SIAM J. SCI. COMPUT.
Vol. 47, No. 1, pp. A430–A453

IMPROVING EFFICIENCY OF PARALLEL ACROSS THE METHOD SPECTRAL DEFERRED CORRECTIONS*

GAYATRI ČAKLOVIĆ†, THIBAUT LUNET‡, SEBASTIAN GÖTSCHEL‡, AND DANIEL RUPRECHT‡

Abstract. Parallel across the method time integration can provide small scale parallelism when solving initial value problems. Spectral deferred corrections (SDCs) with a diagonal sweeper, closely related to iterated Runge–Kutta methods proposed by Van der Houwen and Sommeijer, can use a number of threads equal to the number of quadrature nodes in the underlying collocation method. However, convergence speed, efficiency, and stability depend critically on the coefficients of the used SDC preconditioner. Previous approaches used numerical optimization to find good diagonal coefficients. Instead, we propose an approach that allows one to find optimal diagonal coefficients analytically. We show that the resulting parallel SDC methods provide stability domains and convergence order very similar to those of well established serial SDC variants. Using a model for computational cost that assumes 80% efficiency of an implementation of parallel SDCs, we show that our variants are competitive with serial SDC, previously published parallel SDC coefficients, Picard iteration, and a fourth-order explicit as well as a fourth-order implicit diagonally implicit Runge–Kutta method.

Key words. parallel in time (PinT), spectral deferred correction, parallel across the method, stiff and non-stiff problems, iterated Runge–Kutta methods

MSC codes. 65R20, 65L04, 65L05, 65L20

DOI. 10.1137/24M1649800

See reproducibility of computational results at end of the article.

1. Introduction.

Numerical methods for solving initial-value problems for nonlinear systems of ordinary differential equations (ODEs)

$$\frac{du(t)}{dt} = f(t, u(t)), \quad t \in [0, T], \quad u(0) = u_0 \in \mathbb{R}^{N_{\text{dof}}}, \tag{1.1}$$

are of great importance in many fields of sciences. For ODEs arising from spatial discretization of a partial differential equation (PDE) in a method-of-lines approach, the number of degrees of freedom N_{dof} is often very large. Hence, developing efficient methods for minimizing time-to-solution and computational cost becomes important. Because of the large number of compute cores in modern computers, leveraging concurrency is one of the most effective ways to reduce solution times.

A widely used class of methods for solving (1.1) are Runge–Kutta methods (RKMs), usually represented by Butcher tables of the form

*Submitted to the journal's Numerical Algorithms for Scientific Computing section March 27, 2024; accepted for publication (in revised form) October 18, 2024; published electronically February 11, 2025.

https://doi.org/10.1137/24M1649800

Funding: This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, and Switzerland. This project also received funding from the German Federal Ministry of Education and Research (BMBF) grants 16HPC048 and 16ME0679K and from the European Union - NextGenerationEU.

†Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (caklovicka@gmail.com).

‡Chair Computational Mathematics, Institute of Mathematics, Hamburg University of Technology, 21073 Hamburg, Germany (thibaut.lunet@tuhh.de, sebastian.goetschel@tuhh.de, ruprecht@tuhh.de).

$$\begin{array}{c|c} \boldsymbol{c} & \mathbf{A} \\ \hline & \boldsymbol{b}^\top \end{array}$$

with $\mathbf{A} \in \mathbb{R}^{s\times s}$, $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^s$, and s the number of stages. The Butcher table is a concise way of representing the update from t_0 to $t_0 + \Delta t$, which, for a scalar ODE, reads

$$\text{solve:} \quad u(\boldsymbol{t}_s) - \Delta t \mathbf{A} f(\boldsymbol{t}_s, u(\boldsymbol{t}_s)) = u(t_0 \mathbb{1}), \quad \boldsymbol{t}_s = t_0 \mathbb{1} + \Delta t \boldsymbol{c}, \tag{1.2}$$

$$\text{update:} \quad u(t_0 + \Delta t) \approx u(t_0) + \Delta t \boldsymbol{b}^\top f(\boldsymbol{t}_s, u(\boldsymbol{t}_s)), \tag{1.3}$$

where $\mathbb{1} = (1, \ldots, 1)^\top$ is the vector with unit entries, and $u(\boldsymbol{t}_s)$ is the vector containing solutions at all times in $\boldsymbol{t}_s$.[1] For implicit Runge–Kutta methods (IRKs), where the matrix $\mathbf{A}$ is dense, computing the stages in (1.2) requires solving a system of size $sN_{\text{dof}} \times sN_{\text{dof}}$.

A popular class of IRKs are collocation methods [15, Sec II.7] based on Gaussian quadrature. Collocation methods are attractive since they can be of very high order and are A-stable or L-stable depending on the type of quadrature nodes used. However, since they have a dense matrix $\mathbf{A}$, they are computationally expensive. Diagonally implicit Runge–Kutta methods (DIRKs) with a lower-triangular $\mathbf{A}$ are computationally cheaper, because the implicit systems for the stages can be solved independently. However, compared to collocation, DIRKs have lower order for an equal number of stages and less favorable stability properties.

1.1. Parallelism across the method. The first ideas on exploiting parallelism in the numerical solution of ODEs emerged in the 1960s [25]. Given the massive increase in concurrency in modern high-performance computing, the last two decades have seen a dramatic rise in interest in parallel-in-time methods; see the recent reviews by Ong and Spiteri [28] and Gander [11]. In the terminology established by Gear [13], we focus on "parallelism across the method" where a time integration scheme is designed such that computations within a single time step can be performed in parallel. By contrast, "parallel across the steps" methods like Parareal [23], PFASST [8], and MGRIT [9] parallelize across multiple time steps. Revisionist integral deferred corrections (RIDC) are a hybrid that compute a small number of time steps simultaneously [27]. While parallelism across the method is more limited in the number of cores it can employ, it often provides better parallel efficiency and is easier to implement than parallelization across the steps.

Note that time parallelization is meant to be employed in combination with and not instead of spatial parallelization. While we focus only on temporal parallelization here, parallel spectral deferred corrections (SDCs) have been shown to be capable of extending scaling beyond the saturation of pure spatial parallelization [10]. It is important to keep in mind that parallelizations in space and time are multiplicative. That is, if a code's spatial parallelization saturates at, say, 1000 cores, adding parallelization in time with four cores would allow one to use up to $4 \times 1000 = 4000$ cores.

For collocation methods, parallel across the method variants exist based on diagonalization of $\mathbf{A}$ [3, 22, 26, 29, 39] or based on using a specific GMRES preconditioning [21, 30, 24]. However, those approaches can introduce significant overhead and, particularly for large problems, struggle to outperform sequential time-stepping [4]. Runge–Kutta methods with parallelism across stages or blocks of stages, that is, with diagonal or lower block diagonal Butcher matrices, have also been considered,

[1] For simplicity and to avoid complex notation using tensor products, we describe time integration here only from the scalar perspective.

but the resulting schemes lack stability and are of lower order than their sequential counterparts [17, 18, 19, 31, 37].

Spectral deferred corrections (SDC), introduced in 2000 by Dutt, Greengard, and Rokhlin [7], are an iterative approach for computing the stages of a collocation method by performing multiple "sweeps" through the quadrature nodes with a lower order method. SDC can also be interpreted as a preconditioned fixed-point or Richardson iteration [16, 33]. In standard SDC, the preconditioner applied to (1.2) corresponds to a lower triangular matrix that is inverted by forward substitution and results in a sweep-like type of iteration. Speck [39] suggests using a diagonal preconditioner instead, which allows one to parallelize the iteration update for the stages. However, depending on the entries of the diagonal preconditioner, convergence of parallel SDC can be much slower than convergence of standard SDC. Unbeknownst to the author, this idea had been proposed before by van der Houwen and Sommeijer [44], but in the context of iterated IRK methods instead of SDC.

Links between SDC and RKM are well established [5]. In addition, we show in section 2.1 that SDC is actually equivalent to a specific iterated IRK method by van der Houwen and Sommeijer [44] that uses a lower-triangular preconditioner.

IRK were widely studied in the 1990s and shown to preserve important attributes of the underlying collocation method, such as order and stability [2]. They form the basis of "parallel iterated RK across the steps" (PIRKAS) methods [34, 42, 43], which can be written as block Gauss–Seidel SDC (BGS-SDC) methods [1, 14]. Both PIRKAS and BGS-SDC methods have been combined with parallelism across the method using diagonal preconditioning [44] to form the parallel diagonal implicitly Iterated RK across the steps (PDIRKAS) methods [38, 41, 45, 46, 47], providing two levels of parallelism in time. Similar two-level parallelism in time has been achieved by a combination of PFASST [8] with parallel SDC [36].

The key to fast convergence and thus good performance of either parallel SDC or iterated IRK is the choice of coefficients in the diagonal preconditioner [44, sect. 3]. The authors identify two possible optimization problems to compute good diagonal coefficients for either nonstiff or stiff problems. Both seek to minimize the spectral radius of certain matrices, but since these are ill conditioned, optimization algorithms struggle as the number of parallel stages s increases. This was already acknowledged both by van der Houwen and Sommeijer and Speck [44, 39]. Van der Houwen and Sommeijer proposed using an objective function based on the stability function of the iterated IRK as a remedy. However, this is applicable only to some types of IRK methods and worked only for stiff problems.

1.2. Contributions. We present a generic approach to compute optimized coefficients for diagonal preconditioners in SDC or iterated IRK. It leads to three sets of coefficients that we call `MIN-SR-NS`, `MIN-SR-S`, and `MIN-SR-FLEX`. While `MIN-SR-NS` is suited for nonstiff problems, `MIN-SR-S` and `MIN-SR-FLEX` are designed for stiff problems. We provide analytical expressions for the coefficients of `MIN-SR-NS` and `MIN-SR-FLEX` and a generic approach to generate `MIN-SR-S` coefficients for any type of collocation method. We show that the resulting parallel SDC methods remain accurate and stable compared to state-of-the-art SDC preconditioners, in particular those by van der Houwen and Sommeijer [44] and Weiser [48]. We demonstrate that parallel SDC methods can compete with standard RKM from the literature with respect to computational cost, because they allow one to exploit parallelism across the method without sacrificing speed of convergence. All numerical experiments reported in this paper can be reproduced with the accompanying code [40].

2. Optimal diagonally preconditioned spectral deferred corrections. We start by describing SDC in section 2.1 and develop the new diagonal preconditioners in section 2.2. For details on SDC see Dutt, Greengard, and Rokhlin or Huang, Jia, and Minion [7, 16].

2.1. Spectral deferred corrections as a fixed point iteration. Consider the Picard formulation of the initial value problem (1.1) on $[t_0, t_0 + \Delta t]$,

$$u(t) = u_0 + \int_{t_0}^{t} f(s, u(s))ds, \tag{2.1}$$

for some time step size Δt. By choosing $M \in \mathbb{N}$ collocation nodes $0 \leq \tau_1 < \cdots < \tau_M \leq 1$ and defining $t_m = t_0 + \tau_m \Delta t$, we can write (2.1) for each t_m as

$$u(t_m) = u_0 + \Delta t \int_0^{\tau_m} f(t_0 + s\Delta t, u(t_0 + s\Delta t))ds, \quad m = 1, \ldots, M. \tag{2.2}$$

Let ℓ_i denote the ith Lagrange polynomial associated with the nodes $\tau_1, \ldots, \tau_M$. Using a polynomial approximation of the integrand f turns (2.2) into

$$u_m = u_0 + \Delta t \sum_{i=1}^{M} \left(\int_0^{\tau_m} \ell_i(s)ds \right) f(t_i, u_i), \quad m = 1, \ldots, M, \tag{2.3}$$

where u_m is a discrete approximation of $u(t_m)$. We collect (2.3) for $m = 1, \ldots, M$ in the compact matrix formulation

$$\boldsymbol{u} - \Delta t \mathbf{Q} f(\boldsymbol{u}) = u_0 \mathbb{1}, \tag{2.4}$$

with $\boldsymbol{u} = [u_1, \ldots, u_M]^\top$, $f(\boldsymbol{u}) = [f(t_1, u_1), \ldots, f(t_M, u_M)]^\top$, $\mathbb{1} = [1, \ldots, 1]^\top$. The collocation matrix $\mathbf{Q} \in \mathbb{R}^{M \times M}$ has entries

$$[\mathbf{Q}]_{ij} = \int_0^{\tau_i} \ell_j(s)ds =: q_{i,j}. \tag{2.5}$$

We refer to (2.4) as the collocation problem. This is equivalent to the first RKM step (1.2) since the collocation method is an IRK method with $\mathbf{A} = \mathbf{Q}$, $[\boldsymbol{b}]_i = \int_0^1 \ell_i(s)ds$, and $[\boldsymbol{c}]_i = \tau_i$ [15, Thm. 7.7]. The second RKM step (1.3) is equivalent to the update

$$u(t_0 + \Delta t) \approx u_0 + \Delta t \boldsymbol{b}^\top f(\boldsymbol{u}). \tag{2.6}$$

Note that if $\tau_M = 1$, we can also set $u(t_0 + \Delta t) \approx u_M$ instead.

Remark 2.1. Using the update (2.6) improves the order of the step solution but can also reduce numerical stability [35, Rem. 4]. This is confirmed by numerical experiments (not presented here), as some preconditioners that appear to be A-stable lose this property when performing the collocation update. While this aspect would need further investigation, it motivates us to only use node distributions with $\tau_M = 1$ (i.e., Radau-right and Lobatto) for better numerical stability.

The distribution and type of quadrature nodes control the order of the collocation method [44, Table 2.1]. We use Legendre polynomials, but other types can be used as well. The interested reader can consult the book by Gautschi [12] for details and examples. For example, Lobatto-type nodes for a Legendre distribution produce a method of order $2M - 2$. Radau-II-type nodes for a Legendre distribution produce a

method of order $2M-1$. We refer to Radau-II quadrature as Radau-right to make explicit that it includes the right boundary node $\tau_M = 1$.

SDC solves (2.4) with the preconditioned fixed-point iteration

$$\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \mathbf{P}^{-1}\left[u_0\mathbb{1} - \left(\boldsymbol{u}^k - \Delta t \mathbf{Q} f(\boldsymbol{u}^k)\right)\right], \tag{2.7}$$

where $\mathbf{P}[\boldsymbol{u}] := \boldsymbol{u} - \Delta t \mathbf{Q}_\Delta f(\boldsymbol{u})$, and $\mathbf{Q}_\Delta \in \mathbb{R}^{M\times M}$ is a matrix called the SDC preconditioner. Because $\mathbf{Q}_\Delta$ is typically chosen to be lower triangular, the inversion of $\mathbf{P}$ can be computed node by node by forward substitution. Therefore, one SDC iteration (2.7) is often called a "sweep." It is fully defined by the used preconditioner $\mathbf{P}$. Note that the *SDC preconditioner* $\mathbf{Q}_\Delta$ is problem independent in contrast to $\mathbf{P}$. The generic form of an SDC sweep is

$$\boldsymbol{u}^{k+1} - \Delta t \mathbf{Q}_\Delta f(\boldsymbol{u}^{k+1}) = u_0\mathbb{1} + \Delta t(\mathbf{Q} - \mathbf{Q}_\Delta) f(\boldsymbol{u}^k). \tag{2.8}$$

By setting $\mathbf{Q}_\Delta = 0$ we retrieve the classical Picard iteration (`PIC`). The original SDC method [7] considers $\mathbf{Q}_\Delta$ matrices based on explicit Euler (`EE`) and implicit Euler (`IE`)

$$\mathbf{Q}_\Delta^{\texttt{EE}} = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \Delta\tau_2 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Delta\tau_2 & \Delta\tau_3 & \dots & \Delta\tau_M & 0 \end{bmatrix}, \quad \mathbf{Q}_\Delta^{\texttt{IE}} = \begin{bmatrix} \Delta\tau_1 & 0 & \dots & 0 \\ \Delta\tau_1 & \Delta\tau_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \Delta\tau_1 & \Delta\tau_2 & \dots & \Delta\tau_M \end{bmatrix},$$

where $\Delta\tau_m = \tau_m - \tau_{m-1}$ for $1 \le m \le M$, and $\tau_0 = 0$. The computational cost of solving (2.8) with forward substitutions is the same as that of a DIRK method with $s = M$ stages.

Remark 2.2. We can also define the SDC sweep based on the $\mathbf{A}$ and $\boldsymbol{c}$ Butcher arrays of a DIRK method. Setting $\mathbf{Q}_\Delta = \mathbf{Q} = \mathbf{A}$ retrieves (1.2) from (2.8), independent of $\boldsymbol{u}^0$. The collocation update (2.6) is equivalent to the RKM update (1.3). Hence, any generic SDC implementation based on (2.8) can be used to run any type of DIRK or explicit RK method. Such an approach is implemented in `pySDC` [40].

As first suggested by Speck [39], it is also possible to use a diagonal $\mathbf{Q}_\Delta$. This allows one to compute the sweep update for all nodes in parallel using M threads or processes. Note that if we consider any IRK method introduced by van der Houwen and Sommeijer [44, eq. 3.1a] with a dense Butcher matrix $\mathbf{A}$, the diagonal preconditioned iteration in (1.2) is equivalent to the generic SDC sweep (2.8) with a diagonal $\mathbf{Q}_\Delta$.

Possible choices for the entries of $\mathbf{Q}_\Delta$ that have been suggested are the diagonal elements of $\mathbf{Q}$, that is, $\mathbf{Q}_\Delta = \operatorname{diag}(q_{11}, \dots, q_{MM})$, or $\mathbf{Q}_\Delta = \operatorname{diag}(\tau_1, \dots, \tau_M)$, corresponding to an implicit Euler step from t_0 to t_m (`IEpar`). However, these preconditioners result in slow convergence of the SDC iteration (2.7), making it inefficient [39, 44]. Hence, we focus on finding better coefficients for a diagonal $\mathbf{Q}_\Delta$ for which the resulting SDC iteration converges rapidly.

2.2. Optimal coefficients for diagonal preconditioning. Like van der Houwen and Sommeijer and Speck [44, 39] we start with Dahlquist's test equation

$$\frac{du}{dt} = \lambda u, \quad \lambda \in \mathbb{C},\ t \in [0, T],\ u(0) = 1. \tag{2.9}$$

Applying (2.8) to (2.9) results in the sweep

$$(\mathbf{I} - \Delta t \lambda \mathbf{Q}_\Delta)\boldsymbol{u}^{k+1} = \Delta t \lambda (\mathbf{Q} - \mathbf{Q}_\Delta)\boldsymbol{u}^k + \boldsymbol{u}_0. \tag{2.10}$$

TABLE 1
Spectral radius $\rho(\mathbf{K}_{\text{NS}})$ of the nonstiff iteration matrix for optimal diagonal coefficients found in the literature using $M = 4$ Radau-right nodes.

Coefficients	Reference	Spectral radius $\rho(\mathbf{K}_{\text{S}})$
`VDHS`	van der Houwen and Sommeijer 1991 [44]	0.025
`MIN`	Speck 2018 [39]	0.42
`MIN3`	Speck at al. 2024 [40]	0.0081

Let $\boldsymbol{e}^k := \boldsymbol{u}^k - \boldsymbol{u}$ be the error w.r.t the exact solution of the collocation problem (2.4) and $z := \Delta t\lambda$. The iteration matrix $\mathbf{K}(z)$ governing the error is

$$\boldsymbol{e}^{k+1} = \mathbf{K}(z)\boldsymbol{e}^k, \quad \mathbf{K}(z) = z\left(\mathbf{I} - z\mathbf{Q}_\Delta\right)^{-1}\left(\mathbf{Q} - \mathbf{Q}_\Delta\right), \tag{2.11}$$

To find optimal diagonal coefficients for $\mathbf{Q}_\Delta$, we use the spectral radius of the iteration matrix $\rho(\mathbf{K}(z))$ as the indicator for convergence speed. However, the dependency on $z = \Delta t\lambda$ would make the resulting optimization problem specific. Therefore, we consider the spectral radii of the matrices

$$\mathbf{K}_{\text{NS}} = \lim_{|z|\to 0} \frac{\mathbf{K}(z)}{z}, \quad \mathbf{K}_{\text{S}} = \lim_{|z|\to\infty} \mathbf{K}(z).$$

where (NS) denotes the *nonstiff* and (S) the *stiff* limit of the SDC iteration matrix.

In particular, we aim to find diagonal coefficients such that they become nilpotent to ensure fast asymptotic convergence [48]. Short algebraic calculations yield

$$\mathbf{K}_{\text{NS}} = \mathbf{Q} - \mathbf{Q}_\Delta, \tag{2.12}$$

$$\mathbf{K}_{\text{S}} = \mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q}. \tag{2.13}$$

The $\mathbf{K}_{\text{S}}$ matrix was considered both by van der Houwen and Sommeijer [44] and Speck [39]. Both noticed the difficulty of finding optimal coefficients when minimizing the spectral radius of $\mathbf{K}_{\text{S}}$. Several diagonal coefficients are available in the literature and summarized in Appendix A. Table 1 shows the spectral radius of the resulting $\mathbf{K}_{\text{S}}$. Values for $\rho(\mathbf{K}_{\text{S}})$ depend on the used optimization approach. The `MIN3` coefficients proposed by Speck are similar to `VDHS` but were obtained differently by using an online black box optimization software that, unfortunately, is no longer available.

The matrix in the nonstiff limit $\mathbf{K}_{\text{NS}}$ was only considered by van der Houwen and Sommeijer [44] but discarded because of the difficulty of numerically optimizing the spectral radius of $\mathbf{K}_{\text{NS}}$ and the poor performance of the obtained diagonal coefficients. Both $\mathbf{K}_{\text{S}}$ and $\mathbf{K}_{\text{NS}}$ become very poorly conditioned as M increases, which makes computing the optimal coefficients numerically very difficult. By contrast, we propose an analytical approach to find diagonal coefficients by focusing on nilpotency of $\mathbf{K}_{\text{NS}}$ and $\mathbf{K}_{\text{S}}$ instead.

2.2.1. Preliminaries. Given a set of distinct nodes $0 \le \tau_1 < \cdots < \tau_M \le 1$, we can associate any vector $\boldsymbol{x} \in \mathbb{R}^M$ uniquely with a polynomial $x \in P_{M-1}$ having real coefficients via the mapping

$$\Phi : \boldsymbol{x} \mapsto x(t) = \sum_{j=1}^{M} \boldsymbol{x}_j \ell_j(t), \tag{2.14}$$

where $\ell_i \in P_{M-1}$ are the Lagrange polynomials for the nodes $\tau_1, \ldots, \tau_M$. Inversely, a polynomial $x \in P_{M-1}$ can be mapped to a vector $\boldsymbol{x} \in \mathbb{R}^M$ by evaluating it at the

nodes and setting $\boldsymbol{x}_j = x(\tau_j)$. The bijective mapping Φ defines the isomorphism $\mathbb{R}^M \cong P_{M-1}$. Using the definition of the collocation matrix $\mathbf{Q}$ from (2.5), we obtain

$$\sum_{j=1}^{M} q_{m,j}\boldsymbol{x}_j = \sum_{j=1}^{M} \boldsymbol{x}_j \left(\int_0^{\tau_m} \ell_j(s)\, ds \right) = \int_0^{\tau_m} \sum_{j=1}^{M} \boldsymbol{x}_j \ell_j(s)\, ds = \int_0^{\tau_m} x(s)\, ds \tag{2.15}$$

for $m = 1, \ldots, M$ so that

$$\mathbf{Q}\boldsymbol{x} = \begin{bmatrix} \int_0^{\tau_1} x(s)\, ds \\ \vdots \\ \int_0^{\tau_M} x(s)\, ds \end{bmatrix}. \tag{2.16}$$

Hence, multiplying a vector $\boldsymbol{x} \in \mathbb{R}^M$ by $\mathbf{Q}$ generates a vector that has the associated polynomial integrated from zero to the quadrature nodes τ_m as components. Applying Φ to $\mathbf{Q}\boldsymbol{x}$ fits a polynomial through the points $(\tau_m, \int_0^{\tau_m} x(t)\, dt)$ so that[2]

$$Qx := \sum_{j=1}^{M} \left(\int_0^{\tau_j} x(s)\, ds \right) l_j(t). \tag{2.17}$$

The mappings $\mathbf{Q}$, Q, and Φ commute; see Figure 1.

PROPOSITION 2.3. *For $1 \le n \le M-1$, let $\boldsymbol{\tau}^n \in \mathbb{R}^M$ be the vector $(\tau_1^n, \ldots, \tau_M^n)$ and $\tau^n \in P_{M-1}$ the monomial $t \mapsto t^n$. Then*

$$\boldsymbol{\tau}^n \cong \tau^n. \tag{2.18}$$

Proof. By definition of the mapping, $\Phi\boldsymbol{\tau}^n$ is the polynomial of degree $M-1$ interpolating the points (τ_j, τ_j^n) for $j = 1, \ldots, M$. Since the monomial $\tau^n \in P_{M-1}$ interpolates these points and the interpolating polynomial is unique, we must have $\Phi\boldsymbol{\tau}^n = \tau^n$. ∎

Remark 2.4. Note that we can still apply Φ to vectors $\boldsymbol{\tau}^n \in \mathbb{R}^M$ for $n > M-1$, but we will no longer obtain the monomial τ^n. Instead, we obtain the polynomial

$$p(t) = \sum_{j=1}^{M} \tau_j^n l_j(t) \tag{2.19}$$

of degree $M-1$ that interpolates the points (τ_j, τ_j^n).

$$\begin{array}{ccc} \mathbb{R}^M & \xrightarrow{\mathbf{Q}} & \mathbb{R}^M \\ \Phi \downarrow & & \uparrow \Phi^{-1} \\ P_{M-1} & \xrightarrow{Q} & P_{M-1} \end{array}$$

FIG. 1. *Bijective mapping between $\mathbb{R}^M$ and P_{M-1}.*

[2]Strictly speaking, we should write $(Qx)(t)$ since $Qx \in P^{M-1}$, but we omit the argument t for less cluttered notation.

PROPOSITION 2.5. *With the definitions from Proposition* 2.3 *we have*

$$\mathbf{Q}\boldsymbol{\tau}^n \cong \frac{\tau^{n+1}}{n+1} \tag{2.20}$$

for $1 \le n \le M-2$.

Proof. Using (2.17), we have

$$Q\Phi\boldsymbol{\tau}^n = \sum_{j=1}^{M} \left(\int_0^{\tau_j} s^n \, ds \right) \ell_j(t) = \frac{1}{n+1} \sum_{j=1}^{M} \tau_j^{n+1} \ell_j(t). \tag{2.21}$$

Because $l_j(\tau_m) = 1$ if $j = m$ and zero otherwise, evaluating this polynomial at the nodes τ_j when applying Φ^{-1} recovers the values $\tau_j^{n+1}/(n+1)$ so that

$$\Phi^{-1} Q \Phi \boldsymbol{\tau}^n = \frac{1}{n+1} \boldsymbol{\tau}^{n+1}. \tag{2.22}$$

Because Φ is an isomorphism, we can apply it to both sides of the equation and, since $n+1 \le M-1$, use Proposition 2.3 to get

$$Q\Phi\boldsymbol{\tau}^n = \frac{1}{n+1} \Phi(\boldsymbol{\tau}^{n+1}) = \frac{1}{n+1} \tau^{n+1}. \tag{2.23}$$

Note that $\mathbf{Q}\boldsymbol{\tau}^n \cong Q\Phi\boldsymbol{\tau}^n$ completes the proof. ∎

PROPOSITION 2.6. *Consider a set of nodes* $0 \le \tau_1 < \cdots < \tau_M \le 1$ *with* $\tau_1 > 0$ *and* $m \in \mathbb{N}$. *Let*

$$\mathbf{Q}_{\Delta,m} := \operatorname{diag}\left(\frac{\tau_1}{m}, \ldots, \frac{\tau_M}{m} \right) \tag{2.24}$$

be a diagonal matrix with entries τ_j/m *and* $Q_{\Delta,m} := \Phi \mathbf{Q}_{\Delta,m} \Phi^{-1}$. *For* $1 \le n \le M-2$, *it holds that*

$$\mathbf{Q}_{\Delta,m} \boldsymbol{\tau}^n \cong \frac{\tau^{n+1}}{m}, \tag{2.25}$$

where τ^{n+1} *is again the monomial* $t \mapsto t^{n+1}$.

Proof. We have

$$\mathbf{Q}_{\Delta,m} \boldsymbol{\tau}^n = \begin{bmatrix} \frac{\tau_1}{m} \tau_1^n \\ \vdots \\ \frac{\tau_M}{m} \tau_m^n \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \tau_1^{n+1} \\ \vdots \\ \tau_M^{n+1} \end{bmatrix} = \frac{1}{m} \boldsymbol{\tau}^{n+1}. \tag{2.26}$$

Applying Φ and using Proposition 2.3 yields

$$\mathbf{Q}_{\Delta,m} \boldsymbol{\tau}^n \cong \Phi \mathbf{Q}_{\Delta,m} \boldsymbol{\tau}^n = \Phi \frac{\boldsymbol{\tau}^{n+1}}{m} = \frac{\tau^{n+1}}{m}. \tag{2.27}$$

∎

PROPOSITION 2.7. *For nodes* $0 < \tau_1 < \cdots < \tau_M \le 1$, *we have*

$$\mathbf{Q}_{\Delta,m}^{-1} \boldsymbol{\tau}^{n+1} = m \boldsymbol{\tau}^n \cong m \tau^n = Q_{\Delta,m}^{-1} \tau^{n+1} \tag{2.28}$$

for $1 \le n \le M-1$.

Proof. Because of $\tau_1 > 0$, the matrix $\mathbf{Q}_{\Delta,m}$ is invertible with inverse

$$\mathbf{Q}_{\Delta,m}^{-1} = \operatorname{diag}\left(\frac{m}{\tau_1}, \ldots, \frac{m}{\tau_M}\right), \tag{2.29}$$

and thus

$$\mathbf{Q}_{\Delta,m}^{-1}\boldsymbol{\tau}^{n+1} = \operatorname{diag}\left(\frac{m}{\tau_1}\tau_1^{n+1}, \ldots, \frac{m}{\tau_M}\tau_M^{n+1}\right) = m\boldsymbol{\tau}^n. \tag{2.30}$$

For $1 \le n \le M-1$ we have $\boldsymbol{\tau}^n \cong \tau^n$ by Proposition 2.3. Finally,

$$Q_{\Delta,m}^{-1}\tau^{n+1} = \Phi\mathbf{Q}_{\Delta,m}^{-1}\Phi^{-1}\tau^{n+1} = \Phi\mathbf{Q}_{\Delta,m}^{-1}\begin{bmatrix}\tau_1^{n+1}\\ \vdots \\ \tau_m^{n+1}\end{bmatrix} = m\Phi\begin{bmatrix}\tau_1^{n}\\ \vdots \\ \tau_M^{n}\end{bmatrix} = m\tau^n \tag{2.31}$$

using Proposition 2.3. ∎

These results will be used in the proofs in the next sections.

2.2.2. `MIN-SR-NS` preconditioning. Here we introduce the `MIN-SR-NS` SDC preconditioner $\mathbf{Q}_\Delta = \mathbf{Q}_{\Delta,M}$ with constant coefficients that is suited for non-stiff problems.

THEOREM 2.8. *For any set of collocation nodes $0 < \tau_1 < \ldots < \tau_M \le 1$, the matrix $\mathbf{Q} - \mathbf{Q}_{\Delta,M}$ is nilpotent with index M. For the MIN-SR-NS preconditioner, setting $\mathbf{Q}_\Delta = \mathbf{Q}_{\Delta,M}$, it holds that $\rho(\mathbf{K}_{\mathrm{NS}}) = 0$.*

Proof. Let $\boldsymbol{\tau}^{M-1} \in \mathbb{R}^M$. Then it holds that

$$\mathbf{Q}\boldsymbol{\tau}^{M-1} = \begin{bmatrix}\int_0^{\tau_1} t^{M-1}\, ds\\ \vdots \\ \int_0^{\tau_M} t^{M-1}\, ds\end{bmatrix} = \frac{1}{M}\boldsymbol{\tau}^M \tag{2.32}$$

and

$$\mathbf{Q}_{\Delta,M}\boldsymbol{\tau}^{M-1} = \frac{1}{M}\boldsymbol{\tau}^M \tag{2.33}$$

so that $\boldsymbol{\tau}^{M-1} \in \ker(\mathbf{Q} - \mathbf{Q}_{\Delta,M})$. By Proposition 2.3 and because Φ is an isomorphism, $\tau^{M-1} = \Phi\boldsymbol{\tau}^{M-1}$ is in the kernel of $Q - Q_{\Delta,m}$. For $1 \le n \le M-2$, Propositions 2.5 and 2.6 yield

$$(\mathbf{Q} - \mathbf{Q}_{\Delta,M})\boldsymbol{\tau}^n \cong \left(\frac{1}{n+1} - \frac{1}{M}\right)\tau^{n+1}. \tag{2.34}$$

Now consider any polynomial $p \in P_{M-1}$ with $p(t) = \sum_{j=1}^M p_j t^j$. Then,

$$(Q - Q_{\Delta,M})p = \sum_{j=1}^{M-1} p_j\left(\frac{1}{j+1} - \frac{1}{M}\right)t^{j+1} = \sum_{j=2}^{M} p_{j-1}\left(\frac{1}{j} - \frac{1}{M}\right)t^j \tag{2.35}$$

so that $(Q - Q_{\Delta,M})p \in (P_{M-1} \setminus P_0) \cup \{0\}$. By induction, we get

$$(Q - Q_{\Delta,M})^k p \in (P_{M-1} \setminus P_{k-1}) \cup \{0\}. \tag{2.36}$$

Applying $(Q - Q_{\Delta,M})$ M times yields

$$(Q - Q_{\Delta,M})^M p \in (P_{M-1} \setminus P_{M-1}) \cup \{0\} = \{0\}.$$

As $p \in P_{M-1}$ was arbitrary, we have $(Q - Q_{\Delta,M})^M = 0$. Because Φ is an isomorphism we find that $(\mathbf{Q} - \mathbf{Q}_{\Delta,M})^M = 0$ and therefore $\rho(\mathbf{K}_{\text{NS}}) = 0$. ∎

Remark 2.9. In the proof of Theorem 2.8, one can interpret $p(\tau)$ as a polynomial representation of the collocation error, since the iteration matrix is approximated by $\mathbf{K}_{\text{NS}}$. Hence, each SDC iteration using $\mathbf{Q}_{\Delta,M}$ preconditioning improves the solution quality by removing the lowest order term in the error, up to the point where there is no term left.

Note that the `MIN-SR-NS` coefficients are different from the ones derived by van der Houwen and Sommeijer [44, sect. 3.3.1]. They suggest using a diagonal matrix that satisfies

$$\mathbf{Q}_\Delta^{-1}\boldsymbol{\tau} = \mathbf{Q}^{-1}\boldsymbol{\tau},$$

which leads to $\mathbf{Q}_\Delta = \mathbf{Q}_{\Delta,1}$ and corresponds to using an implicit Euler step between t_0 and the nodes time t_m (`IEpar`). The reason is that they aimed to improve convergence of the nonstiff components of the solution for large time steps, which is different from minimizing the spectral radius of $\mathbf{K}_{\text{NS}}$.

2.2.3. `MIN-SR-S` preconditioning. Here we introduce an SDC preconditioner with constant coefficients that is suited for stiff problems.

DEFINITION 2.10. *Consider a set of collocation nodes $0 < \tau_1 < \cdots < \tau_M \leq 1$. We call a diagonal matrix $\mathbf{Q}_\Delta$ with increasing diagonal entries that minimizes*

$$\left|\det\left[(1-t)\mathbf{I} + t\mathbf{Q}_\Delta^{-1}\mathbf{Q}\right] - 1\right| \quad \forall t \in \{\tau_1, \ldots, \tau_M\} \tag{2.37}$$

a MIN-SR-S preconditioner for SDC. Such a preconditioner finds a local minimum for $\rho(\mathbf{K}_{\text{S}})$.

As mentioned above, using the spectral radius as an objective function makes the optimization problem very challenging to solve numerically. Instead, we search for diagonal coefficients such that $\mathbf{K}_{\text{S}}$ is nilpotent. While we have no guarantee that such coefficients exist for every M, it is known that for $M = 2$ there are two possible minimizers but only one where the coefficients are ordered [44]. If such coefficients exist for any M, the following holds:

$$\forall t \in \mathbb{R}, \quad \det\left[\mathbf{I} + t(\mathbf{Q}_\Delta^{-1}\mathbf{Q} - \mathbf{I})\right] - 1 = 0. \tag{2.38}$$

Since $\det\left[\mathbf{I} + t(\mathbf{Q}_\Delta^{-1}\mathbf{Q} - \mathbf{I})\right] - 1$ is a polynomial in t of degree M, we only need to check (2.38) for $M + 1$ points. Because the equation is trivially satisfied for $t = 0$, checking it for nodes $0 < \tau_1 < \cdots < \tau_M$ is sufficient to show (2.37).

Note that currently there is no theory showing whether (2.37) has one or more solutions or no solution. We use `MINPACK`'s `hybrd` algorithm implemented in `scipy` to find diagonal coefficients that minimize locally the spectral radius of $\mathbf{K}_{\text{S}}$. For $M = 4$, for example, the `MIN-SR-S` coefficients for Radau-right nodes shown in Appendix A give $\rho(\mathbf{K}_{\text{S}}) = 0.00024$. However, this approach does not ensure that the diagonal coefficients are increasingly ordered. Because ordered coefficients led to better stability in our numerical experiments (see the discussion in section 3.2), we used a particular

choice of starting value for the minimization. Because the `MIN-SR-NS` coefficients and the increasingly ordered coefficients minimizing the stiff spectral radius $\rho(\mathbf{K}_S)$ are similar, we used `MIN-SR-NS` as starting values for the optimization finding the `MIN-SR-S` coefficients. This yielded increasingly ordered coefficients up to $M = 4$. For larger values of M, we observed that we can fit a power-law of the form αt^β on $[0,1]$ through the points (τ_i^M, Md_i^M), $i = 1,\dots,M$, where $\boldsymbol{d}^M$ are the increasingly ordered coefficients that minimize $\rho(\mathbf{K}_S)$, and evaluate it to produce a starting value for $M+1$. Hence, we propose the following incremental procedure to provide good starting values for the optimization to compute $\boldsymbol{d}^{M+1}$. Assuming that $\boldsymbol{d}^M$ is known,

1. find values for α and β such that the power-law minimizes the distance to the points (τ_i^M, Md_i^M) in the L_2 norm,
2. compute $\widetilde{\boldsymbol{d}}^{M+1} = \alpha t^\beta/(M+1)$ for $t \in \boldsymbol{\tau}^{M+1}$,
3. find a numerical solution for (2.37) using $\widetilde{\boldsymbol{d}}^{M+1}$ as initial guess.

Iterating this process up to a desired M yielded increasingly ordered coefficients with a very small $\rho(\mathbf{K}_S)$ in all numerical experiments.

Remark 2.11. The assumption $\tau_1 \neq 0$ in Definition 2.10 guarantees that $\mathbf{Q}_\Delta$ is not singular. If the first collocation node is zero, e.g., for Lobatto nodes, the collocation matrix takes the form

$$\mathbf{Q} = \begin{bmatrix} x & \boldsymbol{y}^\top \\ \boldsymbol{q} & \widetilde{\mathbf{Q}} \end{bmatrix},$$

where $\boldsymbol{y}, \boldsymbol{q} = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_{M-1}]^\top \in \mathbb{R}^{M-1}$, and $x \in \mathbb{R}$. Then, (2.4) can be rewritten as

$$\begin{bmatrix} u_2 \\ \vdots \\ u_M \end{bmatrix} - \widetilde{\mathbf{Q}} \begin{bmatrix} f(u_2) \\ \vdots \\ f(u_M) \end{bmatrix} = \begin{bmatrix} u_0 + \boldsymbol{q}_1 f(u_0) \\ \vdots \\ u_0 + \boldsymbol{q}_{M-1} f(u_0) \end{bmatrix}$$

since $u(\tau_1) = u_0$. The matrix $\widetilde{\mathbf{Q}}$ is still a collocation matrix but now based on the nodes $0 < \tau_2 < \dots < \tau_M$. We can apply the approach described in Definition 2.10 for $\widetilde{\mathbf{Q}}$ to determine a diagonal $\widetilde{\mathbf{Q}}_\Delta$ and add a zero coefficient to build the diagonal $\mathbf{Q}_\Delta$ preconditioner for the original node distribution.

2.2.4. `MIN-SR-FLEX` preconditioning. The discussion in section 2.2.3 illustrates the difficulty of finding a single set of diagonal coefficients that minimize $\rho(\mathbf{K}_S)$ analytically. Therefore, here we consider a series of preconditioners that change from one iteration to the next, as was already suggested by Weiser [48, sect. 4.2]. Let $\mathbf{Q}_\Delta^{(k)}$ be the preconditioner used in the kth iteration of SDC. Telescoping the error iteration (2.11) gives

$$\boldsymbol{e}^k = \mathbf{K}^{(k)}(z) \dots \mathbf{K}^{(1)}(z) \boldsymbol{e}^0, \quad \mathbf{K}^{(k)}(z) = z \left(\mathbf{I} - z\mathbf{Q}_\Delta^{(k)}\right)^{-1} \left(\mathbf{Q} - \mathbf{Q}_\Delta^{(k)}\right). \tag{2.39}$$

Using the same calculation as for the iteration matrix in the stiff limit $\mathbf{K}_S$, we get

$$\lim_{|z|\to\infty} \boldsymbol{e}^k = \mathbf{K}_S^{(k)} \dots \mathbf{K}_S^{(1)} \boldsymbol{e}^0, \quad \mathbf{K}_S^{(k)} = \lim_{|z|\to\infty} \mathbf{K}^{(k)}(z) = \mathbf{I} - \left(\mathbf{Q}_\Delta^{(k)}\right)^{-1} \mathbf{Q}.$$

We use this result to introduce the following k-dependent preconditioning.

THEOREM 2.12. *For any set of nodes with $\tau_1 > 0$, we have*

$$\left(\mathbf{I} - \mathbf{Q}_{\Delta,M}^{-1}\mathbf{Q}\right) \dots \left(\mathbf{I} - \mathbf{Q}_{\Delta,2}^{-1}\mathbf{Q}\right)\left(\mathbf{I} - \mathbf{Q}_{\Delta,1}^{-1}\mathbf{Q}\right) = \mathbf{0},$$

where $\mathbf{Q}_{\Delta,m}$ is defined as in (2.24)*. Hence, using successive diagonal preconditioning $\mathbf{Q}_\Delta^{(k)} = \mathbf{Q}_{\Delta,k}$ with $k \in \{1,\dots,M\}$ provides SDC iterations such that $\lim_{|z|\to\infty} \boldsymbol{e}^k = 0$, with $\boldsymbol{e}^k$ being the iteration error defined in* (2.39)*. We call this preconditioning MIN-SR-FLEX.*

Proof. Since $\mathbf{Q} \cong Q$, $\mathbf{Q}_{\Delta,m}^{-1} \cong Q_{\Delta,m}^{-1}$, and $\mathbf{I} \cong I$, we have

$$\left(\mathbf{I} - \mathbf{Q}_{\Delta,m}^{-1}\mathbf{Q}\right) \cong \left(I - Q_{\Delta,m}^{-1}Q\right). \tag{2.40}$$

For any monomial $\tau^n \in P_{M-1}$, Proposition 2.3 yields

$$\left(I - Q_{\Delta,m}^{-1}Q\right)\tau^n = \tau^n - Q_{\Delta,m}^{-1}Q\Phi\boldsymbol{\tau}^n. \tag{2.41}$$

With the help of equation (2.23) we find that

$$\left(I - Q_{\Delta,m}^{-1}Q\right)\tau^n = \tau^n - Q_{\Delta,m}^{-1}\frac{\tau^{n+1}}{n+1}. \tag{2.42}$$

Finally, Proposition 2.7 gives us

$$\left(I - Q_{\Delta,m}^{-1}Q\right)\tau^n = \tau^n - \frac{m\tau^n}{n+1} = \left(1 - \frac{m}{n+1}\right)\tau^n. \tag{2.43}$$

If $m = n+1$, the right-hand side is zero and therefore $\tau^n \in \ker(I - Q_{\Delta,n+1}^{-1})$.

Let $p \in P_{M-1}$, $p(t) = \sum_{j=0}^{M-1} p_j t^j$ be an arbitrary polynomial of degree $M-1$. Then

$$\left(I - Q_{\Delta,1}^{-1}Q\right)p(t) = \frac{1}{2}p_1 t + \frac{2}{3}p_2 t^2 + \dots + \frac{M-1}{M}p_{M-1}t^{M-1} \in (P_{M-1} \setminus P_0) \cup \{0\}. \tag{2.44}$$

Applying $I - Q_{\Delta,2}^{-1}Q$ removes the linear term so that

$$\left(I - Q_{\Delta,2}^{-1}Q\right)\left(I - Q_{\Delta,1}^{-1}Q\right)p \in (P_{M-1} \setminus P_1) \cup \{0\}. \tag{2.45}$$

Repeating this until $k = M$ yields

$$\left(I - Q_{\Delta,M-2}^{-1}Q\right) \circ \dots \circ \left(I - Q_{\Delta,2}^{-1}Q\right) \circ \left(I - Q_{\Delta,1}^{-1}Q\right)(p) \in P_{M-1} \setminus P_{M-1} \cup \{0\}, \tag{2.46}$$

that is, p gets mapped to the zero polynomial. Since p was arbitrary, the mapping must be the zero mapping. Using (2.40) shows that

$$\left(\mathbf{I} - \mathbf{Q}_{\Delta,M}^{-1}\mathbf{Q}\right) \dots \left(\mathbf{I} - \mathbf{Q}_{\Delta,2}^{-1}\mathbf{Q}\right)\left(\mathbf{I} - \mathbf{Q}_{\Delta,1}^{-1}\mathbf{Q}\right) = \mathbf{0}. \qquad \square$$

Note that an observation similar to that stated in Remark 2.9 concerning the error reduction can be made for the `MIN-SR-FLEX` preconditioner from Theorem 2.12. As before, Remark 2.11 holds `MIN-SR-FLEX` preconditioners for any node distribution with $\tau_1 = 0$. Theorem 2.12 defines the preconditioner for a fixed number of iterations $k \le M$ and lets the user choose the coefficients for $k > M$. Since this preconditioning is tailored to stiff problems, we use the `MIN-SR-S` preconditioning from Definition 2.10 in combination with `MIN-SR-FLEX` when $k > M$ in all numerical experiments.

3. Convergence order and stability. This section investigates the convergence order and stability of parallel SDC with a fixed number of sweeps. This approach is easier to compare against classical RKM than using SDC with a residuum-based stopping criterion [39]. We use nodes from a Legendre distribution, but the reader can generate plots for other choices with the provided code [40]. In all cases, the initial solution $\boldsymbol{u}^0$ for the SDC iteration is generated by copying the initial value u_0 to all nodes.

3.1. Convergence order. As a rule of thumb, the SDC order of convergence increases by at least one per iteration. This has been proved for implicit and explicit Euler methods as sweepers [7, Thm. 4.1] as well as for higher order correctors [5, Thm. 3.8]. However, except for equidistant nodes, sweepers of higher order are not guaranteed to provide more than one order per sweep. For `LU`-SDC there is numerical evidence that it increases order by one per sweep [48] but no proof seems to exist. Van der Houwen, Sommeijer, and Couzy [45, Thm. 2.1] show that for any diagonal preconditioner $\mathbf{Q}_\Delta$ the order of the method after K sweeps is $\min(K, p^*)$, where p^* is the order of the underlying collocation method. We confirm this numerically for the Dahlquist test problem (2.9) with $\lambda = i$ and $T = 2\pi$.

Figure 2 shows the L_∞-error against the analytical solution for SDC with the `MIN-SR-NS` preconditioner with $M = 4$ Radau-Right nodes (left) and $M = 5$ Lobatto nodes (right) for $K = 1, 2, \ldots, M$ sweeps. The underlying collocation methods are of orders seven and eight so that the order of SDC is determined by K. For Radau-right nodes, SDC gains one order per sweep for $K = 1$ and $K = 2$, while the third sweep increases the order by two. The same happens for Lobatto nodes when going from $K = 3$ to $K = 4$ sweeps. This unexpected order gain has also been observed for other configurations of the `MIN-SR-NS` preconditioner, but we do not yet have a theoretical explanation for this. Increased order of `MIN-SR-NS` is also observed for more complex problems; see section 4.2.

Figure 3 shows convergence of SDC with `MIN-SR-S` preconditioner, while Figure 4 shows convergence for `MIN-SR-FLEX`. In both cases the order increases by one per sweep but without the additional gains we observed for `MIN-SR-NS`. Also, errors for `MIN-SR-S` and `MIN-SR-FLEX` are generally higher than for `MIN-SR-NS`. This is expected, as `MIN-SR-NS` is optimized for nonstiff problems, and the Dahlquist problem with $\lambda = i$ is not stiff. Note that no results seem to exist in the literature that analyze convergence

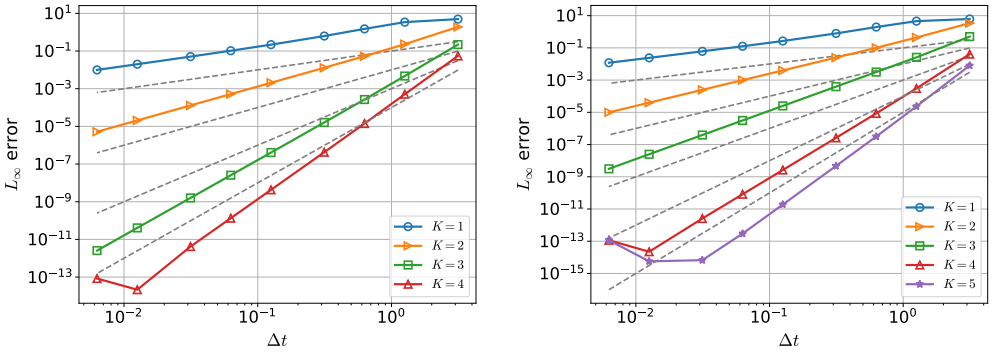

FIG. 2. *Convergence of SDC for the Dahlquist test equation with MIN-SR-NS preconditioner using $K = 1, \ldots, M$ sweeps per time step. Dashed lines with slopes from one to K are shown as a visual guide. Left: $M = 4$ Radau-right nodes. Right: $M = 5$ Lobatto nodes.*

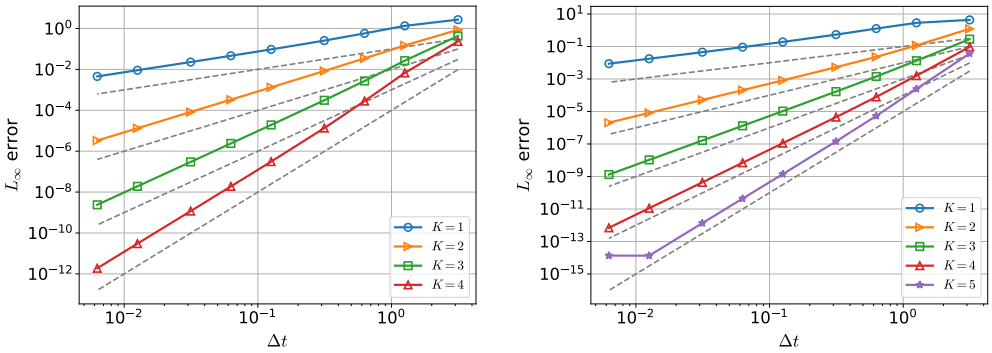

FIG. 3. *Convergence of SDC for the Dahlquist test equation with MIN-SR-S preconditioner using $K = 1, \ldots, M$ sweeps per time step. Dashed lines with slopes from one to K are shown as a visual guide. Left: $M = 4$ Radau-right nodes. Right: $M = 5$ Lobatto nodes.*

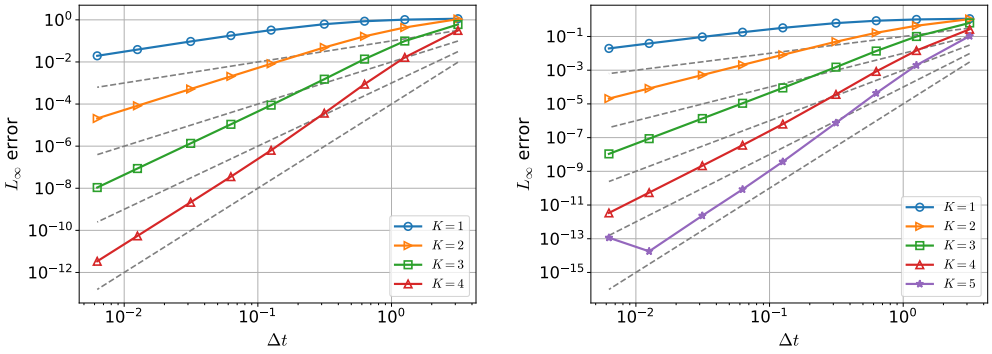

FIG. 4. *Convergence of SDC for the Dahlquist test equation with MIN-SR-FLEX preconditioner using $K = 1, \ldots, M$ sweeps per time step. Dashed lines with slopes from one to K are shown as a visual guide. Left: $M = 4$ Radau-right nodes. Right: $M = 5$ Lobatto nodes.*

order of a nonstationary SDC iteration like `MIN-SR-FLEX`, where the preconditioner changes in every iteration.

3.2. Numerical stability. We investigate stability of parallel SDC by numerically computing the border of its stability region

$$\mathcal{S}_C = \{z \in \mathbb{C} \text{ s.t. } |R(z)| = 1\}, \tag{3.1}$$

where R is the stability function of a given SDC configuration. A method is called A-stable if $\mathcal{S}_C \subset \mathbb{C}^-$, that is, if the stability domain includes the negative complex half-plane. We use $M = 4$ Radau-right nodes from a Legendre distribution for all experiments. Other configurations can again be analyzed using the provided code.

Figure 5 shows stability for SDC with the `PIC` preconditioner (Picard iteration). This is a fully explicit method and, as expected, not A-stable for any K. Interestingly, K-many sweeps reproduce the stability contour of the explicit RKM of order K with K stages. In particular, we recognize the stability regions of explicit Euler for $K = 1$ and of the classical explicit RKM of order 4 for $K = 4$.

Figure 6 shows stability regions for `MIN-SR-NS`. As for the Picard iteration, the method is not A-stable for any number of sweeps, but the stability regions are

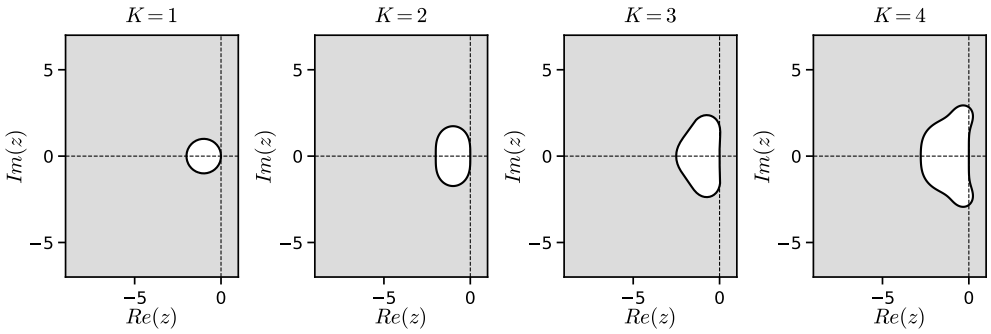

FIG. 5. *Stability region for SDC with $M=4$ Radau-right nodes using the PIC preconditioner for $K=1,2,3,4$. The gray zones are unstable areas in the complex plane.*

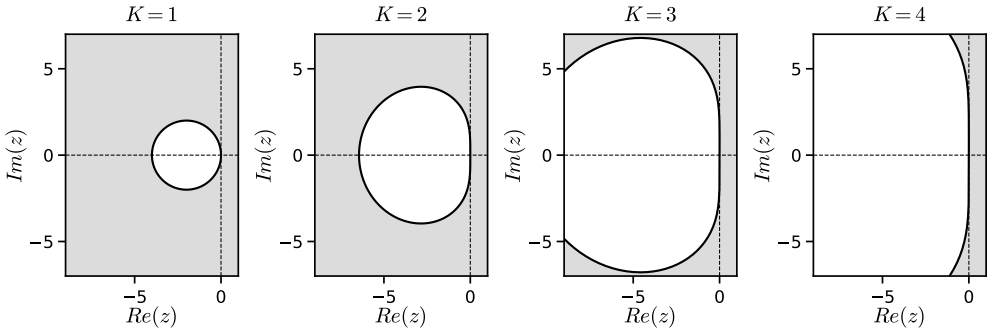

FIG. 6. *Stability region for SDC with $M=4$ Radau-right nodes using the MIN-SR-NS preconditioner for $K=1,2,3,4$. The gray zones are unstable areas in the complex plane.*

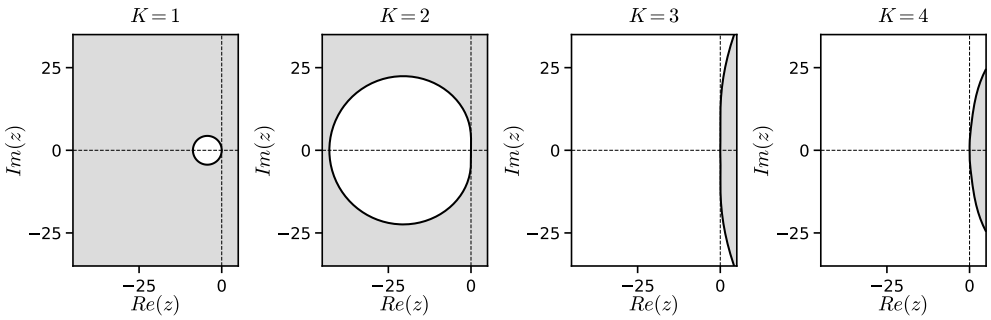

FIG. 7. *Stability region for SDC with $M=4$ Radau-right nodes using the MIN-SR-S preconditioner for $K=1,2,3,4$. The gray zones are unstable areas in the complex plane. Note that the λ ranges are five times larger than those for the previous Figures 5 and 6.*

significantly larger. That `MIN-SR-NS` is not A-stable is not unexpected since the coefficients are optimized for the non-stiff case where $|z| \simeq 1$.

Figures 7 and 8 show stability regions for `MIN-SR-S` and `MIN-SR-FLEX`. While SDC with `MIN-SR-S` appears to be A-stable for $K \geq 3$, SDC with `MIN-SR-FLEX` seems A-stable for any number of sweeps K. However, this only holds for Radau-right nodes. For example, SDC with $M = 5$ Lobatto nodes and $K = 4$ sweeps is not A-stable (stability regions are not shown here). A theoretical investigation is left for later work.

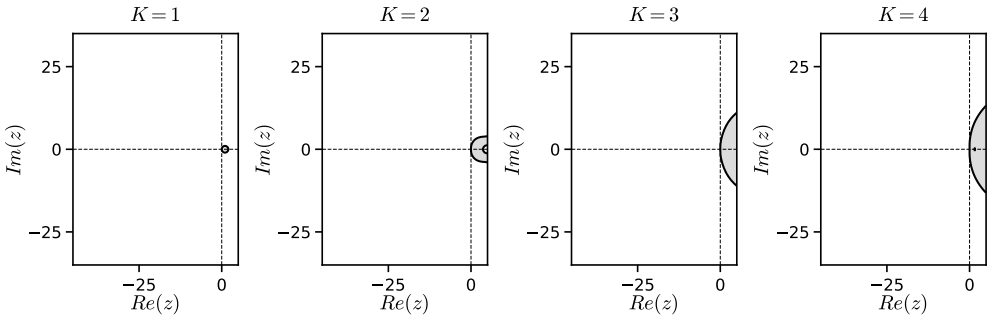

FIG. 8. *Stability region for SDC with $M = 4$ Radau-right nodes using the MIN-SR-FLEX preconditioner for $K = 1, 2, 3, 4$. The gray zones are unstable areas in the complex plane.*

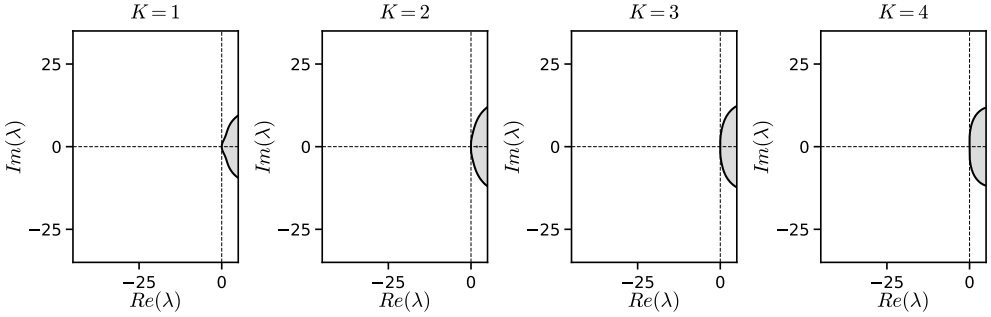

FIG. 9. *Stability region for SDC with $M = 4$ Radau-right nodes using the LU preconditioner for $K = 1, 2, 3, 4$. The gray zones are unstable areas in the complex plane. Note that the stability regions for IE-SDC are very similar.*

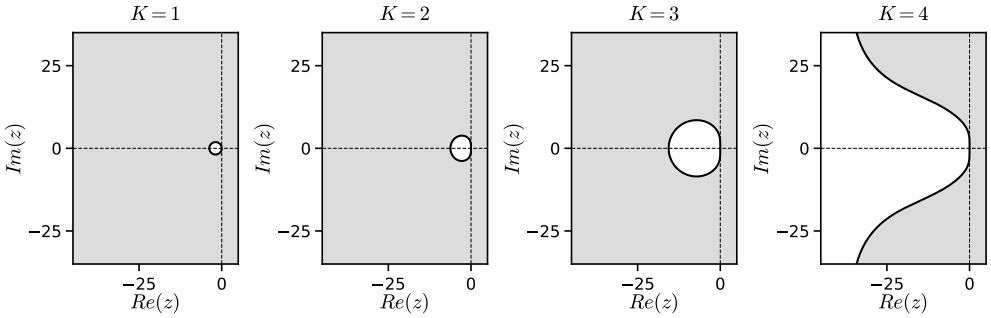

FIG. 10. *Stability region for SDC with $M = 4$ Radau-right nodes using the VDHS preconditioner for $K = 1, 2, 3, 4$. The gray zones are unstable areas in the complex plane.*

For comparison, we show the stability contours of the *LU* preconditioner by Weiser [48] in Figure 9. Stability for SDC with a standard implicit Euler sweeper is very similar and not shown. Note that the `LU` preconditioner is lower triangular and does not allow for parallelism in the sweep. Since stability regions of `MIN-SR-FLEX` and `LU` are similar, we can conclude that, with an optimized choice of coefficients, parallelism in SDC can be obtained without loss of stability. For further comparison, we show the stability contours for the `VDHS` preconditioner, obtained by minimizing the spectral radius of $\mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q}$ [44, sect. 4.3.4] in Figure 10. For $K \in \{1, 2, 3\}$ the stability regions

are very small compared to `LU` or `MIN-SR-FLEX`. Even though the stable regions grow as K increases, it does not include the imaginary axis, making the method unsuitable for oscillatory problems with purely imaginary eigenvalues. Bounded stability regions are also observed for the MIN preconditioner [39] (not shown). Since the spectral radius of $\mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q}$ is significantly larger for `MIN` than `MIN-SR-S` and `VDHS`, we do not consider it in the rest of this paper.

Remark 3.1. Experiments not documented here suggest that when minimizing the spectral radius of the stiff limit $\mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q}$, enforcing monotonically increasing coefficients for `MIN-SR-S` provides a notable improvement in numerical stability.

4. Computational efficiency. We compare computational cost versus accuracy of our three new SDC preconditioners against SDC preconditioners from the literature, the classical explicit $4th$ order (RK4) and the L-stable stiffly accurate implicit RKM ESDIRK4(3)6L[2]SA [20] (ESDIRK43). The two RKM were implemented in `pySDC` by Baumann et al. [1]. Again, all figures in this section can be reproduced using scripts in the provided code [40].

4.1. Estimating computational cost. A fair run-time assessment of parallel SDC requires an optimized parallel implementation, which is the subject of a separate work [10]. Here we instead estimate computational cost by considering the elementary operations of each scheme.

To solve a system of ODEs (1.1), both SDC and RKM need to

1. evaluate the right-hand-side (RHS) $f(u,t)$ for given u, t; and
2. solve the following nonlinear system for some b, t and α:

$$u - \alpha f(u,t) = b. \tag{4.1}$$

We solve (4.1) with an exact Newton iteration, starting from the initial solution u_0 or the previous SDC iterate u_m^{k-1}. We stop iterating when a set problem-dependent tolerance is reached or after 300 iterations. In our numerical experiments, the Jacobian J_f can be computed, and $I - \alpha J_f$ is inverted analytically. In this case, the cost of one Newton iteration is similar to the cost of an RHS evaluation. For all methods, we therefore model that computational cost of a simulation by $N_{\text{Newton}} + N_{\text{RHS}}$. For the Allen–Cahn problem, since the cost of one Newton iteration is bigger than this for one RHS evaluation, we model the computational cost by $2N_{\text{Newton}} + N_{\text{RHS}}$. Note that $N_{\text{Newton}} = 0$ for explicit methods like RK4 or `PIC`-SDC.

For parallel SDC, the computations in every sweep can be parallelized across M threads. To account for unavoidable overhead from communication or competition for resources between threads, we assume a parallel efficiency of $P_{\text{eff}} = 0.8$ or 80%, which is achievable in optimized implementations using compiled languages and `OpenMP` [10]. Thus, in our performance model, we divide the computational cost estimate for SDC by MP_{eff} instead of M. The cost model is validated below against wall-clock time measurements for the Allen–Cahn problem using MPI parallelization. Parallel efficiencies there range between 74% and 93%, depending on time step size, providing additional evidence that our assumption of 80% efficiency is reasonable. No run-times are shown for the Lorenz or Prothero–Robinson problem but can be generated with the provided code [40].

4.2. Lorenz system. We first consider the nonlinear system of ODEs

$$\frac{dx}{dt} = \sigma(y - x), \quad \frac{dy}{dt} = x(\rho - z) - y, \quad \frac{dz}{dt} = xy - \beta z, \tag{4.2}$$

with $(\sigma, \rho, \beta) = (10, 28, 8/3)$. The initial value is $(x(0), y(0), z(0)) = (5, -5, 20)$, and we use a Newton tolerance of 10^{-12}. We set the final time for the simulation to $T = 1.24$, which corresponds to two revolutions around one of the attraction points. A reference solution is computed using an embedded RKM of order 5(4) [6] implemented in `scipy` with an error tolerance of 10^{-14}.

Figure 11 shows error versus time step size for `MIN-SR-NS` for SDC for $K = 1, \ldots, 5$. Since the Lorenz system is not stiff, we compare against Picard iteration (`PIC`), which is known to be efficient for nonstiff problems. We do see the expected order increase by one per iteration as well as the additional order gain for `MIN-SR-NS` starting at $K = 3$, which was seen already for the Dahlquist problem. For the same number of sweeps, this makes `MIN-SR-NS` more accurate than Picard iteration or other SDC methods.

Figure 12 shows error against modeled computational cost for `MIN-SR-S` and a variety of other SDC variants (left) as well as RKM and the `VDHS` preconditioner (right). The parallel and PIC preconditioners significantly outperform classical SDC with explicit Euler or LU sweep. As expected, the preconditioner for nonstiff problems, `MIN-SR-NS`, is the most efficient, although the stiff preconditioners `MIN-SR-S` and `MIN-SR-FLEX` remain competitive. `MIN-SR-NS` also outperforms the `VDHS` preconditioner and ESDIRK43 RKM. For errors above 10^{-4}, explicit RKM4 is more efficient

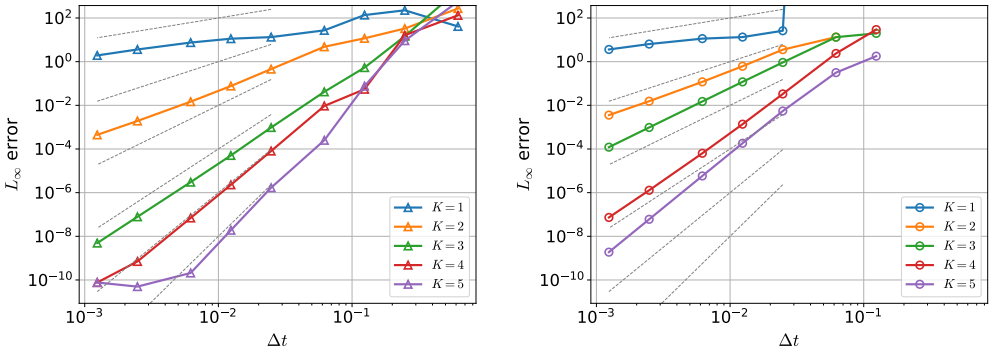

FIG. 11. *Error vs. time step size for SDC for the Lorenz problem using $M = 4$ Radau-right nodes for $K = 1, \ldots, 5$ sweeps per time step. Left: MIN-SR-NS preconditioner. Right: PIC preconditioner. Dashed gray lines with slopes from 1 to 6 are shown as a visual guide.*

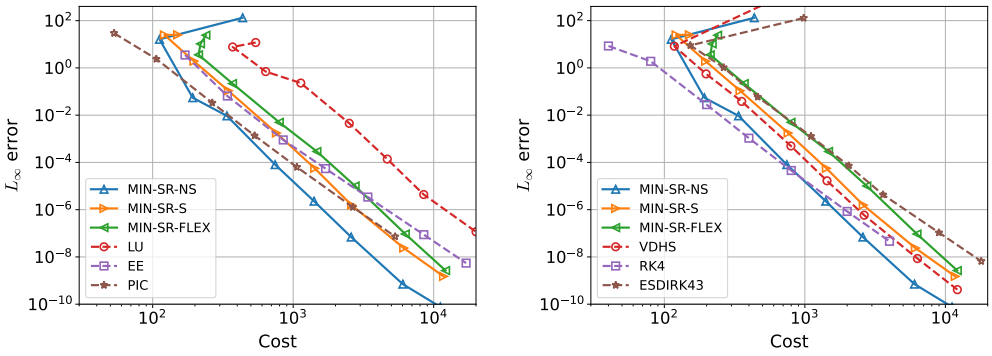

FIG. 12. *Error vs. cost for SDC for the Lorenz problem using $M = 4$ Radau-right nodes and $K = 4$ sweeps. Left: comparison with classical SDC preconditioners. Right: comparison with efficient time integration methods from the literature and SDC with VDHS preconditioner.*

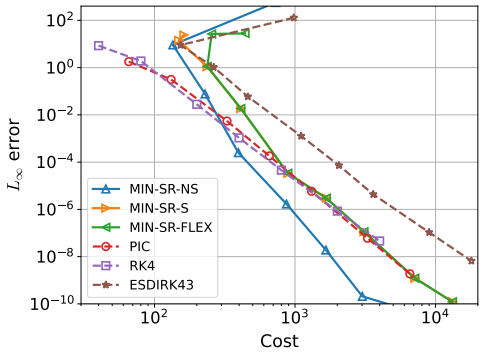

FIG. 13. *Error vs. cost for SDC and Picard iteration for the Lorenz problem using* $M = 4$ *Radau-right nodes and* $K = 5$ *sweeps in comparison to RKM4 and ESIDRK43.*

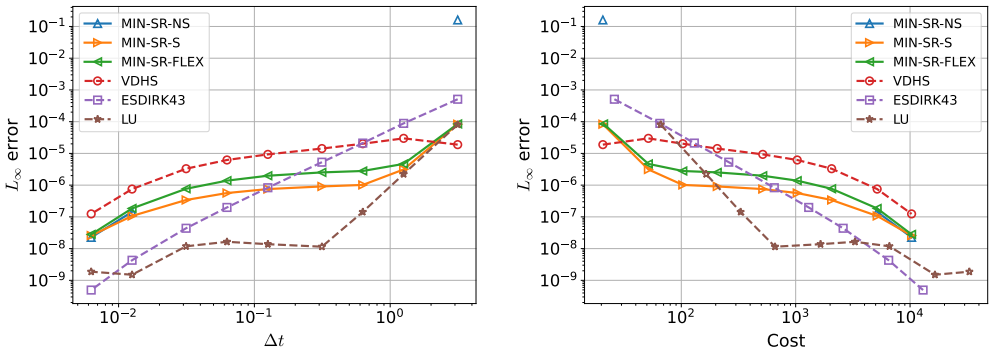

FIG. 14. *Error vs. time step size for parallel SDC and classical time integration schemes for the Prothero–Robinson problem with* $\varepsilon = 10^{-3}$. *SDC uses* $M = 4$ *Radau-right nodes and* $K = 4$ *sweeps. Left: error vs. time step. Right: error vs. cost.*

than `MIN-SR-NS`, but for errors below 10^{-6}, `MIN-SR-NS` outperforms RKM4. When increasing the number of sweeps to $K = 5$, the advantage in efficiency of `MIN-SR-NS` over RKM4 becomes more pronounced; see Figure 13.

4.3. Prothero–Robinson problem. Our second test case is the stiff ODE by Prothero and Robinson [32],

$$\frac{du}{dt} = \frac{u - g(t)}{\varepsilon} + \frac{dg}{dt}. \tag{4.3}$$

The analytical solution for this ODE is $u(t) = g(t)$, and we set $g(t) = \cos(t)$ and $\varepsilon = 10^{-3}$. We also set a Newton tolerance of 10^{-12}.

Figure 14 shows error versus time step size (left) and modeled computational cost (right) for our three parallel SDC methods, nonparallel `LU`-SDC, parallel `VDHS` SDC, and the implicit ESDIRK43 RKM [20]. All SDC variants use $K = 4$ sweeps. The parallel SDC variants all show a noticeable range where the error does not decrease with time step size. This is a known phenomenon for the Prothero–Robinson problem [48, sect. 6.1] and means that a very small time step is required to recover the theoretically expected convergence order. While `MIN-SR-S` outperforms `LU`-SDC in efficiency up to an error of around 10^{-6}, its stalling convergence results in `LU`-SDC being more efficient for very high accuracies.

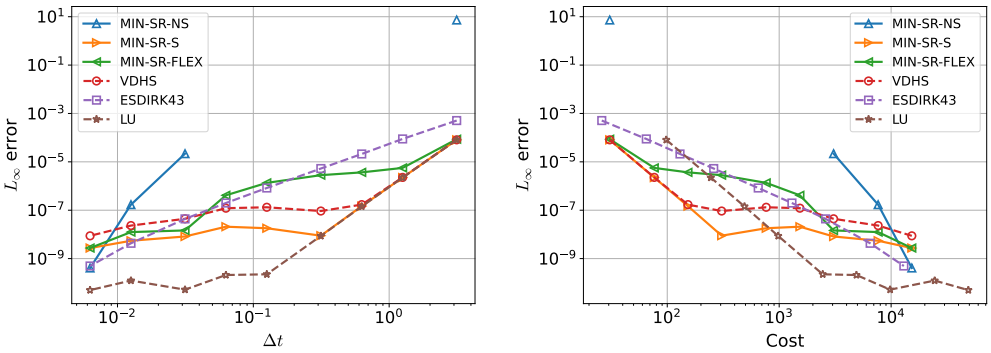

FIG. 15. *Comparison of diagonal SDC and classical time integration schemes on the Prothero–Robinson problem, using* $\varepsilon = 10^{-3}$. *Each SDC configuration uses* $M = 4$ *Radau-right nodes and* $K = 6$ *sweeps. Left: error vs. time step. Right: error vs. cost.*

If we increase the number of sweeps to $K = 6$, the error level at which convergence stalls is reduced, see Figure 15. This also pushes down the error where `LU`-SDC becomes more efficient than `MIN-SR-S` to around 10^{-8}. Nevertheless, in both cases `MIN-SR-S` will be an attractive integrator unless extremely tight accuracy is required.

4.4. Allen–Cahn equation. As a last test problem, we consider the one-dimensional Allen–Cahn equation with driving force

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - \frac{2}{\varepsilon^2} u(1-u)(1-2u) - 6 d_w u(1-u), \quad x \in [-0.5, 0.5], \quad t \in [0, T]. \tag{4.4}$$

using inhomogeneous Dirichlet boundary conditions. The exact solution is

$$u(x,t) = 0.5 \left[1 + \tanh\left(\frac{x - vt}{\sqrt{2}\varepsilon} \right) \right], \quad v = 3\sqrt{2}\varepsilon d_w, \tag{4.5}$$

and we use it to set the initial solution $u(x, 0)$ and the boundary conditions for $x = \pm 0.5$. We set $T = 50$ as a simulation interval and set parameters $\varepsilon = d_w = 0.04$. The spatial derivative is discretized with a second order finite-difference scheme on 2047 grid points, and we solve (4.1) in each Newton iteration with a sparse linear solver from `scipy`. Numerical experiments were run on one compute node of the `JUSUF` cluster (AMD EPYC 7742 2.25 GHz) at Jülich Supercomputing Center, using a locally compiled version of `Python = 3.11.9` (`GCC = 12.3.0`), with `numpy = 2.0.2`, `scipy = 1.14.1`, and `mpi4py = 4.0.0` wrapping `ParaStationMPI = 5.9.2-1`.

Figure 16 shows the absolute error in the L_2 norm at T versus the time step size. All methods converge to an error of around 2×10^{-4}, which corresponds to the space discretization error for the chosen grid. The fastest converging method is SDC with LU, followed by ESDIRK43 and SDC with MIN-SR-FLEX preconditioners.

Figure 17 (left) shows the absolute error in the L_2 norm at T versus modeled computational cost. All methods converge to an error of around $2 \cdot 10^{-4}$, which corresponds to the space discretization error for the chosen grid. The most efficient methods is `MIN-SR-FLEX` parallel SDC, followed by ESDIRK43 and SDC with LU preconditioner. Figure 17 (right) shows the absolute L_2 error versus wall-clock time in seconds. The plot looks very similar when using modeled cost, illustrating that our cost model gives a reasonable approximation of actual cost. Again, `MIN-SR-FLEX` is the most efficient integrator, except for very loose error tolerances of 10^{-1} and

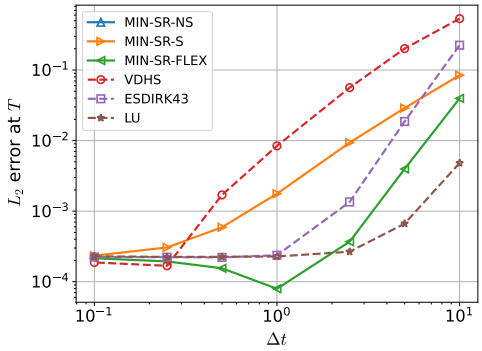

FIG. 16. *Comparison of the accuracy of the diagonal SDC preconditioners and classical time integration schemes for the Allen–Cahn equation. Each SDC configuration uses $M = 4$ Radau-right nodes and $K = 4$ sweeps.*

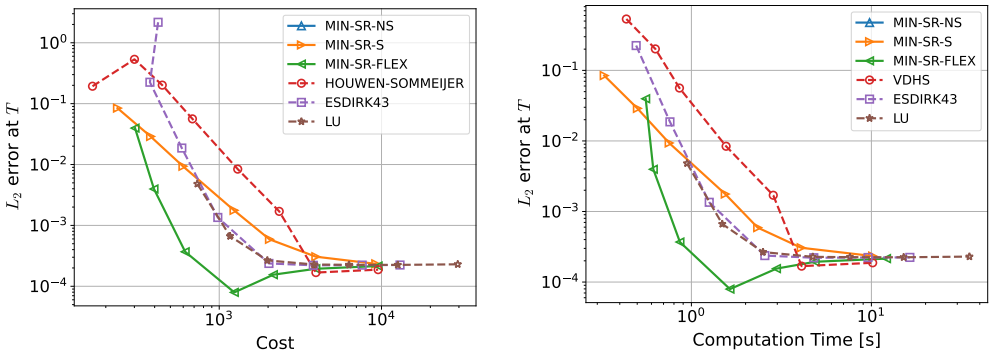

FIG. 17. *Comparison of the efficiency of diagonal SDC preconditioners and classical time integration schemes for the Allen–Cahn equation. Each SDC configuration uses $M = 4$ Radau-right nodes and $K = 4$ sweeps. Left: error vs. modeled computational cost. Right: error vs. computation time.*

above where `MIN-SR-S` is faster. Note that in order for SDC with `LU` preconditioner to catch up to `MIN-SR-FLEX` in terms of performance for a given time step size, the efficiency of the parallel implementation would need to be below 40%.

5. Conclusions. Using a diagonal preconditioner for spectral deferred corrections allows one to exploit small-scale parallelization in time for a number of threads up to the number of quadrature nodes in the underlying collocation formula. However, efficiency and stability of parallel SDC depend critically on the coefficients in the preconditioner. We consider two minimization problems, one for stiff and one for non-stiff time-dependent problems, that can be solved to find an optimized parameter. This allows us to propose three new sets of coefficients, `MIN-SR-NS` for non-stiff problems and `MIN-SR-S` and `MIN-SR-FLEX` for stiff problems. While we use numerical optimization to determine the `MIN-SR-S` diagonal coefficients, we can obtain the `MIN-SR-NS` and `MIN-SR-FLEX` coefficients analytically. `MIN-SR-FLEX` is a nonstationary iteration, where the preconditioner changes in every SDC sweep and is designed to generate a nilpotent error propagation matrix.

We demonstrate by numerical experiments that the three new variants of parallel SDC increase their order by at least one per sweep, similarly to existing nonparallel SDC methods. The variants designed for stiff problems have excellent stability

properties and might well be A-stable, although we do not have a rigorous proof yet. Stability regions of `MIN-SR-FLEX` in particular are very similar to those of the non-parallel `LU`-SDC variant and much larger than those of the iterated implicit Runge–Kutta methods by van der Houwen and Sommeijer [44]. To model computational efficiency, we count the number of right-hand side evaluations and Newton iterations for all schemes and assume 80% parallel efficiency for the implementation of parallel SDC. We compare error against computational effort of our three new parallel SDC variants against parallel SDC variants from the literature, explicit RKM4 and implicit ESDIRK43 for the nonstiff Lorenz system, the stiff Prothero–Robinson problem, and the stiff, one-dimensional Allen–Cahn equation. For all three test problems, the new parallel SDC variants can outperform existing parallel SDC variants, state-of-the-art serial SDC methods, as well as RKM4 and ESDIRK43. Wall-clock time measurements for the Allen–Cahn problem show that our cost model closely tracks actual run-times and that a parallel implementation of SDC can be more efficient than optimized serial SDC methods as well as Runge–Kutta methods.

Appendix A. Optimized diagonal coefficients for the stiff limit. Here, we repeat the coefficients used for our numerical experiments, either numerically computed or obtained from the literature, as well as the resulting spectral radius of the $\mathbf{K}_\mathrm{S}$ matrix. The values are for $M = 4$ Radau-right nodes from a Legendre distribution.

`VDHS` by van der Houwen and Sommeijer [45]

$\boldsymbol{d} = [0.32049937, 0.08915379, 0.18173956, 0.2333628], \quad \rho(\mathbf{K}_\mathrm{S}) = 0.025$

`MIN` by Speck [40]

$\boldsymbol{d} = [0.17534868, 0.0619158, 0.1381934, 0.19617814], \quad \rho(\mathbf{K}_\mathrm{S}) = 0.42$

`MIN3` by Speck et al. [41]

$\boldsymbol{d} = [0.31987868, 0.08887606, 0.18123663, 0.23273925], \quad \rho(\mathbf{K}_\mathrm{S}) = 0.0081$

`MIN-SR-S` introduced in this paper

$\boldsymbol{d} = [0.05363588, 0.18297728, 0.31493338, 0.38516736], \quad \rho(\mathbf{K}_\mathrm{S}) = 0.00024$

Reproducibility of computational results. This paper has been awarded the "SIAM Reproducibility Badge: Code and data available" as recognition that the authors have followed reproducibility principles valued by SISC and the scientific computing community. Code and data that allow readers to reproduce the results in this paper are available at https://zenodo.org/records/13828395.

REFERENCES

[1] T. Baumann, S. Götschel, T. Lunet, D. Ruprecht, and R. Speck, *Adaptive time step selection for spectral deferred corrections*, Numer. Algorithms, (2024), https://doi.org/10.1007/s11075-024-01964-z.

[2] K. Burrage, *Parallel methods for initial value problems*, Appl. Numer. Math., 11 (1993), pp. 5–25, https://doi.org/10.1016/0168-9274(93)90037-R.

[3] J. C. Butcher, *On the implementation of implicit Runge–Kutta methods*, BIT Numer. Math., 16 (1976), pp. 237–240, https://doi.org/10.1007/BF01932265.

[4] G. Caklovic, *ParaDiag and Collocation Methods: Theory and Implementation*, Ph.D. thesis, Karlsruher Institut für Technologie (KIT), 2023, https://doi.org/10.5445/IR/1000164518.

[5] A. Christlieb, B. Ong, and J.-M. Qiu, *Comments on high-order integrators embedded within integral deferred correction methods*, Comm. Appl. Math. Comput. Sci., 4 (2009), pp. 27–56, https://doi.org/10.2140/camcos.2009.4.27.

[6] J. R. Dormand and P. J. Prince, *A family of embedded Runge-Kutta formulae*, J. Comput. Appl. Math., 6 (1980), pp. 19–26, https://doi.org/10.1016/0771-050X(80)90013-3.

[7] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT Numer. Math., 40 (2000), pp. 241–266, https://doi.org/10.1023/A:1022338906936.

[8] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Comm. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132, https://doi.org/10.2140/camcos.2012.7.105.

[9] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36 (2014), pp. C635–C661, https://doi.org/10.1137/130944230.

[10] P. Freese, S. Götschel, T. Lunet, D. Ruprecht, and M. Schreiber, *Parallel Performance of Shared Memory Parallel Spectral Deferred Corrections*, preprint, arXiv:2403.20135, 2024.

[11] M. J. Gander, 50 *years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition, Springer, 2015, pp. 69–113, https://doi.org/10.1007/978-3-319-23321-5_3.

[12] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*, OUP Oxford, 2004.

[13] C. Gear, *Parallel methods for ordinary differential equations*, CALCOLO, 25 (1988), pp. 1–20, https://doi.org/10.1007/BF02575744.

[14] D. Guibert and D. Tromeur-Dervout, *Parallel deferred correction method for CFD problems*, in Parallel Computational Fluid Dynamics 2006, Elsevier, 2007, pp. 131–138.

[15] E. Hairer, G. Wanner, and S. P. Nørsett, *Solving Ordinary Differential Equations* I*: Nonstiff Problems*, 2nd ed., Springer-Verlag, 1993, https://doi.org/10.1007/978-3-540-78862-1.

[16] J. Huang, J. Jia, and M. Minion, *Accelerating the convergence of spectral deferred correction methods*, J. Comput. Phys., 214 (2006), pp. 633–656, https://doi.org/10.1016/j.jcp.2005.10.004.

[17] A. Iserles and S. P. Nørsett, *On the theory of parallel Runge-Kutta methods*, IMA J. Numer. Anal., 10 (1990), pp. 463–488, https://doi.org/10.1093/imanum/10.4.463.

[18] K. R. Jackson, *A survey of parallel numerical methods for initial value problems for ordinary differential equations*, IEEE Trans. Magn., 27 (1991), pp. 3792–3797, https://doi.org/10.1109/20.104928.

[19] K. R. Jackson and S. P. Nørsett, *The potential for parallelism in Runge–Kutta methods. Part* 1*: RK formulas in standard form*, SIAM J. Numer. Anal., 32 (1995), pp. 49–82, https://doi.org/10.1137/0732002.

[20] C. A. Kennedy and M. H. Carpenter, *Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A review*, Technical report NASA/TM–2016–219173, 2016, https://ntrs.nasa.gov/api/citations/20160005923/downloads/20160005923.pdf.

[21] S. Leveque, L. Bergamaschi, A. Martinez, and J. W. Pearson, *Fast Iterative Solver for the All-at-Once Runge–Kutta Discretization*, preprint, arXiv:2303.02090, 2023.

[22] I. Lie, *Some Aspects of Parallel Runge-Kutta Methods*, Technical report T-MC-87-3, Trondheim TU. Inst. Math., 1987, https://cds.cern.ch/record/201368.

[23] J.-L. Lions, Y. Maday, and G. Turinici, *A "parareal" in time discretization of PDE's*, C. R. Acad. Sci. Ser. I Math., 332 (2001), pp. 661–668, https://doi.org/10.1016/S0764-4442(00)01793-6.

[24] P. Munch, I. Dravins, M. Kronbichler, and M. Neytcheva, *Stage-Parallel Fully Implicit Runge–Kutta Implementations with Optimal Multilevel Preconditioners at the Scaling Limit*, SIAM J. Sci. Comput., 46 (2023), pp. S71–S96, https://doi.org/10.1137/22m1503270.

[25] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM, 7 (1964), pp. 731–733, https://doi.org/10.1145/355588.365137.

[26] S. P. Nørsett and H. H. Simonsen, *Aspects of parallel Runge-Kutta methods*, in Numerical Methods for Ordinary Differential Equations, A. Bellen, C. W. Gear, and E. Russo, eds., Springer, 1989, pp. 103–117.

[27] B. W. Ong, R. D. Haynes, and K. Ladd, *Algorithm* 965*: RIDC methods: A family of parallel time integrators*, ACM Trans. Math. Softw., 43 (2016), 8, https://doi.org/10.1145/2964377.

[28] B. W. Ong and R. J. Spiteri, *Deferred correction methods for ordinary differential equations*, J. Sci. Comput., 83 (2020), 60, https://doi.org/10.1007/s10915-020-01235-8.

[29] B. Orel, *Parallel Runge–Kutta methods with real eigenvalues*, Appl. Numer. Math., 11 (1993), pp. 241–250, https://doi.org/10.1016/0168-9274(93)90051-R.

[30] W. Pazner and P.-O. Persson, *Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations*, J. Comput. Phys., 335 (2017), pp. 700–717, https://doi.org/10.1016/j.jcp.2017.01.050.

[31] D. Petcu, *Experiments with an ODE solver on a multiprocessor system*, Comput. Math. Appl., 42 (2001), pp. 1189–1199, https://doi.org/10.1016/S0898-1221(01)00232-2.
[32] A. Prothero and A. Robinson, *On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations*, Math. Comp., 28 (1974), pp. 145–162, https://doi.org/10.1090/S0025-5718-1974-0331793-2.
[33] W. Qu, N. Brandon, D. Chen, J. Huang, and T. Kress, *A numerical framework for integrating deferred correction methods to solve high order collocation formulations of ODEs*, J. Sci. Comput., 68 (2016), pp. 484–520, https://doi.org/10.1007/s10915-015-0146-9.
[34] T. Rauber and G. Rünger, *Parallel Implementations of Iterated Runge-Kutta Methods*, Int. J. High Perform. Comput. Appl., 10 (1996), pp. 62–90, https://doi.org/10.1177/109434209601000103.
[35] D. Ruprecht and R. Speck, *Spectral deferred corrections with fast-wave slow-wave splitting*, SIAM J. Sci. Comput., 38 (2016), pp. A2535–A2557, https://doi.org/10.1137/16M1060078.
[36] R. Schöbel and R. Speck, *PFASST-ER: Combining the parallel full approximation scheme in space and time with parallelization across the method*, Comput. Vis. Sci., 23 (2020), 12, https://doi.org/10.1007/s00791-020-00330-5.
[37] S. I. Solodushkin and I. F. Iumanova, *Parallel numerical methods for ordinary differential equations: A survey*, in CEUR Workshop Proceedings, Vol. 1729, CEUR-WS, 2016, pp. 1–10.
[38] B. Sommeijer, *Parallel-iterated Runge-Kutta methods for stiff ordinary differential equations*, J. Comput. Appl. Math., 45 (1993), pp. 151–168, https://doi.org/10.1016/0377-0427(93)90271-C.
[39] R. Speck, *Parallelizing spectral deferred corrections across the method*, Comput. Vis. Sci., 19 (2018), pp. 75–83, https://doi.org/10.1007/s00791-018-0298-x.
[40] R. Speck, T. Lunet, T. Baumann, L. Wimmer, I. Akramov, G. R. de Souza, and J. Fritz, *Parallel-in-Time/pySDC*, 2024, https://doi.org/10.5281/zenodo.13828395.
[41] P. van der Houwen and B. Sommeijer, *Analysis of parallel diagonally implicit iteration of Runge-Kutta methods*, Appl. Numer. Math., 11 (1993), pp. 169–188, https://doi.org/10.1016/0168-9274(93)90047-U.
[42] P. van der Houwen, B. Sommeijer, and W. van der Veen, *Parallel iteration across the steps of high-order Runge-Kutta methods for nonstiff initial value problems*, J. Comput. Appl. Math., 60 (1995), pp. 309–329, https://doi.org/10.1016/0377-0427(94)00047-5.
[43] P. J. van der Houwen and B. P. Sommeijer, *Parallel iteration of high-order Runge-Kutta methods with stepsize control*, J. Comput. Appl. Math., 29 (1990), pp. 111–127, https://doi.org/10.1016/0377-0427(90)90200-J.
[44] 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, https://doi.org/10.1137/0912054.
[45] P. J. van der Houwen, B. P. Sommeijer, and W. Couzy, *Embedded diagonally implicit Runge-Kutta algorithms on parallel computers*, Math. Comp., 58 (1992), pp. 135–159, https://doi.org/10.2307/2153025.
[46] P. J. van der Houwen, B. P. Sommeijer, and W. Van der Veen, *Parallelism across the steps in iterated Runge-Kutta methods for stiff initial value problems*, Numer. Algorithms, 8 (1994), pp. 293–312, https://doi.org/10.1007/BF02142695.
[47] W. van der Veen, J. de Swart, and P. van der Houwen, *Convergence aspects of step-parallel iteration of Runge-Kutta methods*, Appl. Numer. Math., 18 (1995), pp. 397–411, https://doi.org/10.1016/0168-9274(95)00063-Z.
[48] M. Weiser, *Faster SDC convergence on non-equidistant grids by DIRK sweeps*, BIT Numer. Math., 55 (2015), pp. 1219–1241, https://doi.org/10.1007/s10543-014-0540-y.