On the Use of Half-Implicit Numerical Integration in Multibody Dynamics

Luning Fang
Department of Mechanical Engineering,
University of Wisconsin-Madison,
Madison, WI 53706
e-mail: lfang9@wisc.edu

Alexandra Kissel
Department of Mechanical Engineering,
University of Wisconsin-Madison,
Madison, WI 53706
e-mail: akissel@wisc.edu

Ruochun Zhang
Department of Mechanical Engineering,
University of Wisconsin-Madison,
Madison, WI 53706
e-mail: rzhang294@wisc.edu

Dan Negrut[1]
Department of Mechanical Engineering,
University of Wisconsin-Madison,
Madison, WI 53706
e-mail: negrut@wisc.edu

This work highlights the use of half-implicit numerical integration in the context of the index three differential algebraic equations (DAEs) of multibody dynamics. Although half-implicit numerical integration is well established for ordinary differential equations problems, to the best of our knowledge, no formal discussion covers its use in the context of index three DAEs of multibody dynamics. We wish to address this since when compared to fully implicit methods, half-implicit integration has two attractive features: (i) the solution method does not require the computation of the Jacobian associated with the constraint, friction, contact, or user-defined applied forces; and (ii) the solution is simpler to implement. Moreover, for nonstiff problems, half-implicit numerical integration yields a faster solution. Herein, we outline the numerical method and demonstrate it in conjunction with three mechanisms. We report on convergence order behavior and solution speed. The Python software developed to generate the results reported is available as open in a public repository for reproducibility studies. [DOI: 10.1115/1.4056183]

1 Introduction

Complex multibody systems, e.g., cars, rovers, space structures and robots, often lead to multibody models with topologies that display closed loops. When using absolute Cartesian generalized coordinates, the numerical solution of the index three differential algebraic equations (DAEs) problem associated with these systems has relied on implicit numerical integrators, see, for instance [1–4]; for an overview of the topic, see Ref. [5]. The contribution made herein is the use of half-implicit methods to directly solve the index three DAEs problem; i.e., produce a solution that satisfies the kinematic constraint equations while being compatible with a discretized form of the equations of motion. In the most simple case of a half-implicit integration formula, and the most simple case of a one degree-of-freedom system, the position x_{n+1} and velocity v_{n+1} are obtained at time t_{n+1} from the acceleration a_n as

$$v_{n+1} = v_n + h a_n \tag{1a}$$

$$x_{n+1} = x_n + h v_{n+1} \tag{1b}$$

This is the half-implicit, symplectic Euler method [6]; it is explicit at the velocity level (level one "L1"), but implicit at the position level (level zero "L0"); and, it is a first-order method, but higher order methods can be obtained by combining a suitable explicit integrator for the L1 update, with a suitably chosen implicit integrator for the L0 update. Higher order integrators fall outside the scope of this technical note. Our contribution is with outlining how integrators like the one in Eq. (1) can be used to directly solve the index three DAEs of multibody dynamics.

Several attractive aspects are noted when using a half-implicit solver to handle the index three DAEs of multibody dynamics. First, when using in conjunction with the Absolute Nodal Coordinate Formulation [7], the Jacobian of the internal forces does not need to be evaluated. Additionally, if no kinematic constraints are present in the system, the mass matrix is constant and only needs to be factored once and used throughout the simulation to compute the analog of a_n in Eq. (1). Second, the solution does not require the Jacobian of some hard-to-handle forces that might appear in the model, e.g., friction, contact, impact, constraint, user-defined. The latter are particularly difficult to handle, since in many cases user-defined forces do not have closed-form expression and are provided as a black box, in a user subroutine. Finally, the solution process for half-implicit integrators is straightforward. The Jacobian is easy to assemble and has a sparsity pattern that can be leveraged for linear-time solution [8,9].

This Technical Note is organized as follows. Section 2.1 states the index three DAEs problem of interest; Sections 2.2 and 2.3 outline the half- and fully-implicit schemes, respectively; Section 3 reports convergence and timing results. We close with a discussion of the main highlights and directions of future work.

2 Solution Approach

Herein, we assume the use of absolute coordinates. The location, $\boldsymbol{r}$, of each body is with respect to a global reference frame. The orientation, or pose, of each rigid body is also with respect to the global reference frame, captured by Euler angles $\boldsymbol{\varepsilon} \equiv [\phi - \theta - \psi]^{\mathrm{T}}$ or Euler parameters (quaternions) $\boldsymbol{p} \equiv [e_0, e_1, e_2, e_3]^{\mathrm{T}}$. However, herein we use Lie-group integration to produce the orientation matrix $\boldsymbol{A}$ directly, since this approach is more attractive than using Euler parameters or Euler angles [10]. The discussion herein focuses on rigid body dynamics but the conclusions carry over to absolute nodal coordinate formulation (ANCF)-based flex body dynamics. Finally, the local reference frame (LRF) associated with each body is assumed to be central and principal, so that the generalized mass matrix is constant and diagonal.

2.1 The Constrained Equations of Motion.

The state of a system of nb rigid bodies, each of mass m_i and inertia matrix $\bar{\boldsymbol{J}}_i$, can be described by translational and angular acceleration, $\ddot{\boldsymbol{r}}_i$ and $\dot{\bar{\boldsymbol{\omega}}}_i$; velocity, $\dot{\boldsymbol{r}}_i$ and $\bar{\boldsymbol{\omega}}_i$; and position $\boldsymbol{r}_i$ and orientation matrix $\boldsymbol{A}_i$, where $i = 1, 2, \ldots, \mathrm{nb}$. Herein, a letter with a bar $\bar{\cdot}$ represents a geometric quantity expressed in the body's LRF.

For a constrained problem that describes the relative motion among bodies connected via lower-pair joints, e.g., revolute joint, spherical joint and translational joint, the state of the system needs to satisfy a set of nonlinear equations, described as $\boldsymbol{\Phi}(\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_{\mathrm{nb}}, \boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_{\mathrm{nb}}, t) = \boldsymbol{0}_{\mathrm{nc}}$, where nc is the total number of constraints, and t is time. By applying D'Alembert's principle, one can derive the constrained equations of motion [11] in matrix form as

$$\boldsymbol{M}\ddot{\boldsymbol{r}} + \boldsymbol{\Phi}_{,\boldsymbol{r}}^{T}\boldsymbol{\lambda} = \boldsymbol{f} \tag{2a}$$

[1]Corresponding author.

Manuscript received July 26, 2022; final manuscript received October 31, 2022; published online November 17, 2022. Assoc. Editor: Adam Brink.

$$\bar{\boldsymbol{J}}\dot{\bar{\boldsymbol{\omega}}} + \bar{\boldsymbol{\Pi}}^T(\boldsymbol{\Phi})\boldsymbol{\lambda} = \bar{\boldsymbol{\tau}} \tag{2b}$$

$$\boldsymbol{\Phi}(\boldsymbol{r}, \boldsymbol{A}, t) = \mathbf{0}_{\text{nc}} \tag{2c}$$

where $\boldsymbol{M} \equiv \text{diag}\{m_1\mathbf{I}_3, \ldots, m_{\text{nb}}\mathbf{I}_3\}$, $\bar{\boldsymbol{J}} \equiv \text{diag}\{\bar{\mathbf{J}}_1, \ldots, \bar{\mathbf{J}}_{\text{nb}}\}$, $\boldsymbol{f} \equiv [\mathbf{f}_1^T, \ldots, \mathbf{f}_{\text{nb}}^T]^{\text{T}}$, and $\bar{\boldsymbol{\tau}} \equiv [\bar{\boldsymbol{\tau}}_1^T, \ldots, \bar{\boldsymbol{\tau}}_{\text{nb}}^T]^{\text{T}}$. Here, $\boldsymbol{M}\ddot{\boldsymbol{r}}$ and $\bar{\boldsymbol{J}}\dot{\bar{\boldsymbol{\omega}}}$ are the inertia force and torque, respectively; $\mathbf{f}_i$ is the external force on body i; the external torque $\bar{\boldsymbol{n}}_i$ appears in $\bar{\boldsymbol{\tau}}_i \equiv \bar{\boldsymbol{n}}_i - \tilde{\bar{\boldsymbol{\omega}}}_i\bar{\boldsymbol{J}}_i\bar{\boldsymbol{\omega}}_i$; $\boldsymbol{\Phi}_{,\boldsymbol{r}}^T\boldsymbol{\lambda}$ and $\bar{\boldsymbol{\Pi}}^T(\boldsymbol{\Phi})\boldsymbol{\lambda}$ are the reaction forces and torques from the joints, with Lagrange multipliers $\boldsymbol{\lambda} \in \mathbb{R}^{\text{nc}}$ associated with the kinematic constraints $\boldsymbol{\Phi}$, and $\boldsymbol{\Phi}_{,\boldsymbol{r}}$ and $\bar{\boldsymbol{\Pi}}(\boldsymbol{\Phi})$ being the first-order variations of $\boldsymbol{\Phi}$ with respect to $\boldsymbol{r}$ and $\boldsymbol{A}$. Note that in $\boldsymbol{\Phi}$, the orientation matrix $\boldsymbol{A}$ only shows up in the form of $\boldsymbol{A}\bar{\boldsymbol{s}}$, with $\bar{\boldsymbol{s}}$ being a vector in the local body frame. As shown in Ref. [10], $\bar{\boldsymbol{\Pi}}(\boldsymbol{\Phi})$ has cleaner formulations compared to taking the partial derivatives of $\boldsymbol{\Phi}$ with respect to Euler angles or Euler parameters.

2.2 The Half-Implicit Approach.

At time $t = t_n$, the discretized equations of motion are

$$\boldsymbol{M}\ddot{\boldsymbol{r}}_n + \mathbb{P}_{n,\boldsymbol{r}}^T(\boldsymbol{\Phi})\boldsymbol{\lambda}_n = \boldsymbol{f}_n \tag{3a}$$

$$\bar{\boldsymbol{J}}\dot{\bar{\boldsymbol{\omega}}}_n + \mathbb{P}_{n,\boldsymbol{A}}^T(\boldsymbol{\Phi})\boldsymbol{\lambda}_n = \bar{\boldsymbol{\tau}}_n \tag{3b}$$

where $\mathbb{P}_{,\boldsymbol{r}}(\boldsymbol{\Phi}) \equiv \boldsymbol{\Phi}_{,\boldsymbol{r}}$ and $\mathbb{P}_{,\boldsymbol{A}}(\boldsymbol{\Phi}) \equiv \bar{\boldsymbol{\Pi}}(\boldsymbol{\Phi})$. The half-implicit scheme becomes

$$\dot{\boldsymbol{r}}_{n+1} = \dot{\boldsymbol{r}}_n + h\ddot{\boldsymbol{r}}_n \tag{4a}$$

$$\bar{\boldsymbol{\omega}}_{n+1} = \bar{\boldsymbol{\omega}}_n + h\dot{\bar{\boldsymbol{\omega}}}_n \tag{4b}$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + h\dot{\boldsymbol{r}}_{n+1} \tag{4c}$$

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n \exp(h\tilde{\bar{\boldsymbol{\omega}}}_{n+1}) \tag{4d}$$

Here, Eq. (4d) is a first-order Lie-group integration scheme for the orientation matrix $\boldsymbol{A}$. From Eqs. (4a) and (4c)

$$\ddot{\boldsymbol{r}}_n = (\boldsymbol{r}_{n+1} - \boldsymbol{r}_n - h\dot{\boldsymbol{r}}_n)/h^2, \quad \dot{\bar{\boldsymbol{\omega}}}_n = (\bar{\boldsymbol{\omega}}_{n+1} - \bar{\boldsymbol{\omega}}_n)/h$$

which upon plugging back into Eq. (3) leads to the following nonlinear system of equations

$$\boldsymbol{M}\boldsymbol{r}_{n+1} + \mathbb{P}_{n,\boldsymbol{r}}^T\hat{\boldsymbol{\lambda}}_n + \boldsymbol{b}_n = \mathbf{0} \tag{5a}$$

$$\bar{\boldsymbol{J}}\bar{\boldsymbol{\theta}}_{n+1} + \mathbb{P}_{n,\boldsymbol{A}}^T\hat{\boldsymbol{\lambda}}_n + \boldsymbol{c}_n = \mathbf{0} \tag{5b}$$

$$\boldsymbol{\Phi}(\boldsymbol{r}_{n+1}, \boldsymbol{A}_{n+1}, t_{n+1}) = \mathbf{0} \tag{5c}$$

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n \exp(\tilde{\bar{\boldsymbol{\theta}}}_{n+1}) \tag{5d}$$

The unknowns are $\boldsymbol{r}_{n+1}, \bar{\boldsymbol{\theta}}_{n+1}, \boldsymbol{A}_{n+1}$, and $\hat{\boldsymbol{\lambda}}_n$, where

$$\bar{\boldsymbol{\theta}}_{n+1} \equiv h\bar{\boldsymbol{\omega}}_{n+1}, \quad \hat{\boldsymbol{\lambda}}_n \equiv h^2\boldsymbol{\lambda}_n \tag{6}$$

and both $\boldsymbol{b}_n$ and $\boldsymbol{c}_n$ are evaluated with quantities from the previous time-step t_n

$$\boldsymbol{b}_n \equiv -(\boldsymbol{M}\boldsymbol{r}_n + h\boldsymbol{M}\dot{\boldsymbol{r}}_n + h^2\boldsymbol{f}_n) \tag{7a}$$

$$\boldsymbol{c}_n \equiv h^2(\tilde{\bar{\boldsymbol{\omega}}}_n\bar{\boldsymbol{J}}\bar{\boldsymbol{\omega}}_n - \bar{\boldsymbol{n}}_n) - h\bar{\boldsymbol{J}}\bar{\boldsymbol{\omega}}_n \tag{7b}$$

To solve the nonlinear system in Eq. (5) using the Newton–Raphson approach, we need to evaluate the Jacobian matrix $\boldsymbol{N}^{(k)}$ for each iteration (k), and solve for the corrections $\delta\boldsymbol{q}$

$$\boldsymbol{N}^{(k)}\delta\boldsymbol{q} = \boldsymbol{e}^{(k)}, \quad \boldsymbol{e}^{(k)} \equiv \begin{bmatrix} -\boldsymbol{e}_1^{(k)} \\ -\boldsymbol{e}_2^{(k)} \\ -\boldsymbol{e}_3^{(k)} \end{bmatrix}, \quad \delta\boldsymbol{q} \equiv \begin{bmatrix} \delta\boldsymbol{r} \\ \delta\bar{\boldsymbol{\theta}} \\ \delta\hat{\boldsymbol{\lambda}} \end{bmatrix} \tag{8}$$

Then, at the next iteration $(k+1)$

$$\boldsymbol{r}_{n+1}^{(k+1)} = \boldsymbol{r}_{n+1}^{(k)} + \delta\boldsymbol{r} \tag{9a}$$

$$\bar{\boldsymbol{\theta}}_{n+1}^{(k+1)} = \bar{\boldsymbol{\theta}}_{n+1}^{(k)} + \delta\bar{\boldsymbol{\theta}} \tag{9b}$$

$$\hat{\boldsymbol{\lambda}}_n^{(k+1)} = \hat{\boldsymbol{\lambda}}_n^{(k)} + \delta\hat{\boldsymbol{\lambda}} \tag{9c}$$

$$\boldsymbol{A}_{n+1}^{(k+1)} = \boldsymbol{A}_n \exp\left(\tilde{\bar{\boldsymbol{\theta}}}_{n+1}^{(k+1)}\right) \tag{9d}$$

We derive the Jacobian matrix as

$$\boldsymbol{N}^{(k)} = \begin{bmatrix} \boldsymbol{M} & \mathbf{0} & \mathbb{P}_{n,\boldsymbol{r}}^T(\boldsymbol{\Phi}) \\ \mathbf{0} & \bar{\boldsymbol{J}} & \mathbb{P}_{n,\boldsymbol{A}}^T(\boldsymbol{\Phi}) \\ \mathbb{P}_{n+1,\boldsymbol{r}}^{(k)}(\boldsymbol{\Phi}) & \mathbb{P}_{n+1,\boldsymbol{A}}^{(k)}(\boldsymbol{\Phi}) & \mathbf{0} \end{bmatrix} \tag{10}$$

and the residual as

$$\boldsymbol{e}_1^{(k)} = \boldsymbol{M}\boldsymbol{r}_{n+1}^{(k)} + \mathbb{P}_{n,\boldsymbol{r}}^T(\boldsymbol{\Phi})\hat{\boldsymbol{\lambda}}_n^{(k)} + \boldsymbol{b}_n \tag{11a}$$

$$\boldsymbol{e}_2^{(k)} = \bar{\boldsymbol{J}}\bar{\boldsymbol{\theta}}_{n+1}^{(k)} + \mathbb{P}_{n,\boldsymbol{A}}^T(\boldsymbol{\Phi})\hat{\boldsymbol{\lambda}}_n^{(k)} + \boldsymbol{c}_n \tag{11b}$$

$$\boldsymbol{e}_3^{(k)} = \boldsymbol{\Phi}(\boldsymbol{r}_{n+1}^{(k)}, \boldsymbol{A}_{n+1}^{(k)}, t_{n+1}) \tag{11c}$$

Instead of evaluating $\boldsymbol{N}^{(k)}$ at each iteration, computation cost is reduced by using the approximations $\mathbb{P}_{n+1,\boldsymbol{r}}^{(k)} \approx \mathbb{P}_{n,\boldsymbol{r}}$ and $\mathbb{P}_{n+1,\boldsymbol{A}}^{(k)} \approx \mathbb{P}_{n,\boldsymbol{A}}$, to yield a quasi-Newton symmetric matrix computed only once at each time-step,

$$\boldsymbol{N} = \begin{bmatrix} \boldsymbol{M} & \mathbf{0} & \mathbb{P}_{n,\boldsymbol{r}}^T(\boldsymbol{\Phi}) \\ \mathbf{0} & \bar{\boldsymbol{J}} & \mathbb{P}_{n,\boldsymbol{A}}^T(\boldsymbol{\Phi}) \\ \mathbb{P}_{n,\boldsymbol{r}}(\boldsymbol{\Phi}) & \mathbb{P}_{n,\boldsymbol{A}}(\boldsymbol{\Phi}) & \mathbf{0} \end{bmatrix} \tag{12}$$

Then, $\boldsymbol{N}\delta\boldsymbol{q} = \boldsymbol{e}^{(k)}$ is solved iteratively until the correction $\delta\boldsymbol{q}$ meets the stopping criteria $||\delta\boldsymbol{q}|| < \epsilon_{tol}$. Once the solver converges, the solution $\boldsymbol{\lambda}_n$ is plugged back in Eq. (3) to compute the accelerations $\ddot{\boldsymbol{r}}_n$ and $\dot{\bar{\boldsymbol{\omega}}}_n$. Note that the Jacobians $\mathbb{P}_{n,\boldsymbol{r}}$ and $\mathbb{P}_{n,\boldsymbol{A}}$ in Eq. (12) are derived for all lower-pair joints in Ref. [10]. Thus, the solution approach can be effectively used in conjunction with any constrained multibody system.

2.3 The Fully-Implicit Approach.

The fully-implicit integration scheme used is described in Ref. [10], and only summarily presented here. At time $t = t_{n+1}$, the discretized constrained equations of motion

$$\boldsymbol{M}\ddot{\boldsymbol{r}}_{n+1} + \mathbb{P}_{n+1,\boldsymbol{r}}^T\boldsymbol{\lambda}_{n+1} = \boldsymbol{f}_{n+1} \tag{13a}$$

$$\bar{\boldsymbol{J}}\dot{\bar{\boldsymbol{\omega}}}_{n+1} + \mathbb{P}_{n+1,\boldsymbol{A}}^T\boldsymbol{\lambda}_{n+1} = \bar{\boldsymbol{\tau}}_{n+1} \tag{13b}$$

$$\frac{1}{h^2}\boldsymbol{\Phi}(\boldsymbol{r}_{n+1}, \boldsymbol{A}_{n+1}, t_{n+1}) = \mathbf{0} \tag{13c}$$

are solved iteratively using the Newton–Raphson method to produce $\ddot{\boldsymbol{r}}_{n+1}$, $\dot{\bar{\boldsymbol{\omega}}}_{n+1}$, and $\boldsymbol{\lambda}_{n+1}$. The scaling factor $1/h^2$ in Eq. (13c) is for improving the condition number of the Newton iteration matrix. The velocity- and position-level quantities are updated using the Backward Euler scheme (compare Eqs. (4a) and (4b) to (14a) and (14b)),

$$\dot{\boldsymbol{r}}_{n+1} = \dot{\boldsymbol{r}}_n + h\ddot{\boldsymbol{r}}_{n+1} \tag{14a}$$

$$\bar{\boldsymbol{\omega}}_{n+1} = \bar{\boldsymbol{\omega}}_n + h\dot{\bar{\boldsymbol{\omega}}}_{n+1} \tag{14b}$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + h\dot{\boldsymbol{r}}_{n+1} \tag{14c}$$

$$\boldsymbol{A}_{n+1} = \boldsymbol{A}_n \exp(h\tilde{\bar{\boldsymbol{\omega}}}_{n+1}) \tag{14d}$$

When assembling the Newton iteration matrix for solving Eq. (13), the Jacobians of the reaction forces and torques, $\mathbb{P}^T_{n+1,\mathbf{r}}\boldsymbol{\lambda}_{n+1}$ and $\mathbb{P}^T_{n+1,\mathbf{A}}\boldsymbol{\lambda}_{n+1}$, with respect to $\mathbf{r}$, $\dot{\mathbf{r}}$, $\mathbf{A}$ and $\bar{\boldsymbol{\omega}}$ are dropped to improve the solver's efficiency.

3 Numerical Experiments

The software implemented to generate the results reported herein is available in a public repository for reproducibility studies and further development [12]. Three systems: a double pendulum, a slider–crank, and a four–link mechanism are investigated to compare the performance of the fully- and half-implicit approaches.

The double pendulum is composed of 4 m and 2 m long rectangular bars, connected via revolute joints. The system is released from the configuration shown in Fig. 1(*a*) and swings owing to the gravitational pull. Both bars have the same square-shaped cross section and density of 7.8 g cm^{-3}.

The slider–crank and four–link systems are kinematically driven. The mass and inertia properties of the slider–crank mechanism are in Table 1. We prescribe the crank (body 1 in Fig. 1(*b*)) to rotate about its negative x-axis at a rate of 2π rad s^{-1}. The four–link setup consists of a rotor disk and two bars of the same square-shaped cross section. The density of each body is 1.5 g cm^{-3}. The rotor (body 1 in Fig. 1(*c*)) is prescribed to rotate about its z-axis at a rate of π rad s^{-1}. More parameters are given in Table 2. All simulations were run for 8 s, which allowed for at least three full rotations for the slider–crank and four–link mechanisms.

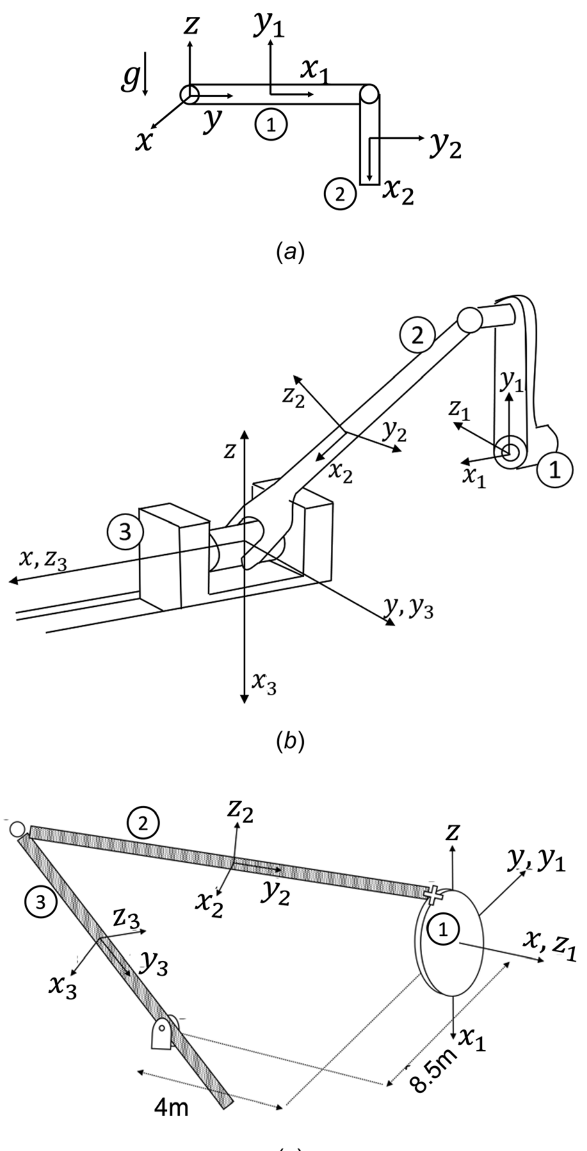


Fig. 1 Mechanisms used in the numerical experiments: (*a*) double pendulum (*b*) slider–crank (*c*) four–link

Table 1 Mass and inertia properties of the slider–crank mechanism (SI units), adopted from Haug [11]

Body	Mass	$I_{\bar{x}\bar{x}}$	$I_{\bar{y}\bar{y}}$	$I_{\bar{z}\bar{z}}$
1	0.12	1×10^{-4}	1×10^{-5}	1×10^{-4}
2	0.5	4×10^{-3}	4×10^{-4}	4×10^{-3}
3	2	1×10^{-4}	1×10^{-4}	1×10^{-4}

Table 2 Properties of the four–link mechanism (SI units).

Rotor radius	Bar 2 length	Bar 3 length	Bar width
2	12.2	7.4	0.1

3.1 Convergence Order Analysis.

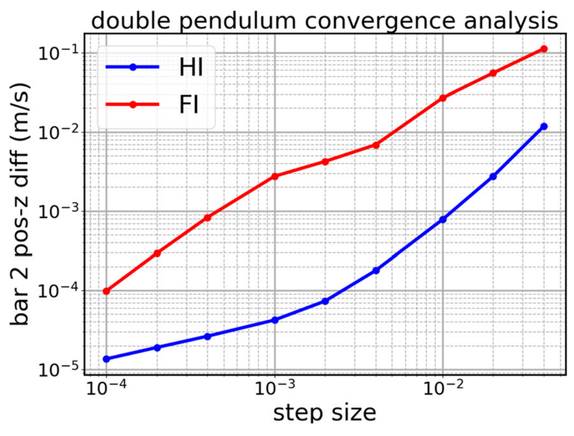
The convergence order analysis starts with producing a reference solution. The position, velocity, and acceleration of the double pendulum were obtained by integrating a set of ordinary differential equations that describe the system in 2D using two coordinates. To that end, we use a highly accurate, 8th-order Runge–Kutta solver – DoPri853 [13]—and a constant step size of $h = 1\times10^{-6}$ s. For the slider–crank and four–link mechanisms, the reference solution was generated by posing the problem as a kinematic one. To this end, only Eq. (2*c*) and its time derivatives were solved using the Newton–Raphson method, which allowed for the reference solution to be obtained within machine precision.

Figure 2 shows on a log–log scale the difference between the fully- and half-implicit integrators and the reference solution plotted over a range of step sizes. Labels "HI" and "FI" stand for "Half Implicit" and "Fully Implicit," respectively. The tolerance of the fully-implicit approach is set to $\epsilon_{\text{tol}} = 10^{-10}/h^2$; for the half-implicit, 10^{-10}. The scaling factor $1/h^2$ is from Eq. (13*c*), in comparison to Eq. (2*c*). The error relative to the reference solution was calculated as $\sqrt{\sum_{i=1}^{n}\left(z_i^{\text{DAE}} - z_i^{\text{ref}}\right)^2/n}$, where n is the total number of time steps in each simulation and z_i^{DAE} and z_i^{ref} are quantities at time-step t_i produced by the DAE solvers and the reference solution, respectively.

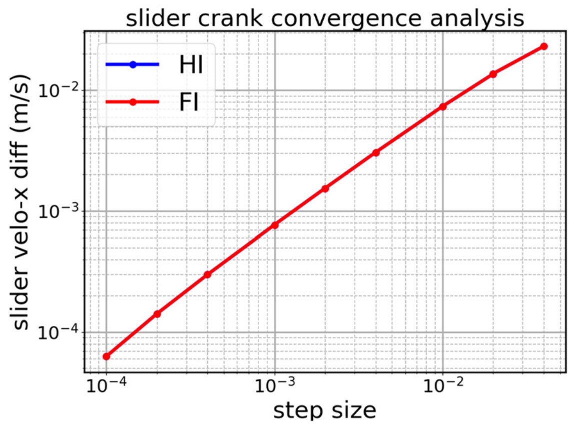
Figure 2(*a*) plots the z-position error for bar 2. The most important feature noted is a slope of 1.0 for both integrators. In other words, if the step size is reduced by a factor of 10, the error will decrease by a factor of 10. Figures 2(*b*) and 2(*c*) plot the velocity order analysis for the slider of the slider–crank mechanism and link body 2 of the four–link mechanism, respectively. All three figures show first-order convergence for both methods, thus meeting expectations. Finally, since the half-implicit integrator is symplectic, it conserves a numerical Hamiltonian, which leads to superior performance in terms of long-term energy preservation [6]. This is evident in Fig. 3—the total energy of the system drains over time using a fully-implicit approach, a highly-damped Backward Euler integrator, but holds steady for the symplectic half-implicit approach.

3.2 Performance Analysis.

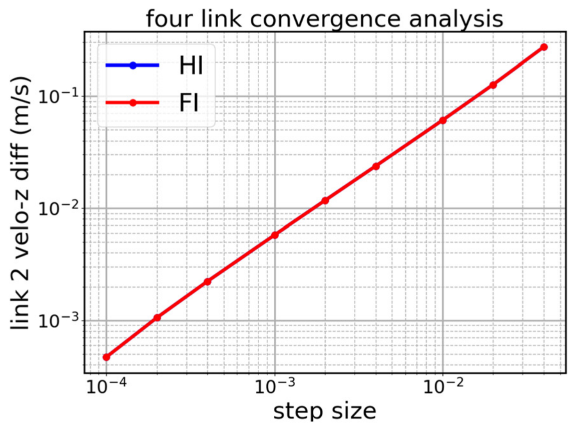
For a fair timing comparison, each integrator had its own ϵ_{tol} to ensure that they achieved similar quality of the solution (relative to the reference solution). More plots of how ϵ_{tol} affects the solution quality are available in Ref. [14]. Note that both solvers use SciPy's built-in sparse solver.



(*a*)



(*b*)



(*c*)

Fig. 2 Order analysis, half-implicit (HI) and fully-implicit solutions (FI): (*a*) Double pendulum, (*b*) slider–crank, and (*c*) four–link

Each timing test was run five times and the average CPU time was recorded for three different step sizes, $h = 1 \times 10^{-4}, 1 \times 10^{-3}$, and 1×10^{-2} s, as shown in Figs. 4(*a*), 5(*a*), and 6(*a*). The average number of iterations for the Newton solver to converge is illustrated in Figs. 4(*b*), 5(*b*), and 6(*b*). A larger step size results in more iterations because the initial starting point for the Newton method is less accurate. The half-implicit approach is faster for all mechanisms and step sizes. This is a direct consequence of the fact that the Newton iteration matrix is easier to evaluate and

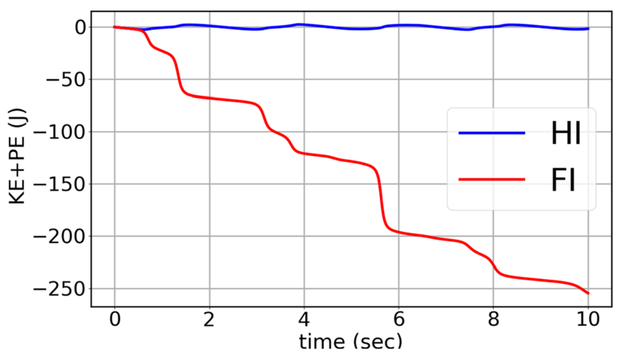


Fig. 3 Energy of double pendulum system over time

more accurate. Indeed, the Newton iteration matrix of the fully-implicit approach requires the computation of the partial derivative of the constraint reaction forces and torques, yet in practice these partial derivatives are ignored since they are exceedingly expensive to evaluate. However, for the half-implicit approach, the constraint reaction forces and torques are expressed at time-step t_n, independent of the position or orientation at t_{n+1}, resulting in a simpler and better approximation of the Newton iteration matrix.

3.3 Scaling Analysis. A scaling analysis was carried out to confirm that the speed-up observed in Sec. 3.2 translates to larger problems. As shown in Fig. 7, N rigid links were connected by spherical joints and released under gravity $\boldsymbol{g}$ pointing in a random direction (not vertically down). At time $t = 0$, the bars were assigned an angular velocity of random magnitude ω_i around their longitudinal axis. Tests were conducted for $N = 1, 2, 4, 6, 8, 16, 32$. The timing results reported in Fig. 8 indicate that the half-implicit approach is consistently faster than the fully-implicit one.

3.4 An Example With Joint Friction. To demonstrate the versatility of the half-implicit scheme, the slider-crank model is enhanced to include joint friction; thus, the external force $\boldsymbol{f}$ in Eq. (2*a*) changes over time. Herein, Coulomb friction is applied in the translational joint between the slider and the ground. The direction of the friction force opposes the relative motion of the slider with respect to the ground; its magnitude is the product of the friction coefficient, μ, and the reaction force in the normal-to-the-motion direction (herein, the z direction). No distinction is made between static and kinetic friction. Keeping with the half-implicit nature of the discretization scheme, the friction force at the current time-step is evaluated explicitly based on the joint normal force from the previous step. Three simulations were carried out using a prescribed motion of the crank, but various friction coefficients – $\mu = 0$, 0.2, and 0.4. In all cases the step size was 1×10^{-3} s. Figure 9 shows the torques required to enforce the prescribed crank motion, in each of the three cases. With increasing μ (and thus joint friction force), the driving torque used to drive the crank experiences larger swings. The sudden jump in the driving torque value results from the change of the slider velocity direction (and that of the friction force).

4 Discussion

The half-implicit integrator for index three DAEs advances the simulation from t_n to t_{n+1} by computing at t_n a set of Lagrange multipliers $\boldsymbol{\lambda}_n$ that enforce the kinematic constraints equations at t_{n+1}; i.e., $\boldsymbol{\Phi}(\boldsymbol{r}_{n+1}, \boldsymbol{A}_{n+1}, t_{n+1}) = \boldsymbol{0}$. With $\boldsymbol{\lambda}_n$ in hand, the acceleration at t_n can be obtained by solving a linear system of equations whose matrix is diagonal and constant throughout the simulation. Once the acceleration at t_n is known, one uses an explicit integrator to compute the velocities at t_{n+1} (see Eqs. (4*a*) and (4*b*)). The

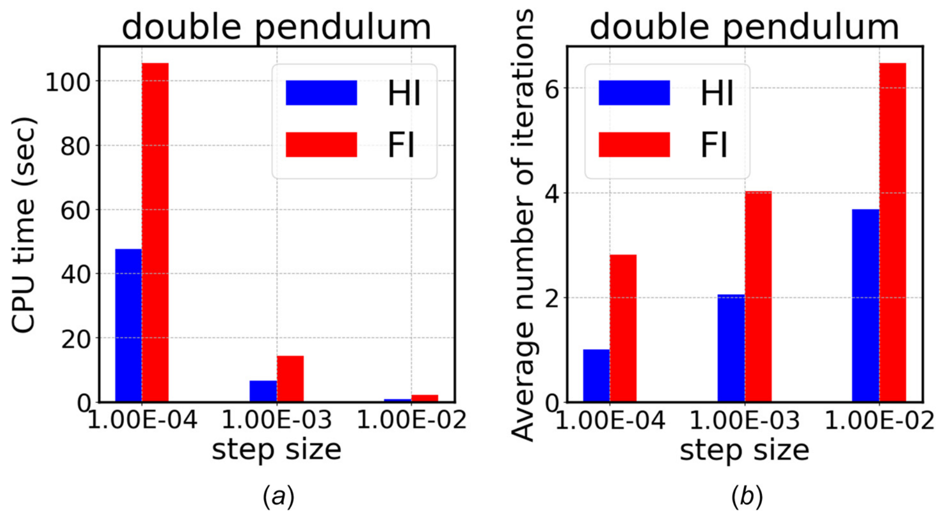


Fig. 4 Double pendulum: (*a*) CPU time and (*b*) number of iterations

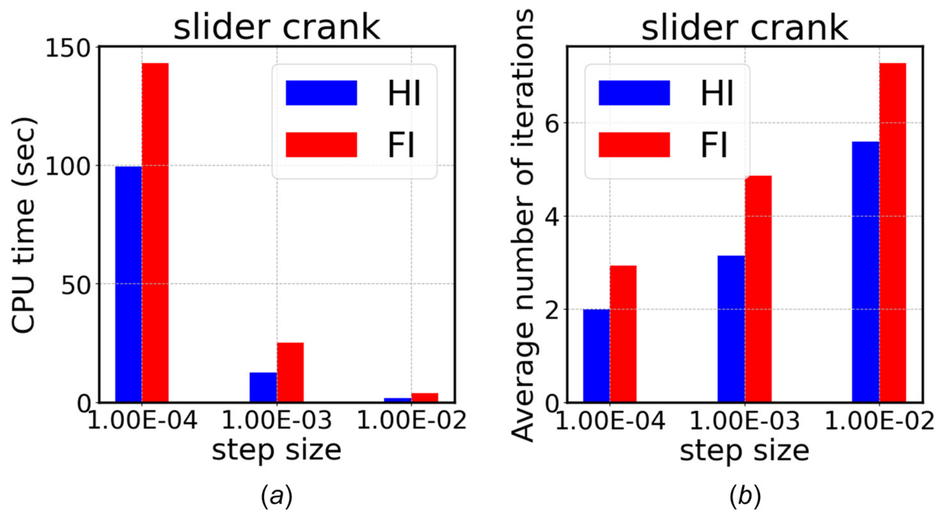


Fig. 5 Slider–crank mechanism: (*a*) CPU time and (*b*) number of iterations

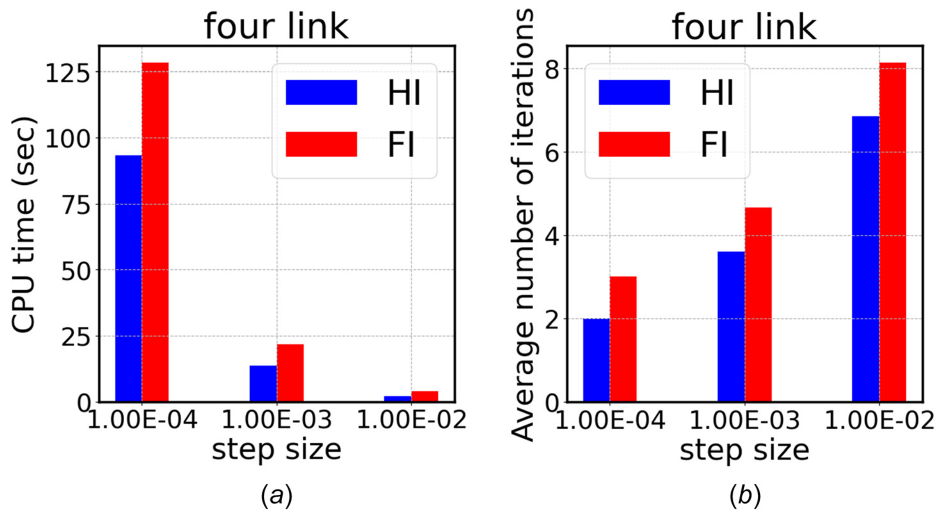


Fig. 6 Four–link mechanism: (*a*) CPU time and (*b*) number of iterations

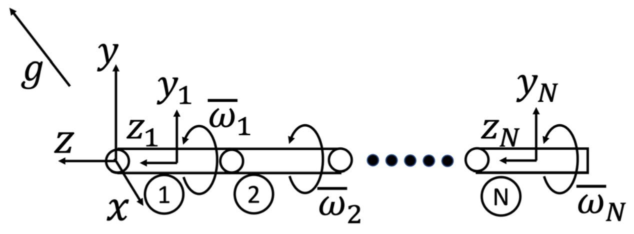


Fig. 7 Problem setup, scaling analysis

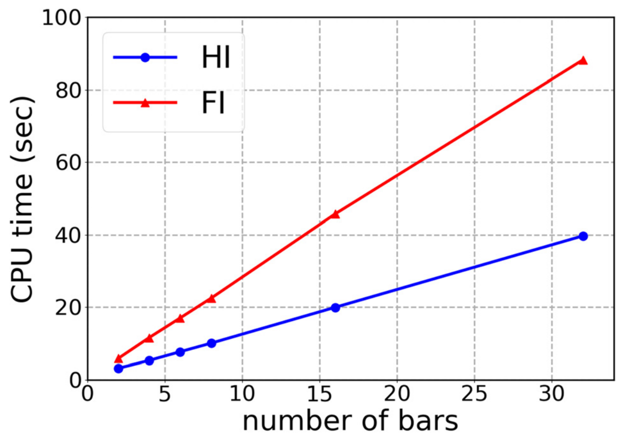


Fig. 8 Scaling analysis results

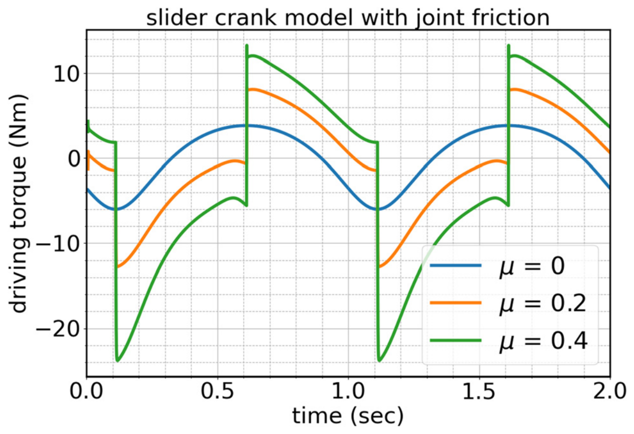


Fig. 9 Driving torque applied to the crank with and without friction at the translational joint of the slider

new position and orientation can be computed using an implicit formula with the new velocities (see Eqs. (4c) and (4d)).

The method works for arbitrary multibody systems: with open and closed loops, arbitrary number of bodies, any collection of joints (lower-pair constraints) or bilateral kinematic constraints. Although not discussed herein, the presence of nonholonomic constraints poses no additional challenges, as derived in Sec. 3.4 in Ref. [14]. In terms of the convergence of the Newton solver, the half-implicit method performs slightly better than the fully-implicit one. This can be traced back to the fact that some entries in the Jacobian matrix of the fully-implicit integrator were dropped—they are both expensive to compute and challenging to produce, e.g., Jacobian of constraint forces, friction forces and contact forces. Should one use the exact Jacobian, convergence can improve; however, the fully-implicit solver will be computationally more expensive. Note that since the half-implicit formulation uses reaction forces and torques from the previous time-step, $\mathbb{P}^T_{n,r}\boldsymbol{\lambda}_n$ and $\mathbb{P}^T_{n,A}\boldsymbol{\lambda}_n$, it does not require the Jacobian of external or internal generalized forces. Consequently, a simpler yet more accurate Newton matrix is obtained, leading to faster convergence.

Finally, the half-implicit approach is expected to work equally well for the index three DAEs problem formulated using Euler angles or Euler parameters as generalized coordinates. However, as shown in Ref. [10], the fully-implicit ***rA*** approach is faster than these alternatives, which justified the choice for Lie-integration in this technical note.

5 Conclusions and Future Work

We discuss the use of a half-implicit integration method for the solution of the index three DAEs of multibody dynamics. The approach stands to benefit the simulation of multibody systems that are not exceedingly stiff. In the case of stiff problems, a fully implicit DAE solution is expected to be superior. We demonstrate the half-implicit approach in conjunction with a first-order Lie-group integrator, yet the method can be equally applied to solutions that rely on Euler parameters or Euler angles. The half-implicit scheme treats the reaction forces and torque explicitly in time, leading to a simpler and better approximation of the Jacobian matrix in the Newton step and a more expeditious solution. The scaling analysis demonstrated that the conclusion above carries to larger systems. Finally, the numerical solution is straightforward to implement, see Ref. [12].

There are two things that remain to be investigated. First, provide a formal proof that the half-implicit approach has a guaranteed convergence order when used in the context of index three DAEs of multibody dynamics. Second, we are currently working on a second-order half-implicit integration formula that relies on Lie-group integration in multibody dynamics. Adaptive step size should be considered to further improve the efficiency of the numerical solution.

Funding Data

- Army Research Office (Award No. W911NF1910431; Funder ID: 10.13039/100000183).
- Directorate for Computer and Information Science and Engineering (Award No. CISE1835674; Funder ID: 10.13039/100000083).
- Division of Civil, Mechanical and Manufacturing Innovation (Award ID: CMMI2153855; Funder ID: 10.13039/100000147).
- Office of Cyber infrastructure (Award No. OAC2209791; Funder ID: 10.13039/100000105).

References

[1] Orlandea, N., Chace, M. A., and Calahan, D. A., 1977, "A Sparsity-Oriented Approach to the Dynamic Analysis and Design of Mechanical Systems – Part I and Part II," ASME Trans. ASME J. Eng. Ind., **99**(3), pp. 773–779.
[2] Negrut, D., Rampalli, R., Ottarsson, G., and Sajdak, A., 2007, "On the Use of the HHT Method in the Context of Index 3 Differential Algebraic Equations of Multibody Dynamics," ASME J. Comput. Nonlinear Dyn., **2**, pp. 207–218.
[3] Arnold, M., and Brüls, O., 2007, "Convergence of the Generalized-α Scheme for Constrained Mechanical Systems," Multibody Syst. Dyn., **18**(2), pp. 185–202.
[4] Wieloch, V., and Arnold, M., 2021, "BDF Integrators for Constrained Mechanical Systems on Lie Groups," J. Comput. Appl. Math., **387**, p. 112517.
[5] Bauchau, O. A., and Laulusa, A., 2008, "Review of Contemporary Approaches for Constraint Enforcement in Multibody Systems," ASME J. Comput. Nonlinear Dyn., **3**(1), p. 011005.
[6] Hairer, E., Lubich, C., and Wanner, G., 2006, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Vol. 31, Springer Science & Business Media, Berlin/Heidelberg, Germany
[7] Shabana, A. A., and Yakoub, R. Y., 2001, "Three Dimensional Absolute Nodal Coordinate Formulation for Beam Elements: Theory," ASME J. Mech. Des., **123**(4), pp. 606–613.

[8] Baraff, D., 1996, “Linear-Time Dynamics Using Lagrange Multipliers,” Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, New Orleans, LO, Aug. 4–9, pp. 137–146.
[9] Serban, R., Negrut, D., Haug, E. J., and Potra, F. A., 1997, “A Topology-Based Approach for Exploiting Sparsity in Multibody Dynamics in Cartesian Formulation,” J. Struct. Mech., **25**(3), pp. 379–396.
[10] Taves, J., Kissel, A., and Negrut, D., 2022, “Constrained Multibody Kinematics and Dynamics in Absolute Coordinates: A Discussion of Three Approaches to Representing Rigid Body Rotation,” ASME J. Comput. Nonlinear Dyn., (in press)
[11] Haug, E. J., 1989, *Computer-Aided Kinematics and Dynamics of Mechanical Systems*, Vol.-I, Prentice Hall, Englewood Cliffs, NJ.
[12] Fang, L., Kissel, A., Zhang, R., and Negrut, D., “Models, Scripts, and Meta-Data: Index 3 DAE Half-Implicit Integration Implementation,” accessed Nov 10, 2022, https://github.com/uwsbel/public-metadata/tree/master/2022/HalfImplicit_JCND
[13] Hairer, E., Norsett, S., and Wanner, G., 2009, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer, Berlin.
[14] Fang, L., Kissel, A., Benatti, S., and Negrut, D., 2021, “Using Half-Implicit Numerical Integration to Solve the Equations of Constrained Multibody Dynamics,” Simulation-Based Engineering Laboratory, University of Wisconsin-Madison, Report No. TR-2021-13, accessed Nov 10, 2022, https://sbel.wisc.edu/wp-content/uploads/sites/569/2022/07/TR-2021-13.pdf