SIAM J. SCI. COMPUT.
Vol. 40, No. 3, pp. B684–B705

AN EXPONENTIAL TIME INTEGRATOR FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATION*

GIJS L. KOOIJ†, MIKE A. BOTCHEV‡, AND BERNARD J. GEURTS§

Abstract. We present an exponential time integration method for the incompressible Navier–Stokes equation. An essential step in our procedure is the treatment of the pressure by applying a divergence-free projection to the momentum equation. The differential-algebraic equation for the discrete velocity and pressure is then reduced to a conventional ordinary differential equation that can be solved with the proposed exponential integrator. A promising feature of exponential time integration is its potential time parallelism within the Paraexp algorithm. We demonstrate that our approach leads to parallel speedup assuming negligible parallel communication.

Key words. exponential time integration, incompressible Navier–Stokes equation, block Krylov subspace method, parallel in time

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

DOI. 10.1137/17M1121950

1. Introduction.

With today's trend toward massively parallel computing, there is a growing interest in parallel-in-time simulations. For the numerical solution of partial differential equations, a parallellization in time could realize additional speedup, when, for example, the maximum speedup with a parallellization in space is approached, e.g., see [40, 56]. Parallel-in-time simulations received increased attention after the introduction of the Parareal method [29, 42]; for earlier work see, e.g., [8, 15, 61, 62]. So far, existing methods have not proven themselves to be particularly efficient for parallel-in-time simulations of flows at high Reynolds numbers. In this article, we introduce a time integration method that paves the way to parallel-in-time simulation of incompressible fluid flows at high Reynolds numbers.

Theoretical studies have shown that the parallel efficiency of Parareal for hyperbolic problems is in practice typically well below the best efficiency one can expect from Parareal [26, 29]. Also, numerical experiments for the Navier–Stokes equation at high Reynolds numbers report limited performance [24, 57]. However, in [18] speedup was achieved for high Reynolds simulations for which a turbulence model was deployed. For some hyperbolic problems a stabilization of Parareal can be applied [9, 14, 28, 51]. There are also reports of cases in which Parareal succeeds, even for problems with little or no dissipation. We mention simulations of, e.g., Hamiltonian systems [13], highly oscillatory PDEs [35], plasma turbulence [54], and gravitational collapse [41].

*Submitted to the journal's Computational Methods in Science and Engineering section March 21, 2017; accepted for publication (in revised form) January 9, 2018; published electronically May 1, 2018.
http://www.siam.org/journals/sisc/40-3/M112195.html

Funding: The first author's work was supported financially by FOM, Foundation for Fundamental Research on Matter, part of NWO, the Netherlands Organisation for Scientific Research. The second author's work was supported by Russian Science Foundation grant 17-11-01376.

†Multiscale Modeling and Simulation, Faculty of EEMCS, University of Twente, Enschede, Netherlands (g.l.kooij@utwente.nl).

‡Mathematics of Computational Science, Faculty of EEMCS, University of Twente, Enschede, Netherlands, and Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Moscow, Russia (botchev@kiam.ru).

§Multiscale Modeling and Simulation, Faculty of EEMCS, University of Twente, Enschede, Netherlands, and Fluid Dynamics Laboratory, Faculty of Applied Physics, Eindhoven University of Technology, Eindhoven, Netherlands (b.j.geurts@utwente.nl).

An interesting alternative, the Paraexp method, has been introduced in [27] for the time-parallel solution of linear initial-value problems (IVPs). With Paraexp, the original problem is decoupled into nonhomogeneous and homogeneous subproblems. Parallel speedup is based on the observation that the homogeneous subproblems can be solved very fast with exponential integrators. These are time integration methods based on the exact integration of linear IVPs. Exponential integrators require efficient algorithms to compute the matrix exponential or its product with a vector. There are many methods for computing the matrix exponential. Particularly, Krylov subspace methods prove to be very suitable for the implementation of exponential integrators [16, 27, 48]. A detailed overview of exponential integrators is given in [36].

An extension of Paraexp to nonlinear IVPs has been introduced in [39]. Preliminary tests show that parallel speedup is realistic for the viscous Burgers equation, even for low viscosity coefficients. The approach in [39] is based on the exponential block Krylov (EBK) method [4]. A difference with the original formulation of Paraexp is that the exponential integrator, here the EBK method, is used to solve both the nonhomogeneous and the homogeneous subproblems. This unified approach is demonstrated to be effective, as the EBK method can also be very competitive in solving nonhomogeneous subproblems compared to conventional time integration methods.

The application of exponential integrators to fluid dynamics is not straightforward. The incompressible Navier–Stokes equation, discretized in space, is a differential-algebraic equation where a time derivative for the pressure is absent. The continuity equation acts as a constraint equation that imposes a divergence-free condition on the velocity field. The pressure is determined such that the velocity field remains divergence-free. Consequently, special care needs to be taken with exponential integration for the advancement of the pressure in time. One possible approach is to reformulate the governing equations by treating the pressure with a divergence-free projection of the Navier–Stokes equation. This reformulation gives a differential equation for the velocity field that can be solved with Krylov-based exponential integrators; see [17, 50]. Other approaches for Krylov-based exponential integration in the context of fluid dynamics include the method of pseudocompressibility to find steady state solutions [53] and a method for the fully compressible Navier–Stokes equation [55].

The EBK method and its potential parallelization with Paraexp have been demonstrated for the viscous Burgers' equation in [39]. Its application to incompressible flows is not trivial because of the structure of the governing equations. In this paper, we discuss how the EBK method can be extended to the incompressible Navier–Stokes equation. This also paves the way for parallel-in-time simulations of incompressible flows with Paraexp. We follow the approach of [17, 50] and treat the pressure by a divergence-free projection. The new time integration method is tested in several numerical experiments including the Taylor–Green vortex and a lid-driven cavity flow. We find that the EBK method can be applied successfully to incompressible flows and also in cases with rather low viscosity. Furthermore, we show that the EBK method can be used within the time-parallel framework outlined in [39]. We provide a simplified model to analyze how much speedup is feasible with the EBK method for parallel-in-time simulations of the incompressible Navier–Stokes equation. This analysis indicates that with our current implementation of the EBK method a moderate parallel speedup can indeed be expected. This method could provide additional speedup on top of the speedup obtained with a conventional parallellization in space only.

The paper is organized as follows. In section 2, we explain the basic principle of the EBK method and discuss how it can be applied to the incompressible Navier–

Stokes equation. In section 3, we test the method on a number of test cases, including the Taylor–Green vortex and lid-driven cavity flow. The potential parallel speedup of the EBK method is explored in section 4. Finally, the discussion and conclusions are presented in section 5.

2. Exponential time integration. In this section, we describe briefly the EBK method and discuss its application to the incompressible Navier–Stokes equation.

2.1. Exponential block Krylov method. Consider the IVP on $t \in [0, T]$,

$$\mathbf{y}'(t) + A\mathbf{y}(t) = \mathbf{b}(t), \quad \mathbf{y}(0) = \mathbf{y}_0, \tag{2.1}$$

where $\mathbf{y} \in \mathbb{R}^n$ is the unknown function, $\mathbf{b}(t) \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n\times n}$. We are mainly interested in IVPs in the context of the method of lines, in which hyperbolic or parabolic partial differential equations are discretized in the spatial dimensions first. The exact solution of (2.1) is given by the variation-of-constants formula,

$$\mathbf{y}(t) = \exp(-tA)\mathbf{y}_0 + \int_0^t \exp(-(t-\tau)A)\mathbf{b}(\tau)\,\mathrm{d}\tau. \tag{2.2}$$

Conventional time integration methods, e.g., the Euler method, use low-order approximations based on finite differences to the matrix exponential function. Exponential integrators avoid such approximations by direct evaluation of the matrix exponential function. In this study, we use the EBK method to solve (2.1). Exponential integrators are in general attractive due to their excellent stability and accuracy properties [36]. The EBK method is demonstrated to be particularly competitive for solving (2.1) compared to implicit schemes, and also with respect to other exponential integrators [39]. The EBK method is based on a so-called block Krylov subspace, which is generated by the action of a matrix on *multiple* vectors simultaneously. For details of the EBK method, we refer the reader to [4].

An essential step for constructing the block Krylov subspace is a polynomial approximation of the source term of the form

$$\mathbf{b}(t) \approx U\mathbf{q}(t), \tag{2.3}$$

where $U \in \mathbb{R}^{n\times m}$, and $\mathbf{q}(t) \in \mathbb{R}^m$ is a polynomial function of t. It is crucial for successful applications of EBK that m is not too large, so that (2.3) is in fact a low rank approximation; see also [4]. A large m, which is the dimension of the block in the Krylov subspace, is undesirable. In general, the dimension of a Krylov subspace should remain small with respect to the original system for good computational efficiency. Approximation (2.3) can be efficiently constructed in many ways [31, 47]. In this paper, we use truncated singular value decomposition (SVD) combined with a piecewise polynomial function $\mathbf{q}(t)$ to obtain (2.3). The error of the approximation (2.3) on a given interval $[0, T]$ can be reduced by retaining more singular values (increasing m). If m is found to be too large, then we can decrease the time interval $[0, T]$ and repeat the construction of (2.3) which often leads to a much smaller m. In principle, the approximation error in (2.3) can be arbitrarily small, assuming $\mathbf{b}(t)$ is smooth (see, for example, [58, section 2.4] for convergence properties of spline interpolation).

We define s nodes on the interval $t \in [0, T]$, $0 = t_0 < t_1 < \cdots < t_s = T$. The polynomial approximation of $\mathbf{b}(t)$ is based on the matrix of samples $\widetilde{B} = [\mathbf{b}_1 \dots \mathbf{b}_s]$, where $\mathbf{b}_i := \mathbf{b}(t_i)$. For optimal efficiency of the EBK method, we use a truncated

SVD of $\widetilde{B}$, which reduces the dimension of the block size m substantially with respect to the original dimension of $\widetilde{B}$. From the SVD, we find an approximation (2.3). Here, $U\mathbf{q}(t)$ represents the m leading singular vectors of the SVD of $\mathbf{b}(t)$. Based on a block of vectors U instead of a single vector, we can define the block Krylov subspace

$$\mathcal{K}_l(A, U) := \text{span}\left\{U, AU, A^2U, \ldots, A^{l-1}U\right\}, \tag{2.4}$$

where l is the Krylov iteration index. A linear basis of the block Krylov subspace is generated by the block Arnoldi or by the block Lanczos process, if A is symmetric (see [52]). After the basis has been computed, the original IVP can be projected onto the block Krylov subspace using the orthogonality of the basis vectors [4]. The dimensions of the projected IVP are typically much smaller than those of the original problem ($lm \ll n$). Because of its strongly reduced problem size, the projected IVP can be solved cheaply and accurately (to any desired tolerance) with a general ODE solver (see "Method 5" in [48]). This step is performed at every Krylov iteration. The Krylov iterations, incrementing l at every iteration, stop when the exponential residual meets the stopping criterion [5]. The accuracy of the numerical solution is thus controlled by the residual in the block Krylov process and by the low rank approximation (2.3). The approximation (2.3) on $t \in [0, T]$ can be improved either by taking more samples s or by saving more singular values m.

2.2. Incompressible Navier–Stokes equation. In section 2.1, we have introduced the EBK method for solving the linear IVP (2.1). In this section, we present an algorithm in which the EBK method is used for exponential time integration of the incompressible Navier–Stokes equation. We consider the incompressible Navier–Stokes equation on a d-dimensional space domain Ω ($d = 2$ or $d = 3$) with boundary $\partial\Omega$. The governing equations in $\Omega \times [0, T]$ are

$$\vec{u}_t + (\vec{u} \cdot \nabla)\vec{u} = -\nabla p + \nu\Delta\vec{u}, \quad \vec{u}(0) = \vec{u}_0, \tag{2.5}$$

$$\nabla \cdot \vec{u} = 0, \tag{2.6}$$

with appropriate boundary conditions on $\partial\Omega$. Here, $\vec{u} \in \mathbb{R}^d$ is the velocity vector, p the pressure and ν the kinematic viscosity. We follow the method of lines approach by discretizing in space first. There are many suitable discretization techniques for the Navier–Stokes equation, with popular choices being finite volume and finite element methods [23, 65]. These discretization methods typically result in a semidiscrete system of the general form

$$G\mathbf{u}' + N(\mathbf{u})\mathbf{u} + B^T\mathbf{p} + A_\nu\mathbf{u} = \mathbf{f}, \quad \mathbf{u}(0) = \mathbf{u}^0, \tag{2.7}$$

$$B\mathbf{u} = \mathbf{g}, \tag{2.8}$$

where G is the mass matrix, $N(\mathbf{u})$ is the convective operator, B the divergence operator, B^T the gradient operator, and A_ν the viscous operator. We emphasize that time remains an independent variable and $\mathbf{u} = \mathbf{u}(t)$ is still a function of time with derivative $\mathbf{u}'$. Note that some discretization methods may slightly deviate from the general form. For example, for inf-sup unstable finite element spaces an additional stabilization term is added to the continuity equation [23]. For some discretization methods, it holds that the mass matrix G is the identity matrix. The vectors $\mathbf{f}$ and $\mathbf{g}$ may depend on the boundary conditions on the velocity field and thus are not necessarily zero. We consider problems with n_u degrees of freedom (DOFs) associated with the velocity field components, and n_p with the pressure field, i.e., $(\mathbf{u}, \mathbf{p}) \in \mathbb{R}^{dn_u \times n_p}$.

The DOFs n_u and n_p may differ, depending on the spatial discretization. The pressure and velocity components are located at different gridpoints when, for example, different finite element spaces are used for the velocity and the pressure, or when a staggered grid is used in a finite volume method.

Equations (2.7) and (2.8) form essentially a differential-algebraic equation in which the discretized continuity equation (2.8) can be seen as an algebraic constraint on the velocity field. The differential-algebraic nature of this equation is different than the basic ODE, given by (2.1). We note that the compressible Navier–Stokes equations are less complicated in that sense, because a time derivative is present also in the continuity equation. Therefore existing exponential integrators can be readily applied to compressible Navier–Stokes equations [55], while for incompressible flow this is more challenging.

Equation (2.7) cannot be integrated directly with the EBK method, because of (a) the nonlinear convection term and (b) the algebraic constraint on $\mathbf{u}$ in the form of the discrete continuity equation. We now present an approach on how to handle these issues.

The nonlinearity of the problem can be dealt with by incorporating the EBK method into an iterative procedure [39]. Namely, the nonlinear term is linearized about a state $\bar{\mathbf{u}}$, specified later, as follows:

$$N(\mathbf{u}_{k+1})\mathbf{u}_{k+1} \approx N(\mathbf{u}_k)\mathbf{u}_k + N(\bar{\mathbf{u}})[\mathbf{u}_{k+1} - \mathbf{u}_k], \tag{2.9}$$

where k is the iteration index. Instead of $N(\bar{\mathbf{u}})$, one could use the Jacobian matrix of the nonlinear term as well. In many codes the matrix $N(\mathbf{u})$ is readily available, which makes the implementation of the linearization above easier in practice. Using this linearization the momentum equation becomes

$$G\mathbf{u}'_{k+1} + [A_\nu + N(\bar{\mathbf{u}})]\mathbf{u}_{k+1} + B^T\mathbf{p}_{k+1} = \mathbf{f}_k, \tag{2.10}$$

$$B\mathbf{u}_{k+1} = \mathbf{g}. \tag{2.11}$$

We will work with $\bar{\mathbf{u}}$ as the average of $\mathbf{u}_k$ on $[0, T]$, such that the matrix $[A_\nu + N(\bar{\mathbf{u}})]$ is independent of time, similar to (2.1). It should be noted that $\mathbf{u}_k(t)$ is the solution at iteration k on the complete interval $[0, T]$. In practice, the solution is stored at multiple temporal points as a matrix $U_k = [\mathbf{u}^1 \dots \mathbf{u}^s]$. The iterative process is usually started with a constant vector $\mathbf{u}_0 = \mathbf{u}(0)$, and for $\bar{\mathbf{u}}$, we normally take a time-average $\bar{\mathbf{u}} = \frac{1}{T}\int_0^T \mathbf{u}_k(t)\,dt$. The right-hand side (RHS) $\mathbf{f}_k$ contains the nonlinear remainder,

$$\mathbf{f}_k := \mathbf{f} - [N(\mathbf{u}_k) - N(\bar{\mathbf{u}})]\mathbf{u}_k. \tag{2.12}$$

Next, we deal with the pressure in (2.10). We manipulate (2.10) in such a way that the pressure is eliminated.

Multiplying (2.10) with BG^{-1}, we find

$$B\mathbf{u}'_{k+1} + BG^{-1}\left([A_\nu + N(\bar{\mathbf{u}})]\mathbf{u}_{k+1} + B^T\mathbf{p}_{k+1}\right) = BG^{-1}\mathbf{f}_k. \tag{2.13}$$

To simplify this equation, we use the time derivative of the continuity equation (2.11), which reads

$$B\mathbf{u}'_{k+1} = \mathbf{g}'. \tag{2.14}$$

Substituting relation (2.14) into (2.13) gives us an algebraic equation for the pressure,

$$BG^{-1}B^T\mathbf{p}_{k+1} = \left[BG^{-1}(\mathbf{f}_k - (A_\nu + N(\bar{\mathbf{u}}))\mathbf{u}_k) - B\mathbf{g}'\right], \tag{2.15}$$

which is essentially a discrete Poisson equation, since $BG^{-1}B^T$ represents a discrete Laplace operator. Substituting the solution of (2.15) into (2.10) removes the pressure from the momentum equation,

$$\mathbf{u}'_{k+1} + P[A_\nu + N(\bar{\mathbf{u}})]\mathbf{u}_{k+1} = P\mathbf{f}_k - G^{-1}B^T\left(BG^{-1}B^T\right)^{-1}\mathbf{g}', \tag{2.16}$$

where, to simplify notation, we have introduced the projection operator,

$$P := G^{-1} - G^{-1}B^T\left(BG^{-1}B^T\right)^{-1}BG^{-1}. \tag{2.17}$$

Note that P projects discrete velocity fields onto a (discretely) divergence-free space that satisfies the condition $B\mathbf{u} = 0$. We note that for some discretizations the explicit calculation and storage of G^{-1} is not practical, but in general the action of G^{-1} on a vector is easily computed with direct or iterative linear solvers. The matrix P is also never calculated explicitly for the same reason. Our procedure of treating the pressure in the momentum equation is in the same vein as classic pressure correction or fractional step methods (see, e.g., [10, 34, 37, 59, 63]). In fact, the projection operator in [63] is identical to (2.17) in the case of $G = I$.

The projection P is essentially the discrete equivalent of the Leray projection [11, 25]. It is easy to verify that the projection P has the analogous properties

1. $P^2\mathbf{u} = P\mathbf{u}$ for all $\mathbf{u} \in \mathbb{R}^{dn_u}$,
2. $BP\mathbf{u} = 0$ for all $\mathbf{u} \in \mathbb{R}^{dn_u}$,
3. $P\mathbf{u} = \mathbf{u}$ for all $\left\{\mathbf{u} \in \mathbb{R}^{dn_u} | B\mathbf{u} = 0\right\}$,
4. $PB^T\mathbf{p} = 0$ for all $\mathbf{p} \in \mathbb{R}^{n_p}$.

Equation (2.16) is a linear IVP that can be solved with the EBK method; cf. (2.1). A similar approach for exponential integrators for the incompressible Navier–Stokes equation is given by [50]. In our case, the projection operator P is derived directly from the semidiscretization in (2.7).

At each iteration k the EBK method is employed to find $\mathbf{u}_{k+1}$ by solving (2.16). Therefore, EBK can be seen as an inner iterative process and iterations k as outer iterations. The convergence of the outer iterations has been studied in [46, 66] (in a slightly different context of waveform relaxation methods). The convergence is generally slower if the nonlinear term becomes larger, and it will be investigated whether the convergence is still sufficient for our application [39]. The outer iterations can be stopped if they are converged within a certain tolerance,

$$\|\mathbf{u}_{k+1} - \mathbf{u}_k\|_\infty < \texttt{tol}, \tag{2.18}$$

which not only shows the stagnation in iterations but also controls the nonlinear residual [39]. Generally, the tolerance for the nonlinear residual should be in the same order as the tolerance for the exponential residual, i.e., the stopping criterion for the inner iterations of the EBK methods. After the outer iterations are ended, we form the matrix $U = [\mathbf{u}^1 \dots \mathbf{u}^s]$ that comprises s samples of the solution $\mathbf{u}(t)$ on the interval $[0, T]$. This time interval is typically much larger than the time step size Δt of traditional time integration methods restricted by a Courant–Friedrichs–Lewy (CFL) condition. In numerical experiments in [39], we find $T/\Delta t = \mathcal{O}(10^3)$, for example.

2.3. Other exponential integrators. In the previous section, we have discussed how the incompressible NS equation can be rewritten in a form that fits (2.1). In this case, we have $A = P[A_\nu + N(\bar{\mathbf{u}})]$, where P is a projection matrix. In principle, one could apply several exponential integrators to this equation. A stimulating

overview of existing schemes is given by [36, 43, 44]. We mention, for example, the exponential integrators for stiff systems proposed by [12]. The schemes in [12] are highly suitable for partial differential equations in periodic domains that can be discretized with spectral methods. The resulting matrices are diagonal, which allows a trivial computation of the matrix exponential. These schemes are, however, less suitable for the incompressible Navier–Stokes equation in general domains, where the matrix is generally not diagonal. The method relies on the explicit calculation of the matrix exponential e^{Ah}, where h is a time step size, and the (pseudo)inverse of A, which would be very expensive in terms of computation time and memory consumption.

In our application, we have to deal with the projection matrix P, which we do not wish to calculate explicitly. Krylov-based methods are very attractive in this case, because they allow a matrix-free implementation. In other words, they only require the calculation of the matrix-vector product of A with a vector—not the calculation of the actual matrix itself. Exponential time integration for the Navier–Stokes equation has not received much attention so far, but a few Krylov-based exponential integrators in the context of fluid dynamics have been proposed. An application for the *compressible* Navier–Stokes equations is, for example, discussed in [55]. The solution of the compressible formulation is easier in a sense, because the divergence-free velocity constraint is absent. Steady solutions to the incompressible Navier–Stokes equation can be found using the method of pseudocompressibility [53]. The application to the incompressible Navier–Stokes equation is more complicated. The equations have to be rewritten, because the mass matrix is not invertible. This approach is taken by [17, 50] and is very similar to the one we discussed in section 2.2. The main difference is that in our approach the Navier–Stokes equation is discretized in space first. The projection matrix then follows naturally from rewriting the semidiscrete equations and is consistent with the boundary conditions of the original equations.

An important aspect of the EBK method is the potential parallelization in time based on the Paraexp algorithm, which is demonstrated in [39]. As shown in [39], the maximum parallel efficiency is normally bounded by $1/K_P$, where K_P is the number of iterations of the time-parallel method. That means that if, for example, two iterations are required, the parallel efficiency is already down to at most 50%. Because the EBK method requires iterations anyway because of the nonlinearity, those iterations can be intertwined. The upper bound on the parallel efficiency can then be relaxed to K_S/K_P, where K_S is the number of iterations of the sequential EBK method. In practice, it safe to assume that the parallel version requires more iterations, i.e., $K_P > K_S$. In the ideal case, K_S would be very close to K_P, which would then allow for near-optimal parallel efficiency. This makes the EBK method, in comparison to other exponential integrators, an interesting candidate for parallelization in time.

3. Incompressible flow simulations. In section 2, we covered the basics of the EBK method and how it can be applied to the incompressible Navier–Stokes equation. In this section, we now present several numerical experiments with our time integration method. These experiments involve a Taylor–Green vortex and lid-driven cavity flow.

The governing equations are discretized in space with a finite element method implemented in the MATLAB package IFISS [20, 21]. In our numerical experiments, we are using *Taylor–Hood* Q_2–Q_1 elements. This means that the velocity components are approximated with continuous piecewise quadratic polynomials, and the pressure with continuous piecewise linear polynomials. Taylor–Hood elements are *inf-sup* stable [23].

The projection operator (2.17) requires the solution of a linear system. Because the test and trial functions are nonorthogonal, the mass matrix G is not diagonal in our case. Therefore, it is not practical to explicitly construct the discrete Laplace operator, $BG^{-1}B^T$. Instead of constructing this matrix, we simply solve the block-coupled system,

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^* \\ 0 \end{pmatrix}, \tag{3.1}$$

where $\mathbf{v}, \mathbf{v}^* \in \mathbb{R}^{dn_u}$, and $\boldsymbol{\lambda} \in \mathbb{R}^{n_p}$. The linear system is solved with a direct solver in MATLAB. It is not hard to verify that the solution of (3.1) is equivalent to the action of the projection operator, $\mathbf{v} = P\mathbf{v}^*$, as defined in (2.17). With a block Gaussian elimination, the system can be reduced to

$$\begin{bmatrix} G & B^T \\ 0 & BG^{-1}B^T \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{v}^* \\ BG^{-1}\mathbf{v}^* \end{pmatrix}. \tag{3.2}$$

After back-substitution, we find the solution for $\mathbf{v}$,

$$\mathbf{v} = G^{-1}\mathbf{v}^* - G^{-1}B^T \left(BG^{-1}B^T\right)^{-1} BG^{-1}\mathbf{v}^*. \tag{3.3}$$

This solution is indeed identical to $\mathbf{v} = P\mathbf{v}^*$, according to the definition of P (2.17). If the problem size of (3.1) becomes too large for direct solvers, iterative solvers can be used. In practice, variations of the Uzawa algorithm are a possible choice for solving the saddle-point problem (3.1) by decoupling $\mathbf{v}$ and $\boldsymbol{\lambda}$; see [45]. Alternatively, block-coupled solvers are feasible with efficient preconditioners [1, 2, 19, 22]. For a recent overview see [23].

To ensure that the initial condition satisfies the discrete continuity equation at $t = 0$, the velocity field is projected onto a divergence-free space. We correct the initial condition by solving the linear system

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{pmatrix} \mathbf{u}^0 \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{u}}^0 \\ \mathbf{g}(0) \end{pmatrix}, \tag{3.4}$$

where $\tilde{\mathbf{u}}^0$ is the uncorrected initial condition. As a result, the initial condition satifies $B\mathbf{u}^0 = \mathbf{g}(0)$.

3.1. Taylor–Green vortex. In the first test case, we consider the so-called Taylor–Green vortex, for which an exact solution to the unsteady Navier–Stokes equation is known, given by

$$\vec{u}(x, y, t) = \begin{pmatrix} -\sin(\pi x)\cos(\pi y)\exp(-2\pi^2\nu t) \\ \cos(\pi x)\sin(\pi y)\exp(-2\pi^2\nu t) \end{pmatrix}. \tag{3.5}$$

We solve the problem numerically on a square domain, $\Omega = [-1, 1]\times[-1, 1]$, with time-dependent Dirichlet boundary conditions and initial condition consistent with (3.5). In other words, the values of the boundary conditions and initial condition follow from the evaluation of the exact solution at the boundaries or at time zero.

The RHS in (2.16) is calculated on N_t nodes (time moments), which is necessary to construct the low rank polynomial approximation in (2.3). The nodes are equally spaced in time with a distance Δt. The step size Δt is not restricted by stability conditions, but we use the Courant number to get a natural estimate of an appropriate

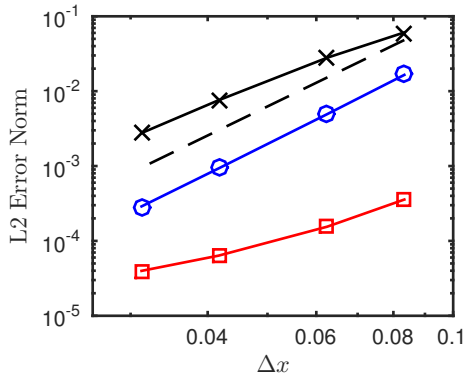


FIG. 1. *Taylor–Green vortex: error in x-velocity at $t = 1$ versus spatial resolution.* $\times$, $\nu = 10^{-3}$; $\circ$, $\nu = 10^{-2}$; $\square$, $\nu = 10^{-1}$; $--$, $O(\Delta x^4)$.

Δt. Here, we choose Δt to satisfy the condition $C = u\Delta x/\Delta t \leq 1$, where $u = 1$ (initially) and Δx is the spatial distance between the finite element nodes. Also, we preserve $m = 4$ leading singular values from the SVD, which appears to be sufficient for our purpose. This initial choice of parameters is based on prior experiments [39] but could still be optimized. We discuss the influence of m and N_t later in section 3.2.

Equation (2.16) is integrated by the EBK method using a residual tolerance of $\texttt{tol} = 10^{-3}$, which is sufficient in this case. The outer iterations are stopped when the stopping criterion (2.18) is reached, using the same tolerance of 10^{-3}. To reach the tolerance, typically four outer iterations in total are required.

We integrate from $t = 0$ to $t = 1$ and measure the numerical error in the x-component of the velocity at the final time, measured in the relative L^2-norm, $\|u - u_{\text{ref}}\|/\|u_{\text{ref}}\|$. In this case, the numerical error is mostly determined by the spatial discretization. Figure 1 shows the error varying with the spatial resolution for several values of ν. The error is generally larger when ν decreases, which is related to the tendency of central discretization schemes to produce small "wiggles" at higher mesh Reynolds numbers, i.e., for advection-dominated flows. Similar wiggly behavior has been observed using the default Crank–Nicolson scheme of IFISS. In such cases, the accuracy can be improved by using a finer mesh or by using a higher-order upwind-type discretization of the advection term [32]. Furthermore, the error shows approximately fourth-order convergence with Δx, which is one order higher than expected from typical error estimates [23]. This form of superconvergence might be attributed to the high regularity of the exact solution [64].

We have demonstrated the convergence of the underlying spatial discretization of the governing equations. The total numerical error in this test case is dominated by the spatial discretization. We can examine the time integration error in isolation by comparing the solution to a reference solution. Here, we calculate a highly accurate reference solution with a simple Euler method followed by Richardson extrapolation. Figure 2 shows the time integration error against the number of modes, m. The error is measured in the relative L^2-norm, $\|u - u_{ref}\|/\|u_{ref}\|$. It shows that increasing m can lead to a strong reduction in the error if the tolerance is sufficiently low. The influence of the EBK parameters on the error are examined in more detail in the next section.

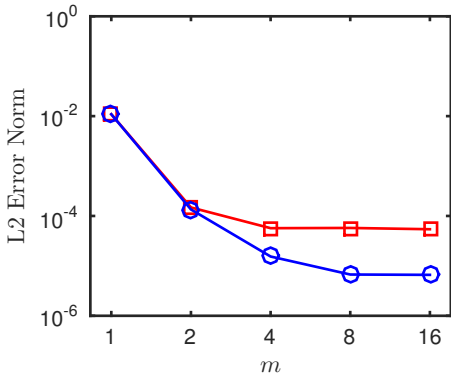


FIG. 2. *Taylor–Green vortex for* $\nu = 10^{-2}$, $N_t = 17$, *and* $\Delta x = 1/16$*: error in* x*-velocity at* $t = 1$ *versus* m. □, `tol` $= 10^{-2}$; ○, `tol` $= 10^{-3}$.

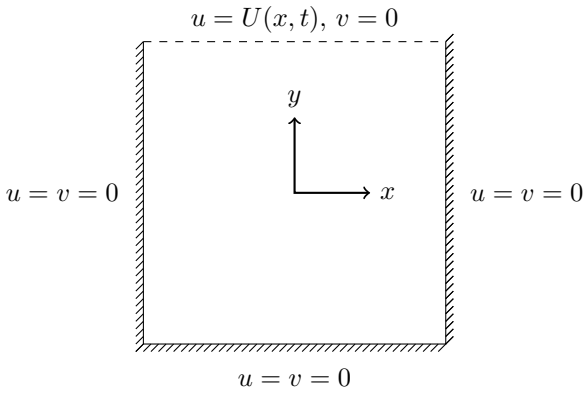


FIG. 3. *Geometry and boundary conditions of a lid-driven cavity flow. The domain is* $\Omega = [-1,1] \times [-1,1]$.

3.2. Lid-driven cavity flow. Lid-driven cavity flow is a well-known benchmark problem used to test solvers for the Navier–Stokes equation; see, for example, [6, 30]. In this section, we analyze the performance of the EBK method in simulations of lid-driven cavity flow. In this case, the exact solution to the Navier–Stokes equation is not known, but we compare EBK-based results with the Crank–Nicolson method, which is implemented as the default timestepping method in IFISS [38].

We study lid-driven cavity flow in a square domain, $\Omega = [-1,1] \times [-1,1]$. No-slip boundary conditions apply at the side and bottom walls. The velocity in the x-direction at the top boundary, the "lid" of the cavity, is prescribed by the function $U(x,t)$. The geometry and the boundary conditions of the problem are sketched in Figure 3.

3.2.1. Stokes flow. First, we simulate unsteady Stokes flow in a cavity, i.e., ignoring the nonlinear convection term in (2.7), which would approximate laminar flow at very low Reynolds numbers. The initial condition is $\vec{u} = 0$, and the lid is accelerated gradually from zero to a steady velocity, spinning up the flow in the cavity. The lid velocity is given by

$$U(x,t) = \left(1 - x^2\right)\left(1 + x^2\right)\left(1 - e^{-t}\right). \tag{3.6}$$

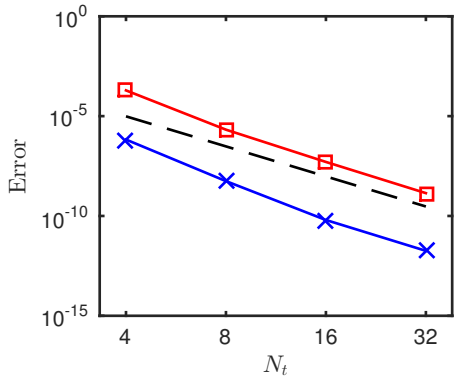


FIG. 4. *Numerical error at $t = 1$ versus number of samples N_t. □, relative L^2-error of x-velocity; ×, L^∞-error of continuity equation ($\|B\mathbf{u} - \mathbf{g}\|_\infty$),—$O(N_t^{-5})$. Stokes flow in a lid-driven cavity with $\nu = 0.02$.*

Note that the boundary conditions are *regularized* to prevent corner singularities, i.e., the velocity discontinuities at the top corners; see, for example, [7]. Otherwise very high spatial resolution is required to accurately resolve the solution near those points. In this case, the time-dependency of the RHS in (2.10) comes only from the transient boundary condition (3.6), not from the nonlinear term.

Without the presence of a nonlinear term, no outer iterations are needed, and we can focus on certain aspects of the EBK method in more detail. The accuracy of the EBK method mainly depends on the low rank approximation of the source term (2.3). The main parameters are (a) the number of source term samples, N_t, and (b) the number of modes from the truncated SVD, m.

As explained in section 2.1, the numerical error of the EBK method depends on the residual tolerance and the error in low rank approximation of the RHS (2.3). In the following test, we use a low residual tolerance of $\texttt{tol} = 10^{-10}$, such that the numerical error mainly depends on the accuracy of the low rank approximation. For the spatial discretization, we use 16×16 finite elements, which amounts to 33×33 gridpoints for the velocity field components.

We can define a Reynolds number based on $Re = U_{ref}L/\nu$, where $U_{ref} = 1$ is the maximum lid velocity and $L = 2$ is the cavity length. In this test case, the viscosity coefficient is $\nu = 0.02$, which corresponds to $Re = 100$. We integrate from $t = 0$ to $t = 1$. A reference solution is computed with the Crank–Nicolson method using a small time step size and Richardson extrapolation. The error in the x-velocity is measured in the relative L^2-norm. Figure 4 shows the numerical error at $t = 1$. As expected, the accuracy increases with the number of samples N_t. Also, the error in the continuity equation decreases with N_t, because the RHS (2.10) is approximated more accurately. Both errors display fifth-order convergence with N_t here. Because the approximation (2.3) is based on cubic spline interpolation, we should expect at least a fourth-order convergence.

A time-consuming part in the EBK method is the block Arnoldi process, which generates an orthonormal basis of the block Krylov subspace. At every Krylov iteration, the action of the matrix $P(A + J(\bar{\mathbf{u}}))$ on m vectors is required. This matrix action is not a simple matrix-vector multiplication, as the matrix $P(A + J(\bar{\mathbf{u}}))$ is not calculated explicitly. Because P involves the solution of a discrete Poisson problem (see (2.17)), each Krylov iteration is at least as expensive as a Poisson solve. An

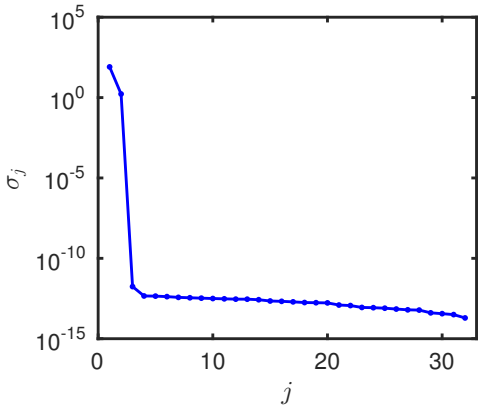


FIG. 5. *Singular values σ_j for $N_t = 32$. Stokes flow in a lid-driven cavity with $\nu = 0.02$.*

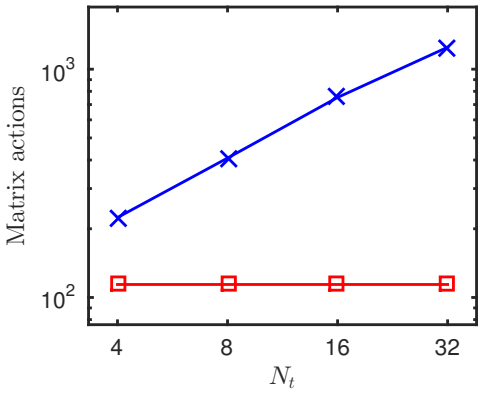
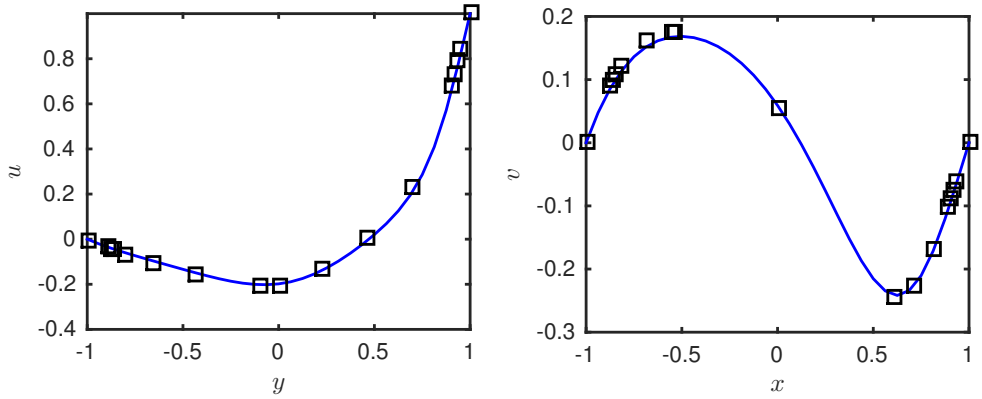


FIG. 6. *Actions of the matrix $P[A+J(\bar{\mathbf{u}})]$ versus number of samples N_t. $\times, m = N_t$; $\square$, $m = 2$. Stokes flow in a lid-driven cavity with $\nu = 0.02$.*

SVD shows, however, that the singular values may rapidly decrease in magnitude; see Figure 5, for example. The difference between the second and third values is several orders of magnitude in this example. Thus, in this case, we can save considerable computational work by truncating the SVD with $m = 2$ without losing any accuracy in practice. Using $m = 2$, for example, we achieve a relative error of 1.671E-09; see Figure 4. If we use $m = 32$ instead, the error reduces slightly to 1.391E-09. In other words, the truncation of the SVD (in this case with $m = 2$) does not lead to a significant loss of accuracy.

Figure 6 shows the number of matrix actions required by EBK against the number of samples. The number stays constant for $m = 2$ but increases strongly when increasing $m = N_t$, i.e., no truncation. In this case, a significant number of matrix actions can be saved with a truncated SVD, which improves the computational efficiency of the EBK method significantly, with practically no effect on the accuracy of the solution. In some cases, the number of matrix actions can be reduced from 1248 to 114; see Figure 6. This shows that a truncated SVD is instrumental in realizing good computational efficiency of the EBK method in practice.



(a) x-velocity along the vertical centerline. (b) y-velocity along the horizontal centerline.

FIG. 7. *Steady state solution of lid-driven cavity flow at* $Re = 100$*, shown by the velocity components along lines through the geometric center of the cavity. —, EBK* (33×33 *grid);* □*, Ghia, Ghia, and Shin* [30].

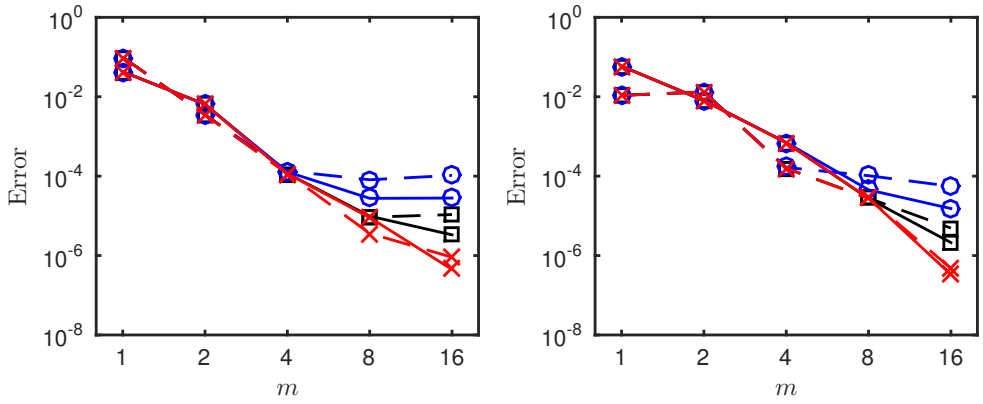
TABLE 1
Error in L^2*-norm of the steady state solution of lid-driven cavity flow at* $Re = 100$.

T_{end}	Error u	Error v
10	7.96E-3	1.55E-2
20	1.36E-3	2.32E-3
40	1.72E-4	3.12E-4
80	4.63E-5	8.54E-5

3.2.2. Steady state. Steady state solutions to lid-driven cavity flow are well-studied in the literature [6, 30]. In the following test case, we use the EBK method to solve the *steady* Navier–Stokes equation. We verify that the EBK method indeed converges to the correct steady state. In this case, we use an equidistant grid of 16×16 elements, i.e., 33×33 gridpoints. The flow is initialized as a solution to Stokes flow. Using this initial condition, the Navier–Stokes equation is integrated over a long time interval $t \in [0, T_{end}]$ until the solution reaches a steady state.

Based on the results in Figure 5, we use $m = 4$ to capture the relevant modes. Also, we use $N_t = 33$, which corresponds to a CFL number of $C = 5$ in case of $T_{end} = 10$. Also, we use tol $= 10^{-3}$ for controlling the exponential and the nonlinear residuals. The precise choice of parameters matters very little in this case. We have observed that the accuracy of the time integration method has a marginal influence on the error of the steady state solution.

The steady solution at $Re = 100$ is plotted in Figure 7, for which we used $T_{end} = 10$. The results obtained with the EBK method are in good agreement with the reference data from [30]. The EBK method yields the correct steady state solution. We also compare the EBK method to the default steady Navier–Stokes solver of IFISS, which provides us with a reference solution (on the same mesh). The errors of the velocity profiles, measured in the relative L^2-norm, are given in Table 1. In general, the EBK method agrees well with the default steady Navier–Stokes solver in IFISS. The final time appears to be the main parameter influencing the error. Increasing T_{end} decreases the error as the solution becomes closer to the true steady state. Long time intervals are needed because the transient solution approaches the steady state rather slowly, especially in cases with low viscosity.



(a) $Re = 100$ (b) $Re = 1000$

FIG. 8. *Error at $t = 1$ versus m for several tolerances (`tol`) of the exponential and nonlinear residual. Solid: error in u, dashed: error in v. $\circ$, `tol` $= 10^{-2}$; $\square$, `tol` $= 10^{-3}$; $\times$, `tol` $= 10^{-4}$.*

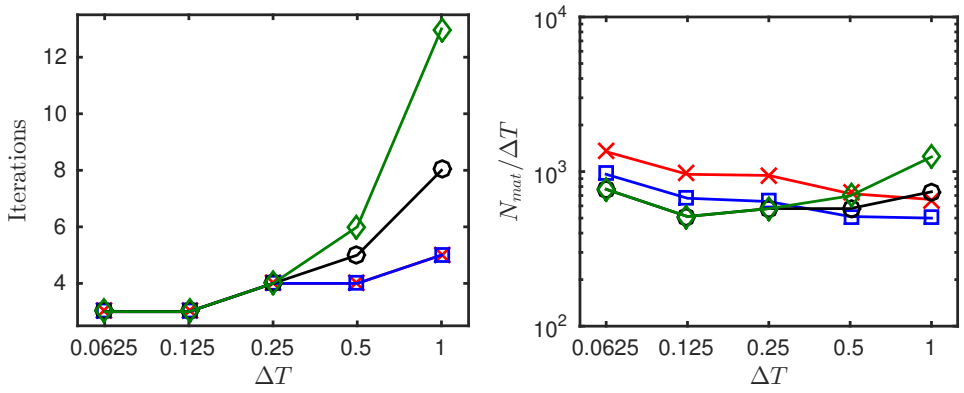
3.2.3. Oscillating lid. In this section, we examine the transient solution of a lid-driven cavity flow. In this case, the lid velocity oscillates in time. As such, the solution is intrinsically unsteady and will not tend to a steady state asymptotically. The lid velocity is prescribed as

$$U(x,t) = \left(1 - x^2\right)\left(1 + x^2\right)\left(1 + \tfrac{1}{2}\sin(2\pi t)\right). \tag{3.7}$$

We use a 33×33 grid, which was shown in section 3.2.2 to be an adequate resolution for resolving the steady state solution at $Re = 100$. In this case, we want to calculate a time-accurate solution, and the RHS in (2.16) is evaluated at $N_t = 33$ points on the interval $t \in [0, T_{end}]$. This corresponds to a Courant number of approximately $C = 0.5$, which is below the typical stability limit of explicit timestepping methods. We calculate the solution at $T_{end} = 1$, i.e., after one oscillation of the lid. An accurate reference solution is computed with the CN method, which is the default timestepping method of IFISS. As in section 3.2.2, we measure the error in the u-profile along the vertical centerline of the cavity, and in the v-profile along the horizontal centerline. The errors are measured in the relative L^2-norm.

The errors are shown in Figure 8 for two different Reynolds numbers. The error decreases as the number of SVD modes increases, until the error is limited by the tolerance of the EBK method. This shows that a stricter tolerance is needed for higher m in order to achieve a higher accuracy. For example, at $m = 4$ and $Re = 100$ the choice of the tolerance does not influence the total accuracy, and a tolerance of 10^{-2} would suffice. Note that the matrix-vector multiplications are expensive in the EBK method, because they involve the projection matrix P. The number of matrix actions is proportional to m. Therefore it would be beneficial not to take m too large. The results for $Re = 100$ and $Re = 1000$ are both very similar, which suggests that the Reynolds number has a limited influence on the accuracy of the EBK method.

Instead of computing the solution on the entire interval $[0, T_{end}]$ directly, one could divide the interval into multiple subintervals of size ΔT. The solutions on the subintervals are then computed sequentially. The number of outer iterations depends largely on the size of the time interval ΔT. Figure 9 shows the number of iterations as a function of the time interval size ΔT. As ΔT increases, more iterations are required



(a) Number of outer iterations (b) Number of matrix actions

FIG. 9. *Number of outer iterations and matrix actions versus the time interval size ΔT, using $m = 4$, and* `tol` $= 10^{-2}$. *The matrix actions are compensated with ΔT ($N_{mat}/\Delta T$). $\times$, $Re = 100$; $\square$, $Re = 300$; $\circ$, $Re = 1000$; $\diamond$, $Re = 3000$.*

to achieve a given tolerance. The number of matrix actions, which are an important indication of the computational costs, does not necessarily increase by increasing ΔT, however. Also, the Reynolds number plays a role. As the Reynolds number increases, the nonlinearity becomes stronger and more iterations are needed. At $Re = 3000$ we see a significant increase for $\Delta T > 0.25$, for example. We also measure the total number of matrix actions, N_{mat}, which involves the matrix $P[A+J(\bar{\mathbf{u}})]$. This provides a reasonable estimate of the computational cost. For $Re = 1000$ and $Re = 3000$ the number of matrix actions starts to increase from $\Delta T > 0.25$, due to the considerable increase in the number of iterations. In practice, the optimal ΔT can be determined by experimentation. These results suggest that it is preferable to choose ΔT as large as possible, without the number of iterations becoming excessively large.

4. Parallellization in time. In this section, we describe how the EBK method can be parallelized. We provide a basic model to study the potential scaling for the incompressible Navier–Stokes equation and determine the conditions under which EBK can become competitive in terms of computing time.

4.1. Parallel algorithm. The EBK method can be parallelized in time naturally with the Paraexp method, which is based on "parallel exponential propagation." The Paraexp method was introduced as an efficient parallel-in-time algorithm for nonhomogeneous differential equations [27]. One advantage of Paraexp is that its parallel performance does not deteriorate for hyperbolic PDEs or parabolic PDEs with little dissipation. Although the well-known Parareal method is known to work successfully for some hyperbolic problems, e.g., [9, 14, 28, 41, 51], convection-dominated flows were generally found to be highly challenging for Parareal [57]. The EBK method allows for a natural extension of the Paraexp method to nonlinear PDEs. Kooij, Botchev, and Geurts [39] show that realistic speedup can be expected for the advection-diffusion equation and the viscous Burgers equation.

The idea behind the Paraexp method is that a linear problem can be decoupled into independent subproblems using the principle of superposition. In our case, we solve (2.16) in parallel at every outer iteration. This equation is of the form

$$\mathbf{y}'(t) = \mathcal{A}\mathbf{y}(t) + \mathbf{g}(t), \quad \mathbf{y}(0) = 0, \quad t \in [0, T]. \tag{4.1}$$

Here, without loss of generality, we assume that the problem is transformed to have a zero initial condition. This means using a substitution $\mathbf{y}(t) = \widehat{\mathbf{y}}(t) - \mathbf{y}_0$, where $\widehat{\mathbf{y}}(t)$ is the solution to the original problem with the nonzero initial condition $\mathbf{y}_0$. In the case of nonlinear problems, (4.1) is updated at every outer iteration. The matrix $\mathcal{A}$ contains the Jacobian matrix of the nonlinear term averaged over $t \in [0, T]$, and the source term $\mathbf{g}(t)$ contains the nonlinear remainder. The Jacobian matrix is time-averaged, such that $\mathcal{A}$ does not depend on t. The time interval is divided into $\mathcal{P}$ nonoverlapping subintervals, $0 = T_0 < T_1 < \cdots < T_{\mathcal{P}} = T$. Based on these subintervals, we define $\mathbf{g}_j(t)$ as the piecewise function

$$\mathbf{g}_j(t) = \begin{cases} \mathbf{g}(t) & \text{for } T_{j-1} \le t < T_j, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Next, the original problem can be written as the combination of $\mathcal{P}$ independent problems,

$$\mathbf{v}_j'(t) = \mathcal{A}\mathbf{v}_j(t) + \mathbf{g}_j(t), \quad \mathbf{v}_j(0) = 0, \quad t \in [0, T]. \tag{4.3}$$

These problems can be solved in parallel on $\mathcal{P}$ processors and do not require any communication during the solution procedure. It is easy to see that the solution to (4.1) is the sum of the solutions to the subproblems (4.3),

$$\mathbf{y}(t) = \sum_{j=1}^{\mathcal{P}} \mathbf{v}_j(t). \tag{4.4}$$

Parallel speedup is expected on the observation that the source term is zero in all but one subinterval. The closed-form solution in the homogeneous part of (4.3) is given by

$$\mathbf{v}_j(t) = \exp\left((t - T_j)\mathcal{A}\right)\mathbf{v}_j(T_j), \quad t \ge T_j, \tag{4.5}$$

which is also called the *exponential propagation* of the solution. The matrix exponential in (4.5) can be evaluated very efficiently [48]. The computation time of (4.5) is typically negligible compared to the time needed for solving the nonhomogeneous part of (4.3) [27]. For a fast evaluation of (4.5), efficient algorithms for calculating the matrix exponential or its action on vectors are needed. These methods are typically much faster than traditional timestepping methods. Each parallel processor only deals with a part of the original nonhomogeneous problem. Parallel speedup is based on this premise. In the original formulation of the Paraexp method the nonhomogeneous part is solved by an off-the-shelf ODE solver. With the EBK method, we can take a unified approach by solving both the homogeneous and the nonhomogeneous part of (4.3) with the same time integrator. Such a unified approach is shown to be more effective than an original implementation of the Paraexp method [39].

4.2. Speedup and efficiency. We consider strong scaling when integrating on a fixed interval $t \in [0, \Delta T]$. The wall clock time of the sequential algorithm is $\mathcal{T}_1 = \tau_1 K_1$, where τ_1 is the time of one EBK solve on the interval ΔT and K_1 is the number of outer iterations. The wall clock time of the parallel algorithm is $\mathcal{T}_{\mathcal{P}} = \tau_{\mathcal{P}} K_{\mathcal{P}} = (\tau_{nh} + \tau_h + \tau_c) K_{\mathcal{P}}$, where $\tau_{\mathcal{P}}$ is the wall clock time of one outer iteration, τ_{nh} the solver time for the nonhomogeneous part of the IVP, and τ_h the time for the homogeneous part. Here, τ_c denotes the communication time between the parallel processes per

outer iteration and $K_{\mathcal{P}}$ the number of outer iterations of the parallel EBK method. In the case of $\mathcal{P} = 1$, we simply have $\tau_{nh} = \tau_1$ and $\tau_h = \tau_c = 0$. After every outer iteration, the solutions of the subproblems are summed and the source term, containing the nonlinear terms, is updated with the latest solution. These operations are included in τ_c. The parallel efficiency is then calculated as

$$\eta = \frac{\mathcal{T}_1}{\mathcal{P}\mathcal{T}_{\mathcal{P}}} = \frac{\tau_1 K_1}{\mathcal{P}(\tau_{nh} + \tau_h + \tau_c)K_{\mathcal{P}}}. \tag{4.6}$$

This expression shows that ideal efficiency of $\eta \approx 1$ is obtained under several conditions. First, the time spent on the nonhomogeneous part should be proportional to $\mathcal{P}$, $\tau_{nh} = \tau_1/\mathcal{P}$. Second, the time spent on the homogeneous part should be much less than the nonhomogenous part, $\tau_h \ll \tau_{nh}$. Third, the communication time should be relatively small, $\tau_c \ll \tau_h + \tau_{nh}$. Finally, the required number of outer iterations should not increase, $K_{\mathcal{P}} \leq K_1$.

Note that τ_h represents the cost of solving the homogeneous part. We use a direct solver to solve (3.1) when applying the projection operator. For larger systems, an iterative solver is needed and τ_h will vary depending on the number of iterations. Assuming an appropriate preconditioner is used, the number of iterations should not vary greatly. So, we do not expect this to cause significant load imbalances in time. If possible load imbalances would occur in practice, they could be mitigated by adapting the size of the subintervals; see [27].

Under some assumptions, we can propose a simple model for predicting the maximal speedup that can be achieved. In the ideal case of perfect parallel efficiency, we have $\tau_h \ll \tau_{nh}$, but in practice τ_h needs to be taken into account. We can write τ_h as a fraction α of τ_{nh}, i.e., $\tau_h = \alpha\tau_{nh}$. In practice, α is greater than zero, depending on the number of subintervals/processors $\mathcal{P}$, i.e., $\alpha = \alpha(\mathcal{P}) > 0$. Based on observations shown later, we expect α to be around 0.1. The parallel speedup is

$$S = \frac{\mathcal{T}_1}{\mathcal{T}_{\mathcal{P}}} = \frac{\tau_1 K_1}{((1 + \alpha(\mathcal{P}))\tau_{nh} + \tau_c)K_{\mathcal{P}}}. \tag{4.7}$$

The Jacobian matrix is time-averaged over each subinterval individually. It is safe to assume that $K_{\mathcal{P}} \leq K_1$, because the nonlinear correction, using a time-averaged Jacobian matrix, is more accurate on smaller subintervals. For now, we assume that the communication is negligible, $\tau_c = 0$, and that $\tau_{nh} = \tau_1/\mathcal{P}$, corresponding to the fact that τ_1 relates to EBK on the whole interval and $\tau_{\mathcal{P}}$ to EBK on one of the subintervals. We assume that $\tau_{\mathcal{P}} < \tau_1$, because the nonhomogeneity is limited to a smaller subinterval ($\Delta T/\mathcal{P}$). Under these assumptions, the expression for the speedup can be simplified to

$$S = \frac{\mathcal{P}}{1 + \alpha(\mathcal{P})}. \tag{4.8}$$

A low value for α is important for high parallel speedup. In a worst-case scenario, we could have $\tau_h + \tau_{nh} = \tau_1$. Assuming $\tau_{nh} = \tau_1/\mathcal{P}$, we would then find $\alpha = \mathcal{P} - 1$ and hence no speedup at all with $S = 1$. The value $\alpha = \mathcal{P} - 1$ is a maximum value that can be expected realistically. This situation occurs if traditional time-stepping methods, instead of exponential integrators, are used for calculating the exponential propagation (4.5). In other words, the homogeneous part of (4.3) is not solved substantially faster than the nonhomogeneous part. However, in our case we

TABLE 2

Computation time of the serial execution of the EBK and the CN method for the Taylor–Green vortex at $Re = 100$. The (maximum) mesh Reynolds number Re_h is also indicated. The last column contains the ratio between the CPU time of the EBK and the CN method.

		EBK		CN		
Δx	Re_h	Error	CPU	Error	CPU	Ratio
1/32	3.125	1.924E-4	67.8 s	2.023E-4	16.9 s	4.01

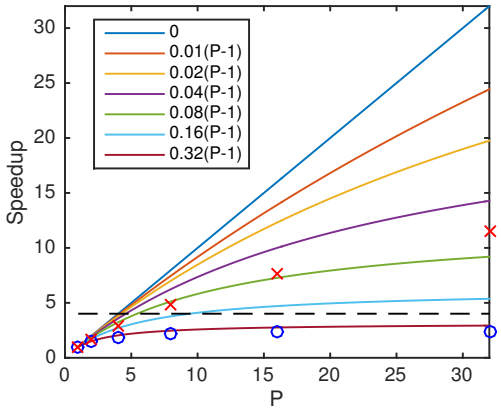


FIG. 10. *Parallel speedup for several $\alpha(\mathcal{P})$, where $\alpha = \tau_h/\tau_{nh}$. The dashed line indicates the break-even point of the EBK method with the CN method. ∘, Jacobian matrix averaged over each subinterval. ×, Jacobian matrix averaged over the complete time interval.*

use EBK and this effect does not play a role. This is a key motivation for adhering to EBK as a method for time-parallel integration.

The wall clock times of the EBK and the CN method from the experiment of the Taylor–Green vortex (see section 3.1) are listed in Table 2 for $\Delta x = 1/32$. These computations are performed for $\Delta T = 10$ and $Re = 100$. The time required by the EBK method corresponds to τ_1 here. For the EBK method, we use $N_t = 33$, $m = 4$, and $\texttt{tol} = 10^{-3}$. For the CN method, we use the adaptive scheme described in [38] with a tolerance of 10^{-7}. Using these parameters, both methods have a very similar accuracy, as shown in Table 2.

The EBK method is generally slower than the CN method, by a factor of 4 typically. In other words, the parallel speedup needs to be at least 4 for the EBK method to break even with the CN method in terms of computation time. The parallel speedup is illustrated for several $\alpha(\mathcal{P})$ in Figure 10, taking $\alpha(P) = c(\mathcal{P} - 1)$ and considering the dependence of S on a constant c. The case $\alpha = 0$ corresponds to ideal speedup, i.e., the homogeneous parts have zero cost. As the cost of the homogeneous part increases with respect to the nonhomogeneous ones, α increases and the parallel efficiency decreases. The curves move away from the ideal line as α increases. This figure illustrates that the EBK method can break even only when α is sufficiently small.

When we assume that $\alpha = c(\mathcal{P} - 1)$, where $c > 0$ is a constant, we can examine the speedup as $\mathcal{P}$ approaches infinity. In that case, the limit of the speedup is

$$\lim_{\mathcal{P}\to\infty} S = \frac{\mathcal{P}}{1 + c(\mathcal{P} - 1)} = \frac{1}{c}. \tag{4.9}$$

When the Jacobian matrix is averaged over each local subinterval, the EBK method performs close to $c \approx 0.32$. That means the maximum speedup is $1/c = 3.125$ and it will not be able to break even with the CN method. However, the Jacobian matrix can be averaged over the total interval. The homogeneous problem can then be solved as a single problem, because the matrix does not vary per subinterval. In that case, the performance is close to $c \approx 0.08$. In other words the maximum speedup is $1/c = 12.5$ and it is possible to surpass the CN method in terms of computation time, as shown in Figure 10.

We note that for the parallel EBK method, the total time interval, $[0, T]$, is fixed. The interval is divided into smaller subintervals, according to the number of processors: $\Delta T = T/\mathcal{P}$. Because the total time interval does not increase with the number of processors, we do not expect the number of outer iterations to increase. On the contrary, the number of outer iterations is likely to decrease, when the Jacobian matrix is averaged over each local subinterval. The linearization is then more accurate, which improves the convergence of the nonlinear iterative process. This idea is supported by numerical experiments in [39]. Also, the number of samples N_t per subinterval decreases with $\mathcal{P}$, such that $N_t = 32/\mathcal{P} + 1$, where $\mathcal{P} \leq 32$. Using an equidistant temporal grid, the solution is then always calculated at the same temporal nodes. This also means that the number of SVD modes must be reduced for very high $\mathcal{P}$, because m cannot be larger than N_t. In this experiment, we have used $m = \min(4, N_t)$. This might explain why we observe a slight additional speedup at $P = 32$, above the theoretical projection of $\alpha = 0.08(\mathcal{P} - 1)$.

The parallel efficiency appears to remain relatively low, because the homogeneous parts are not solved substantially faster than the nonhomogeneous parts. The EBK method may be improved by the so-called shift-and-invert (SaI) technique [4]. The SaI technique is based on a rational Krylov subspace, which improves the convergence by emphasizing small eigenvalues [49, 60]. The expected faster convergence would make the EBK method more competitive with the CN method already at lower numbers of processors. Also, we might get closer to the ideal situation in which $\tau_h \ll \tau_{nh}$. This would increase α and therefore reduce the parallel speedup. The realization of the SaI technique is not straightforward, because of the differential-algebraic nature of the Navier–Stokes equation. The possible implementation and improvements of the SaI technique will be explored in future studies.

5. Conclusions. The EBK method is an exponential time integration method which we have extended in this paper to the incompressible Navier–Stokes equation. The pressure is treated by introducing a divergence-free projection operator, whereas the nonlinear convection term is taken care of by a process of outer across-time iterations. In these iterations, the nonlinear term is handled as a source term, evaluated with the solution of the previous iteration (on the complete time interval). This approach is very similar to waveform relaxations.

The spatial discretization of the Navier–Stokes equation is carried out by a finite element method. In several numerical experiments, we demonstrate that the EBK method can produce highly accurate time solutions. Furthermore, a major advantage is that the EBK method is suitable for parallel-in-time simulations. With a simplified model we have analyzed the potential speedup. We establish that parallel speedup for parallel-in-time simulations of the incompressible Navier–Stokes equation is indeed feasible with the EBK method. The efficiency of the EBK method itself might be further improved by using a rational Krylov subspace [3, 33] with the SaI technique [60]. Future research will be directed toward an actual application of the

time-parallel method, measuring the performance benefit as function of problem size and number of processors.

REFERENCES

[1] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137, https://doi.org/10.1017/S0962492904000212.

[2] M. Benzi and Z. Wang, *A parallel implementation of the modified augmented Lagrangian preconditioner for the incompressible Navier-Stokes equations*, Numer. Algorithms, 64 (2013), pp. 73–84, https://doi.org/10.1007/s11075-012-9655-x.

[3] M. Berljafa and S. Güttel, *Generalized rational Krylov decompositions with an application to rational approximation*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 894–916, https://doi.org/10.1137/140998081.

[4] M. A. Botchev, *A block Krylov subspace time-exact solution method for linear ordinary differential equation systems*, Numer. Linear Algebra Appl., 20 (2013), pp. 557–574.

[5] M. A. Botchev, V. Grimm, and M. Hochbruck, *Residual, restarting, and Richardson iteration for the matrix exponential*, SIAM J. Sci. Comput., 35 (2013), pp. A1376–A1397.

[6] O. Botella and R. Peyret, *Benchmark spectral results on the lid-driven cavity flow*, Comput. & Fluids, 27 (1998), pp. 421–433.

[7] C.-H. Bruneau and M. Saad, *The 2D lid-driven cavity problem revisited*, Comput. & Fluids, 35 (2006), pp. 326–348.

[8] K. Burrage, *Parallel and Sequential Methods for Ordinary Differential Equations*, Clarendon Press, New York, 1995.

[9] F. Chen, J. S. Hesthaven, and X. Zhu, *On the Use of Reduced Basis Methods to Accelerate and Stabilize the Parareal Method*, Springer, New York, 2014, pp. 187–214, https://doi.org/10.1007/978-3-319-02090-7_7.

[10] A. J. Chorin, *Numerical solution of the Navier–Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[11] A. J. Chorin, *On the convergence of discrete approximations to the Navier–Stokes equations*, Math. Comp., 23 (1969), pp. 341–353.

[12] S. Cox and P. Matthews, *Exponential time differencing for stiff systems*, J. Comput. Phys., 176 (2002), pp. 430–455, https://doi.org/10.1006/jcph.2002.6995.

[13] X. Dai, C. Le Bris, F. Legoll, and Y. Maday, *Symmetric parareal algorithms for Hamiltonian systems*, Math. Model. Numer. Anal., 47 (2013), pp. 717–742.

[14] X. Dai and Y. Maday, *Stable parareal in time method for first- and second-order hyperbolic systems*, SIAM J. Sci. Comput., 35 (2013), pp. A52–A78, https://doi.org/10.1137/110861002.

[15] J. J. de Swart, *Parallel Software for Implicit Differential Equations*, Ph.D. thesis, University of Amsterdam, 1997.

[16] V. Druskin and L. Knizhnerman, *Extended Krylov subspaces: Approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 755–771, https://doi.org/10.1137/S0895479895292400.

[17] W. Edwards, L. Tuckerman, R. Friesner, and D. Sorensen, *Krylov methods for the incompressible Navier–Stokes equations*, J. Comput. Phys., 110 (1994), pp. 82–102, https://doi.org/10.1006/jcph.1994.1007.

[18] A. Eghbal, A. G. Gerber, and E. Aubanel, *Acceleration of unsteady hydrodynamic simulations using the parareal algorithm*, J. Comput. Sci., 19 (2017), pp. 57–76, https://doi.org/10.1016/j.jocs.2016.12.006.

[19] H. Elman and D. Silvester, *Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46.

[20] H. C. Elman, A. Ramage, and D. J. Silvester, *Algorithm* 866: *IFISS, a Matlab toolbox for modelling incompressible flow*, ACM Trans. Math. Software, 33 (2007), 14.

[21] H. C. Elman, A. Ramage, and D. J. Silvester, *IFISS: A computational laboratory for investigating incompressible flow problems*, SIAM Rev., 56 (2014), pp. 261–273.

[22] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations*, Numer. Math., 90 (2002), pp. 665–688.

[23] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, New York, 2014.

[24] P. F. Fischer, F. Hecht, and Y. Maday, *A Parareal in time semi-implicit approximation of the Navier–Stokes equations*, in Domain Decomposition Methods in Science and Engineering, Springer, New York, 2005, pp. 433–440.
[25] H. Fujita and T. Kato, *On the Navier–Stokes initial value problem.* I, Arch. Ration. Mech. Anal., 16 (1964), pp. 269–315.
[26] M. J. Gander, *Analysis of the parareal algorithm applied to hyperbolic problems using characteristics*, Bol. Soc. Esp. Mat. Apl., 42 (2008), pp. 21–35.
[27] M. J. Gander and S. Güttel, *PARAEXP: A parallel integrator for linear initial-value problems*, SIAM J. Sci. Comput., 35 (2013), pp. C123–C142.
[28] M. J. Gander and M. Petcu, *Analysis of a Krylov subspace enhanced parareal algorithm for linear problems*, in Paris-Sud Working Group on Modelling and Scientific Computing 2007–2008, ESAIM Proc. 25, EDP Sciences, Les Ulis, France, 2008, pp. 114–129, https://doi.org/10.1051/proc:082508.
[29] M. J. Gander and S. Vandewalle, *Analysis of the Parareal time-parallel time-integration method*, SIAM J. Sci. Comput., 29 (2007), pp. 556–578, https://doi.org/10.1137/05064607X.
[30] U. Ghia, K. Ghia, and C. Shin, *High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method*, J. Comput. Phys., 48 (1982), pp. 387–411.
[31] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, *How to find a good submatrix*, in Matrix Methods: Theory, Algorithms and Applications, World Scientific, Hackensack, NJ, 2010, pp. 247–256, https://doi.org/10.1142/9789812836021_0015.
[32] P. M. Gresho and R. L. Lee, *Don't suppress the wiggles—they're telling you something!*, Comput. & Fluids, 9 (1981), pp. 223–253.
[33] S. Güttel and L. Knizhnerman, *A black-box rational Arnoldi variant for Cauchy-Stieltjes matrix functions*, BIT, 53 (2013), pp. 595–616, https://doi.org/10.1007/s10543-013-0420-x.
[34] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids, 8 (1965), 2182.
[35] T. Haut and B. Wingate, *An asymptotic parallel-in-time method for highly oscillatory PDEs*, SIAM J. Sci. Comput., 36 (2014), pp. A693–A713.
[36] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.
[37] R. I. Issa, *Solution of the implicitly discretised fluid flow equations by operator-splitting*, J. Comput. Phys., 62 (1986), pp. 40–65.
[38] D. A. Kay, P. M. Gresho, D. F. Griffiths, and D. J. Silvester, *Adaptive time-stepping for incompressible flow part* ii: *Navier–Stokes equations*, SIAM J. Sci. Comput., 32 (2010), pp. 111–128.
[39] G. L. Kooij, M. A. Botchev, and B. J. Geurts, *A block Krylov subspace implementation of the time-parallel Paraexp method and its extension for nonlinear partial differential equations*, J. Comput. Appl. Math., 316 (2017), pp. 229–246, https://doi.org/10.1016/j.cam.2016.09.036.
[40] R. Krause and D. Ruprecht, *Hybrid space–time parallel solution of Burgers' equation*, in Domain Decomposition Methods in Science and Engineering XXI, Springer, New York, 2014, pp. 647–655.
[41] A. Kreienbuehl, P. Benedusi, D. Ruprecht, and R. Krause, *Time-parallel gravitational collapse simulation*, Commun. Appl. Math. Comput. Sci., 12 (2017), pp. 109–128.
[42] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'edp par un schéma en temps "pararéel,"* C. R. Math. Acad. Sci. Paris Sér. I, 332 (2001), pp. 661–668, https://doi.org/10.1016/S0764-4442(00)01793-6.
[43] 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, https://doi.org/10.1016/j.cam.2013.07.027.
[44] V. T. Luan and A. Ostermann, *Parallel exponential Rosenbrock methods*, Comput. Math. Appl., 71 (2016), pp. 1137–1150, https://doi.org/10.1016/j.camwa.2016.01.020.
[45] Y. Maday, D. Meiron, A. T. Patera, and E. M. Rønquist, *Analysis of iterative methods for the steady and unsteady Stokes problem: Application to spectral element discretizations*, SIAM J. Sci. Comput., 14 (1993), pp. 310–337.
[46] U. Miekkala and O. Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 459–482, https://doi.org/10.1137/0908046.

[47] A. Y. Mikhalev and I. V. Oseledets, *Iterative representing set selection for nested cross approximation*, Numer. Linear Algebra Appl., 23 (2016), pp. 230–248, https://doi.org/10.1002/nla.2021.
[48] 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.
[49] I. Moret and P. Novati, *RD rational approximations of the matrix exponential*, BIT, 44 (2004), pp. 595–615.
[50] C. K. Newman, *Exponential Integrators for the Incompressible Navier–Stokes Equations*, Ph.D. thesis, Virginia Polytechnic Institute and State University, Blacksburg, 2004.
[51] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, Comput. & Fluids, 59 (2012), pp. 72–83.
[52] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[53] Y. Saad and B. Semeraro, *Application of Krylov Exponential Propagation to Fluid Dynamics Equations*, Technical Report 91.06, Research Institute for Advanced Computer Science, 1991.
[54] D. Samaddar, D. Newman, and R. Sánchez, *Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm*, J. Comput. Phys., 229 (2010), pp. 6558–6573, https://doi.org/10.1016/j.jcp.2010.05.012.
[55] J. C. Schulze, P. J. Schmid, and J. L. Sesterhenn, *Exponential time integration using Krylov subspaces*, Internat. J. Numer. Methods Fluids, 60 (2009), pp. 591–609.
[56] R. Speck, D. Ruprecht, M. Emmett, M. Bolten, and R. Krause, *A space-time parallel solver for the three-dimensional heat equation*, in Parallel Computing: Accelerating Computational Science and Engineering (CSE), Vol. 25, IOS Press, Amsterdam, 2014, pp. 263–272.
[57] J. Steiner, D. Ruprecht, R. Speck, and R. Krause, *Convergence of Parareal for the Navier–Stokes equations depending on the Reynolds number*, in Numerical Mathematics and Advanced Applications—ENUMATH 2013, Springer, New York, 2015, pp. 195–202.
[58] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, 3rd ed., Texts Appl. Math. 12, Springer, New York, 2002.
[59] R. Témam, *Sur l'approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionnaires,* I, II, Arch. Ration. Mech. Anal., 33 (1969), pp. 377–385.
[60] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput., 27 (2006), pp. 1438–1457.
[61] P. J. van der Houwen, *Parallel step-by-step methods*, Appl. Numer. Math., 11 (1993), pp. 69–81.
[62] P. J. van der Houwen and B. P. Sommeijer, *Iterated Runge–Kutta methods on parallel computers*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 1000–1028.
[63] J. J. I. M. van Kan, *A second-order accurate pressure-correction scheme for viscous incompressible flow*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 870–891.
[64] J. Wang and X. Ye, *Superconvergence of finite element approximations for the Stokes problem by projection methods*, SIAM J. Numer. Anal., 39 (2002), pp. 1001–1013, http://www.jstor.org/stable/3061942.
[65] P. Wesseling, *Principles of Computational Fluid Dynamics*, Springer Ser. Comput. Math. 29, Springer, New York, 2001.
[66] J. White, F. Odeh, A. L. Sangiovanni-Vincentelli, and A. Ruehli, *Waveform Relaxation: Theory and Practice*, Technical Report UCB/ERL M85/65, EECS Department, University of California, Berkeley, 1985, http://www.eecs.berkeley.edu/Pubs/TechRpts/1985/543.html.

ERRATUM

The correct affiliation for the second author is as follows:

Mathematics of Computational Science, Faculty of EEMCS, University of Twente, Enschede, Netherlands, and Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, and Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Moscow, Russia (botchev@kiam.ru).