AN ITERATIVE METHOD FOR CONSTRAINED DYNAMIC PROBLEMS – A CASE STUDY

J. Wang[1], G. Hou[2], and C. Modnak[3]

[1]*Department of Mathematics and Statistics, Old Dominion University, Norfolk, VA 23529, USA*
Email: j3wang@odu.edu
[2]*Department of Mechanical and Aerospace Engineering, Old Dominion University, Norfolk, VA 23529, USA*
Email: ghou@odu.edu
[3]*Department of Mathematics and Statistics, Old Dominion University, Norfolk, VA 23529, USA*
Email: cmodn001@odu.edu

Received 6 May 2011; accepted 18 May 2011

ABSTRACT

We propose an iterative scheme to numerically solve a class of constrained dynamic problems in the form of differential-algebraic equations. This algorithm allows efficient decoupling of the solution procedure and can be combined with any differential equation solvers. We use a simple, yet nontrivial, example to demonstrate the application of this method. Through both mathematical analysis and numerical tests, we show that this iterative scheme achieves fast convergence and ensures an accurate and efficient solution procedure for such constrained dynamic problems.

Keywords: constrained dynamic problem, iterative algorithm, ODE solver

1 INTRODUCTION

In this paper, we are concerned with numerical calculation of constrained dynamic problems in the form of

$$\begin{bmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ D_1 & D_2 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \lambda \end{bmatrix} + \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \lambda \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ d \end{bmatrix} \tag{1}$$

Here x and y are vectors with the same or different dimensions, A_j and C_j $(j = 1, 2)$ are square matrices, and r_j $(j = 1, 2)$ are vectors, of corresponding dimensions; λ can be either a vector or scalar depending on the specific constraints, and B_j and D_j $(j = 1, 2)$ are corresponding matrices or vectors. We have used the double dots to denote the second derivative with respect to time. Equation (1) constitutes a typical system of differential-algebraic equations (Kunkel and Mehrmann 2006). Such problems arise in many areas of science and engineering such as multi-body dynamics (Haug 1992) and fluid-structure interactions (Unger *et al.* 2007). In addition, formulation similar to equation (1) can be found in the two-field structural domain decomposition methods (Park and Felippa 2000), as well as in the simultaneous analysis and

constrained optimization problems (Knoll and Keyes 2004). In many of these applications, λ typically represents the so-called Lagrange multipliers (Belegundu and Chandrupatla 1999) which reflect the nature of the constraints.

Difficulties of numerical study on these problems stem from the algebraic constraints which impose additional challenges to traditional ordinary differential equation (ODE) solvers. It has been reported (Hairer and Wanner 1996; Kunkel and Mehrmann 2006) that the numerical solution may easily deviate from the exact solution if the constraints are not carefully enforced during numerical integration. It is also observed that many differential-algebraic systems behave like highly stiff differential equations which demand special treatment; in particular, extremely small stepsizes are required if explicit time-marching methods (such as regular Runge-Kutta) are employed. Hence, it is commonly believed that such dynamic problems should be solved by implicit methods, among which the backward difference formulas (Ascher and Petzold 1998; Brenan *et al.* 1989) and the implicit Runge-Kutta methods (Roche 1989) have been the most successful numerical approaches to solve differential-algebraic equations. Based on these two methods and their variations, several commercial or freely-available numerical codes have been developed, including the DASSL (Brenan *et al.* 1989), RADAU5 (Hairer and Wanner 1996), and MEBDFDAE (Cash 2000). Nevertheless, these fully implicit methods in general make the computation expensive, especially when large-scale problems are encountered and high-order accuracy is needed. In addition, there are several more sophisticated, and more specialized, numerical methods developed for differential-algebraic equations (see, *e.g.*, Fox *et al.* 2000; Gear 1971; Rangan 2003). Stability issues of some of these numerical have also been investigated (see, *e.g.*, Arnold 1993; Prothero and Robinson 1974). We refer to (Cash 2003; Hairer and Wanner 1996) for comprehensive reviews on numerical methods for differential-algebraic equations.

In this paper, we introduce an iterative method which effectively overcomes some of the aforementioned limitations in numerical study of constrained dynamic problems, and which allows simple, traditional ODE solvers to be used with success. Essentially, this method decouples the computation of x and y, while explicitly enforces the constraints at each time step. The iterative scheme is *local* in the sense that the iterations are performed on each single step, say, from t_n to t_{n+1}, instead of the global time domain. This feature ensures fast convergence and high accuracy, and possesses the flexibility that it can be combined with any ODE solvers. Furthermore, this iterative approach allows different ODE solvers and even different meshes to be used for x and y, so as to meet their (usually different) computational requirements. In this paper, we will focus our attention on a relatively simple, yet nontrivial, example to demonstrate the analysis and application of this method. More general numerical formulation and convergence analysis will be presented in another study.

The remainder of the present paper is organized as follows. In Section 2, the iterative algorithm is presented and the procedure of implementation is described. In Section 3, an example is presented as a case study to illustrate the application of the proposed method, followed by a detailed convergence analysis in Section 4. Results from numerical simulation are presented in Section 5 to verify the algorithm and analysis. Finally, conclusions are drawn and some discussion is made in Section 6.

2 ITERATIVE ALGORITHM

The iterative method proposed in this paper is based on a natural splitting of the leading coefficient matrix in equation (1). Specifically, the leading terms (i.e., those with the second

derivatives of the unknowns) in equation (1) are decomposed as

$$\begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & B_2 \\ 0 & D_2 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \lambda \end{bmatrix} + \begin{bmatrix} 0 & 0 & B_1 \\ 0 & 0 & 0 \\ D_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \lambda \end{bmatrix} \tag{2}$$

Suppose that at $t = t_n$, the variables x, y as well as their time derivatives, and λ, are all known. We advance the solution from $t = t_n$ to $t = t_{n+1}$ through an iterative procedure presented below. The interval $[t_n\,,\,t_{n+1}]$ is partitioned into a set of subintervals. Assume the ith iterative solution is already obtained on $[t_n\,,\,t_{n+1}]$. At the $(i+1)$th iteration, we solve the system

$$\begin{bmatrix} A_1 & 0 & 0 \\ 0 & A_2 & B_2 \\ 0 & D_2 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}^{i+1} \\ \ddot{y}^{i+1} \\ \lambda^{i+1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & B_1 \\ 0 & 0 & 0 \\ D_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}^{i} \\ \ddot{y}^{i} \\ \lambda^{i+1} \end{bmatrix} + \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{i+1} \\ y^{i+1} \\ \lambda^{i+1} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ d \end{bmatrix}$$

which yields two separated equations,

$$\begin{bmatrix} A_2 & B_2 \\ D_2 & 0 \end{bmatrix} \begin{bmatrix} \ddot{y}^{i+1} \\ \lambda^{i+1} \end{bmatrix} + \begin{bmatrix} C_2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y^{i+1} \\ \lambda^{i+1} \end{bmatrix} = \begin{bmatrix} r_2 \\ d - D_1 \ddot{x}^i \end{bmatrix} \tag{3}$$

and

$$A_1\,\ddot{x}^{i+1} + C_1\,x^{i+1} \;=\; r_1 - B_1\lambda^{i+1} \tag{4}$$

Equations (3) and (4) have decoupled the computation of x and y so that they can be solved separately at each time step. This builds the ground for a local iterative procedure to advance the solution from $t = t_n$ to $t = t_{n+1}$. For $i > 0$, we carry out the following steps to obtain the solution at the $(i+1)$th iteration.

Step 1 Compute y^{i+1} and λ^{i+1} on $[t_n\,,\,t_{n+1}]$ by solving equation (3) with the initial conditions

$$y^{i+1}(t_n) \;=\; y(t_n)\,, \qquad \dot{y}^{i+1}(t_n) \;=\; \dot{y}(t_n) \tag{5}$$

The numerical solution of (3)(5) is then reported at $t = t_{n+1}$. We calculate the backward errors

$$\varepsilon_y^{i+1} \;=\; \big|\big|y^{i+1}(t_{n+1}) - y^i(t_{n+1})\big|\big| \tag{6}$$

and

$$\varepsilon_\lambda^{i+1} \;=\; \big|\big|\lambda^{i+1}(t_{n+1}) - \lambda^i(t_{n+1})\big|\big| \tag{7}$$

Step 2 Compute x^{i+1} on $[t_n\,,\,t_{n+1}]$ by solving equation (4) with the initial conditions

$$x^{i+1}(t_n) \;=\; x(t_n)\,, \qquad \dot{x}^{i+1}(t_n) \;=\; \dot{x}(t_n) \tag{8}$$

Then calculate the backward error

$$\varepsilon_x^{i+1} \;=\; \big|\big|x^{i+1}(t_{n+1}) - x^i(t_{n+1})\big|\big| \tag{9}$$

Step 3 Check the convergence. If

$$\max\left(\varepsilon_x^{i+1}\,,\,\varepsilon_y^{i+1}\,,\,\varepsilon_\lambda^{i+1}\right) \;\le\; \varepsilon_0 \tag{10}$$

where ε_0 is the given error tolerance, then the convergence has been achieved. Start the iterative procedure for the next time step, $n = n + 1$. Otherwise, set $i = i + 1$ and return to *Step 1*.

In summary, we propose an iterative method to numerically solve a class of constrained dynamic problems in the form of (1). This method offers the following advantages:

- The algorithm is straightforward to implement, and can be easily combined with any ODE solvers.

- The algorithm ensures the accuracy of the solution by explicitly enforcing the algebraic constraint *at each time step*. This point will be demonstrated by rigorous error analysis and numerical simulation results through our case study.

- The solution procedures for x and y are decoupled, which allows us to employ different ODE solvers and even different meshes for x and y . Interpolations can be applied in case the meshes for x and y do not coincide. Such a decoupling technique and the flexibility are important in many practical applications, where the equations for x and y have different computational requirements in terms of accuracy and efficiency and thus are best solved by respective disciplinary codes.

3 AN EXAMPLE

To demonstrate the application of the proposed iterative algorithm, we consider the following model problem as a case study:

$$\left(\begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{Bmatrix} - \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} - \begin{Bmatrix} 6 \\ 7 \end{Bmatrix} \right)^T \begin{Bmatrix} \delta x_1 \\ \delta x_2 \end{Bmatrix} = 0 \tag{11}$$

$$\left(\begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{Bmatrix} - \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} - \begin{Bmatrix} 10 \\ 4 \end{Bmatrix} \right)^T \begin{Bmatrix} \delta y_1 \\ \delta y_2 \end{Bmatrix} = 0 \tag{12}$$

for arbitrary δx_j and δy_j ($j = 1,\ 2$), but subject to the constraint

$$x_2 \ = \ 2y_1 \tag{13}$$

This problem can be regarded as a simplified modeling of a two-structure joint system where the structural properties (the Young's modulus and Poisson ratio; see Haug 1992) and external forcing strength are all set to idealized numbers. Initially the displacements and velocities are all zero; i.e., the initial conditions of this problem are given as

$$x_1(0) \ = \ \dot{x}_1(0) \ = \ 0\,, \qquad x_2(0) \ = \ \dot{x}_2(0) \ = \ 0 \tag{14}$$

$$y_1(0) \ = \ \dot{y}_1(0) \ = \ 0\,, \qquad y_2(0) \ = \ \dot{y}_2(0) \ = \ 0 \tag{15}$$

Based upon the Theorem of Lagrange Multipliers (Belegundu and Chandrupatla 1999), we obtain

$$\begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \end{Bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} + \begin{Bmatrix} 6 \\ 7 \end{Bmatrix} - \begin{Bmatrix} 0 \\ \lambda \end{Bmatrix} \tag{16}$$

$$\begin{bmatrix} 5 & 2 \\ 2 & 4 \end{bmatrix} \begin{Bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{Bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} + \begin{Bmatrix} 10 \\ 4 \end{Bmatrix} + \begin{Bmatrix} 2\lambda \\ 0 \end{Bmatrix} \tag{17}$$

with the constraints:

$$\ddot{x}_2(t) \ = \ 2\ddot{y}_1(t)\,, \qquad \dot{x}_2(t) \ = \ 2\dot{y}_1(t)\,, \qquad x_2(t) \ = \ 2y_1(t) \tag{18}$$

Using equations (16-18), we can assemble a differential-algebraic system in the form of

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 1 \\ 0 & 0 & 5 & 2 & -2 \\ 0 & 0 & 2 & 4 & 0 \\ 0 & 1 & -2 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{y}_1 \\ \ddot{y}_2 \\ \lambda \end{Bmatrix} = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ y_1 \\ y_2 \\ \lambda \end{Bmatrix} + \begin{Bmatrix} 6 \\ 7 \\ 10 \\ 4 \\ 0 \end{Bmatrix} \tag{19}$$

We will apply the numerical formulation (3) and (4) to solve equation (19). This means that we first solve the equation

$$\begin{bmatrix} 5 & 2 & -2 \\ 2 & 4 & 0 \\ -2 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \ddot{y}_1^{i+1} \\ \ddot{y}_2^{i+1} \\ \lambda^{i+1} \end{Bmatrix} - \begin{bmatrix} 1 & 2 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} y_1^{i+1} \\ y_2^{i+1} \\ \lambda^{i+1} \end{Bmatrix} = \begin{Bmatrix} 10 \\ 4 \\ -\ddot{x}_2^i \end{Bmatrix} \tag{20}$$

to update y^{i+1} and λ^{i+1} on $[t_n\,,\,t_{n+1}]$. Then we compute x^{i+1} by solving

$$\begin{bmatrix} 4 & 1 \\ 1 & 3 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1^{i+1} \\ \ddot{x}_2^{i+1} \end{Bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} x_1^{i+1} \\ x_2^{i+1} \end{Bmatrix} + \begin{Bmatrix} 6 \\ 7 \end{Bmatrix} - \begin{Bmatrix} 0 \\ \lambda^{i+1} \end{Bmatrix} \tag{21}$$

We note that equation (20) can be further simplified by employing an LU decomposition approach (Golub and Van Loan 1996). The details are provided in the Appendix. As a result of equations (72) and (73), the computation of y and λ can also be separated. Based on the procedure described in Section 2, the iterative scheme will be implemented by the following steps.

Step 1 Compute y^{i+1} on $[t_n\,,\,t_{n+1}]$ by solving equation (72),

$$\begin{cases} \ddot{y}_1^{i+1} &= \dfrac{1}{2}\ddot{x}_2^i \\ \ddot{y}_2^{i+1} &= -\dfrac{1}{2}y_2^{i+1} + 1 - \dfrac{1}{4}\ddot{x}_2^i \end{cases} \tag{22}$$

with the initial conditions (5). Note that $\ddot{x}_2^i$ is known from the previous iteration. Particularly, to start the first iteration with $i = 0$, an initial guess is made by setting

$$x^0(t) = x(t_n)\,, \qquad \ddot{x}^0(t) = \ddot{x}(t_n)\,, \qquad \text{for} \quad t_n \le t \le t_{n+1} \tag{23}$$

The solution of (22) and (5) is then reported at $t = t_{n+1}$. For $i > 0$, we calculate the backward error ε_y^{i+1} according to (6).

Step 2 The Lagrange multiplier λ^{i+1} for $t_n \le t \le t_{n+1}$ is updated using equation (73),

$$\lambda^{i+1} = \frac{1}{2}\,y_1^{i+1} + \frac{1}{2}y_2^{i+1} - 4 + \ddot{x}_2^i \tag{24}$$

For $i > 0$, we calculate the backward error $\varepsilon_\lambda^{i+1}$ according to (7).

Step 3 Based on equation (21), the value of x^{i+1} on $[t_n\,,\,t_{n+1}]$ can then be found by solving

$$\begin{Bmatrix} \ddot{x}_1^{i+1} \\ \ddot{x}_2^{i+1} \end{Bmatrix} = \frac{1}{11}\begin{bmatrix} 5 & 1 \\ 2 & 7 \end{bmatrix} \begin{Bmatrix} x_1^{i+1} \\ x_2^{i+1} \end{Bmatrix} + \begin{Bmatrix} 1 \\ 2 \end{Bmatrix} + \frac{1}{11}\begin{Bmatrix} 1 \\ -4 \end{Bmatrix} \lambda^{i+1} \tag{25}$$

with the initial conditions (8). The solution is then reported at $t = t_{n+1}$. For $i > 0$, we calculate the backward error ε_x^{i+1} according to (9).

Step 4 Check the convergence based on the criterion in (10).

4 CONVERGENCE ANALYSIS

In this section, we show the convergence of the proposed iterative method for the model problem through a careful error analysis. Let us define the error functions

$$\begin{aligned} e^i_{x_j}(t) &= x_j(t) - x^i_j(t) \\ e^i_{y_j}(t) &= y_j(t) - y^i_j(t) \\ e^i_{\lambda}(t) &= \lambda(t) - \lambda^i(t) \end{aligned}$$

for $j = 1,\ 2$ and $t \in [t_n\,,\, t_{n+1}]$. They measure the difference between the exact solution and the numerical solution at the ith iteration. We will focus our attention on numerical errors due to the iterative procedure. For convenience of discussion, we assume the numerical solution at $t = t_n$ is exact, i.e.,

$$e^i_{x_j}(t_n) = 0\,, \qquad \dot{e}^i_{x_j}(t_n) = 0 \tag{26}$$

$$e^i_{y_j}(t_n) = 0\,, \qquad \dot{e}^i_{y_j}(t_n) = 0 \tag{27}$$

for $j = 1,\ 2$ and $i = 0,\ 1,\ 2, \cdots$. To quantify the errors, we will employ the maximum norm, denoted by $||\cdot||$, for any continuous function f on the interval $[t_n\,,\, t_{n+1}]$,

$$||f|| = \max_{t_n \le t \le t_{n+1}} |f(t)|$$

Suppose that the ith iterative solution is known. At the $(i+1)$th iteration, The functions $e^{i+1}_{y_j}$ and e^{i+1}_{λ} satisfy the following differential-algebraic equations, according to equations (22) and (24),

$$\ddot{e}^{i+1}_{y_1} = \frac{1}{2}\,\ddot{e}^i_{x_2} \tag{28}$$

$$\ddot{e}^{i+1}_{y_2} = -\frac{1}{2}\,e^{i+1}_{y_2} - \frac{1}{4}\,\ddot{e}^i_{x_2} \tag{29}$$

$$e^{i+1}_{\lambda} = \frac{1}{2}\,e^{i+1}_{y_1} + \frac{1}{2}\,e^{i+1}_{y_2} + \ddot{e}^i_{x_2} \tag{30}$$

With the initial condition (27), the solution to equations (28)-(30) can be easily found as

$$e^{i+1}_{y_1} = \frac{1}{2}\,e^i_{x_2} \tag{31}$$

$$e^{i+1}_{y_2} = -\frac{1}{4}\,e^i_{x_2} + \frac{1}{4\sqrt{2}} \int_{t_n}^{t} e^i_{x_2}(\tau)\,\sin\big(\frac{t-\tau}{\sqrt{2}}\big)\,d\tau \tag{32}$$

$$e^{i+1}_{\lambda} = \frac{1}{8}\,e^i_{x_2} + \ddot{e}^i_{x_2} + \frac{1}{8\sqrt{2}} \int_{t_n}^{t} e^i_{x_2}(\tau)\,\sin\big(\frac{t-\tau}{\sqrt{2}}\big)\,d\tau \tag{33}$$

Note that

$$\sqrt{2}\int_{t_n}^{t} \sin\big(\frac{t-\tau}{\sqrt{2}}\big)\,d\tau = 1 - \cos\big(\frac{t-t_n}{\sqrt{2}}\big) = \frac{1}{2!}\,\frac{1}{2}\,(t-t_n)^2 - \frac{1}{4!}\,\frac{1}{4}\,(t-t_n)^4 + \cdots$$

We thus can easily obtain the estimates

$$||e_{y_1}^{i+1}|| = \frac{1}{2}\,||e_{x_2}^{i}|| \tag{34}$$

$$||e_{y_2}^{i+1}|| \leq \big(\frac{1}{4}+\frac{1}{32}\,\Delta t^2\big)\,||e_{x_2}^{i}|| + O(\Delta t^4) \tag{35}$$

$$||e_{\lambda}^{i+1}|| \leq \big(\frac{1}{8}+\frac{1}{64}\,\Delta t^2\big)\,||e_{x_2}^{i}|| + ||\ddot{e}_{x_2}^{i}|| + O(\Delta t^4) \tag{36}$$

Based on (25), the error functions $e_{x_j}^{i+1}$ satisfy the differential equations

$$\left\{ \begin{array}{c} \ddot{e}_{x_1}^{i+1} \\ \ddot{e}_{x_2}^{i+1} \end{array} \right\} = A \left\{ \begin{array}{c} e_{x_1}^{i+1} \\ e_{x_2}^{i+1} \end{array} \right\} + \frac{1}{11} \left\{ \begin{array}{c} e_{\lambda}^{i+1} \\ -4\,e_{\lambda}^{i+1} \end{array} \right\} \tag{37}$$

where

$$A = \frac{1}{11}\left[\begin{array}{cc} 5 & 1 \\ 2 & 7 \end{array} \right]$$

The Jordan Canonical Form of A is given by

$$D = \left[\begin{array}{cc} r_1 & \\ & r_2 \end{array} \right] = \frac{1}{11}\left[\begin{array}{cc} 6-\sqrt{3} & \\ & 6+\sqrt{3} \end{array} \right]$$

Let $P = \big(p_{ij}\big)_{2\times 2}$ be an invertible matrix such that $D = P\,A\,P^{-1}$. Let also $Q = \big(q_{ij}\big)_{2\times 2} = P^{-1}$ denotes the inverse of P. With the initial condition (26), the solution of (37) is given by

$$e_{x_1}^{i+1} = \frac{q_{11}}{\sqrt{r_1}}\int_{t_n}^{t} f_1(\tau)\,\sinh[\sqrt{r_1}\,(t-\tau)]\,d\tau + \frac{q_{12}}{\sqrt{r_2}}\int_{t_n}^{t} f_2(\tau)\,\sinh[\sqrt{r_2}\,(t-\tau)]\,d\tau \tag{38}$$

$$e_{x_2}^{i+1} = \frac{q_{21}}{\sqrt{r_1}}\int_{t_n}^{t} f_1(\tau)\,\sinh[\sqrt{r_1}\,(t-\tau)]\,d\tau + \frac{q_{22}}{\sqrt{r_2}}\int_{t_n}^{t} f_2(\tau)\,\sinh[\sqrt{r_2}\,(t-\tau)]\,d\tau \tag{39}$$

where

$$\begin{aligned} f_1(\tau) &= \frac{1}{11}\big(p_{11}-4\,p_{12}\big)\,e_{\lambda}^{i+1}(\tau) \\ f_2(\tau) &= \frac{1}{11}\big(p_{21}-4\,p_{22}\big)\,e_{\lambda}^{i+1}(\tau) \end{aligned} \tag{40}$$

It is straightforward to find that

$$\begin{aligned} \ddot{e}_{x_2}^{i+1} = {} & \sqrt{r_1}\,q_{21}\int_{t_n}^{t} f_1(\tau)\,\sinh[\sqrt{r_1}\,(t-\tau)]\,d\tau + q_{21}\,f_1(t) + \\ & \sqrt{r_2}\,q_{22}\int_{t_n}^{t} f_2(\tau)\,\sinh[\sqrt{r_2}\,(t-\tau)]\,d\tau + q_{22}\,f_2(t) \end{aligned} \tag{41}$$

Using equations (39)(40), we obtain

$$\begin{aligned} ||e_{x_2}^{i+1}|| &\leq \frac{1}{2}\,\Delta t^2\,|q_{21}|\,||f_1(t)|| + \frac{1}{2}\,\Delta t^2\,|q_{22}|\,||f_2(t)|| + O(\Delta t^4) \\ &\leq \frac{1}{22}\,\Delta t^2\,\big(\,|q_{21}|\,|p_{11}-4\,p_{12}| + |q_{22}|\,|p_{21}-4\,p_{22}|\,\big)\,||e_{\lambda}^{i+1}|| + O(\Delta t^4) \end{aligned} \tag{42}$$

where we have applied the following result in the first inequality of (42),

$$\frac{1}{\sqrt{r}}\int_{t_n}^{t}\sinh[\sqrt{r}(t-\tau)]\,d\tau \;=\; \frac{1}{r}\left[\,\cosh[\sqrt{r}(t-t_n)]-1\,\right] \;=\; \frac{1}{2!}(t-t_n)^2+\frac{1}{4!}\,r(t-t_n)^4+\cdots$$

Using the results in (41)(42), we now have

$$\begin{aligned}||\ddot{e}_{x_2}^{i+1}|| \;\leq\; & \frac{1}{22}\,\Delta t^2\left(\,r_1\,|q_{21}|\,|p_{11}-4\,p_{12}|+r_2\,|q_{22}|\,|p_{21}-4\,p_{22}|\,\right)||e_{\lambda}^{i+1}||\;+\\ & \frac{1}{11}\left(\,|q_{21}|\,|p_{11}-4\,p_{12}|+|q_{22}|\,|p_{21}-4\,p_{22}|\,\right)||e_{\lambda}^{i+1}||+O(\Delta t^4)\end{aligned} \tag{43}$$

Let us denote

$$\begin{aligned}E \;&=\; \frac{1}{22}\left(\,r_1\,|q_{21}|\,|p_{11}-4\,p_{12}|+r_2\,|q_{22}|\,|p_{21}-4\,p_{22}|\,\right)\\ F \;&=\; \frac{1}{11}\left(\,|q_{21}|\,|p_{11}-4\,p_{12}|+|q_{22}|\,|p_{21}-4\,p_{22}|\,\right)\end{aligned}$$

We can then combine the results in (36) and (42)(43) to obtain

$$\begin{aligned}||e_{x_2}^{i+1}|| \;&\leq\; \frac{1}{2}\,\Delta t^2\,F\,||e_{\lambda}^{i+1}||+O(\Delta t^4)\\ &\leq\; \frac{F}{16}\,\Delta t^2\,||e_{x_2}^{i}||+\frac{F}{2}\,\Delta t^2\,||\ddot{e}_{x_2}^{i}||+O(\Delta t^4)\end{aligned} \tag{44}$$

and

$$\begin{aligned}||\ddot{e}_{x_2}^{i+1}|| \;&\leq\; (\Delta t^2\,E+F)\,||e_{\lambda}^{i+1}||+O(\Delta t^4)\\ &\leq\; \left[\,\left(\frac{E}{8}+\frac{F}{64}\right)\Delta t^2+\frac{F}{8}\,\right]||e_{x_2}^{i}||+(\Delta t^2\,E+F)\,||\ddot{e}_{x_2}^{i}||+O(\Delta t^4)\end{aligned} \tag{45}$$

In a similar way, we find

$$||e_{x_1}^{i+1}|| \;\leq\; \frac{\widetilde{F}}{16}\,\Delta t^2\,||e_{x_2}^{i}||+\frac{\widetilde{F}}{2}\,\Delta t^2\,||\ddot{e}_{x_2}^{i}||+O(\Delta t^4) \tag{46}$$

where

$$\widetilde{F} \;=\; \frac{1}{11}\left(\,|q_{11}|\,|p_{11}-4\,p_{12}|+|q_{12}|\,|p_{21}-4\,p_{22}|\,\right)$$

Now multiply both sides of (45) by a *positive* number s, whose value is to be determined, then add the result to (44) to obtain

$$\begin{aligned}||e_{x_2}^{i+1}||+s\,||\ddot{e}_{x_2}^{i+1}|| \;\leq\; & \left\{\,\frac{F}{16}\,\Delta t^2+s\left[\,\left(\frac{E}{8}+\frac{F}{64}\right)\Delta t^2+\frac{F}{8}\,\right]\right\}||e_{x_2}^{i}||\;+\\ & \left[\,\frac{F}{2}\,\Delta t^2+s\,(\Delta t^2\,E+F)\,\right]||\ddot{e}_{x_2}^{i}||+O(\Delta t^4\,,\;s\,\Delta t^4)\end{aligned} \tag{47}$$

Let us set

$$s \;=\; \frac{\frac{F}{2}\,\Delta t^2+s\,(\Delta t^2\,E+F)}{\frac{F}{16}\,\Delta t^2+s\left[\,\left(\frac{E}{8}+\frac{F}{64}\right)\Delta t^2+\frac{F}{8}\,\right]} \tag{48}$$

This results in a quadratic equation for s,

$$\big[\,(\frac{E}{8}+\frac{F}{64})\,\Delta t^2+\frac{F}{8}\,\big]\,s^2\;+\;\big[\,(\frac{F}{16}-E)\,\Delta t^2-F\,\big]\,s-\frac{F}{2}\,\Delta t^2\;=\;0 \tag{49}$$

whose two roots can be determined by the quadratic formula. Since $E>0$, $F>0$, it is easy to observe that equation (49) has two real roots: one positive with order $O(1)$, and one negative with order $O(\Delta t^2)$. Let β denote the positive root. Also denote

$$\gamma\;=\;\frac{F}{16}\,\Delta t^2+\beta\,\Big[\,(\frac{E}{8}+\frac{F}{64})\,\Delta t^2+\frac{F}{8}\,\Big] \tag{50}$$

Since β and γ are both positive and are in the order of $O(1)$, the combination of (47)(48) and (50) yields

$$\begin{aligned}||e_{x_2}^{i+1}||\;\le\;||e_{x_2}^{i+1}||+\beta\,||\ddot{e}_{x_2}^{i+1}||\;&\le\;\gamma\,\big(\,||e_{x_2}^{i}||+\beta\,||\ddot{e}_{x_2}^{i}||\,\big)\;+\;O(\Delta t^4)\\ &\le\;\gamma^2\,\big(\,||e_{x_2}^{i-1}||+\beta\,||\ddot{e}_{x_2}^{i-1}||\,\big)\;+\;O(\Delta t^4)\\ &\le\;\cdots\\ &\le\;\gamma^{i+1}\,\big(\,||e_{x_2}^{0}||+\beta\,||\ddot{e}_{x_2}^{0}||\,\big)\;+\;O(\Delta t^4)\end{aligned} \tag{51}$$

as well as

$$||\ddot{e}_{x_2}^{i+1}||\;\le\;\beta^{-1}\,\big(\,||e_{x_2}^{i+1}||+\beta\,||\ddot{e}_{x_2}^{i+1}||\,\big)\;\le\;\frac{\gamma^{i+1}}{\beta}\,\big(\,||e_{x_2}^{0}||+\beta\,||\ddot{e}_{x_2}^{0}||\,\big)\;+\;O(\Delta t^4) \tag{52}$$

where $e_{x_2}^0$ and $\ddot{e}_{x_2}^0$ measure the start-up error of the iterative procedure. Based on the results in (34)-(36), (46) and (51)(52), it is clear that the iterative procedure is convergent if

$$\gamma\;<\;1 \tag{53}$$

where γ is defined in (50). The value of γ also measures the rate of convergence: the smaller γ, the faster convergence. The result in (51) shows that when the convergence is achieved, the numerical error resulting from the iterative procedure is of order $O(\Delta t^4)$. Although the truncation errors of the ODE solver are not considered in this error analysis, the estimate in (51) does indicate that the iterative method will introduce an error of at most $O(\Delta t^4)$ provided the convergence can be achieved.

It is then straightforward to carry out the algebraic evaluations for the model problem to obtain

$$E\;\approx\;0.12\,,\qquad F\;\approx\;0.36\,,\qquad \beta\;\approx\;8.00-1.20\,\Delta t^2+O(\Delta t^4)$$

and

$$\gamma\;\approx\;0.36+0.13\,\Delta t^2+O(\Delta t^4) \tag{54}$$

Hence, the iterative procedure will converge even for relatively large Δt, though smaller values of Δt should be used for better accuracy. For sufficiently small Δt, the convergence rate is approximately 0.36.

i	1	2	3	4	5	6	7	8	9	10
$\Delta t = 1.0$	0.014	0.50	0.45	0.42	0.40	0.37	0.35	0.31	0.25	0.25
$\Delta t = 0.5$	0.0016	0.57								

Table 1: Convergence ratios $\varepsilon_x^{i+1}/\varepsilon_x^i$ $(i = 1, 2, \cdots)$ with the tolerance $\varepsilon_0 = 10^{-6}$. For $\Delta t = 1.0$ and 0.5, it takes 11 and 3 iterations, respectively, to achieve the convergence.

5 NUMERICAL RESULTS

In this section, we verify the proposed iterative method by conducting numerical calculation to the model problem. We choose the computational domain as $[0, 10]$. We consider different values of Δt in the numerical tests. Meanwhile, we divide each interval $[t_n\,,\, t_{n+1}]$ into M subintervals when implementing the iterative procedure. In what follows we set $M = 10$ unless otherwise noted. We employ the trapezoidal rule, which is one of the most popular second-order methods, as the ODE solver in most of these tests.

We first investigate the convergence property of the iterative method. To that end we pick a point between 0 and 10, say $t = 2$, and record the convergence history. We find that when $\Delta t = 1.0$, it takes 8 iterations to achieve the convergence with a given tolerance $\varepsilon_0 = 10^{-4}$, and 11 iterations for the convergence with a smaller tolerance, $\varepsilon_0 = 10^{-6}$. When $\Delta t = 0.5$, however, it only takes 2 and 3 iterations to satisfy the 10^{-4} and 10^{-6} requirements, respectively. Similarly, we find that at $t = 3$, it takes 8 iterations when $\Delta t = 1.0$, and 3 iterations when $\Delta t = 0.5$, to achieve the convergence with $\varepsilon_0 = 10^{-6}$. This clearly shows the fast convergence of the iterative method, especially when the mesh is refined. Furthermore, we can check the convergence rates based on the recorded convergence history. For illustration, we present the results for x only; similar results hold for y and λ. Numerically, the convergence rate for x can be estimated by calculating the ratios $\varepsilon_x^{i+1}/\varepsilon_x^i$ for $i = 1, 2, \cdots$, where ε_x^{i+1} is defined in equation (9). Table 1 displays the calculated ratios at $t = 2$ with the tolerance $\varepsilon_0 = 10^{-6}$ for $\Delta t = 1.0$ and 0.5. We observe similar convergence pattern in both cases; except for $i = 1$, the ratios $\varepsilon_x^{i+1}/\varepsilon_x^i$ $(i = 2, 3, \cdots)$ are in a range reasonably consistent with the analytical prediction of the convergence rate in equation (54). The differences could be attributed to the coupling between the iterative errors and the local truncation errors from the ODE solver, the latter of which was not considered in our convergence analysis. For $\Delta t = 1.0$, the ratios keep decreasing when i (the number of iterations) is increasing, showing the error is decreasing faster and faster until the convergence is achieved. The ratios with $i = 1$ are distinct from others, for both choices of Δt; the much lower values in the $i = 1$ case indicate a vast improvement of accuracy after the first iteration against the initial guess (see equation 23). Similar convergence results have been obtained with various choices of Δt and ε_0.

Next, we verify the overall accuracy of this numerical approach by checking the numerical error at the end of the computation, $t = 10$. Since the analytical solution is not available, the order of accuracy for x can be estimated by using the Richardson extrapolation:

$$R_x(\Delta t) \;=\; \left|\left| \frac{x_{\Delta t}(10) - x_{\Delta t/2}(10)}{x_{\Delta t/2}(10) - x_{\Delta t/4}(10)} \right|\right|$$

where $x_{\Delta t}(10)$ denotes the final numerical solution of $x(10)$ with the stepsize Δt. In a similar way we can define $R_y(\Delta t)$ and $R_\lambda(\Delta t)$. Table 2-a shows the values of R_x, R_y and R_λ for two rounds of numerical runs with Δt being 1.0 and 0.1, respectively. We clearly see that fully second-order accuracy is achieved in the end of the computation, which is consistent

Δt	$R_x(\Delta t)$	$R_y(\Delta t)$	$R_\lambda(\Delta t)$
1.0	4.11	4.11	4.03
0.1	4.00	4.00	4.00

(a)

Δt	$R_x(\Delta t)$	$R_y(\Delta t)$	$R_\lambda(\Delta t)$
1.0/M	1.99	2.01	2.01
0.1/M	2.00	2.00	2.03

(b)

Table 2: Order of accuracy for the two numerical approaches: (a) trapezoidal rule with iterations; and (b) trapezoidal rule without iterations, where $M = 10$. Results show fully second-order accuracy in part (a), and only first-order accuracy in part (b).

Δt	0.1	0.05	0.025
$x_2 - 2y_1$	$3.6 \cdot 10^{-8}$	$1.1 \cdot 10^{-8}$	$2.9 \cdot 10^{-9}$

(a)

Δt	0.1/M	0.05/M	0.025/M	0.0125/M
$x_2 - 2y_1$	689.5	744.8	774.5	784.8

(b)

Table 3: Test of the algebraic constraint $x_2 = 2y_1$ for the two numerical approaches: (a) trapezoidal rule with iterations; and (b) trapezoidal rule without iterations, where $M = 10$. Results show that the constraint is accurately matched in part (a), and seriously violated in part (b).

with the accuracy of the trapezoidal rule. In contrast, if the iterative procedure is *not* used and the trapezoidal rule *alone* is applied to the model problem (correspondingly, with a stepsize $\Delta t/M$, for a fair comparison), then only first-order accuracy can be achieved. See Table 2-b. The degradation of accuracy, also known as the order reduction (Gourlay 1970; Prothero and Robinson 1974), in the latter case is due to the fact that many traditional ODE solvers (such as the trapezoidal rule) are usually unable to well maintain the algebraic constraint in the course of time marching, thus leading to inaccurate or even completely incorrect numerical solution. This point is further justified below.

For the model problem, the algebraic constraint is given by equation (13). To test how well this constraint is satisfied in the numerical solution, we compute $x_2 - 2y_1$ at $t = 10$ for both cases: the trapezoidal rule with or without iterations. The results are presented in Table 3. Clearly, for the combined trapezoidal rule and iterative method, the value of $x_2 - 2y_1$ is approaching zero with second-order convergence, confirming that the algebraic constraint is accurately matched. In contrast, when using the trapezoidal rule alone, the value of $x_2 - 2y_1$ is in the magnitude of several hundred (with first-order convergence measured), indicating that the numerical solution has seriously deviated from the exact solution. The significance of using the iterative method is thus clear: it ensures both *correct* and *accurate* numerical solution to such constrained dynamic problems.

One more evidence is provided in Table 4, where we present the estimates for the order of accuracy when our iterative algorithm is combined with another common ODE solver: the second-order regular (i.e., explicit) Runge-Kutta method. We again set $\Delta t = 1.0$, and consider two choices of the subintervals: $M = 20$ and $M = 200$, respectively. Table 4 shows the measurements at the end of computation, $t = 10$, where second-order accuracy is observed for all the variables. One minor exception is that when $M = 20$, the order for λ is slightly below two ($\sqrt{3.56} \approx 1.89$). Nevertheless, as the subintervals are refined ($M = 200$), almost fully second-order accuracy ($\sqrt{3.87} \approx 1.97$) is observed for λ.

	$R_x(\Delta t)$	$R_y(\Delta t)$	$R_\lambda(\Delta t)$
$\Delta t = 1.0\,,\ M = 20$	4.34	4.04	3.56
$\Delta t = 1.0\,,\ M = 200$	4.68	4.36	3.87

Table 4: Order of accuracy for the iterative scheme combined with the second-order explicit Runge-Kutta method. Results show second-order accuracy for x, y and λ.

Finally, we verify that the iterative method can be combined with different ODE solvers for x and y, a flexibility important in many practical applications. For a simple illustration, we apply the first-order explicit Euler method for x and the trapezoidal rule for y, combined with the iterative procedure. We find similar results as those presented in Tables 1, 2-a and 3-a. Particularly, we obtain in the end of the computation,

$$R_x \approx 4.14\,, \qquad R_y \approx 4.09\,, \qquad R_\lambda \approx 4.02$$

with $\Delta t = 0.1$ and $\varepsilon_0 = 10^{-6}$, confirming fully second-order accuracy. This shows that although the Euler method is only first-order accurate, the overall accuracy can still attain second-order by combining the Euler with the trapezoidal rule and using the proposed iterative procedure. Consequently, computational effort can be saved (for x in this particular case), compared to the previous tests with the trapezoidal rule or the Runge-Kutta method applied to both x and y, without degrading the overall accuracy. It is to be seen if this observation holds true for other problems and for the combination of higher-order ODE solvers.

We mention again that there are several implicit and sophisticated numerical methods for differential-algebraic equations, such as the modified extended backward difference formulas (Cash 2000), the implicit Runge-Kutta methods (Roche 1989), and others (*e.g.*, Fox *et al.* 2000; Gear 1971; Rangan 2003). These methods can possibly produce very accurate results for the model problem we present in this paper. Nevertheless, we emphasize the significance of our proposed iterative algorithm is that it offers a straightforward improvement of simple, traditional ODE solvers (such as the forward Euler, trapezoidal rule, regular Runge-Kutta, etc.) so that they can be applied to compute challenging differential-algebraic equations. When directly and individually implemented to constrained dynamic problems, these traditional ODE solvers encounter significant difficulty in either losing accuracy or generating incorrect solutions. In contrast, when combined with the proposed iterative procedure, these methods can produce both correct and accurate results. Furthermore, this combination is straightforward and there is no change to the details of the original ODE solvers.

6 CONCLUSIONS AND DISCUSSION

We have presented an iterative method to numerically solve a class of constrained dynamic problems in the form of differential-algebraic equations, and demonstrated the application of this method through a case study. We have conducted a careful analysis on the convergence of the iterative method in terms of the model problem. The analytical predictions have been verified by numerical results.

The main advantage of this iterative method is to ensure an accurate and efficient solution procedure for challenging differential-algebraic equations based on traditional ODE solvers, with the decoupling of the computation for x and y and with the algebraic constraints strictly main-

tained in the course of time marching. This iterative method is not an independent differential-algebraic equation solver, in the sense that an underlying ODE solver is needed to obtain the solution. Nevertheless, this iterative method can be easily combined with any ODE solvers, and the implementation of the iterative procedure is straightforward. It thus allows simple and traditional ODE solvers to be applied to constrained dynamic problems in a routine and reliable way.

As described in Section 2, the proposed iterative method is based on splitting the leading coefficient matrix in equation (1). Essentially, this means we impose the algebraic constraint on y when decoupling the computations for x and y (see equations 3 and 4). Such splitting approach is certainly not unique. For example, if we impose the constraint on x and split the leading terms in equation (1) correspondingly, then we obtain

$$\begin{bmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & 0 \\ D_1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}^{i+1} \\ \ddot{y}^{i+1} \\ \lambda^{i+1} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & B_2 \\ 0 & D_2 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}^{i} \\ \ddot{y}^{i} \\ \lambda^{i+1} \end{bmatrix} + \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x^{i+1} \\ y^{i+1} \\ \lambda^{i+1} \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ d \end{bmatrix}$$

which results in two separated equations,

$$\begin{bmatrix} A_1 & B_1 \\ D_1 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x}^{i+1} \\ \lambda^{i+1} \end{bmatrix} + \begin{bmatrix} C_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x^{i+1} \\ \lambda^{i+1} \end{bmatrix} = \begin{bmatrix} r_1 \\ d - D_2 \ddot{y}^{i} \end{bmatrix} \tag{55}$$

and

$$A_2\, \ddot{y}^{i+1} + C_2\, y^{i+1} \;=\; r_2 - B_2\, \lambda^{i+1} \tag{56}$$

An iterative procedure can be similarly implemented based on equations (55) and (56). Other iterative methods can be possibly constructed by imposing the algebraic constraint on both x and y, with differing weights, which corresponds to a linear combination of the two iterative formulas, (3, 4) and (55, 56). It would be interesting to ask if there is an "optimal" iterative method that can best balance the accuracy and efficiency for constrained dynamic problems in the form of (1). We expect the answer would depend on the choice of the underlying ODE solver. Furthermore, such splitting techniques and iterative methods can be applied in exactly the same way to equation (1) with time-dependent coefficients; i.e., when the coefficient matrices A_j, B_j, C_j and D_j and the right-hand side vectors are all functions of t. We plan to carefully explore these extensions in another paper.

In addition, we expect the iterative approaches can be applied to some of the nonlinear constrained dynamic problems. Particularly, many real-life dynamic problems from fluid-structure interactions (Unger *et al.* 2007), multiple rigid body dynamics (Haug 1992), and several other scientific and engineering fields (Brenan *et al.* 1989; Knoll and Keyes 2004; Kunkel and Mehrmann 2006; Park and Felippa 2000), involve second-order differential equations for unknowns from two different sub-systems, in the form of

$$F_1(x,\, \dot{x},\, \ddot{x},\, \lambda) \;=\; 0 \tag{57}$$
$$F_2(y,\, \dot{y},\, \ddot{y},\, \lambda) \;=\; 0 \tag{58}$$

subject to the constraint

$$G(x,\, y) = 0 \tag{59}$$

The functions F_1, F_2 and G are generally nonlinear. Nevertheless, we can always separate the linear parts from equations (57 - 59), and write this nonlinear system in the same form as equation (1) where the right-hand side vector now contains all the nonlinear terms. The iterative formulation such as (3, 4) and (55, 56) can then be similarly applied, with the nonlinear terms r_1, r_2 and d on the right side evaluated at i, the previous iterate. The coefficient matrices in the linear parts can be adjusted if necessary, with corresponding changes on the right-hand side, to ensure the convergence. We expect this approach works at least for problems with weak or moderate nonlinearity, though the accuracy and stability of the algorithm will need to be carefully investigated. This will provide another interesting topic in our future work.

7 APPENDIX

We describe the procedure to decouple equation (20) using an LU decomposition approach (Golub and Van Loan 1996). Although the algebraic calculation is carried out for this specific problem, the formulation is applicable to various problems of this type.

First, we note that equation (17) can be simplified as

$$\left\{ \begin{array}{c} \ddot{y}_1 \\ \ddot{y}_2 \end{array} \right\} = \left[\begin{array}{cc} \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{8} & \frac{3}{8} \end{array} \right] \left\{ \begin{array}{c} y_1 \\ y_2 \end{array} \right\} + \left\{ \begin{array}{c} 2 \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} \frac{1}{2} \\ -\frac{1}{4} \end{array} \right\} \lambda \tag{60}$$

Equation (20) can then be rewritten as

$$\left[\begin{array}{ccc} 1 & 0 & -1/2 \\ 0 & 1 & 1/4 \\ -2 & 0 & 0 \end{array} \right] \left\{ \begin{array}{c} \ddot{y}_1^{i+1} \\ \ddot{y}_2^{i+1} \\ \lambda^{i+1} \end{array} \right\} + \left[\begin{array}{ccc} -1/4 & -1/4 & 0 \\ 1/8 & -3/8 & 0 \\ 0 & 0 & 0 \end{array} \right] \left\{ \begin{array}{c} y_1^{i+1} \\ y_2^{i+1} \\ \lambda^{i+1} \end{array} \right\} = \left\{ \begin{array}{c} 2 \\ 0 \\ -\ddot{x}_2^i \end{array} \right\} \tag{61}$$

which can be cast in a standard form as

$$\left[\begin{array}{cc} A & B \\ D & 0 \end{array} \right] \left\{ \begin{array}{c} \ddot{v} \\ \lambda \end{array} \right\} = \left[\begin{array}{cc} C & 0 \\ 0 & 0 \end{array} \right] \left\{ \begin{array}{c} v \\ \lambda \end{array} \right\} + \left\{ \begin{array}{c} r \\ d \end{array} \right\} \tag{62}$$

where

$$A = \left[\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right], \quad C = \left[\begin{array}{cc} -1/4 & -1/4 \\ 1/8 & -3/8 \end{array} \right], \quad B = \left\{ \begin{array}{c} -1/2 \\ 1/4 \end{array} \right\}, \quad r = \left\{ \begin{array}{c} 2 \\ 0 \end{array} \right\}$$

and

$$D = \left(\begin{array}{cc} -2 & 0 \end{array} \right), \qquad d = -\ddot{x}_2^i$$

The LU decomposition of the leading coefficient matrix in equation (62) provides the following relation,

$$\left[\begin{array}{cc} A & B \\ D & 0 \end{array} \right] = \left[\begin{array}{cc} A & 0 \\ D & -DA^{-1}B \end{array} \right] \left[\begin{array}{cc} I & A^{-1}B \\ 0 & I \end{array} \right] \tag{63}$$

By introducing two intermediate variables, v^* and λ^*, equation (62) can be solved in two steps. In the forward elimination step, we have

$$\left[\begin{array}{cc} A & 0 \\ D & -DA^{-1}B \end{array} \right] \left\{ \begin{array}{c} v^* \\ \lambda^* \end{array} \right\} = \left[\begin{array}{cc} C & 0 \\ 0 & 0 \end{array} \right] \left\{ \begin{array}{c} v \\ \lambda \end{array} \right\} + \left\{ \begin{array}{c} r \\ d \end{array} \right\} \tag{64}$$

which implies

$$Av^* = Cv + r \tag{65}$$

and

$$DA^{-1}B\lambda^* \;=\; Dv^* - d \tag{66}$$

In the backward substitution step, we have

$$\begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix} \begin{Bmatrix} \ddot{v} \\ \lambda \end{Bmatrix} = \begin{Bmatrix} v^* \\ \lambda^* \end{Bmatrix} \tag{67}$$

which implies

$$\ddot{v} = v^* - A^{-1}B\lambda^* \tag{68}$$

and

$$\lambda = \lambda^* \tag{69}$$

Substitute equations (65) and (66) into equation (68) to obtain

$$\begin{aligned} \ddot{v} \;&=\; v^* - A^{-1}B\left(DA^{-1}B\right)^{-1}(Dv^* - d) \\ &=\; \left[I - A^{-1}B\left(DA^{-1}B\right)^{-1}D\right]A^{-1}Cv \\ &\quad +\; \left[I - A^{-1}B\left(DA^{-1}B\right)^{-1}D\right]A^{-1}r + A^{-1}B\left(DA^{-1}B\right)^{-1}d \end{aligned}$$

For our problem, $A = I$ and that helps to simplify the above equations as

$$\ddot{v} = \left[I - B\left(DB\right)^{-1}D\right]Cv + \left[I - B\left(DB\right)^{-1}D\right]r + B\left(DB\right)^{-1}d \tag{70}$$

and

$$\lambda = \left(DB\right)^{-1}DCv + \left(DB\right)^{-1}Dr - \left(DB\right)^{-1}d \tag{71}$$

We can then easily carry out the algebraic calculation to obtain

$$DB = 1\,, \qquad\qquad DC = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix}$$

and

$$\left[I - B\left(DB\right)^{-1}D\right] = \begin{bmatrix} 0 & 0 \\ 1/2 & 1 \end{bmatrix}, \qquad B\left(DB\right)^{-1} = \begin{Bmatrix} -1/2 \\ 1/4 \end{Bmatrix}$$

Therefore, we have

$$\begin{Bmatrix} \ddot{y}_1 \\ \ddot{y}_2 \end{Bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & -1/2 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} + \begin{bmatrix} 0 & 0 \\ 1/2 & 1 \end{bmatrix} \begin{Bmatrix} 2 \\ 0 \end{Bmatrix} - \begin{Bmatrix} -1/2 \\ 1/4 \end{Bmatrix} \ddot{x}_2 \tag{72}$$

and

$$\lambda = \frac{1}{2}y_1 + \frac{1}{2}y_2 - 4 + \ddot{x}_2 \tag{73}$$

Equations (72) and (73) are used in *Steps 1* and *2* of the proposed iterative algorithm applied to the case study (see Section 3).

8 ACKNOWLEDGMENT

This work was partially supported by the National Science Foundation (under Grant No. DMS-0813691) and the Thomas F. and Kate Miller Jeffress Trust (under Grant No. J-964).

REFERENCES

Arnold M (1993). Stability of numerical methods for Differential-Algebraic Equations of higher index. *Applied Numerical Mathematics* 13, pp. 5-14.

Ascher UM and Petzold LR (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia, USA.

Belegundu AD and Chandrupatla TR (1999). *Optimization Concepts and Applications in Engineering*. Prentice Hall.

Brenan KE, Campbell SL and Petzold LR (1989). *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Elsevier Science Publishing, North-Holland.

Cash JR (2000). Modified extended backward differentiation formulae for the numerical solution of stiff initial-value problems in ODEs and DAEs. *Journal of Computational and Applied Mathematics* 125, pp. 117-130.

Cash JR (2003). Efficient numerical methods for the solution of stiff initial-value problems and differential algebraic equations. *Proceedings of the Royal Society of London A* 459, pp. 797-815.

Fox B, Jennings LS and Zomaya AY (2000). Numerical computation of differential-algebraic equations for the approximation of artificial satellite trajectories and planetary ephemerides. *Journal of Applied Mechanics* 67, pp. 574-580.

Gear C (1971). Simultaneous numerical solution of differential-algebraic equations. *IEEE Transactions on Circuit Theory* 18, pp. 89-95.

Golub GH and Van Loan CF (1996). *Matrix Computations*. The Johns Hopkins University Press.

Gourlay AR (1970). A note on trapezoidal methods for the solution of initial value problems. *Mathematics of Computation* 24, pp. 629-633.

Hairer E and Wanner G (1996). *Solving Ordinary Differential Equations II: Stiff and Differential Algebraic Problems*. Springer.

Haug EJ (1992). *Intermediate Dynamics*. Prentice Hall.

Knoll DA and Keyes DE (2004). Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *Journal of Computational Physics* 193, pp. 357-397.

Kunkel P and Mehrmann V (2006). *Differential-Algebraic Equations: Analysis and Numerical Solution*. European Mathematical Society Publishing House, Switzerland.

Park KC and Felippa CA (2000). A variational principle for the formulations of partitioned structural systems. *International Journal for Numerical Methods in Engineering* 47, pp. 395-418.

Prothero A and Robinson A (1974). On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation* 28, pp. 145-162.

Rangan A (2003). Deferred correction methods for low index differential algebraic equations. *BIT Numerical Mathematics* 43, pp. 1-18.

Roche M (1989). Implicit Runge-Kutta methods for differential algebraic equations. *SIAM Journal on Numerical Analysis* 26, pp. 963-975.

Unger R, Haupt MC and Horst P (2007). Application of Lagrange multipliers for coupled problems in fluid and structural interactions. *Computers and Structures* 85, pp. 796-809.