., 25 (2008), pp. 114–129.

[25] M. Gander and S. Vandewalle, *On the superlinear and linear convergence of the parareal algorithm*, Domain Decomposition Methods in Science and Engineering XVI, Lecture Notes in Computational Science and Engineering 55, Springer-Verlag, 2007, pp. 291–298.

[26] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, USA, 1996.

[27] I. P. Gavrilyuk and V. L. Makarov, *Exponentially convergent parallel discretization methods for the first order evolution equations*, Comput. Meth. Appl. Math., 1 (2001), pp. 333–355.

[28] I. P. Gavrilyuk and V. L. Makarov, *Exponentially convergent algorithms for the operator exponential with applications to inhomogeneous problems in Banach spaces*, SIAM J. Numer. Anal., 43 (2005), pp. 2144–2171.

[29] S. Güttel, *Rational Krylov Methods for Operator Functions*, Ph.D. Thesis, Technische Universität Bergakademie Freiberg, Germany, 2010.

[30] S. Güttel, *Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection*, To appear in GAMM Mitteilungen, 2013.

[31] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*, Springer-Verlag, Second revised edition, 1996.

[32] N. J. Higham, *Functions of Matrices. Theory and Computation*, SIAM, Philadelphia, USA, 2008.

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

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

[35] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 2010, pp. 209–286.

[36] W. Huisinga, L. Pesce, R. Kosloff and P. Saalfrank, *Faber and Newton polynomial integrators for open-system density matrix propagation*, J. Chem. Phys., 110 (1999), pp. 5538–5547.

[37] L. A. Knizhnerman, *Calculation of functions of unsymmetric matrices using Arnoldi's method*, USSR Comput. Maths. Math. Phys., 31 (1991), pp. 1–9.

[38] J.-L. Lions and Y. Maday and G. Turinici, *A parareal in time discretization of PDE's*, C. R. Acad. Sci. Paris, 332 (2001), pp. 661–668.

[39] M. López-Fernández, C. Lubich, C. Palencia and A. Schädle, *Fast Runge-Kutta approximation of inhomogeneous parabolic equations*, Numer. Math., 102 (2005), pp. 277–291.

[40] M. L. Minion, *A Hybrid Parareal Spectral Deferred Corrections Method*, Comm. App. Math. and Comp. Sci., 5 (2011), pp. 265–301.

[41] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.

[42] I. Moret and P. Novati, *The computation of functions of matrices by truncated Faber series*, Numer. Funct. Anal. Optim., 22 (2001), pp. 697–719.

[43] I. Moret and P. Novati, *RD-rational approximations of the matrix exponential*, BIT, 44 (2004), pp. 595–615.

[44] D. L. Powers and R. Jeltsch, *Problem 74-5: On the norm of a matrix exponential*, SIAM Rev., 17 (1975), pp. 174–176.

[45] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, ICS-Preprint No. 2011-05, Lugano, 2011.

[46] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Linear Algebra Appl., 58 (1984), pp. 391–405.

[47] A. Ruhe, *Rational Krylov algorithms for nonsymmetric eigenvalue problems*, IMA Vol. Math. Appl., 60 (1994), pp. 149–164.

[48] Y. Saad, *Analysis of some Krylov subspace approximations to the exponential operator*, SIAM

J. Numer. Anal., 29 (1992), pp. 209–228.

[49] T. Schmelzer and L. N. Trefethen, *Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals*, Electron. Trans. Numer. Anal., 29 (2007), pp. 1–18.

[50] D. Sheen, I. H. Sloan and V. Thomée, *A parallel method for time-discretization of parabolic problems based on contour integral representation and quadrature*, Mathematics of Computation, 69 (1999), pp. 177–195.

[51] D. Sheen, I. H. Sloan and V. Thomée, *A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature*, IMA Journal of Numerical Analysis, 23 (2003), pp. 269–299.

[52] R. M. Smith and A. G. Hutton, *The numerical treatment of advection: A performance comparison of current methods*, Numer. Heat Transfer, 5 (1982), pp. 439–461.

[53] A. Talbot, *The accurate numerical inversion of Laplace transforms*, J. Inst. Math. Appl., 23 (1979), pp. 97–120.

[54] L. N. Trefethen and M. Embree, *Spectra And Pseudospectra: The Behavior of Nonnormal Matrices And Operators*, Princeton University Press, 2005.

[55] L. N. Trefethen, J. A. C. Weideman and T. Schmelzer, *Talbot quadratures and rational approximations*, BIT, 46 (2006), pp. 653–670.

[56] C. Van Loan, *The sensitivity of the matrix exponential*, SIAM J. Numer. Anal., 14 (1977), pp. 971–981.