SEMI-IMPLICIT KRYLOV DEFERRED CORRECTION METHODS FOR DIFFERENTIAL ALGEBRAIC EQUATIONS

SUNYOUNG BU, JINGFANG HUANG, AND MICHAEL L. MINION

Abstract. In the recently developed Krylov deferred correction (KDC) methods for differential algebraic equation initial value problems [33], a Picard-type collocation formulation is preconditioned using low-order time integration schemes based on spectral deferred correction (SDC), and the resulting system is solved efficiently using Newton-Krylov methods. KDC methods have the advantage that methods with arbitrarily high order of accuracy can be easily constructed which have similar computational complexity as lower order methods. In this paper, we investigate semi-implicit KDC (SI-KDC) methods in which the stiff component of the preconditioner is treated implicitly and the non-stiff parts explicitly. For certain types of problems, such a semi-implicit treatment can significantly reduce the computational cost of the preconditioner compared to fully implicit KDC (FI-KDC) methods. Preliminary analysis and numerical experiments show that the convergence of Newton-Krylov iterations in the SI-KDC methods is similar to that in FI-KDC, and hence the SI-KDC methods offer a reduction in overall computational cost for such problems.

1. Introduction

Many numerical techniques have been developed for the accurate and efficient solution of ordinary and partial differential equation initial value problems with algebraic constraints, including linear multi-step methods, Runge-Kutta methods, and operator splitting techniques [6, 10, 16, 20, 30, 46, 54, 59, 61, 62]. Their applications include (among others) numerical simulations in fluid and solid mechanics, circuits design, electrical power systems, and diffusion-reaction processes in biological and chemical systems. In this paper, we focus on constructing efficient higher-order methods for a special class of differential algebraic equations (DAEs) based on a semi-implicit Krylov deferred correction (KDC) technique.

Classical deferred and defect correction methods for ODEs were first proposed by Pereyra and Zadunaisky [51, 61, 62], in which higher-order accurate solutions of initial value ODEs are iteratively built by approximating an equation for the error (or defect) to increase the accuracy of a provisional solution. In 2000, Dutt et al. [23] presented a new variation on the deferred/defect correction strategy for ODEs by introducing Gaussian quadrature nodes and using a Picard integral equation form of the correction equation. The resulting *spectral deferred correction* (SDC) schemes can, in principle, achieve arbitrary order of accuracy for both stiff and non-stiff problems, and (unlike linear-multistep methods) the linear stability properties of higher-order versions of the methods are similar to those of lower-order versions.

2000 *Mathematics Subject Classification.* 65B10, 65F10, 65L20, 65L80, 65N35.

The work of Huang and Bu was supported by NSF grants 0811130 and 0941235. Part of the work was done when Bu was a visiting member of the Institute for Mathematics and Its Applications at the University of Minnesota. The work of Minion was supported by NSF grant 0854961.

Furthermore, SDC methods based on a semi-implicit or IMEX formulation have been constructed by applying an operator splitting approach to the correction equations [14, 45, 46, 47]. Semi-implicit methods treat non-stiff terms in the equations explicitly and stiff terms implicitly, which can significantly reduce the overall cost for stiff problems compared to fully implicit methods. Higher-order semi-implicit SDC schemes do not suffer from the stability limits of linear multistep schemes based on BDF [3, 7, 25], nor the difficulty of satisfying the coupling constraints in IMEX Runge-Kutta methods [8, 12, 13, 19, 37, 49, 57]. In fact, higher-order SDC methods with multiple implicit or explicit terms and varying time steps (multirate methods) have also been developed [14, 15, 41].

Notice that when the SDC iterations converge for an ODE, the method is equivalent to a fully implicit Runge-Kutta (i.e. collocation) method and inherits the excellent stability and accuracy properties of such methods. The benefit of using the SDC approach is that it does not require the solution of coupled implicit equations between the nodes in the collocation formulation as a direct application of a nonlinear solver would. Also, semi-implicit versions of SDC can be used to further reduce the computational cost of the SDC iterates compared to fully implicit methods for problems which can be readily split into stiff and non-stiff terms. Unfortunately, when SDC methods are applied to very stiff ODEs, the convergence of the SDC iterates can be quite slow for a range of time-steps for which the method is stable. In [32], it is shown analytically that the SDC iteration is equivalent to a preconditioned Neumann series expansion for linear ODEs, where the preconditioner is the low-order deferred correction procedure, and the slow convergence of the SDC iterates is due to the existence of a few "bad" eigenvalues in stiff ODE and DAE systems, as will be further discussed in Sec. 2.2. This type of order reduction has been previously observed and analyzed for both SDC and some Runge-Kutta methods (see e.g. [12, 13, 37, 42]). A procedure based on Krylov subspace methods is introduced in [32, 33] that accelerates the convergence of SDC methods for stiff problems and hence eliminates order reduction. Adapting the terminology from the nonlinear solver community, one correction sweep of the SDC iteration in the Krylov accelerated SDC methods becomes a preconditioner for the solution of the full collocation formulation, as detailed in Sec. 2.

In [33], SDC methods are applied to DAE systems of the form

$$F\left(y(t), y'(t), t\right) = 0 \tag{1.1}$$

with the goal of constructing stable high-order solvers for DAEs. The SDC procedure in [33] relies on a Picard type formulation of Eq. (1.1) written in terms of the variable $Y(t) = y'(t)$

$$F\left(y_0 + \int_0^t Y(\tau)d\tau, Y(t), t\right) = 0. \tag{1.2}$$

We refer to Eq. (1.2) as the "yp-formulation" and discuss alternatives to this formulation below. Unfortunately, it is demonstrated in [33] that in the most straightforward application of SDC to the "yp-formulation", the SDC iterations may not converge, even for arbitrarily small time step. However, using the Krylov subspace approach introduced in [32] for ODEs, efficient methods using SDC and the yp-formulation for DAEs are constructed and evaluated in [33]. These so-called *Krylov deferred correction* (KDC) methods converge to the solution of an implicit Gauss Runge-Kutta (GRK) method while requiring only the solution of implicit equations similar to those one would find in a first-order method. Furthermore, it is shown in [33] that the implicit equations that arise in [33] can be better conditioned than those arise from a straightforward application of Newton's method to the GRK formulation. The KDC methods are shown in [33] to compare favorably with other popular approaches on standard test problems, especially when high precision is required.

In the numerical implementations in [32, 33], explicit low-order time stepping schemes are used in the KDC technique for problems with either non-stiff or mildly stiff differential parts, while implicit schemes are applied to those with stiff parts. However, as in the case for ODEs, for problems which can be appropriately split, it is typically advantageous to apply a semi-implicit time integration scheme. The focus of this paper is to study semi-implicit time integration schemes within KDC methods. One case in which DAEs can be readily split is for equations of the form

$$\begin{cases} y'(t) = f(y(t), z(t), t), \\ 0 = g(y(t), z(t), t), \end{cases} \tag{1.3}$$

where the right hand side of the differential equations for $y(t)$ can be split into stiff and non-stiff components. Another possibility explored here is the use of a linearly implicit approach on the differential equation for $y'(t)$. We study the convergence properties of the resulting semi-implicit KDC (SI-KDC) methods in terms of the number of iterates needed to converge to the GRK solution and show that, compared with the fully implicit KDC (FI-KDC) methods, the convergence of the SI-KDC is similar. Since the resulting algebraic equation system in the SI-KDC discretization is simpler than that in FI-KDC, the SI-KDC formulation can hence be more efficient overall. Unfortunately, we also demonstrate that finding an efficient semi-implicit splitting for the SI-KDC technique becomes more complicated for higher-index DAE systems with both algebraic and differential components.

This paper is organized as follows. In Sec. 2, we briefly describe KDC methods, by introducing the Picard integral collocation formulation, spectral deferred correction (SDC) technique, and Newton-Krylov methods. In Sec. 3, semi-implicit KDC methods are discussed, and different semi-implicit preconditioning techniques and their convergence properties are analyzed for DAE systems of different index. In Sec. 4, preliminary numerical results are presented to compare different SI-KDC methods with fully implicit schemes. Finally in Sec. 5, we summarize our results and discuss further applications of the SI-KDC methods.

2. Krylov Deferred Correction Methods

In this section, we discuss the Krylov deferred correction (KDC) technique for DAEs of the form (1.1). The subsections give a review of the discretization scheme used in KDC methods, the formulation of the error equation, and how Krylov subspace methods are used in the iteration.

2.1. **Picard Integral Equation and Spectral Integration.**

In the KDC methods, unlike traditional numerical methods based on the differential form of the equations, we first set $Y(t) = y'(t)$ as the new unknown, and consider the Picard type integral equation (1.2).

To discretize the integral equation (1.2) in one time step $[0, \Delta t]$, we linearly map the Gaussian nodes originally defined on $[-1, 1]$ to $[0, \Delta t]$, and denote the p nodes by $\mathbf{t} = [\mathbf{t_1}, \mathbf{t_2}, \cdots, \mathbf{t_p}]^{\mathbf{T}}$. Similarly, we denote the solution $y(t)$ and the derivative values $Y(t)$ at these nodes by $\mathbf{y} = [y_1, y_2, \cdots, y_p]^T$ and $\mathbf{Y} = [Y_1, Y_2, \cdots, Y_p]^T$ respectively. Given the discretized $\mathbf{Y}$, a degree $p-1$ interpolating polynomial $P(t)$ can be constructed to approximate each component of the solution $Y(t)$ using standard techniques. We can then approximate $\int_0^{t_m} Y(\tau)d\tau$ using $\int_0^{t_m} P(\tau)d\tau$ and evaluate this degree p polynomial to obtain at $\mathbf{t}$ the approximate function values of $\mathbf{y}$. We refer to this procedure as spectral integration, and represent the linear mapping from $\mathbf{Y}$ to $\mathbf{y}$ by a matrix $\Delta t S$ where the spectral integration matrix S

is independent of the step-size Δt and can be precomputed. Using the spectral integration matrix, we derive the collocation formulation

$$\mathbf{F}(\mathbf{y}_0 + \Delta t S \otimes \mathbf{Y}, \mathbf{Y}, \mathbf{t}) = \mathbf{0}, \tag{2.1}$$

which will be symbolically denoted as $\mathbf{H}(\mathbf{Y}) = \mathbf{0}$. In the formula, $\mathbf{y}_0 = [y_0, y_0, \cdots, y_0]^T$ is the vector of initial values, and $\otimes$ is the tensor product (i.e. $\Delta t S$ is applied to each component of $\mathbf{Y}$).

Instead of the "yp-formulation", the original SDC method for ODEs in [23] is based on the traditional Picard integral equation or "y-formulation". Methods based on the Picard formulation have also been developed for two point boundary value problems in [27]. The "y-formulation" for ODEs can be generalized for DAE systems of the form (1.3) by

$$\begin{cases} y(t) = y_0 + \int_0^t f(y(\tau), z(\tau), \tau) d\tau, \\ 0 = g(y(t), z(t), t). \end{cases} \tag{2.2}$$

However, for an arbitrary DAE system of the form Eq. (1.1), the discretization of the "y-formulation" in the current setting would require a differentiation matrix rather than an integration matrix. Since spectral integration is numerically better conditioned than spectral differentiation [27, 58], we focus here on the "yp-formulation".

It is also possible to formulate the integration matrix S using Radau or Lobatto type quadrature nodes instead of Gaussian nodes and calculate the Legendre polynomial coefficients accordingly. The Radau Ia quadrature nodes use the left end point (i.e. $t_1 = 0$), the Radau IIa nodes use the right end point (i.e. $t_p = \Delta t$), and the Lobatto quadrature nodes include both end points. Also, Chebyshev polynomials and the corresponding quadrature nodes may be used instead of Legendre polynomial based nodes, which allow the fast Fourier transform (FFT) to be used for acceleration (FFT is more efficient than existing fast Legendre transforms). Detailed analytical and numerical comparisons of different polynomials and nodes will be reported later. For a discussion of the choice of nodes for the spectral deferred correction methods for ODEs, the readers are referred to [42].

For DAEs, it is pointed out in [30] that the Gaussian collocation formulation encounters "order reduction". When p Gaussian nodes are applied to an index one DAE system, numerical order for the algebraic component is only p, while for Radau IIa nodes, the order is $2p - 1$. We therefore focus on the Radau IIa nodes in our numerical implementations for higher-index DAEs in this paper. Interested readers are referred to [30] (Table 2.3, p18) for further details on the convergence of collocation formulations for different index DAE problems.

2.2. **Error Equation and Spectral Deferred Corrections.**

Notice that for scalar equations, Eq. (2.1) is typically nonlinear with p unknowns as compared to the 1-unknown equation encountered when using the backward Euler (or BDF) method. For N dimensional vector DAEs, the number of unknowns becomes pN as compared to N. Therefore direct application of Newton's method utilizing Gauss elimination for the required linear solves would require $O((pN)^3)$ operations for the collocation formulation with p points, while $O(N^3)$ operations for BDF methods. For this reason, although superior in accuracy and optimal in step-size, very high-order collocation methods for Eq. (2.1) are rarely used in practice. In the following, we discuss how the SDC procedure can be adapted to DAEs to produce efficient methods for solving the high-order collocation formulations.

Assume a provisional solution $\tilde{\mathbf{Y}} = [\tilde{Y}_1, \tilde{Y}_2, \cdots, \tilde{Y}_p]^T$ is obtained at the Gaussian type nodes $\mathbf{t}$ in one time step $[0, \Delta t]$, using a low-order method or other approximation scheme and denote the corresponding interpolating polynomial approximation to the solution as $\tilde{Y}(t)$, one can define an equation for the error $\delta(t) = Y(t) - \tilde{Y}(t)$ by

$$F\left(y_0 + \int_0^t \left(\tilde{Y}(\tau) + \delta(\tau)\right) d\tau, \tilde{Y}(t) + \delta(t), t\right) = 0, \tag{2.3}$$

and the corresponding discretized system becomes

$$\mathbf{F}(\mathbf{y}_0 + \Delta t S \otimes (\tilde{\mathbf{Y}} + \boldsymbol{\delta}), \tilde{\mathbf{Y}} + \boldsymbol{\delta}, \mathbf{t}) = \mathbf{0}. \tag{2.4}$$

We symbolically denote the relation between $\tilde{\mathbf{Y}}$ and $\boldsymbol{\delta}$ as $\mathbf{G}(\tilde{\mathbf{Y}}, \boldsymbol{\delta}) = \mathbf{0}$, i.e., given $\tilde{\mathbf{Y}}$, solving Eq. (2.4) requires finding a specific value $\boldsymbol{\delta}$. Note that finding $\mathbf{Y}$ (or the corresponding $\tilde{\mathbf{Y}} + \boldsymbol{\delta}$) such that $\mathbf{G}(\mathbf{Y}, \mathbf{0}) = \mathbf{0}$ is equivalent to solving $\mathbf{H}(\mathbf{Y}) = \mathbf{0}$ (where $\mathbf{H}$ is defined after Eq. (2.1), which is numerically challenging as the unknowns at different times are coupled. Instead, as Eq. (2.3) gives the identity

$$F\left(y_0 + \int_0^{t_{m+1}} \tilde{Y}(\tau) d\tau + \left(\int_0^{t_m} + \int_{t_m}^{t_{m+1}}\right) \delta(\tau) d\tau, \tilde{Y}(t_{m+1}) + \delta(t_{m+1}), t_{m+1}\right) = 0,$$

a simple time-marching discretization of this equation similar to the explicit (forward) Euler method for ODEs gives a low-order solution $\tilde{\boldsymbol{\delta}} = [\tilde{\boldsymbol{\delta}}_1, \tilde{\boldsymbol{\delta}}_2, \cdots, \tilde{\boldsymbol{\delta}}_p]^T$ by solving

$$F\left(y_0 + [\Delta t S \otimes \tilde{\mathbf{Y}}]_{m+1} + \sum_{l=1}^{m+1} \Delta t_l \tilde{\delta}_{l-1}, \tilde{Y}_{m+1} + \tilde{\delta}_{m+1}, t_{m+1}\right) = 0, \tag{2.5}$$

where $\Delta t_{l+1} = t_{l+1} - t_l$ and t_0 and δ_0 are set to 0. Note that this update formula is in general implicit even though an "explicit" time-marching scheme is used. Similarly, a time-marching scheme based on backward Euler method is given by

$$F\left(y_0 + [\Delta t S \otimes \tilde{\mathbf{Y}}]_{m+1} + \sum_{l=1}^{m+1} \Delta t_l \tilde{\delta}_l, \tilde{Y}_{m+1} + \tilde{\delta}_{m+1}, t_{m+1}\right) = 0. \tag{2.6}$$

These two methods differ only in the way the time integral of $\delta(t)$ is approximated. Eq. (2.5) is equivalent to the rectangle rule using the left endpoint while Eq. (2.6) is the rectangle rule using the right endpoint.

In the SDC methods, the low-order solution $\tilde{\boldsymbol{\delta}}$ is added to the provisional solution $\tilde{\mathbf{Y}}$ in order to form a better approximation, and this iteration continues for a prescribed number of times or until a prescribed error tolerance is achieved. It has been shown that for ODE problems [34], for sufficiently small time step the formal order of accuracy of $\tilde{\mathbf{Y}}$ will increase by one after each SDC iteration using a first-order method. Unfortunately, for general DAE problems of higher-index, it is demonstrated numerically that the analogous SDC iteration procedure is divergent for many DAE systems [33]. It is shown in [32] that for linear systems of ODEs, the spectral deferred correction technique is equivalent to a preconditioned Neumann series expansion, where the preconditioner is the low-order deferred correction procedure. Furthermore, the preconditioned system in the ODE case can be written in the form

$$(I - C)x = b,$$

where C scales like Δt, and hence one can prove that for small Δt, all the eigenvalues of C are located inside the unit disc on the complex plane and the Neumann series

$$x = b + Cb + C^2 b + \cdots$$

is convergent. However for DAE problems, the analogous procedure produces a matrix C that may have eigenvalues with magnitude greater than 1 independent of the step-size Δt, and hence the standard SDC procedure becomes divergent (see

[33] for details). However, this lack of convergence can be removed by applying appropriate Newton-Krylov methods to the iterations.

2.3. **Newton-Krylov Method and Preconditioners.** Newton-Krylov methods are designed for solving nonlinear algebraic equations of the form $M(x) = 0$ with N equations and unknowns. Assume an initial approximate solution x_0 is known, Newton's method is used to iteratively compute a sequence of quadratically convergent approximations (assuming the Jacobian matrix J_M is nonsingular at the solution)

$$x_{n+1} = x_n - \delta x,$$

where δx is the solution of the linear equation

$$J_M(x_n)\delta x = M(x_n)$$

solved using Krylov subspace methods such as the GMRES, BiCGStab, and TFQMR methods [11, 55, 38] (as J_M is in general non-symmetric). The iterations in Newton's method and the Krylov subspace methods can then be intertwined by reducing the residual of the linear equation by a prescribed factor, and then restarting the Newton iterations. The resulting methods are usually called the Newton-Krylov methods.

Notice that when

$$J_M(x_n) = \pm I - C,$$

where most eigenvalues of C are clustered close to 0, because of the rapid decay of most eigenmodes in $C^q b$, the numerical rank of the Krylov subspace

$$K_q(J_M, b) = \{b, Cb, C^2 b, \cdots, C^q b\}$$

is low and the Newton-Krylov iterations converge rapidly. This is true even for cases when there are a few eigenvalues located outside the unit circle (which causes the divergence of the SDC methods for DAEs) or inside but close to the unit circle (the order reduction of the SDC methods for stiff ODE problems).

2.4. **Krylov Deferred Correction Methods.** In general, an efficient numerical implementation of a Newton-Krylov method depends on: (a) a formulation of the problem $M(x) = 0$ such that J_M is close to the identity matrix $\pm I$, and (b) an efficient procedure for computing the matrix vector product Cb (or equivalently $J_M b$). We now discuss these two points in the context of KDC methods.

For (a), one common technique to improve the convergence of a Krylov subspace method is to apply a "preconditioner" to the original system. Traditionally, such preconditioners are chosen as sparse matrices close to J_M^{-1} [21]. Dense integral operators have also been used as preconditioners (see e.g. [39]), which are efficiently applied to an arbitrary vector using fast convolution algorithms such as the fast multipole method [28]. For DAE problems, the low-order time stepping methods in Eqs. (2.5-2.6) can be written in matrix form as

$$\mathbf{F}(\mathbf{y}_0 + \Delta t S \otimes \tilde{\mathbf{Y}} + \Delta t \tilde{S} \otimes \tilde{\boldsymbol{\delta}}, \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}, \mathbf{t}) = \mathbf{0}, \tag{2.7}$$

where $\Delta t\tilde{S}$ is the lower triangular representation of the rectangle rule approximation of the spectral integration operator $\Delta t S$. Specifically, for Eq. (2.5)

$$\Delta t \tilde{S}_E = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ \Delta t_1 & 0 & \cdots & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdots & 0 & 0 \\ \Delta t_1 & \Delta t_2 & \cdots & \Delta t_{p-1} & 0 \end{bmatrix} \tag{2.8}$$

and for Eq. (2.6)

$$\Delta t\tilde{S}_I = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & \Delta t_1 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdots & 0 & 0 \\ 0 & \Delta t_1 & \cdots & \Delta t_{p-2} & 0 \\ 0 & \Delta t_1 & \cdots & \Delta t_{p-2} & \Delta t_{p-1} \end{bmatrix}. \tag{2.9}$$

As before, we denote the relation between $\tilde{\mathbf{Y}}$ and $\tilde{\boldsymbol{\delta}}$ in Eq. (2.7) as an implicit function of the form $\tilde{\mathbf{G}}(\tilde{\mathbf{Y}}, \tilde{\boldsymbol{\delta}}) = \mathbf{0}$, and represent the corresponding explicit function as $\tilde{\boldsymbol{\delta}} = \tilde{\mathbf{H}}(\tilde{\mathbf{Y}})$, where the provisional solution $\tilde{\mathbf{Y}}$ is the independent *input* variable and the *output* function value is $\tilde{\boldsymbol{\delta}}$, derived by solving Eq. (2.7) using one SDC iteration. It can be seen that if $\mathbf{Y}$ satisfies the collocation formulation $\mathbf{H}(\mathbf{Y}) = \mathbf{0}$, then $\mathbf{G}(\mathbf{Y}, \mathbf{0}) = \mathbf{0}$ and $\tilde{\mathbf{G}}(\mathbf{Y}, \mathbf{0}) = \mathbf{0}$, i.e., for the *input* $\mathbf{Y}$, the *output* function value $\tilde{\mathbf{H}}(\mathbf{Y}) = \mathbf{0}$. Therefore solving the collocation formulation $\mathbf{H}(\mathbf{Y}) = \mathbf{0}$ is equivalent to finding the zero of the **new** function $\tilde{\mathbf{H}}(\cdot)$. In [33], it is shown that because the lower-order method solves a "nearby" problem, the Jacobian of $\tilde{\mathbf{H}}$ is closer to identity than that of $\mathbf{H}$, and hence $\tilde{\mathbf{H}}(\cdot) = \mathbf{0}$ can be thought of as a preconditioned version of $\mathbf{H}(\cdot) = \mathbf{0}$. Specifically, applying the implicit function theorem, the Jacobian matrix $J_{\tilde{H}}$ of $\tilde{\mathbf{H}}$ is given by

$$\begin{aligned} J_{\tilde{\mathbf{H}}} = \frac{\partial \tilde{\boldsymbol{\delta}}}{\partial \mathbf{Y}} &= -\left(\frac{\partial \mathbf{F}}{\partial \mathbf{Y}} + \frac{\partial \mathbf{F}}{\partial \mathbf{y}}\Delta t\tilde{S}\right)^{-1}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{Y}} + \frac{\partial \mathbf{F}}{\partial \mathbf{y}}\Delta t S\right) \\ &= -I + \left(\frac{\partial \mathbf{F}}{\partial \mathbf{Y}} + \frac{\partial \mathbf{F}}{\partial \mathbf{y}}\Delta t\tilde{S}\right)^{-1}\left(\frac{\partial \mathbf{F}}{\partial \mathbf{y}}\Delta t(\tilde{S} - S)\right). \end{aligned}$$

When $\frac{\partial \mathbf{F}}{\partial \mathbf{Y}}$ is non-singular, since $\tilde{S}$ is an approximation of S, when Δt is small, $J_{\tilde{\mathbf{H}}}$ is close to $-I$, and Newton-Krylov methods can be applied directly to find the zero of the preconditioned system $\tilde{\mathbf{H}}(\cdot) = \mathbf{0}$. For comparison, the Jacobian matrix of $\mathbf{H} = 0$ is given by

$$J_{\mathbf{H}} = \frac{\partial \mathbf{H}}{\partial \mathbf{Y}} = \left(\frac{\partial \mathbf{F}}{\partial \mathbf{Y}} + \frac{\partial \mathbf{F}}{\partial \mathbf{y}}\Delta t S\right). \tag{2.10}$$

In regards to point (b), when a forward difference approximation technique is adapted as in most Jacobian-free Newton-Krylov (JFNK) solvers, for any vector v, we can approximate $J_{\tilde{\mathbf{H}}}(x)v$ by

$$D_h\tilde{\mathbf{H}}(x : v) = \left(\tilde{\mathbf{H}}(x + hv) - \tilde{\mathbf{H}}(x)\right)/h$$

for some properly chosen parameter h (h may be complex). Note that computing the function $\tilde{\mathbf{H}}$ in this formulation is simply a deferred correction iteration described succinctly in Eq. (2.7). This difference approximation technique as well as the choice of h have been carefully studied previously and the readers are referred to [36] for details.

The results in [32] show that the KDC method for DAEs converges more efficiently (to the Gauss Runge-Kutta solution) using a low-order preconditioning iteration compared with a direct solution of the coupled collocation formulation. In the numerical implementation, the KDC method consists of two components: a Newton-Krylov method that can be applied directly to solve the preconditioned collocation formulation $\tilde{\mathbf{H}}(\cdot) = \mathbf{0}$; and the "function evaluation" required for the Newton-Krylov method, which is simply one deferred correction iteration for the given provisional solution. A pseudo code is listed in Figure 1 for one time step

```
Krylov Deferred Correction Method
[Comment:] We consider one step from 0 to $\Delta t$. We assume the initial
condition $y_0$ is given at $t = 0$, and the Gaussian nodes on $[0, \Delta t]$ and the
corresponding integration matrix $S$ are precomputed.

Step 1: Initial Guess
    Apply a low-order time stepping method to compute an approximate
        solution $\mathbf{Y}^{[0]}$ of the original differential equation at the Gaussian
        type nodes.

Step 2: Find the zero of $\tilde{\mathbf{H}}(\cdot)$
    Use existing JFNK solver to solve the preconditioned system $\tilde{\mathbf{H}}(\cdot) = 0$
        with initial guess $\mathbf{Y}^{[0]}$ as the input variable.

    [Comment:] Most JFNK solvers only require a "function evaluation
        subroutine" (provided below) to compute $\tilde{\mathbf{H}}(\mathbf{v})$ for any provided
        input variable $\mathbf{v}$.

Step 3: Output $y(\Delta t)$
    Compute and output $y(\Delta t) = y_0 + \int_0^{\Delta t} Y(\tau) d\tau$ by using Gaussian
        quadrature on the computed values $\mathbf{Y}$ at the Gaussian nodes from
        Step 2.

Function evaluation subroutine
    Input: A vector $\tilde{\mathbf{Y}}$ provided by JFNK solver.
        Using $\tilde{\mathbf{Y}}$ as the provisional solution, solve Eq. (2.7), which is
            equivalent to one SDC iteration with a low-order scheme.
    Output: the low-order error approximation $\tilde{\boldsymbol{\delta}}$.
End Function evaluation subroutine
```

Figure 1. Outline of the KDC method

from 0 to Δt, utilizing (a) existing JFNK solvers and (b) low-order time stepping schemes for the original and error equations. Both (a) and (b) have been well studied and documented in the literature.

Note that KDC methods are composed of two Newton procedures: (a) a Jacobian-free Newton-Krylov method is applied to find the zero of the preconditioned nonlinear system $\tilde{\mathbf{H}}(\cdot)$ in *Step 2*; and (b) for each "function evaluation" $\tilde{\mathbf{H}}(\mathbf{v})$ (which is an SDC sweep), a Newton-type method is applied to solve the (generally) nonlinear system in each substep of the lower-order time stepping method. We refer to the Newton iterations in (a) as the outer iterations and those in (b) as the inner iterations. For (a), existing JFNK solvers can be easily adapted, and for (b), we modify the Newton solvers from existing efficient low-order time stepping schemes for the original differential equations and solve the error equation. Clearly, each "function evaluation" in general requires the efficient solution of p decoupled nonlinear systems (one at each substep). The purpose of introducing the semi-implicit KDC methods is to optimize the low-order time-marching schemes to improve the efficiency of the inner Newton iterations for problems where an appropriate splitting can be found.

3. Semi-implicit Preconditioning Techniques for Stiff DAEs

In this section we discuss the semi-implicit approach in the low-order time-stepping procedure which forms the preconditioner in the KDC methods. The underlying idea is simply that one may treat the non-stiff terms explicitly and stiff terms implicitly. For ODEs, it is often straightforward to determine an appropriate semi-implicit splitting of the equations by examining the eigenvalues of a linearized problem. However the situation is less clear for DAEs, and one of the pertinent points in this section is that even for very simple examples, there can be several different logical ways to semi-implicitly split a DAE and it is not always clear *apriori* which splitting will provide the most efficient method.

In the first part of this section, we briefly introduce the modification to the KDC method needed to employ a semi-implicit approach. We then consider different options for producing a splitting for simple linear DAE systems of index 1 and 2.

3.1. **Semi-implicit KDC Technique.**

We first assume that the DAE system of interest can be split into two parts

$$F(y(t),y'(t),t)=F_E(y(t),y'(t),t)+F_I(y(t),y'(t),t)=0 \tag{3.1}$$

where F_E represents the non-stiff component and F_I the stiff component. We leave the non-trivial task of finding an appropriate splitting to later sections.

As before, we introduce $Y(t)=y'(t)$ as the new unknown to get

$$F_E\left(y_0+\int Y(\tau)d\tau, Y(t),t\right)+F_I\left(y_0+\int Y(\tau)d\tau, Y(t),t\right)=0. \tag{3.2}$$

This Picard type integral equation can be directly discretized using the spectral integration matrix S to yield the collocation formulation

$$\mathbf{F}_E(\mathbf{y}_0+\triangle tS\otimes\mathbf{Y},\mathbf{Y},\mathbf{t})+\mathbf{F_I}(\mathbf{y_0}+\triangle \mathbf{tS}\otimes\mathbf{Y},\mathbf{Y},\mathbf{t})=\mathbf{0} \tag{3.3}$$

where $\mathbf{Y}=[\mathbf{Y}_1,\mathbf{Y}_2,...,\mathbf{Y}_p]^T$ is the desired solution which approximates $Y(t)=y'(t)$ at the quadrature nodes $\mathbf{t}=[\mathbf{t_1},\mathbf{t_2},\cdots,\mathbf{t_p}]^{\mathbf{T}}$ in one time step $[0,\Delta t]$. We further define the error as $\delta(t)=Y(t)-\tilde{Y}(t)$ where $\tilde{Y}$ is a provisional solution to the DAE system. Eq. (3.3) can then be rewritten in terms of $\delta(t)$ as

$$F_E\left(y_0+\int(\tilde{Y}(\tau)+\delta(\tau))d\tau,\tilde{Y}+\delta,t\right)+F_I\left(y_0+\int(\tilde{Y}(\tau)+\delta(\tau))d\tau,\tilde{Y}+\delta,t\right)=0. \tag{3.4}$$

To improve the provisional solution $\tilde{Y}(t)$, low-order methods can be applied to derive an approximation of the error denoted by $\tilde{\delta}$. When the explicit Euler method ($\tilde{S}_E$ in Eq. (2.8)) is applied to the non-stiff part and the backward Euler method ($\tilde{S}_I$ in Eq. (2.9)) to the stiff one, the low-order method can be rewritten in the matrix form

$$\mathbf{F}_E(\mathbf{y}_0+\triangle tS\otimes\tilde{\mathbf{Y}}+\triangle t\tilde{S}_E\otimes\tilde{\boldsymbol{\delta}},\tilde{\mathbf{Y}}+\tilde{\boldsymbol{\delta}},\mathbf{t})+\mathbf{F}_I(\mathbf{y}_0+\triangle tS\otimes\tilde{\mathbf{Y}}+\triangle t\tilde{S}_I\otimes\tilde{\boldsymbol{\delta}},\tilde{\mathbf{Y}}+\tilde{\boldsymbol{\delta}},\mathbf{t})=\mathbf{0}.$$

This equation gives the preconditioned function $\tilde{\boldsymbol{\delta}}=\tilde{\mathbf{H}}_{SI}(\tilde{\mathbf{Y}})$, and the application of the Newton-Krylov methods is then straightforward. This technique is referred to as the semi-implicit KDC (SI-KDC) technique. Following the discussions in Sec. 2.3, the Jacobian matrix of $\tilde{\mathbf{H}}_{SI}$ is obtained by

$$J_{\tilde{\mathbf{H}}_{SI}}=-\left(\frac{\partial\mathbf{F}}{\partial\mathbf{Y}}+\frac{\partial\mathbf{F}_E}{\partial\mathbf{y}}\Delta t\tilde{S}_E+\frac{\partial\mathbf{F}_I}{\partial\mathbf{y}}\Delta t\tilde{S}_I\right)^{-1}\left(\frac{\partial\mathbf{F}}{\partial\mathbf{Y}}+\frac{\partial\mathbf{F}}{\partial\mathbf{y}}\Delta tS\right) \tag{3.5}$$

which is closer to $-I$ compared with the Jacobian of the original collocation formulation in Eq. (2.10), since $\tilde{S}_E$ and $\tilde{S}_I$ are approximations of S, and Δt is small.

As the semi-implicit KDC discretization scheme converges to the solution of the collocation formulation in Eq. (3.3), its accuracy is not significantly different from results derived using fully implicit preconditioning techniques (or a direct application of Newton's method to the collocation formulation). It will, however, change the condition number of the original system and different preconditioning techniques (choices of F_E and F_I) usually result in very different convergence properties in the (outer) Newton-Krylov iterations. Also, the preconditioning strategies can significantly change the efficiency of the inner Newton iterations (or even make such iterations unnecessary) for special stiff DAE systems. In the following, using simple linear index 1 and index 2 DAE systems of the form

$$\left\{\begin{aligned} x_t &= a_{11}x + a_{12}y + a_{13}z \\ y_t &= a_{21}x + a_{22}y + a_{23}z \\ 0 &= a_{31}x + a_{32}y + a_{33}z, \end{aligned}\right. \tag{3.6}$$

we demonstrate how the choice of splitting can effect the efficiency of the SI-KDC method. In particular, we focus on the impact on the convergence of the outer Newton-Krylov iterations and efficiency of the inner process in one SDC iteration.

3.2. **Index One DAE System.** We first focus on a simple index one version of Eq. (3.6) by setting $a_{33} \neq 0$ and $\frac{1}{a_{33}} \approx O(1)$, and assume all other constants a_{ij} are $O(1)$ except for a_{22} which is a large negative number. The term with coefficient a_{22} hence represents the stiff component and all other terms are non-stiff. As discussed in Sec. 2, we apply the "yp-formulation" to the differential variables x and y instead of the traditional "y-formulation". For the algebraic variable z, there are several ways that this can be done, and we examine three possibilities here.

We first focus on a scheme based on applying the "yp-formulation" to z, and the corresponding version of the error equation (2.7) becomes

$$\begin{bmatrix} \tilde{\mathbf{X}} + \tilde{\boldsymbol{\delta}}_1 \\ \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}_2 \\ \mathbf{0} \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \Delta t S \tilde{\mathbf{X}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}_1 \\ \mathbf{y_0} + \Delta t S \tilde{\mathbf{Y}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}_2 \\ \mathbf{z_0} + \Delta t S \tilde{\mathbf{Z}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}_3 \end{bmatrix} \tag{3.7}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \tag{3.8}$$

and $\tilde{S}$ is either $\tilde{S}_I$ or $\tilde{S}_E$, representing different preconditioning schemes for different terms. The equation for $\tilde{\boldsymbol{\delta}}$ can then be explicitly written as

(3.9)
$$\begin{bmatrix} I - \Delta t \tilde{S} a_{11} & -\Delta t \tilde{S} a_{12} & -\Delta t \tilde{S} a_{13} \\ -\Delta t \tilde{S} a_{21} & I - \Delta t \tilde{S} a_{22} & -\Delta t \tilde{S} a_{23} \\ -\Delta t \tilde{S} a_{31} & -\Delta t \tilde{S} a_{32} & -\Delta t \tilde{S} a_{33} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\delta}}_1 \\ \tilde{\boldsymbol{\delta}}_2 \\ \tilde{\boldsymbol{\delta}}_3 \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \Delta t S \tilde{\mathbf{X}} \\ \mathbf{y_0} + \Delta t S \tilde{\mathbf{Y}} \\ \mathbf{z_0} + \Delta t S \tilde{\mathbf{Z}} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{X}} \\ \tilde{\mathbf{Y}} \\ 0 \end{bmatrix}$$

where I is an identity matrix. Clearly, $\tilde{S}_I$ should be applied to the stiff term with coefficient a_{22}. We further assume that we want the provisional solution to remain on the manifold defined by the algebraic equation constraint [1], by applying $\tilde{S}_I$ to $\{a_{31}, a_{32}, a_{33}\}$ terms (as the backward Euler method is well-defined and exact for a single algebraic equation). The explicit low-order scheme $\tilde{S}_E$ can then be applied to

[1]Once convergence is reached, the algebraic constraint will be satisfied by construction, hence here we are only considering enforcing the constraint for each KDC iterate.

all remaining terms. When $\tilde{S}_I$ and $\tilde{S}_E$ are respectively the backward and forward Euler methods, the corresponding semi-implicit time-marching discretization for Eq. (3.9) becomes

$$\begin{bmatrix} \tilde{\boldsymbol{\delta}}_1^{j+1} - a_{11}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_1^{l-1} - a_{12}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_2^{l-1} - a_{13}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_3^{l-1} \\ \tilde{\boldsymbol{\delta}}_2^{j+1} - a_{21}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_1^{l-1} - a_{22}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_2^{l} - a_{23}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_3^{l-1} \\ -a_{31}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_1^{l} - a_{32}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_2^{l} - a_{33}\sum_{l=1}^{j+1}\Delta t_l \tilde{\boldsymbol{\delta}}_3^{l} \end{bmatrix}$$
$$= A \begin{bmatrix} \mathbf{x_0} + [\Delta t S \otimes \tilde{\mathbf{X}}]_{j+1} \\ \mathbf{y_0} + [\Delta t S \otimes \tilde{\mathbf{Y}}]_{j+1} \\ \mathbf{z_0} + [\Delta t S \otimes \tilde{\mathbf{Z}}]_{j+1} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{X}}_{j+1} \\ \tilde{\mathbf{Y}}_{j+1} \\ 0 \end{bmatrix}$$

where $\Delta t_{l+1} = t_{l+1} - t_l$. Note that to march from t_j to t_{j+1} in this specific semi-implicit low-order time stepping procedure, the linear system for the unknowns becomes

$$\begin{bmatrix} \tilde{\boldsymbol{\delta}}_1^{j+1} \\ \tilde{\boldsymbol{\delta}}_2^{j+1} - a_{22}\Delta t_{j+1}\tilde{\boldsymbol{\delta}}_2^{j+1} \\ -a_{31}\Delta t_{j+1}\tilde{\boldsymbol{\delta}}_1^{j+1} - a_{32}\Delta t_{j+1}\tilde{\boldsymbol{\delta}}_2^{j+1} - a_{33}\Delta t_{j+1}\tilde{\boldsymbol{\delta}}_3^{j+1} \end{bmatrix} = RHS$$

where all the known quantities are collected in RHS. As the unknowns are “decoupled”, one can first solve the two “single variable” equations for $\tilde{\boldsymbol{\delta}}_1^{j+1}$ and $\tilde{\boldsymbol{\delta}}_2^{j+1}$, and then solve the last equation for $\tilde{\boldsymbol{\delta}}_3^{j+1}$. Therefore the evaluation of the new function $\tilde{\mathbf{H}}_{SI}$ is less expensive than evaluating $\tilde{\mathbf{H}}_{FI}$ in the FI-KDC scheme where $\tilde{S}_I$ is applied to all terms and a 3×3 linear system must be solved.

As for the outer Newton-Krylov iterations, we compare the Jacobian matrix of the resulting semi-implicit KDC scheme

$$\left(E - \Delta t \tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix} - \Delta t \tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \right)^{-1} (\Delta t S \otimes A - E)$$

with that from the fully-implicit KDC approach

$$\left(E - \Delta t \tilde{S}_I \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \right)^{-1} (\Delta t S \otimes A - E), \quad \text{where} \quad E = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

It can be shown analytically that the eigenvalues of the SI-KDC Jacobian matrix are similarly distributed to those from FI-KDC for sufficiently small Δt, as $\Delta t \tilde{S}_I a_{22}$ is the dominant part in both matrices. Therefore the convergence properties of the Jacobian-Free Newton-Krylov methods are similar for both SI-KDC and FI-KDC methods. In Sec. 4, we present numerical results for the eigenvalue distributions for different preconditioning techniques.

Note that applying $\tilde{S}_I$ to more terms in Eq. (3.9) will generate schemes with similar convergence properties. However the evaluation of the functions $\tilde{\mathbf{H}}$ may become more expensive as the unknowns may no longer decouple and a larger system has to be solved. Also, it is possible to modify the requirement that the provisional solution satisfies the algebraic equation, e.g., we can apply $\tilde{S}_E$ to $\{a_{31}, a_{32}\}$ terms, however this will significantly change the eigenvalue distribution compared with the FI-KDC scheme.

In our second formulation, instead of applying the "yp-formulation" to the algebraic variable z, we use z directly

$$\begin{bmatrix} \tilde{\mathbf{X}} + \tilde{\boldsymbol{\delta}}_1 \\ \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}_2 \\ \mathbf{0} \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \Delta t S \tilde{\mathbf{X}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}_1 \\ \mathbf{y_0} + \Delta t S \tilde{\mathbf{Y}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}_2 \\ \tilde{\mathbf{z}} + \tilde{\boldsymbol{\delta}}_3 \end{bmatrix}. \tag{3.10}$$

In the numerical implementation, this second form avoids the introduction of the variable $\tilde{\mathbf{Z}}$, but does not change significantly the distribution of eigenvalues of the Jacobian associated with this formulation (and hence the convergence of the Newton-Krylov scheme).

Finally, we note that it is unnecessary to introduce the variable $\tilde{\mathbf{Z}}$, and we can eliminate the variable $\tilde{\boldsymbol{\delta}}_3$ in Eq. (3.10) by working with z directly. We can therefore work on the simpler system

$$\begin{bmatrix} \tilde{\mathbf{X}} + \tilde{\boldsymbol{\delta}}_1 \\ \tilde{\mathbf{Y}} + \tilde{\boldsymbol{\delta}}_2 \\ \mathbf{0} \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \Delta t S \tilde{\mathbf{X}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}_1 \\ \mathbf{y_0} + \Delta t S \tilde{\mathbf{Y}} + \Delta t \tilde{S} \tilde{\boldsymbol{\delta}}_2 \\ \tilde{\mathbf{z}} \end{bmatrix}. \tag{3.11}$$

An immediate advantage of this formulation is that given the provisional solutions $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$, we can use a semi-implicit scheme to derive low-order solutions of $\tilde{\boldsymbol{\delta}}_1$, $\tilde{\boldsymbol{\delta}}_2$ and $\mathbf{z}$ at each node point, and define a reduced size function $[\tilde{\boldsymbol{\delta}}_1, \tilde{\boldsymbol{\delta}}_2] = \tilde{\mathbf{H}}_{RS}(\tilde{\mathbf{X}}, \tilde{\mathbf{Y}})$. Due to the reduced system size, the (outer) Newton-Krylov method becomes more efficient while requiring less memory. For our specific index 1 system, detailed algebraic manipulation of the function $\tilde{\mathbf{H}}_{RS}$ returns its Jacobian matrix

$$\begin{bmatrix} I - \Delta t \tilde{S} a_{11} + \Delta t \tilde{S} \frac{a_{13}a_{31}}{a_{33}} & -\Delta t \tilde{S} a_{12} + \Delta t \tilde{S} \frac{a_{13}a_{32}}{a_{33}} \\ -\Delta t \tilde{S} a_{21} + \Delta t \tilde{S} \frac{a_{23}a_{31}}{a_{33}} & I - \Delta t \tilde{S} a_{22} + \Delta t \tilde{S} \frac{a_{23}a_{32}}{a_{33}} \end{bmatrix}^{-1}$$
$$\left(\Delta t S \otimes \begin{bmatrix} a_{11} - \frac{a_{13}a_{31}}{a_{33}} & a_{12} - \frac{a_{13}a_{32}}{a_{33}} \\ a_{21} - \frac{a_{23}a_{31}}{a_{33}} & a_{22} - \frac{a_{23}a_{32}}{a_{33}} \end{bmatrix} - \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \right).$$

In this formulation, $\tilde{S}_I$ is associated with the a_{22} term. The requirement that the provisional solution satisfies the algebraic equation constraint at all nodes is equivalent to applying $\tilde{S}_I$ to terms associated with coefficient factors a_{31} and a_{32}, and $\tilde{S}_E$ can be applied to all remaining terms. Note that for small Δt, the Jacobian matrix $\tilde{\mathbf{H}}_{RS}$ approaches the negative identity matrix.

Finally, as in the first formulation, it may not be necessary to enforce the requirement that the provisional solution always stays on the manifold described by the algebraic equation. Hence, under appropriate conditions, $\tilde{S}_E$ can be applied to terms with coefficient factors a_{31} and a_{32}, especially when the coefficients $\frac{a_{13}a_{31}}{a_{33}}$, $\frac{a_{13}a_{32}}{a_{33}}$, $\frac{a_{23}a_{31}}{a_{33}}$, and $\frac{a_{23}a_{32}}{a_{33}}$ are $O(1)$.

In summary, even for a simple example, there are multiple ways in which the SI-KDC method can be formulated, and the choice can affect the computational cost of applying the preconditioner. In Sec. 4, some numerical tests are presented to show that the choice can also affect the convergence of the (outer) Newton-Krylov iterates in the SI-KDC methods. Therefore the choice of splitting must be carefully considered and will depend on the problem at hand.

3.3. **Index Two DAE System.** We demonstrate here that the task of finding proper semi-implicit splittings becomes even more involved for higher-index DAE systems. In this section, focusing on the scheme where the "yp-formulation" is applied to both differential and algebraic variables, we again consider Eq. (3.6) but now with

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \tag{3.12}$$

so that the index is two. Again we assume that a_{22} creates the stiffness in the equation and study the convergence and stability properties of different preconditioning techniques.

Following the presentation for the index 1 case above, for the first formulation, Eq. (3.9) becomes

$$\begin{bmatrix} I - \Delta t\tilde{S}a_{11} & -\Delta t\tilde{S}a_{12} & -\Delta t\tilde{S}a_{13} \\ -\Delta t\tilde{S}a_{21} & I - \Delta t\tilde{S}a_{22} & -\Delta t\tilde{S}a_{23} \\ -\Delta t\tilde{S}a31 & -\Delta t\tilde{S}a_{32} & 0 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\delta}}_1 \\ \tilde{\boldsymbol{\delta}}_2 \\ \tilde{\boldsymbol{\delta}}_3 \end{bmatrix} = A \begin{bmatrix} \mathbf{x_0} + \Delta tS\tilde{\mathbf{X}} \\ \mathbf{y_0} + \Delta tS\tilde{\mathbf{Y}} \\ \mathbf{z_0} + \Delta tS\tilde{\mathbf{Z}} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{X}} \\ \tilde{\mathbf{Y}} \\ \mathbf{0} \end{bmatrix}.$$

where $\tilde{S}$ is either $\tilde{S}_I$ or $\tilde{S}_E$ representing the low-order approximation of the integration operator. Clearly, we have to apply $\tilde{S}_I$ to the stiff component with coefficient a_{22}. If we want to enforce the condition that the provisional solution stays on the manifold defined by the algebraic equation, $\tilde{S}_I$ should be applied to both a_{31} and a_{32} terms. Also, we can apply $\tilde{S}_E$ to a_{11}, a_{12}, and a_{21} terms. In the following, we discuss different strategies for a_{13} and a_{23} terms, corresponding to terms related with the algebraic variable z in the system.

Our first observation is that, unlike in the first formulation for index one DAE systems, $\tilde{S}_E$ can no longer be applied to both a_{13} and a_{23} terms, as doing so generates a singular linear system when marching from t_j to t_{j+1}, since all coefficients for $\tilde{\boldsymbol{\delta}}_3^{j+1}$ are zero. Three remaining possibilities are to apply $\tilde{S}_I$ to both terms (case II); or $\tilde{S}_E$ to a_{13} and $\tilde{S}_I$ to a_{23} (case EI); or $\tilde{S}_I$ to a_{13} and $\tilde{S}_E$ to a_{23} (case IE). Applying the implicit function theorem, we can derive the Jacobian matrix for each function $\tilde{\mathbf{H}}$ in the KDC framework. The Jacobian matrix J_{II} for case II is given by

$$\left(E - \Delta t\tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \Delta t\tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & a_{13} \\ 0 & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta tS \otimes A - E),$$

J_{EI} is

$$\left(E - \Delta t\tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \Delta t\tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & 0 \\ 0 & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta tS \otimes A - E),$$

and J_{IE} is

$$\left(E - \Delta t\tilde{S}_E \otimes \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & 0 & a_{23} \\ 0 & 0 & 0 \end{bmatrix} - \Delta t\tilde{S}_I \otimes \begin{bmatrix} 0 & 0 & a_{13} \\ 0 & a_{22} & 0 \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta tS \otimes A - E).$$

Similarly, repeating this procedure for the FI-KDC scheme, we get the Jacobian matrix J_{FI}

$$\left(E - \Delta t\tilde{S}_I \otimes \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix} \right)^{-1} (\Delta tS \otimes A - E).$$

It is possible to understand the properties of the four different preconditioning techniques by studying the condition numbers for simple 3×3 matrices representing the linear system to be solved during each step when marching from t_j to t_{j+1}. In the following, we assume the stiff component $I - \Delta t\tilde{S}_I a_{22}$ in the matrix

$$\begin{bmatrix} I - \Delta t\tilde{S}a_{11} & -\Delta t\tilde{S}a_{12} & -\Delta t\tilde{S}a_{13} \\ -\Delta t\tilde{S}a_{21} & I - \Delta t\tilde{S}a_{22} & -\Delta t\tilde{S}a_{23} \\ -\Delta t\tilde{S}a_{31} & -\Delta t\tilde{S}a_{32} & 0 \end{bmatrix} \tag{3.13}$$

is about order λ, and the magnitude of other terms is either order ϵ when $\Delta t\tilde{S}_I$ is applied, or 0 when an explicit time stepping scheme is used. The matrices for cases II, EI, and IE, and the fully implicit FI-KDC formulation are

$$\begin{bmatrix} 1 & 0 & \epsilon \\ 0 & \lambda & \epsilon \\ \epsilon & \epsilon & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & \lambda & \epsilon \\ \epsilon & \epsilon & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & \epsilon \\ 0 & \lambda & 0 \\ \epsilon & \epsilon & 0 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} 1 \pm \epsilon & \epsilon & \epsilon \\ \epsilon & \lambda & \epsilon \\ \epsilon & \epsilon & 0 \end{bmatrix},$$

respectively. For $\lambda = 10^3$ and $\epsilon = 10^{-3}$, the condition number of the Jacobian matrix for each formulation is similar to that of the corresponding matrix above, given by $9.99 \cdot 10^8$, $1.00 \cdot 10^{12}$, $1.00 \cdot 10^9$ and $9.99 \cdot 10^8$, respectively, and the corresponding eigenvalues are almost identical. We therefore conclude that the convergence and stability properties of case II and IE are similar to FI-KDC, while case EI is not a proper preconditioner as it is more ill-conditioned. Numerical experiments presented in Sec. 4.5 on a similar problem show that the number of iterations in the outer Newton-Krylov methods for both cases II and IE are approximately the same as that of the FI-KDC. However as the unknowns can be decoupled in the IE formulation (during the substep from t_j to t_{j+1}, we can first solve the second equation for $\tilde{\delta}_2$ at t_{j+1}, then the third equation for $\tilde{\delta}_1$, and finally the first equation for $\tilde{\delta}_3$), hence case IE is the most efficient preconditioning approach.

In summary, a good semi-implicit preconditioning technique should reduce the amount of work required for evaluating the corresponding function $\tilde{\mathbf{H}}$ without significantly changing the convergence properties of the outer Newton-Krylov methods. This is possible for many stiff DAE systems, especially for those with nonlinear non-stiff components and linear stiff parts. However, the optimal semi-implicit preconditioner is problem dependent and requires an understanding of the underlying properties of the system. Also, in order to fully exploit the efficiency of the new KDC methods, optimized strategies have to be developed for the selection of adaptive step-size, order of the method, proper Newton-Krylov methods, as well as several different parameters. As the discussion here indicates, optimizing the performance of KDC methods for a given class of problems is an open research problem.

4. Preliminary Numerical Results

In this section, we present several numerical examples to illustrate the performance of the SI-KDC methods and validate the analyses presented in the previous section.

4.1. **Linear Index One DAE System.**

In the first example, we consider a stiff linear index one DAE system

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}' = \begin{bmatrix} 2 & 0 & -1 & 1 \\ 0 & -10^4 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 - \exp(t) \\ y_3 \\ y_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \exp(t) \\ 0 \\ 0 \end{bmatrix} \tag{4.1}$$

where the analytic solution is [cos(t), exp(t), sin(t), -cos(t)]. Following the presentation in Sec. 3.2, the first splitting of the equations is formed by decomposing the matrix on the right hand side of Eq. (4.1) into two parts:

$$\begin{bmatrix} 2 & 0 & -1 & 1 \\ 0 & -10^4 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -10^4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

The first piece of the matrix splitting represents the terms to be handled explicitly in the SI-KDC preconditioner, and the second represents those to be handled implicitly. We treat the algebraic constraint implicitly to keep the solution on the manifold defined by this algebraic equation in each iteration.

To test the efficiency of the semi-implicit formulation, we compare the eigenvalue distribution of the matrix $J_{SI}+I$ derived from Eq. (3.5) to that of $J_{FI}+I$ from the FI-KDC method. The particular discretization used is for Δt= 0.2 and 5 Radau IIa nodes. In Fig. 4.1, the eigenvalues of the respective Jacobians are plotted, and it is evident that those from the SI-KDC approach are almost identical to those from FI-KDC. As the convergence of the outer Newton-Krylov schemes is determined by the eigenvalue distributions, this would indicate the SI-KDC method will converge to the collocation formulation at the same rate as FI-KDC. However, since we can easily derive $\tilde{\delta}_3$ from the 3rd equation and then $\tilde{\delta}_1$ from the first equation, the SI-KDC method will have much lower computational cost than the FI-KDC approach.

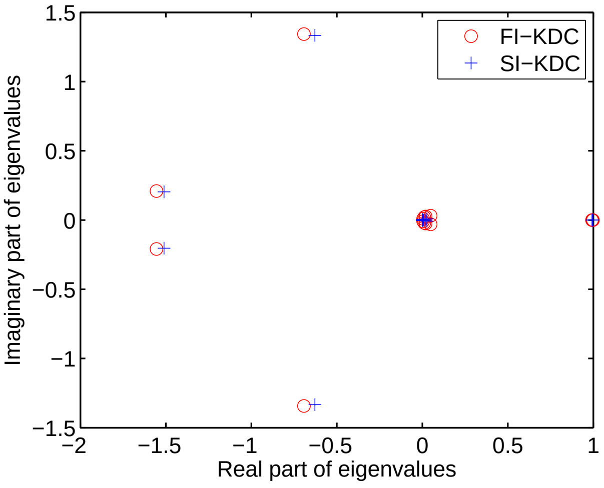

FIGURE 2. Comparing eigenvalue distributions of SI-KDC with FI-KDC for linear index 1 DAE system. Note the different scales on the x-axes.

One concern when using the KDC type methods is the storage and number of operations required by the Krylov subspace methods. When GMRES is used, the required memory and number of multiplications grow linearly and quadratically with iterations, respectively. For large scale partial differential equation applications, spatial data must be stored at each of the p nodes and at each of the GMRES iterations, and the required memory storage may quickly exceed existing computer architecture capacities. Hence alternative Krylov methods are often preferred, such as the restarted GMRES, Bi-conjugate gradients stabilized (BiCGStab), and transpose free quasi minimal residual (TFQMR) methods. In Fig. 3, we compare the convergence of the GMRES with BiCGStab and TFQMR. In the experiments, we use 5 Radau points and time step-size $\Delta t = 0.1$ for $t \in [1, 1.1]$. Our numerical results show very similar convergence rates for these methods, however for the BiCGStab and TFQMR based methods the required memory is bounded, and the number of multiplications grows linearly with iteration number.

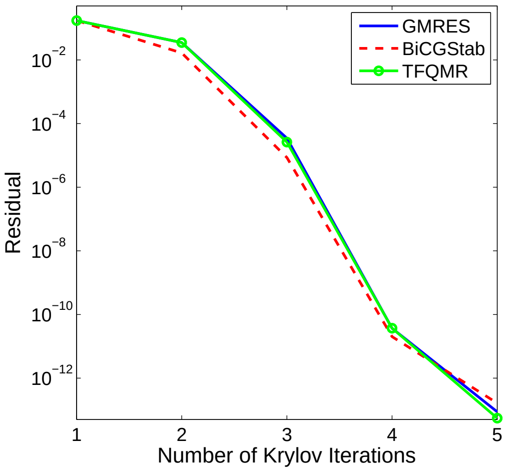

FIGURE 3. Comparing different Krylov subspace methods.

As mentioned in Sec. 3.2, other SI-KDC formulations can be considered for this index 1 DAE system. In the following, we study the convergence and efficiency of the third approach in which the error equation formulation is not applied to the algebraic variable and the “reduced-size” function given by $[\tilde{\boldsymbol{\delta}}_1, \tilde{\boldsymbol{\delta}}_2, \tilde{\boldsymbol{\delta}}_3] = \tilde{\mathbf{H}}_{RS}(\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2, \tilde{\mathbf{Y}}_3)$ has only 3 unknowns instead of 4 in the first formulation. In the experiment, we march from $t = 0.0$ to $t = 10.0$ for different step-sizes and plot the error as functions of the (left) step-sizes and (right) number of function calls in Fig. 4. It can be seen that the SI-KDC scheme using the third formulation with reduced system size uses fewer function evaluations and hence is more efficient for the same accuracy requirement.

4.2. **Nonlinear Index One DAE System.** In our second example, we consider a stiff nonlinear DAE problem

(4.2)

$$\begin{bmatrix} y_1 - \cos(t) \\ y_2 - \sin(t) \\ 0 \end{bmatrix}' = \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & -10^6 & 0 \\ 0 & 0 & 0 \end{bmatrix} + U \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} U^T \right) \begin{bmatrix} (y_1 - \cos(t))y_2 \\ y_2 - \sin(t) \\ y_3 - t \end{bmatrix}$$

where U is an orthogonal matrix. For this system, the non-stiff component is nonlinear and the stiff part linear, we therefore apply the explicit $\tilde{S}_E$ to the non-stiff component and the implicit $\tilde{S}_I$ to the stiff part. Notice that only one *linear* solve is required in the inner Newton iterations in the SI-KDC scheme whereas a nonlinear problem would arise in the FI-KDC scheme. In the following, we compare

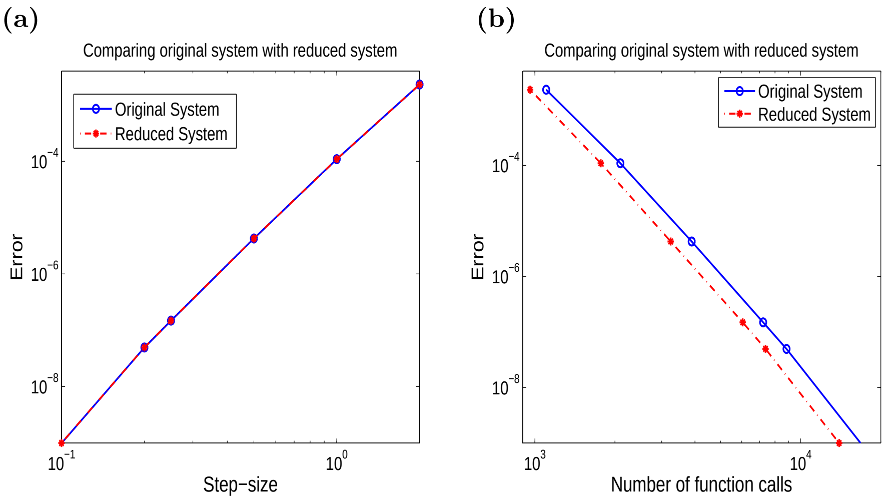

FIGURE 4. The error as functions of stepsize (left) and number of function calls (right).

the convergence behavior and CPU time of the SI-KDC scheme with those from FI-KDC where implicit time stepping schemes are applied to all terms in the system. In Fig. 5, we compute the solution from $t_0 = 0.2$ to $t_{final} = 0.25$ with step-size $\Delta t = 0.05$ using 5 Radau IIa points, and examine the residual after each (outer) Krylov iteration for both SI-KDC and FI-KDC methods. It can be seen that the convergence behavior in the SI-KDC scheme is very similar to that in FI-KDC.

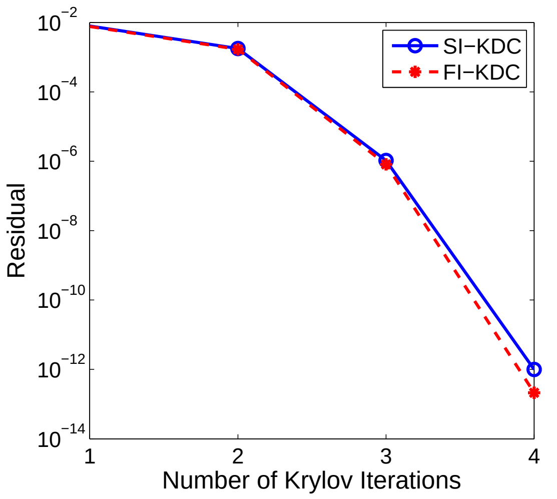

FIGURE 5. Comparing the convergence of SI-KDC and FI-KDC methods for nonlinear index 1 DAE system.

To compare the CPU time and the number of function evaluations, we use different number of nodes (3, 5, 7, 9 and 12) and march from $t = 0$ to $t_{final} = 10.0$ with step-size $\Delta t = 1.0$. In Fig. 6, we plot the error as functions of the (left) CPU time and (right) number of function evaluations (each access to Eq. (4.2) is defined as one function call) for each method. Each data point represents the result for a specific number of nodes. Clearly, for the same accuracy requirement, the SI-KDC scheme is much faster than FI-KDC, since only one linear solve is required when marching from t_j to t_{j+1} for the SI-KDC method, while a nonlinear equation system must be solved in the FI-KDC approach. In Fig. 7, we plot how the error changes

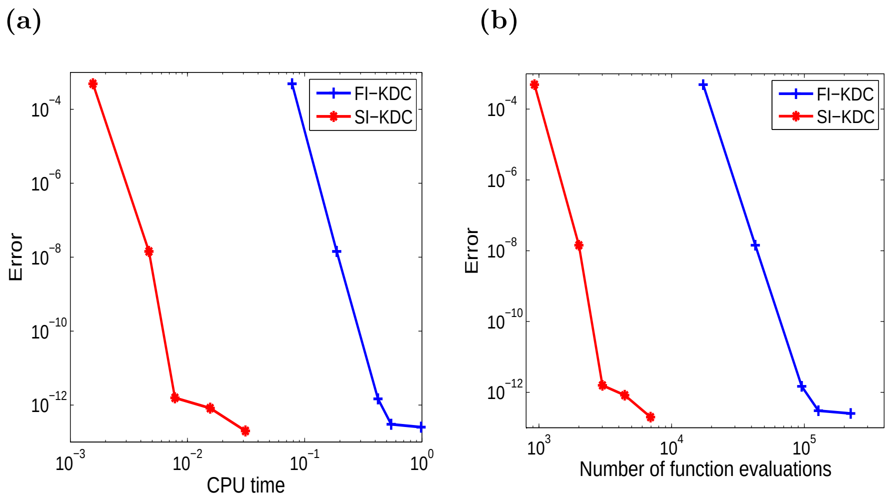

FIGURE 6. The error as functions of CPU time (left) and number of function evaluations (right).

as a function of the number of function evaluations using different time step sizes (Δt= 0.25, 0.5, 1.0, and 2.0) and number of node points (3, 4, and 5) for both SI-KDC and FI-KDC when marching from $t_0 = 0$ to $t_{final} = 10.0$. Each data point represents the number of function evaluations and accuracy for a specific set of Δt and number of nodes. We can see that for the same accuracy requirement, the SI-KDC scheme is much faster than FI-KDC; and for both methods, higher-order schemes (more nodes) are more efficient than lower-order ones for higher-accuracy requirements.

4.3. **Electrical Power System.**

Differential algebraic equations have been widely used in the study and engineering of the bulk transfer of electrical power (see e.g., [5, 48, 60]). Typical electrical power system networks include a large number of dynamic and static components such as generators, exciters, governors, loads, transformers, and other power electronic devices, where the dynamics and constraints for each individual component are often modeled by a system of DAEs. As the power systems exhibit a wide-range of time varying dynamics that may span several orders of magnitude, their efficient numerical simulations are considered challenging. In this section, to evaluate the performance of the SI-KDC approach, we consider a simple power stabilizer system which has 9 buses and 3 generators with constant power loads. Each generator has 2 states, so the number of differential states and algebraic equations are 6 and 18, respectively. This system can be described by the

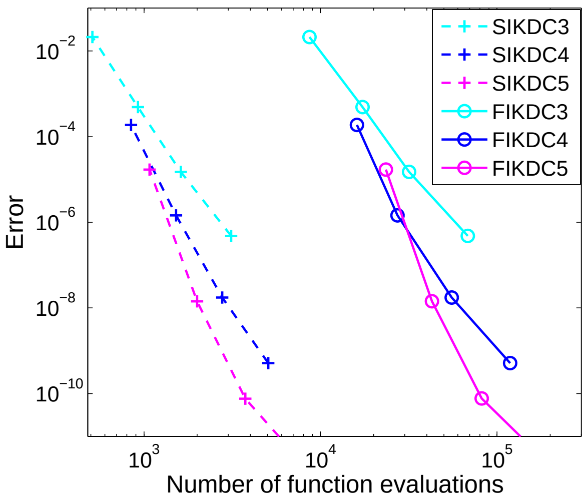

FIGURE 7. Comparing SI-KDC with FI-KDC for different step sizes and number of nodes.

following index 1 DAE system:

$$\delta' = \Omega_b(\omega - 1),$$

$$\omega' = (P_m - P_e - D(\omega - 1))/M,$$

$$V_i \sum (V_j(G_{ij}\cos(\theta_i - \theta_j) + B_{ij}\sin(\theta_i - \theta_j))) + P_{gi} - P_{di} = 0,$$

$$V_i \sum (V_j(G_{ij}\sin(\theta_i - \theta_j) - B_{ij}\cos(\theta_i - \theta_j))) + Q_{gi} - Q_{di} = 0,$$

where the differential variables δ and ω are the internal state vectors (generators and loads), and V_i and θ_i are the algebraic variables for voltage magnitude and phase. In the system, D is a coefficient representing a damping factor, and when D has a large magnitude, the system becomes stiff with the term $-D(\omega-1)$ being linear and stiff and other terms are nonlinear and non-stiff. Here M is an inertia constant; P_m denotes the mechanical power; $P_e = f(V_i, \theta_i)$ is the electrical power; $P_g = f(\delta_i, \theta_i)$ and $Q_g = f(\delta_i, \theta_i)$ represent respectively the active and reactive power injected in the network by generators; P_d and Q_d are respectively the active and reactive power absorbed from the network by loads; and G_{ij} and B_{ij} are respectively a real and an imaginary part of an admittance matrix to represent the current status between load i and load j. Also, the first two sets of differential equations model the dynamics of the generators and loads, and the remaining algebraic equations represent the fast power balance dynamics on the sparsely connected distribution network of power lines and buses. To study the dynamics of the system, we assume a one-phase fault on a line between bus 2 and bus 7 occurs at $t = 1$, and clears out at $t = 2$. When the fault occurs, the shunt admittances of the network are modified and the admittance matrix is recomputed. We neglect further details of this model and techniques to split other systems to stiff and non-stiff in general, and refer interested readers to [5, 48, 60].

In the simulation, we require that the provisional solution stays on the manifold defined by the algebraic constraints, by applying implicit schemes to all terms in the algebraic equation. For the differential equations, we apply explicit schemes to both non-stiff components and algebraic variables, and implicit schemes only to the stiff components. In Fig. 8, we first show how the accuracy of the SI-KDC method depends on the number of Radau IIa nodes and different time step sizes, by plotting the accuracy as a function of the number of total nonlinear solves (one nonlinear solve is required for each substep from t_j to t_{j+1}). It can be seen that (a) for a

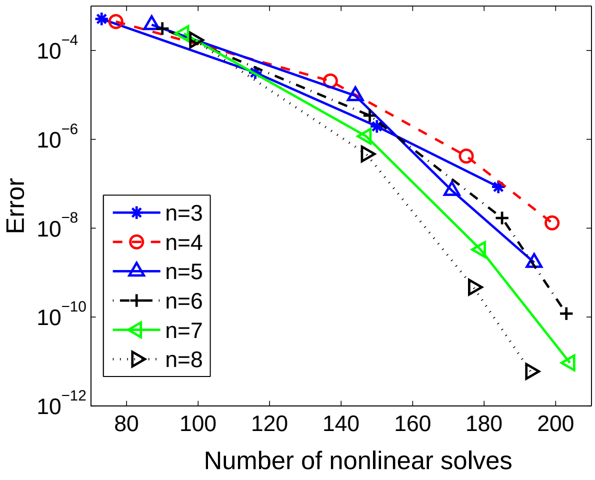

Figure 8. Accuracy of SI-KDC method vs. number of nonlinear solves for different number of nodes.

fixed number of nodes, smaller time step sizes (more nonlinear solves) are required for higher-accuracy requirements, and (b) higher-order methods (more nodes) are in general more efficient than lower-order ones for higher-accuracy requirements.

In Fig 9, using 6 Radau IIa nodes for each time step from $t = 0$ to $t = 1.8$, to study the convergence properties of the outer Newton-Krylov methods in the SI-KDC approach, we show the residual after each low-order SDC iteration (one $\tilde{\mathbf{H}}_{SI}$ evaluation) (left plot), and how the accuracy depends on the number of total nonlinear solves required to march from t_j to t_{j+1} (right plot). These results are compared with those from the FI-KDC approach. Clearly, the FI-KDC approach is optimal in stability and has (slightly) better convergence rates in the outer Newton-Krylov iterations, however the residual after each $\tilde{\mathbf{H}}$ evaluation (SDC iteration) in the SI-KDC method decays in a very similar way as in FI-KDC. As explicit schemes are applied to the non-stiff components and algebraic variables in the differential equations, the size of nonlinear system from the SI-KDC scheme is smaller than that from FI-KDC method, and therefore the SI-KDC preconditioning technique is more efficient than FI-KDC for this specific application.

There exist many numerical simulation tools and methods for power systems [2, 40, 60], including the techniques based on splitting the DAE systems to differential and algebraic parts and solving them separately using ODE solvers for the differential parts and a Newton-type method (e.g. Newton-Raphson) for algebraic components. In the following, we compare the performance of our SI-KDC approach with a MATLAB based package called "PSAT", a power system solver based on the Newton-Raphson methods and trapezoidal rules [44]. In Fig. 10, we examine the accuracy of the SI-KDC approach for different time step sizes and

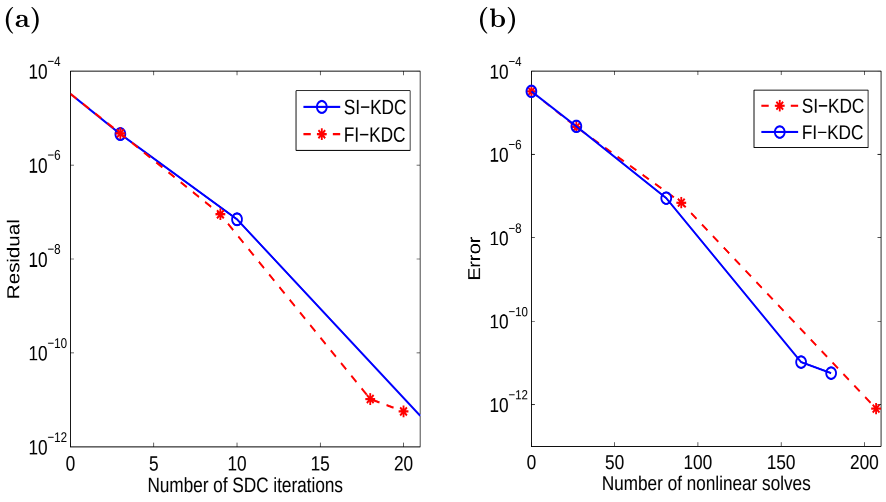

FIGURE 9. (left) Residual after each SDC iterations, and (right) accuracy vs. number of nonlinear solves.

number of nodes, and compare the results with those from PSAT. In the figure, each curve represents the results for a fixed time step-size, and each point on a curve represents a different number of nodes used in the simulation, ranging from 3 to 10. Clearly, for a fixed step-size, more nodes generate higher-accuracy results, and higher-order methods are more efficient for a prescribed accuracy requirement. Also, compared with PSAT, the SI-KDC requires far fewer nonlinear solves for the same accuracy requirement. In our simulation, fixed step sizes are used for both SI-KDC and PSAT.

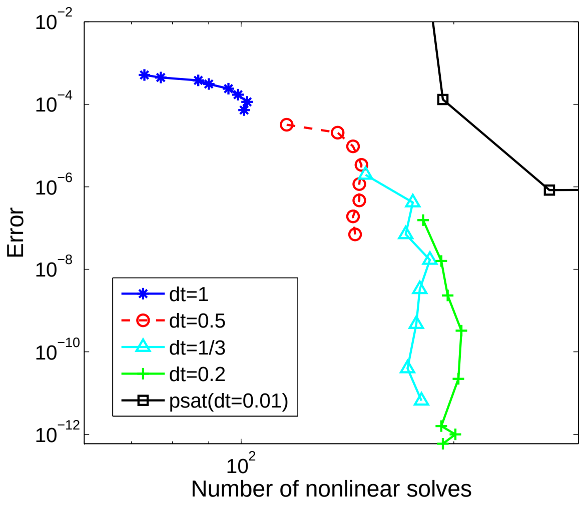

FIGURE 10. Comparing the accuracy and efficiency of SI-KDC with PSAT.

4.4. **Transistor Amplifier Problem.** In our next example, we consider the transistor amplifier problem in [1] which is a stiff DAE system of index 1 consisting of 8 equations given by

$$M\frac{dy}{dt} = f(y), \quad y(0) = y_0, \quad y'(0) = y_0',$$

with $y \in \mathbb{R}^8$ and $0 \le t \le 0.2$. The matrix M is of rank 5 and is given by

$$M = \begin{pmatrix} -C_1 & C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_1 & -C_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -C_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -C_3 & C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_3 & -C_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -C_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -C_5 & C_5 \\ 0 & 0 & 0 & 0 & 0 & 0 & C_5 & -C_5 \end{pmatrix}.$$

The function f is defined as

$$f(y) = \begin{pmatrix} -\frac{U_e(t)}{R_0} + \frac{y_1}{R_0} \\ -\frac{U_b}{R_2} + y_2(\frac{1}{R_1} + \frac{1}{R_2}) - (\alpha - 1)g(y_2 - y_3) \\ -g(y_2 - y_3) + \frac{y_3}{R_3} \\ -\frac{U_b}{R_4} + \frac{y_4}{R_4} + \alpha g(y_2 - y_3) \\ -\frac{U_b}{R_6} + y_5(\frac{1}{R_5} + \frac{1}{R_6}) - (\alpha - 1)g(y_5 - y_6) \\ -g(y_5 - y_6) + \frac{y_6}{R_7} \\ -\frac{U_b}{R_8} + \frac{y_7}{R_8} + \alpha g(y_5 - y_6) \\ \frac{y_8}{R_9} \end{pmatrix}$$

where g and U_e are auxiliary functions given by $g(x) = \beta(e^{\frac{x}{U_F}} - 1)$ and $U_e(t) = 0.1\sin(200\pi t)$. A fully implicit KDC scheme was applied to this problem in [33], and numerical results show that for the same accuracy requirements, the efficiency of the uniform FI-KDC solver is comparable to existing optimized adaptive solvers, including the MEBDFI and RADAU packages (see [1] for discussions of the two methods), especially when higher-accuracy is required.

For this example, as the stiffness also comes from the nonlinear components ($g(x)$ terms), we consider two different SI-KDC strategies for discretizing these terms. In the first scheme denoted by "SI-KDC-S" (S for Simple), we apply an explicit method to g terms, and an implicit scheme to the remaining linear terms. In the second method, as the derivative of $g(x)$ is easily available for this specific example, we employ a "linearly-implicit" approach by rewriting $g(y)$ as

$$\begin{aligned} g(y) &= g(y) - g'(y_0)(y - y_0) + g'(y_0)(y - y_0) \\ &= (g(y) - g'(y_0)y) + g'(y_0)y. \end{aligned}$$

We then apply an implicit scheme to the linearized terms, and an explicit scheme to the remaining nonlinear terms. We denote the second method as "SI-KDC-J" due to the use of the Jacobian matrix.

In Fig. 11, we show how the residual changes as a function of the number of SDC iterations and compare with the FI-KDC scheme using two representative time steps of size $\Delta t = 0.0016666$: in the left panel $t \in [0.003333, 0.005]$ and in the right $[0.008333, 0.01]$. In our simulation, we use 8 Radau IIa nodes, and the numerical discretization error from the GRK formulation is approximately $1.5e-4$. It can be seen that more SDC iterations ($\tilde{\mathbf{H}}$ evaluations) are required by the SI-KDC-S scheme than either the SI-KDC-J or the FI-KDC method. Also, in the right panel, the SI-KDC-J converges very similarly to the FI-KDC method. It is worth noting also that when the stepsize is chosen to be 0.005 or larger for this problem, one can observe overflow in the solution for both SI-KDC schemes, due to the over-flow in the SI predictor.

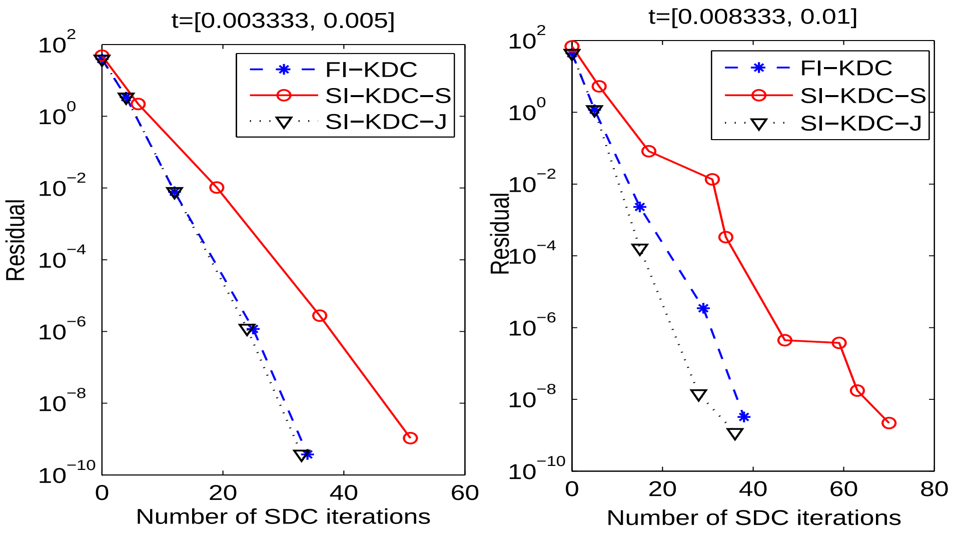

FIGURE 11. Magnitude of the residual versus iteration for FI-KDC, SI-KDC-S, and SI-KDC-J schemes.

However, as only one linear solve is required when marching from t_i to t_{i+1} in each SDC iteration in the SI-KDC, while a nonlinear Newton type solver (we adapted a package from [36] in our code) has to be introduced in FI-KDC, one $\tilde{\mathbf{H}}$ evaluation in SI-KDC is less expensive than that in FI-KDC. As a result, the SI-KDC preconditioning strategies are often preferred as they can be more efficient in CPU time, even though fewer number of SDC iterations may be required by the FI-KDC scheme in most cases. This is further explained in Fig. 12, in which we use 8 Radau IIa nodes, and solve from $t = 0$ to 0.01 using different time step-sizes. In the left plot, we show the error versus time step-size for the SI-KDC-S, SI-KDC-J, and FI-KDC methods. As these methods solve the same GRK formulation, it is not surprising that the error for a given time step is almost identical for each approach. In the right plot, we show the CPU time and accuracy of these methods. It can be seen that (1) compared with the FI-KDC scheme, both SI-KDC methods are more efficient for the same accuracy requirements; and (2) as the Jacobian matrix can be easily evaluated in this example, the SI-KDC-J scheme outperforms the SI-KDC-S strategy, due to the reduced number of iterations when using the linearized approximation.

We therefore conclude that the SI-KDC schemes may also be applied to problems with nonlinear mildly stiff components, especially when the number of the corresponding "bad" eigenvalues which cause the stiffness is small, and no overflow happens for the selected stepsizes in the computation. We refer interested readers to [32] for further examples where explicit KDC methods work on mildly stiff problems, while the original SDC methods may be divergent.

4.5. **Linear Index Two DAE Systems.**

Finally in this section, to numerically validate the analyses in Sec. 3.3, we study the SI-KDC techniques for two different

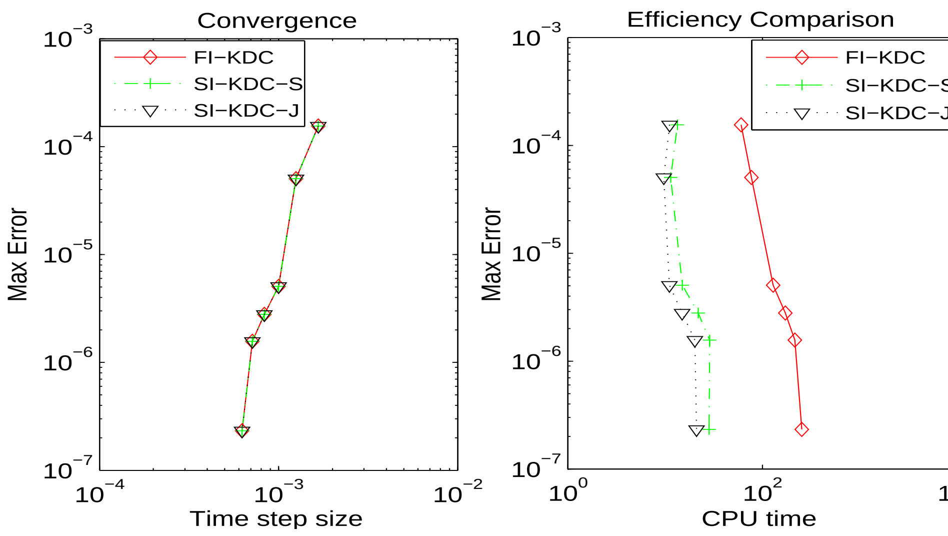

FIGURE 12. Comparing the order and CPU time of the FI-KDC and SI-KDC schemes.

index two DAE systems. We first consider the system (see [6])

$$\begin{cases} x_1' = (\alpha - \frac{1}{2-t})x_1 + (2-t)\alpha z + \frac{3-t}{2-t}\exp^t = f_1(x_1,z), \\ x_2' = \frac{1-\alpha}{t-2}x_1 - 10^4 x_2 + (\alpha-1)z + (10^4+1)\exp^t = f_2(x_1,x_2,z), \\ 0 = (t+2)x_1 + (t^2-4)x_2 - (t^2+t-2)\exp^t = g(x_1,x_2) \end{cases}$$

with $\alpha \sim O(1)$. Several semi-implicit approaches can be applied as discussed in Sec. 3.3. The SI-KDC II approach applies implicit schemes to the algebraic variable z in both differential equations, and the resulting low-order stepping scheme can be succinctly represented as

$$\begin{cases} X_1^{j+1} = f_1(x_1^j, z^{j+1}), \\ X_2^{j+1} = f_2(x_1^j, x_2^{j+1}, z^{j+1}), \\ 0 = g(x_1^{j+1}, x_2^{j+1}) \end{cases} \tag{4.3}$$

where the superscript j represents the node point t_j, and the original equation is used instead of the error equation form to simplify the notations. The SI-KDC IE formulation applies an implicit scheme to z in the first equation, and an explicit method to z in the second equation as in

$$\begin{cases} X_1^{j+1} = f_1(x_1^j, z^{j+1}), \\ X_2^{j+1} = f_2(x_1^j, x_2^{j+1}, z^j), \\ 0 = g(x_1^{j+1}, x_2^{j+1}). \end{cases} \tag{4.4}$$

Similarly, the SI-KDC EI formulation is given by

$$\begin{cases} X_1^{j+1} = f_1(x_1^j, z^j), \\ X_2^{j+1} = f_2(x_1^j, x_2^{j+1}, z^{j+1}), \\ 0 = g(x_1^{j+1}, x_2^{j+1}), \end{cases} \tag{4.5}$$

and the FI-KDC scheme uses the discretization

$$\left\{\begin{array}{l} X_1^{j+1}=f_1(x_1^{j+1},z^{j+1}),\\ X_2^{j+1}=f_2(x_1^{j+1},x_2^{j+1},z^{j+1}),\\ 0=g(x_1^{j+1},x_2^{j+1}). \end{array}\right. \tag{4.6}$$

As we discussed in Sec. 3.3, the SI-KDC EI preconditioning technique is ill-conditioned, which is validated by the eigenvalue distribution of the matrix $J_{EI}+I$ plotted in the left of Fig. 13, in comparison with those from $J_{IE}+I$. In the right plot of Fig. 13, we compare the eigenvalue distributions of the SI-KDC IE with the fully implicit approach in Eq. (4.6), it can be seen that the eigenvalues are very similarly distributed, therefore the convergence properties of the SI-KDC IE approach is similar to those of the FI-KDC method. In Table. 1, we show the condition number of the Jacobian matrix for different low-order stepping schemes and different number of nodes. Not surprisingly, the condition number of the SI-KDC EI matrix is huge and increases very rapidly as the number of nodes increases.

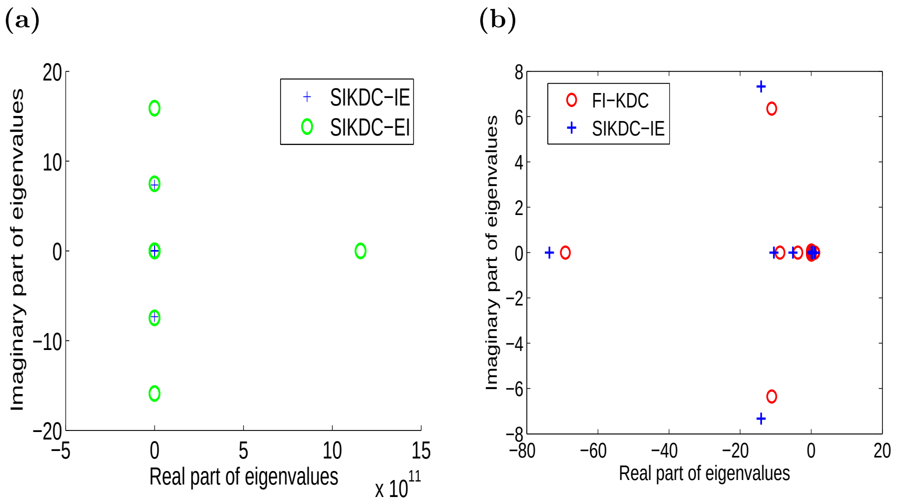

FIGURE 13. Comparing the eigenvalue distributions for (left) SI-KDC IE and SI-KDC EI, and (right) SI-KDC IE and FI-KDC.

	SI-KDC II	SI-KDC IE	SI-KDC EI	FI-KDC
n=3	1.0961e+10	1.0895e+10	3.7478e+17	9.2462e+09
n=4	1.0488e+10	1.0372e+10	3.6052e+19	9.4638e+09
n=5	1.0406e+10	1.0229e+10	2.7762e+21	9.7303e+09
n=8	1.0691e+10	1.0248e+10	1.1832e+27	1.0405e+10
n=10	1.1015e+10	1.0323e+10	8.7312e+30	1.0821e+10
n=15	1.2053e+10	1.0491e+10	2.2292e+38	1.1951e+10
n=20	1.3376e+10	1.0603e+10	4.8088e+43	1.3307e+10

TABLE 1. Condition number of Eq. (3.13) for different node numbers and preconditioners.

Note that for special systems, SI-KDC EI can be stable. Consider the index two system

$$\begin{cases} x_1' = x_1 = f_1(x_1), \\ x_2' = 2x_1 - 10^5 x_2 + z + (10^5+1)\exp(t) = f_2(x_1, x_2, z), \\ 0 = x_1 + x_2 = g(x_1, x_2) \end{cases}$$

where the algebraic variable z does not appear in the first equation. For this problem, the eigenvalue distribution of the matrix $J_{EI} + I$ is almost identical to that of the FI-KDC as shown in Fig. 14, and the SI-KDC EI approach becomes stable. In our numerical simulation, we use 7 Radau nodes for each time step, and

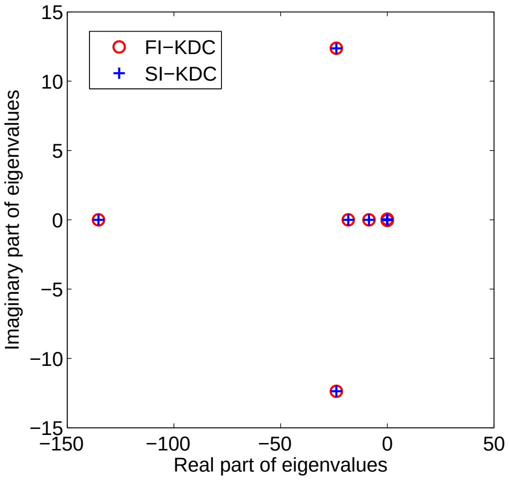

FIGURE 14. Comparing SI-KDC and FI-KDC for index 2 linear DAE.

march from $t = 0.2$ to $t = 1.2$ using step-size $\Delta t = 0.1$. As the system is linear, no Newton iterations are required and we use the GMRES method to solve the preconditioned system. In Fig. 15, we compare the residual after each GMRES step for both the SI-KDC EI and FI-KDC methods for one time step. Again, the convergence of the SI-KDC EI approach is very similar to that of the FI-KDC.

We have also studied other higher-index DAE systems and our analysis and numerical experiments show that designing optimal semi-implicit schemes for stiff DAE systems are highly problem dependent, and requires detailed study of the linearized system.

5. CONCLUSION

In this paper, semi-implicit KDC (SI-KDC) techniques are introduced for stiff DAE systems and compared with fully-implicit KDC (FI-KDC) methods. The SI-KDC technique treats the non-stiff components in the SDC preconditioner using explicit methods and solves the stiff parts using implicit schemes. Our analysis and numerical experiments show that when proper semi-implicit splittings are used, the eigenvalues from the Jacobian matrix of the preconditioned function $\tilde{\mathbf{H}}_{SI}$ are similarly distributed as those from the FI-KDC method. However, when marching from t_j to t_{j+1}, the SI-KDC preconditioner only requires the solution of a

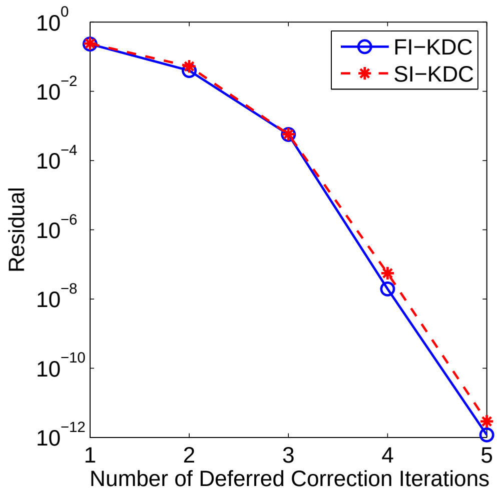

FIGURE 15. Comparing the SI-KDC and the FI-KDC for index 2 linear DAE.

simplified system, compared with the generally fully nonlinear system in the FI-KDC discretization. The SI-KDC methods are therefore numerically less expensive than the FI-KDC methods for the same accuracy requirements. Our analysis also shows that unlike the ODE case, the existence of algebraic equations and algebraic variables makes the design of optimal semi-implicit schemes a challenging task for higher-index DAE systems, and requires detailed analysis and understanding of the underlying system. It is therefore unrealistic to develop general purpose SI-KDC numerical solvers for higher-index DAEs with optimal efficiency.

Note that the SI-KDC methods can be generalized to many partial differential equation systems with algebraic constraints. In particular, we are currently working on the SI-KDC techniques for the Navier-Stokes equations and a two-scale partial differential equation model for water treatment processes. Results along these directions will be reported in the future.

REFERENCES

1. http://pitagora.dm.uniba.it/~ testset/
2. V. AJJARAPU, *Computational Techniques for Voltage Stability Assessment and Control*, Springer, 2007.
3. G. AKRIVIS, M. CROUZEIX, AND C. MAKRIDAKIS, *Implicit-explicit multistep methods for quasi-linear parabolic equations*, Numer. Math., 82:521–541, 1999.
4. B. K. ALPERT, AND V. ROKHLIN, *A fast algorithm for the evaluation of Legendre expansions*, SIAM J. on Sci. and Stat. Computing, 12,158-179, 1991.
5. P. M. ANDERSON, *Analysis of Faulted Power Systems*, Wiley-IEEE Press, 1995.
6. U. M. ASCHER, AND L. R. PETZOLD, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998.
7. U. M. ASCHER, S. J. RUUTH, AND B. WETTON, *Implicit-Explicit Methods For Time-Dependent PDEs*, SIAM J. Numer. Anal, 32, 797–823, 1997.
8. U. M. ASCHER, S. J. RUUTH, AND R. J. SPITERI, *Implicit-Explicit Runge-Kutta Methods for Time-Dependent Partial Differential Equations* Appl. Numer. Math, 25, 151–167, 1997.
9. K. ATKINSON, *An Introduction to Advanced Numerical Analysis*, 2nd edition, John Wiley, 1989.

10. W. Auzinger, H. Hofstätter, W. Kreüzer, and E. Weinmuller, *Modified defect correction algorithms for ODEs. Part I: General theory*, Numer. Algorithms, 36: 135-156, 2004.
11. R. Barrett, et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd Edition, SIAM, Philadelphia, 1994.
12. S. Boscarino, *Erroro analysis of IMEX Runge-Kutta methods derived from differential-algebraic systems*, SIAM J. Numer. Anal., Vol. 45, No. 4, pp. 1600–1621, 2007.
13. S. Boscarino, *On an accurate third order implicit-explicit Runge-Kutta method for stiff problems*, Appl. Numer. Math, Vol. 59, pp. 1515–1528, 2009.
14. A. Bourlioux, A.T. Layton, and M.L. Minion, *High-Order Multi-implicit spectral deferred correction methods for problems of reactive flow*, J.Comput. Phys., 189, 351–376, 2003.
15. E. Bouzarth, M. Minion, *A multirate integrator for regularized stokeslets*, J. Comp. Phys. 229 (2010) 4208–4224.
16. K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, 1995.
17. S. Bu, J. Huang, and M.M. Minion, *Semi-implicit Krylov Deferred Correction Methods for Ordinary Differential Equations*, Proceedings of the American Conference on Applied Mathematics, Houston, May, 2009.
18. M. P. Calvo, and C. Palencia, *Avoiding the order eduction of Runge-Kutta methods for linear initial boundary value problems*, Math. Comput., 71, 1529-1543, 2002.
19. M. P. Calvo, J. de Frutos, and J. Novo, *Linearly implicit Runge-Kutta methods for advection-reaction-diffusion equations*, Appl. Numer. Math., 37:535–549, 2001.
20. C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, 1988.
21. K. Chen, A. Iserles, and P. G. Ciarlet (Editors), *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, 2005.
22. K. Dekker, and J. G. Verwer, *Stability of Runge-Kutta methods for stiff nonlinear differential equations.* CWI Monographs. North-Holland, 1984.
23. A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40(2) 241-266, 2000.
24. A. Dutt, M. Gu, and V. Rokhlin, *Fast Algorithms for Polynomial Interpolation, Integration, and Differentiation*, SIAM J. on Num. Anal., 33(5), 1689-1711, 1996.
25. J. Frank, W. Hundsdorfer and J. G. Verwer, *Stability of Implicit-Explicit Linear Multistep Methods*, Applied Numerical Mathematics: Transactions of IMACS, Vol.25,2–3, 1997
26. D. Gottlieb, and S. S. Orszag, *Numerical Analysis of Spectral Methods*, SIAM, Philadelphia, 1977.
27. L. Greengard, *Spectral Integration and Two-Point Boundary Value Problems*, SIAM J. Num. Anal. 28, 1071-1080 1991.
28. L. Greengard, and V. Rokhlin, *A Fast Algorithm for Particle Simulations*, J. Comput. Phys., 73, 325–348, 1987.
29. E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer-Verlag, 2002.
30. E. Hairer, C. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer-Verlag, 1989.
31. E. Hairer, and G. Wanner, *Solving Ordinary Differential Equations II*, Springer, 1996.
32. J. Huang, J. Jia, and M. Minion, *Accelerating the Convergence of Spectral Deferred Correction Methods*, J. of Comp. Physics, 214(2), 633–656 , 2006.
33. J. Huang, J. Jia, M. Minion, *Arbitrary Order Krylov Deferred Correction Methods for Differential Algebraic Equations*, Comput. Phys., 221,(2), 739–760, 2007.
34. J. Jia, J. Hunag, *Krylov deferred correction accelerated method of lines transpose for parabolic problems*, J. Comput Phys, 227(3),1739-1753, 2008.
35. C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, 1995.
36. C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, 2003.
37. C. A. Kennedy and M. H. Carpenter, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations.*, Appl. Numer. Math., 44:139–181, 2003.
38. D.A. Knoll, D.E. Keyes, *Jacobian-free Newton Krylov methods: A survey of approaches and applications*, J. Comput. Phys. 193 (2004) 357-397.
39. P. Kolm, S. Jiang, and V. Rokhlin, *Quadruple and octuple layer potentials in two dimensions. I. Analytical apparatus*, Appl. Comput. Harmon. Anal. 14, no. 1, 2003.

40. A. Kurita, H. Okubo, K. Oki, S. Agematsu, D. B. Klapper, N. W. Miller, W. W. Price , J. J. Jr.Sanchez-Gasca, K. A. Wirgau, T. D. Younkins , *Multiple time-scale power system dynamic simulation* , Power Systems, IEEE Transactions on On page(s): 216- 223, Vol. 8, Issue: 1, Feb. 1993.
41. A. T. Layton, and M. L. Minion, *Conservative Multi-Implicit Spectral Deferred Correction Methods for Reacting Gas Dynamics*, J. Comput. Phys, 194(2), 697-714, 2004.
42. A. T. Layton, and M. L. Minion, *Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations*, BIT, 45(2), 341-373, 2005.
43. A. T. Layton and M. L. Minion, *Implications of the Choice of Predictors for Semi-Implicit Picard Integral Deferred Correction Methods*, Comm.App. Math. and Comp. Sci., 2(1),1–34, 2007.
44. F. Milano, *An Open Source Power System Analysis Toolbox*, Power Systems, IEEE Transactions on, Vol. 20, No.3, 2005.
45. M. L. Minion, *Higher-order Semi-implicit Projection Methods in Numerical Simulations of Incompressible Flows*, Papers from the workshop held in Half Moon Bay, CA, June 19-21, 2001. also LLNL Technical Report UCRL-JC-145295.
46. M. L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Comm. Math. Sci., 1:471–500, 2003.
47. M. L. Minion, *Semi-Implicit Projection Methods for Incompressible Flow based on Spectral Deferred Corrections*, Appl. Numer. Math., 48 (3-4), 369-387, 2004.
48. N. Mohan, *First Course on Power Systems*, MNPERE, 2006.
49. L. Pareschi and G. Russo, *Implicit-Explicit Runge-Kutta schemes for stiff systems of differential equations*, volume 3, pages 269–287. Nova Science, 2000.
50. J. O. Pessanha and A. A. Paz, *Testing a differential-algebraic equation solver in Long-term Voltage Stability Simulation*, Mathematical Problems in Engineering, 2006.
51. V. Pereyra, *Iterated Deferred Correction for Nonlinear Boundary Value Problems*, Numer. Math. 11, 111–125 1968.
52. L. R. Petzold, *A Description of DASSL: A Differential-Algebraic System Solver*, SAND82-8637, Sandia National Lab, 1982.
53. A. Rangan, *Adaptive Solvers for Partial Differential and Differential-Algebraic Equations*, Ph.D. Thesis, University of California at Berkeley , 2003.
54. A. Rangan, *Deferred Correction Methods for Low Index Differential Algebraic Equations*, BIT, Vol.43, No.1,1-18, 2003.
55. Y. Saad, and M. H. Schultz. *GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems*, SIAM J. Sci. Stat. Comp., 7:856–869, 1986.
56. J. M. Sanz-Serna, J. G. Verwer, and W. H. Hundsdorfer, *Convergence and Order Reduction of Runge-Kutta Schemes Applied to Evolutionary Problems in Partial Differential Equations*, Numer. Math., 50, 405-418, 1986.
57. J. W. Shen and X. Zhong, *Semi-implicit Runge-Kutta schemes for the non-autonomous differential equations in reactive flow computations*, Proceedings of the 27th AIAA Fluid Dynamics Conference, AIAA, June 1996.
58. L. N. Trefethen, and M. R. Trummer, *An instability phenomenon in spectral methods*, SIAM J. Numer. Anal, 24, 1008-1023, 1987
59. P. K. Vijalapura, J. Strain, and S. Govindjee, *Fractional step methods for index-1 differential-algebraic equations*, J. of Comp. Phys., 203(1), 305-320, 2005.
60. D. Yong, V. Ajjarapu, *A Decoupled Time-Domain Simulation Method via Invariant Subspace Partition for Power System Analysis*, Power Systems, IEEE Transactions on On page(s): 11- 18, Volume: 21, Issue: 1, Feb. 2006
61. P. E. Zadunaisky, *A method for the estimation of errors propagated in the numerical solution of a system of ordinary differential equations*, The Theory of Orbits in the Solar System and in Stellar Systems. Proceedings of International Astronomical Union, Symposium 25, 1964.
62. P. E. Zadunaisky, *On the Estimation of Errors Propagated in the Numerical Integration of Ordinary Differential Equations*, Numer. Math. 27, 21–40 1976.

Department of Mathematics, University of North Carolina, CB # 3250, Phillips Hall, Chapel Hill NC 27599-3250, USA
E-mail address: `agatha@email.unc.edu`

Department of Mathematics, University of North Carolina, CB # 3250, Phillips Hall, Chapel Hill NC 27599-3250, USA
E-mail address: `huang@email.unc.edu`

Department of Mathematics, University of North Carolina, CB # 3250, Phillips Hall, Chapel Hill NC 27599-3250, USA
E-mail address: `minion@email.unc.edu`