Nonlinear Dyn (2016) 85:263–280
DOI 10.1007/s11071-016-2682-8

ORIGINAL PAPER

A two-loop procedure based on implicit Runge–Kutta method for index-3 DAE of constrained dynamic problems

Le Zhang · Dingguo Zhang

Received: 9 March 2015 / Accepted: 13 February 2016 / Published online: 1 March 2016

Abstract In this paper, a procedure is proposed to use implicit Runge–Kutta method as the integration method for the solution of index-3 differential–algebraic equation which comes from constrained dynamic problems. The position constraints of a multibody system are usually a set of nonlinear algebraic equations. The iteration of the position constraint equations is embedded into the iteration of the nonlinear algebraic equations that come from the implicit Runge–Kutta method. These two iterations construct a two-loop structure. By using the coordinate partitioning technique, the independent coordinates are picked out from the set of coordinates of the whole system. The basic unknowns of the proposed method are the independent velocities and positions. A symplectic implicit Runge–Kutta method is used as the integration method in the two-loop procedure so that the independent velocities and positions can be obtained. The iteration of the implicit Runge–Kutta method is the outer loop in the two-loop procedure. A fixed-point iteration is brought in the outer loop to avoid the numerical differentiation that is used in Newton's method to get the Jacobian matrix of the right-hand function. Newton's method is used in the inner loop to solve the nonlinear algebraic equations of the position constraints, and then the dependent position coordinates can be obtained. The dependent velocities may be determined by solving a set of linear algebraic equations. The variable step size strategy of the two-loop procedure, which is based on the estimation of the error and the convergence criterion, is proposed for the practical application of this method. Two numerical examples are presented to demonstrate the efficiency of the proposed method.

Keywords DAE · State-space-based method · Implicit integration method · Iteration

L. Zhang · D. Zhang (✉)
School of Science, Nanjing University of Science and Technology, Nanjing 210094, China
e-mail: zhangdg419@mail.njust.edu.cn

L. Zhang
e-mail: zhangleatnjust@gmail.com

1 Introduction

Constrained dynamic systems can be treated as the combination of two coupled subsystems: a structural part and the constraints which are acting on the structure. A set of differential–algebraic equations (DAEs) contain two parts: a set of ordinary differential equations (ODEs) and a set of algebraic equations. The ODEs in the DAEs describe the dynamic behavior of the structural subsystem, and the algebraic equations in the DAEs describe the constraints. So DAEs are often used to describe constrained dynamic systems. Multibody systems are a typical class of constrained systems, and dynamic problems of multibody systems can be described in a general way by using DAEs as the mathematical models [1].

In multibody systems, different bodies are connected by joints, and the motions of the bodies are restrained by these joints. When the dynamic problems of multibody systems are modeled, the mathematical models of the joints are a set of algebraic equations, and these algebraic equations are called position constraints. The DAEs of holonomic multibody systems consist of a set of ODEs which describe the dynamic behavior of the bodies and the position constraints. Besides the position constraints, there exist velocity constraints and acceleration constraints in multibody systems. For holonomic systems,the velocity constraints come from the differentiation of the position constrains, and the acceleration constraints are the result of the differentiation of the velocity constraints. These two sets of equations are both linear algebraic equations. In most multibody systems, the algebraic equations of the position constraints are nonlinear, so the DAEs of multibody system dynamics are usually index-3 [2]. For index-3 DAEs, the numerical methods which are designed for low index problems fail, because the constraints may be violated. Several numerical methods have been proposed to solve index-3 DAEs. Some methods are introduced in [3], and a comparative study can be found in [4].

Baumgarte's stabilization method is the most famous method for index-3 DAEs. In Baumgarte's stabilization method, a stabilization term is introduced to control the violation of the constraints. The position constraint equations and the velocity constraint equations are not solved in Baumgarte's method, so the constraints are not satisfied exactly in this method.

There are several other methods in which the constraint equations are directly solved so that the constraints are exactly satisfied. One of these methods is the project method. The project method introduces extra Lagrange multipliers to make sure that the velocity constraints and the position constraints are satisfied. In this method, all the generalized coordinates are integrated at each time step. The extra Lagrange multipliers also need to be solved in the project method, so this method is inefficient.

There is another method called the state-space-based method which can make sure that all the constraints in multibody systems are satisfied strictly. The ODEs and the algebraic equations are both solved in the state-space-based method to get the result. In the state-space-based method, a technique called coordinate partitioning is used to divide the coordinates of the system into two sets: One is the set of the independent coordinates, and the other is the set of the dependent coordinates. Integration methods are used to solve the ODEs, and in this process, only the independent coordinates are integrated to get the independent velocities and positions. Then, the independent positions are used in an iterative method to solve the nonlinear algebraic equations so that the dependent positions can be obtained, and the dependent velocities can be obtained by solving a set of linear velocity constraint equations. If the DAEs are solved by using the state-space-based method, different kinds of numerical methods designed for ODEs and algebraic equations need to be handled together to form a resultant algorithm to solve the index-3 DAEs.

Any integration method that works for ODEs can be used as the integration method in the numerical method for the index-3 DAEs. If the system is a stiff system, an implicit integration method is needed. Here arises the problem of how to use the implicit integration method in the numerical method, because the implicit integration method brings an extra set of nonlinear algebraic equations. This set of nonlinear algebraic equations should be handled together with the nonlinear algebraic equations of the position constraints. The two different sets of nonlinear algebraic equations both need to be iterated, so there exist two loops of iteration in the numerical method for the index-3 DAEs. One loop is the iteration of the nonlinear algebraic equations that describe the position constraints of the system, and the other loop is for the nonlinear algebraic equations that come from the implicit integration method. These two loops can be combined to form a bigger loop, and if the integration method is HHT method, this method is called the HHT-I3 method [5,6]. In state-space-based methods, these two loops can be executed separately, and it means that the dependent velocities and positions in the loop of the implicit integration method are treated as knowns. The loop of iterating the nonlinear position constraint equations and the solving of the linear velocity constraint equations can be embedded into the loop of iterating the implicit integration method, and then, we get a two-loop procedure. There are different two-loop procedures based on different implicit integration methods.

Different implicit integration methods have been used in the state-space-based method as integration methods; for example: implicit Runge–Kutta method

[7,8], Rosenbrock–Nystrom formula [9], backward differentiation formulas [10], and the Newmark method [11]. The nonlinear equations of implicit integration methods are usually solved by using Newton's method. The Jacobian matrix of the nonlinear function is needed in Newton's method. In most conditions, this matrix cannot be obtained analytically if the DAEs come from multibody system dynamics, and consequently, a numerical differential procedure is needed to get the Jacobian matrix. Fisette and Vaneghem [11] proposed a method to simplify the progress of calculating the Jacobian matrix of the implicit integration method. Haug et al. pointed out that the method used in Ref. [11] ignored some terms; this iterative method is actually a quasi-Newton method, and they improved the method to make sure the iterative method is Newton's method [12].

The calculation of the Jacobian matrix of the nonlinear algebraic equations of the implicit integration method is difficult if the model is complex. Shabana and Hussein [13,14] proposed a two-loop procedure with the Newmark method or HHT method as the integration method, and a fixed-point iterative method is used to avoid the calculation of the extra Jacobian matrix. The numerical accuracy of the Newmark method or HHT method is of order 2 or 1. For the problems of multibody system dynamics, sometimes a high-order integration method is needed to integrate the coordinates.

Runge–Kutta methods are a class of integration methods that are widely used to solve ODEs, and an implicit Runge–Kutta method can be introduced to integrate the coordinates in the two-loop procedure. The accuracy and efficiency of the two-loop procedure can be improved by applying a high-order Runge–Kutta method. Under certain circumstances, the Newmark method is symplectic. Explicit Runge–Kutta methods cannot be symplectic, while there are symplectic Runge–Kutta methods which are implicit. We can keep the integration method symplectic by choosing a symplectic implicit Runge–Kutta method as the integration method in the two-loop procedure.

In this paper, we use a symplectic implicit Runge–Kutta method as the integration method in the two-loop procedure. A fixed-point iterative method is introduced to solve the nonlinear algebraic equations that come from the implicit Runge–Kutta method, and the structure of the two-loop procedure is redesigned because the nonlinear algebraic equations of the implicit Runge–Kutta method are different from the nonlinear algebraic equations of the Newmark method. As a state-space-based method, the new two-loop procedure can make sure that the violation of the constraint equations is extremely small. It offers advantages over existing methods because the numerical difference is no more needed when the implicit Runge–Kutta method is used and it is stable for long-time simulation. Compared to the two-loop procedure based on the Newmark method, this new two-loop procedure is more efficient. Section 2 first introduces the DAE of multibody system dynamics and the two-loop procedure with the Newmark method as the integration method. Section 3 introduces a fixed-point iteration for the implicit Runge–Kutta method and proposes a new two-loop procedure with the implicit Runge–Kutta method as the integration method. Two numerical examples are presented in Sect. 4 to illustrate the specialities of the new two-loop procedure.

2 DAEs of multibody system dynamics and state-space-based methods

The form of the DAEs of multibody system dynamics is shown in this section, and the characteristic of the DAEs is introduced. We discuss the multibody systems as examples of constrained dynamic systems in this paper. The methods discussed in this paper are especially suited for but not limited to the dynamic problems of multibody systems. The methods mentioned here can also be used to solve the DAEs that come from other constraint dynamic problems. The basic idea of the state-space-based methods and the two-loop procedure with the Newmark method as the integration method are reviewed in this section.

2.1 DAEs of multibody system dynamics

In this paper, we analyze the dynamics of multibody systems with holonomic and non-scleronomic constraints. The DAEs of this kind of constraint systems are [1,15]:

$$\begin{cases} M\ddot{q} + C_q^T \lambda = Q \\ C(q,t) = 0 \end{cases} \tag{1}$$

The DAEs (1) have a form of mixed set of the second-order ODEs (2) and the algebraic equations (3):

$$M\ddot{q} + C_q^T \lambda = Q \tag{2}$$

$$C(q, t) = 0 \tag{3}$$

Equation (2) is the set of ODEs that describe the dynamic behavior of the system; Eq. (3) is the set of position constraint equations, and in multibody systems these equations are usually nonlinear. In these equations, M is the mass matrix, q is the vector of the system coordinates, λ is the vector of the Lagrange multipliers, Q is the vector of the generalized force, C is the vector of the position constraint functions, t is time, and C_q is the Jacobian matrix of the position constraint equations, $C_q = \partial C/\partial q$.

The dynamic behavior of the system is governed by Eqs. (2) and (3) together. We set the number of the ordinary differential equations in Eq. (2) as n_q and the number of the algebraic equations in Eq. (3) as n_c. n_q equals to the number of positions in q, and n_c is the number of the Lagrange multipliers λ.

By differentiating the position constraint equations with respect to time, we get the velocity constraint equations:

$$C_q \dot{q} = -C_t \tag{4}$$

where $C_t = \partial C/\partial t$.

By differentiating the velocity constraint equations with respect to time, we get the acceleration constraint equations.

$$C_q \ddot{q} = Q_d \tag{5}$$

where $Q_d = -((C_q \dot{q})_q \dot{q} + 2C_{qt}\dot{q} + C_{tt})$.

Most DAEs with the form of Eq. (1) can only be solved numerically. If the solutions resulted from ODE (2) satisfy the position constraint equations (3), the velocity constraint equations (4) and the acceleration constraint equations (5) will be satisfied automatically. However, the position constraint equations (3) cannot be satisfied exactly when the DAEs are solved by using numerical methods. The algebraic equations in Eqs. (3), (4), and (5) may be violated when numerical methods which are constructed for the ODE are used to solve Eq. (2), and this may lead to the failure of the solution.

2.2 The basic idea of the state-space-based methods

The state-space-based methods are based on the acceleration constraint equation, the velocity constraint equations, and the position constraint equations. So the result of the state-space-based methods can satisfy the constraint on position level, velocity level, and acceleration level at the same time.

The degree of freedom of a holonomic system is the number of the coordinates minus the number of the independent constraints, so the degree of freedom of a multibody system is less than the number of the position coordinates that appear in the corresponding DAE. We set the number the degree of freedom of the system as n_{DOF}, $n_{DOF} = n_q - n_c$. We can use n_{DOF} independent coordinates to describe the behavior of the system. The dependent coordinates can be determined by solving the constraint equations. In the state-space-based methods, we integrate n_{DOF} independent accelerations to get the independent velocities and positions of the next time step, and then use the velocity constraint equation (4) and the position constraint equation (3) to get the dependent velocities and positions, respectively. Independent and dependent accelerations can be calculated together by solving Eq. (6).

$$\begin{cases} M\ddot{q} + C_q^T \lambda = Q \\ C_q \ddot{q} = Q_d \end{cases} \tag{6}$$

The results are:

$$\lambda = \left(C_q M^{-1} C_q^T\right)^{-1} \left(C_q M^{-1} Q - Q_d\right) \tag{7}$$

$$\begin{aligned} \ddot{q} &= M^{-1}Q - M^{-1}C_q^T \lambda \\ &= M^{-1}Q - M^{-1}C_q^T \left(C_q M^{-1} C_q^T\right)^{-1} \\ &\quad \left(C_q M^{-1} Q - Q_d\right) \end{aligned} \tag{8}$$

If we consider the dynamic equation (2), the position constraint equation (3), the velocity constraint equation (4), and the acceleration constraint equation (5) together, the unknowns are the acceleration $\ddot{q}$, the velocity $\dot{q}$, the position q, and the Lagrange multipliers λ. The number of the acceleration $\ddot{q}$, the velocity $\dot{q}$, and the position q are all n_q, and the number of the Lagrange multipliers λ is n_c. The total number of the unknowns is $(3 \times n_q + n_c)$. In Eqs. (2)–

(5), there are $(n_q + 3 \times n_c)$ equations. The difference between the numbers of unknowns and the equations is $(2 \times n_q - 2 \times n_c)$, which equals $(2 \times n_{DOF})$. Here, we need the same number of additional equations so that the unknowns can be obtained. In the state-space-based method, the integration method gives the extra $(2 \times n_{DOF})$ equations, and thus the number of the unknowns equals the number of the equations.

For a holonomic system, n_q ordinary differential equations in Eq. (2) and n_c acceleration constraint equations in Eq. (5) are used together, as Eq. (6), to get n_q accelerations in $\ddot{q}$ and n_c Lagrange multipliers in λ; $(2 \times n_q - 2 \times n_c)$ algebraic equations that come from integration method are solved to get $(n_q - n_c)$ independent velocities in $\dot{q}_i$ and $(n_q - n_c)$ independent positions in q_i; n_c position constraint equations in Eq. (3) are solved to get n_c dependent positions in q_d; and n_c velocity constraint equations in Eq. (4) are solved to get n_c dependent velocities in $\dot{q}_d$.

To use the state-space-based methods, first we need to find the independent coordinates. This procedure is called coordinate partitioning. Coordinate partitioning can be done by decomposing the Jacobian matrix C_q. LU decomposition, QR decomposition, and singular value decomposition (SVD) can all be used in the coordinate partitioning technique, but the results of QR decomposition and SVD are not the same as the results of LU decomposition. The results of QR decomposition and SVD are linear combinations of the coordinates which can be treated as independent generalized coordinates, while LU decomposition picks n_{DOF} original coordinates as the independent coordinates. In this paper, we choose LU decomposition to get the independent coordinates of the constrained systems.

After the process of coordinate partitioning, we need to integrate the independent accelerations to get the independent velocities and the independent positions. The methods used to integrate the ODE can also be used in the state-space-based methods to integrate the independent coordinates. If the system is a stiff system, we need to use implicit methods to integrate the independent coordinates. Different from explicit methods, implicit methods cannot get the solution directly. If the integral methods are implicit methods, there exist extra algebraic equations to be solved.

2.3 Fixed-point iteration of the Newmark method and the two-loop procedure based on the Newmark method

For a dynamic system without constraints, the mathematical model is a set of second-order ODEs:

$$M\ddot{q} = Q \tag{9}$$

The accelerations can be obtained by:

$$\ddot{q} = M^{-1}Q \tag{10}$$

when time is fixed,$M = M(\dot{q}, q)$,$Q = Q(\dot{q}, q)$. We set $F = M^{-1}Q$, at time t_{n+1} we have:

$$\ddot{q}_{n+1} = F(q_{n+1}, \dot{q}_{n+1}) \tag{11}$$

At time t_n, the position q_n and the velocity $\dot{q}_n$ are known, and the acceleration $\ddot{q}_n$ can be calculated by using Eq. (10). At t_{n+1}, q_{n+1}, $\dot{q}_{n+1}$, and $\ddot{q}_{n+1}$ are all unknowns, q_{n+1} and $\dot{q}_{n+1}$ in Eq. (11) can be eliminated by the Newmark method:

$$\begin{cases} q_{n+1} = q_n + h\dot{q}_n + \frac{h^2}{2}((1-2\beta)\ddot{q}_n + 2\beta\ddot{q}_{n+1}) \\ \dot{q}_{n+1} = \dot{q}_n + h((1-\gamma)\ddot{q}_n + \gamma\ddot{q}_{n+1}) \end{cases} \tag{12}$$

Equations (11) and (12) are combined to a single equation:

$$\ddot{q}_{n+1} = F(\ddot{q}_{n+1}) \tag{13}$$

Equation (13) is a set of algebraic equations, and the unknown in Eq. (13) is the acceleration $\ddot{q}_{n+1}$. The function $F = F(q, \dot{q})$ is a set of nonlinear functions of q and $\dot{q}$ when the equation comes from dynamic problems of multibody systems, so Eq. (13) is a set of nonlinear algebraic equations.

In the Newmark method, the position q_{n+1} and the velocity $\dot{q}_{n+1}$ are treated as functions about acceleration $\ddot{q}_{n+1}$: $q = q(\ddot{q})$, $\dot{q} = \dot{q}(\ddot{q})$. By using the Newmark method, the second-order ODEs (9) is transformed to the nonlinear algebraic equation (13). Equation (13) can be solved by using Newton's method which needs the Jacobian matrix $\partial F/\partial\ddot{q}_{n+1}$; meanwhile, the form of Eq. (13) can also be used as the

format of the fixed-point iteration directly. Once the acceleration $\ddot{q}_{n+1}$ is obtained by solving (13), the position q_{n+1} and the velocity $\dot{q}_{n+1}$ can be calculated by the Newmark method (12). Thus, the integration in a time step is done.

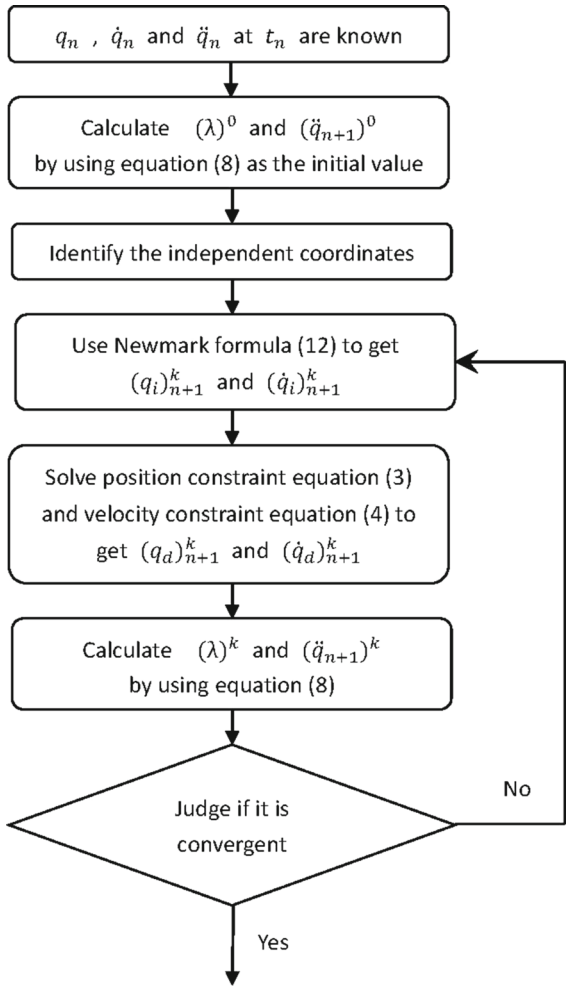
Equation (13) is the combination of Eq. (11) and the Newmark method (12). Besides the format of Eq. (13) itself, the fixed-point iteration of the nonlinear algebraic equation (13) can also be based on Eq. (11) and the Newmark method (12). The iterative process is separated into two steps:

Step 1 Use the Newmark method (12) to calculate the position $(q_{n+1})^k$ and the velocity $(\dot{q}_{n+1})^k$;
Step 2 Use Eq. (11) to calculate the acceleration $(\ddot{q}_{n+1})^k$.

The superscript k represents the kth step. Step 1 together with step 2 forms the kth step in the iteration of the nonlinear algebraic equation (13). For ODEs (9), this two-step iterative process is equivalent to the iterative process based on the format of Eq. (13) itself. The values of the position $(q_{n+1})^k$ and the velocity $(\dot{q}_{n+1})^k$ are calculated during the two-step iterative process, while these values are not needed in the iteration of Eq. (13) itself.

Based on the two-step fixed-point iteration of the Newmark method, the two-loop procedure with the Newmark method as integration method is constructed to solve Eq. (6). The ODEs about the independent coordinates are picked out from Eq. (8), and the two-step fixed-point iterative process of the Newmark method is used to solve these ODEs. The iteration of the nonlinear algebraic equation (3) and the solving of the linear algebraic equation (4) are arranged between the two steps of the two-step process of the Newmark method. Equation (12) gives the value of independent position $(q_i)^k_{n+1}$ and the velocity $(\dot{q}_i)^k_{n+1}$, and then, the dependent position $(q_d)^k_{n+1}$ and the velocity $(\dot{q}_d)^k_{n+1}$ are updated before the acceleration $(\ddot{q}_i)_{n+1}$ is calculated. The outer loop of the two-loop procedure is the iterative process of the Newmark method, and the inner loop is based on a Newton–Raphson algorithm for Eq. (3).

The flow chart for the two-loop procedure using the Newmark method as the integration method in a time step is shown as follows:



Theoretically speaking, the coordinate partitioning process is needed in each time step, but in most conditions, the Jacobian matrix C_q does not change very much in a short time, so in most cases, the decomposition of C_q will pick out the same independent coordinates in a period of time. Therefore, the coordinate partitioning can be executed once in a time step or several time steps.

Only the value of the acceleration is needed in the process as the initial value; the initial value of the acceleration can be predicted by interpolation or using the acceleration $\ddot{q}_n$ of the previous time step. For a given independent acceleration $(\ddot{q}_i)_{n+1}$, the corresponding independent velocity $(\dot{q}_i)_{n+1}$ and position $(q_i)_{n+1}$ can be calculated by Eq. (12). The dependent velocity $(\dot{q}_d)_{n+1}$ and position $(q_d)_{n+1}$ are given by the inner

loop. The acceleration must be updated at the last of the outer loop when both the independent and dependent velocity $\dot{q}_{n+1}$ and position q_{n+1} have been obtained.

Variable step size strategy, convergence criteria, and the choice of the maximum time step size of the two-loop procedure with the Newmark method have been proposed in Ref. [14].

Besides the strategy of using the Newmark method in the two-loop procedure, Shabana also proposed a different format to use the independent coordinates to compute the dependent coordinates in Ref. [13].

$$\begin{bmatrix} C_q \\ I_{in} \end{bmatrix} \dot{q} = \begin{bmatrix} -C_t \\ \dot{q}_i \end{bmatrix} \tag{14}$$

$$\begin{bmatrix} C_q \\ I_{in} \end{bmatrix} \triangle q = \begin{bmatrix} -C \\ 0 \end{bmatrix} \tag{15}$$

where I_{in} is a Boolean matrix that contains zeros and ones. By using the matrix I_{in}, the independent and dependent coordinates do not need to be realigned. The details can be found in Ref. [10,13]. Format (14) and (15) are used in this paper to update the dependent position and velocity.

3 Two-loop procedure based on implicit Runge–Kutta method

In this section, we propose a new two-loop procedure with an implicit Runge–Kutta method as the integration method. The fixed-point iterative method for the implicit Runge–Kutta method is introduced in Sect. 3.1, and the whole structure of the new two-loop procedure is presented in Sect. 3.2. The variable step size strategy, convergence criteria, and the determining of the maximum time step can be found in Sect. 3.3.

3.1 The fixed-point iterative method for implicit Runge–Kutta method

Implicit Runge–Kutta method is widely used in solving the stiff ODEs [16]. No explicit Runge–Kutta method can be symplectic [17], so an advantage of implicit Runge–Kutta method is that this method may be symplectic. If the integration method is symplectic, the result will remain numerically stable over a long simulation time. We choose Sun's RadauII symplectic Runge–Kutta method [18] of 3-stage order 5 as the integration method in this paper. The coefficients of this Runge–Kutta method can be found in Table 1.

Table 1 Sun's RadauII symplectic Runge–Kutta method of 3-stage order 5 [18]

$\frac{4-\sqrt{6}}{10}$	$\frac{16-\sqrt{6}}{72}$	$\frac{328-167\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{450}$
$\frac{4+\sqrt{6}}{10}$	$\frac{328+167\sqrt{6}}{1800}$	$\frac{16+\sqrt{6}}{72}$	$\frac{-2-3\sqrt{6}}{450}$
1	$\frac{85-10\sqrt{6}}{180}$	$\frac{85+10\sqrt{6}}{180}$	$\frac{1}{18}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Different from the Newmark method, Runge–Kutta method is designed for the first-order ODE:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y) \tag{16}$$

To use implicit Runge–Kutta method to solve ODE (9), ODE (9) should first be transformed to Eq. (16) by setting $y = [\dot{q} \quad q]^T$.

The general formula of a s-stage implicit Runge–Kutta method for the first-order ordinary differential equation (16) is:

$$K_i = f\left(x_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij} K_j\right) \quad i = 1, 2, \ldots, s \tag{17a}$$

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i K_i \tag{17b}$$

If the right-hand function $f(x, y)$ is a nonlinear function of x and y, Eq. (17a) is a set of nonlinear algebraic equations with $K_i (i = 1, 2, \ldots, s)$ as unknowns. This set of nonlinear algebraic equations can be solved by using Newton's method, but there needs the extra Jacobian matrix $\partial f/\partial K_i$. Even though the Jacobian matrix $\partial f/\partial K_i$ may be simplified, the numerical differential is still necessary [16].

For implicit Runge–Kutta methods, $K_i (i = 1, 2, \ldots, s)$ are functions of y_{n+1}. If we eliminate $K_i (i = 1, 2, \ldots, s)$ in Eq. (17b), Eq. (17b) becomes a set of nonlinear algebraic equations with y_{n+1} as unknowns:

$$y_{n+1} = f(y_{n+1}) \tag{18}$$

Equation (18) is a set of nonlinear algebraic equations with position q_{n+1} and velocity $\dot{q}_{n+1}$ as unknowns. The benefit of using Eq. (18) instead of Eq. (17a) is that the dimension of the nonlinear algebraic equation is reduced from $(2 \times s \times n_q)$ in Eq. (17a) to $(2 \times n_q)$ in Eq. (18).

Here, we propose a fixed-point strategy to solve Eq. (18). By introducing this fixed-point iterative method, the Jacobian matrix $\partial f/\partial y$ is no longer needed. The process of integrating ODE (16) in a time step is:

1. predict y_{n+1} by a low-order integral method;
2. predict K_i by using $K_i = f(t_n + c_i h, y_{n+1})$;
3. while(.NOT.converged) do
4. for $i = 1, 2, \ldots, s$ do
5. calculate K_i by Eq. (17a);
6. end do;
7. calculate y_{n+1} by using (17b);
8. end do.

This process of using implicit Runge–Kutta method is just for the set of ODE (9). Step 1 gives the initial value of the fixed-point iteration, and an explicit one-step integration method, for example, explicit Euler's method, is recommended. The convergence criterion used in step 3 is based on $y = [\dot{q} \quad q]^T$: When the error of y is smaller than a given norm, the iteration is finished.

The fixed-point iterative method for implicit Runge–Kutta method is different from the fixed-point iteration method for the Newmark method because the nonlinear algebraic equations of these two implicit integration methods are different. The unknowns in the nonlinear algebraic equations of implicit Runge–Kutta method are a collection of position q_{n+1} and velocity $\dot{q}_{n+1}$, and the acceleration $\ddot{q}_{n+1}$ does not appear in the nonlinear algebraic equation of the implicit Runge–Kutta method.

The outer loop in the two-loop procedure is based on the fixed-point iteration of Eq. (18), but the ODEs solved in the outer loop are the ODEs picked out from Eq. (8) which correspond to the independent coordinates. The convergence criterion of the outer loop is based on both the independent and dependent coordinates. The details will be described in Sect. 3.3.

3.2 Structure of the new two-loop procedure

After the coordinate partitioning process, the ODEs which need to be integrated can be picked out from Eq. (8):

$$\begin{aligned} \ddot{q}_i &= F(t, \dot{q}, q) \\ &= F(t, \dot{q}_i, \dot{q}_d, q_i, q_d) \end{aligned} \tag{19}$$

Here, we set $y = [\dot{q} \quad q]^T$ and $y_i = [\dot{q}_i \quad q_i]^T$ and $y_d = [\dot{q}_d \quad q_d]^T$. Then, Eq. (19) becomes a set of first-order ODEs:

$$\frac{\mathrm{d}y_i}{\mathrm{d}t} = f(t, y_i, y_d) \tag{20}$$

Equation (20) is translated to a set of algebraic equations when the implicit Runge–Kutta method is applied. Using the fixed-point iteration for implicit Runge–Kutta method introduced in Sect. 3.1 to handle Eq. (20), we have the nonlinear algebraic equation when time is fixed:

$$y_i = f(y_i, y_d) \tag{21}$$

The construction of the new two-loop procedure is based on the fixed-point iteration of Eq. (21). The unknown in Eq. (21) is y_i, and y_d should be evaluated when Eq. (21) is being solved. The iteration of the nonlinear algebraic equations of position constraint (3) and the solving of the linear algebraic equations of velocity constraint (4) give the value of y_d.

After the coordinate partitioning process, Eq. (3) becomes:

$$C(q_i, q_d) = 0 \tag{22}$$

Equation (4) becomes:

$$[C_{qi} \quad C_{qd}] \begin{bmatrix} \dot{q}_i \\ \dot{q}_d \end{bmatrix} = -C_t \tag{23}$$

where $[C_{qi} \quad C_{qd}]$ is the result of the rearrangement of the Jacobian matrix C_q and $C(q_i, q_d)$ is the result of the rearrangement of the function $C(q)$. In this paper, we use Eqs. (14) and (15) to avoid the rearrangement of C_q and $C(q)$.

In Eqs. (22) and (23), y_i is known while y_d is unknown. Equation (23) gives an explicit function of $\dot{q}_i$:$\dot{q}_d = g_1(\dot{q}_i)$. Equation (22) is an implicit function. Once y_i is given, y_d can be obtained by using numerical methods, so we have $q_d = g_2(q_i)$. We have a function of y_i:

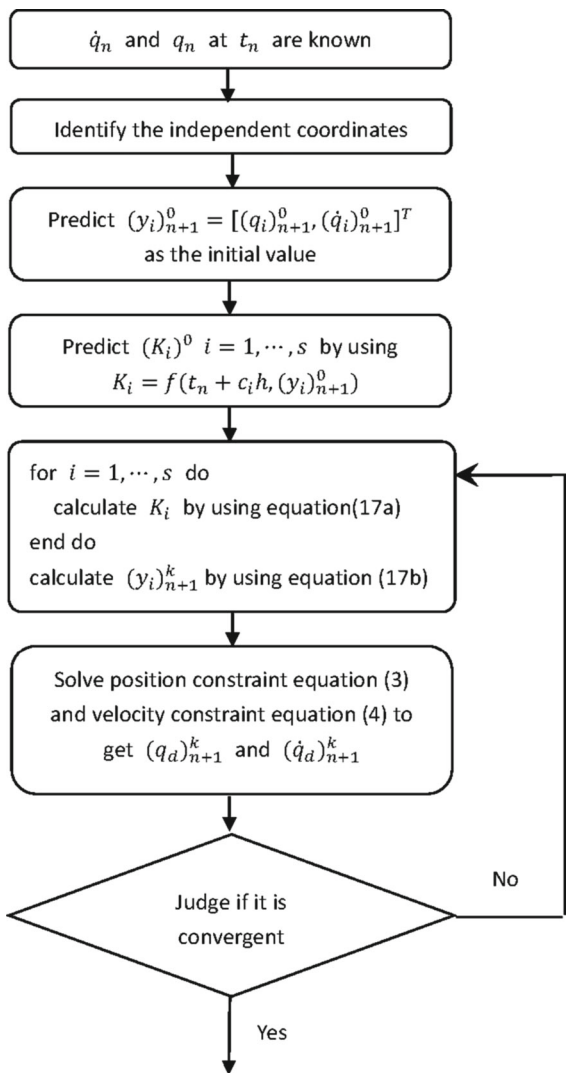
$$y_d = \begin{bmatrix} \dot{q}_d \\ q_d \end{bmatrix} = \begin{bmatrix} g_1(\dot{q}_i) \\ g_2(q_i) \end{bmatrix} = g(y_i) \tag{24}$$

If the solving of Eqs. (3) and (4) is not embedded in the solving of Eq. (20), y_d in Eq. (21) is treated as constant value in a time step so that Eq. (21) is solvable, and it means that y_i is variable while y_d is not variable in this time step. By embedding the solving of Eqs. (3) and (4) into the fixed-point iteration, y_d in Eq. (20) and (21) is eliminated by using Eq. (24), and Eq. (21) becomes

$$y_i = f(y_i) \tag{25}$$

So in the two-loop procedure, both y_i and y_d are variable in a time step.

The flow chart of the new two-loop procedure in a time step is:



The convergence criterion of the new two-loop procedure is based on both y_i and y_d, and it should be handled together with the estimate of the local error, the choice of new size of the time step, and the determining of the maximum size of the time step. The details are discussed in Sect. 3.3.

3.3 Variable step size strategy, convergence criteria, and maximum time step

For practical use of the new two-loop procedure, the variable step size strategy is needed. The variable step size strategy is based on the estimation of the local error of the new two-loop procedure. At the same time, the convergence criterion is also based on the estimation of the local error. The determination of the local error and the new time step size of the Runge–Kutta method can be found in [17]. In this paper, we expand the method to the new two-loop procedure. The new two-loop procedure is proposed for the DAEs, so the dependent velocities and positions which are not the result of the Runge–Kutta method are also used in the calculation of the local error and the new step size.

To avoid the confusion caused by the symbols, we rewrite Eq. (16) as:

$$\frac{\mathrm{d}z}{\mathrm{d}x} = f(x, z)$$

It is difficult to get the local error of an arbitrary Runge–Kutta method by analytic method, so we use Richardson extrapolation to calculate the local error. The process includes two steps: Use time step h to integrate one time to get z_{n+1}, and then, use time step $h/2$ to integrate twice to get $\hat{z}_{n+1}$. The convergence is based on the following condition:

$$\begin{aligned} &|z_{n+1,j} - \hat{z}_{n+1,j}| \le sc_j \\ &sc_j = Atol_j + max(|z_{n,j}|, |z_{n+1,j}|) \cdot Rtol_j \end{aligned} \tag{26}$$

In component-wise (26), sc is the desired tolerance prescribed by the user, $Atol$ represents the absolute errors and $Rtol$ represents the relative errors. We take:

$$err = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(\frac{z_{n+1,j} - \hat{z}_{n+1,j}}{sc_j}\right)^2} \tag{27}$$

as the measurement of the errors. If $err \leq 1$, the step size h is accepted and the convergence is achieved.

By adding y_d in $z_{n+1,j}$ and $\hat{z}_{n+1,j}$, Eqs. (26) and (27) can be applied to calculate the local error of the new two-loop procedure. Here, $z_{n+1,j}$ and $\hat{z}_{n+1,j}$ do not represent the independent coordinate y_i which are given by the Runge–Kutta method, but include the updated dependent velocities and positions included in y_d: $z = [y_i, y_d]^T$. So in the new two-loop procedure, n in Eq. (27) equals n_q rather than n_{DOF}.

The error can be presented as:

$$err \approx c \cdot h^{q+1} \tag{28}$$

We hope the optimal time step size h_{opt} may be:

$$1 \approx c \cdot h_{opt}^{q+1} \tag{29}$$

So we get:

$$h_{opt} = h \cdot (1/err)^{1/(q+1)} \tag{30}$$

It is necessary to multiply h_{opt} by a safety factor fac; meanwhile, a upper limit factor $facmax$ and a lower limit factor $facmin$ are set to prevent the time step size from changing too rapidly. So the new step size is given by:

$$h_{new} = h \cdot min(facmax, max(facmin, fac \cdot (1/err)^{1/(q+1)})) \tag{31}$$

The convergence criterion mentioned above is used for the whole two-loop procedure, so a convergence criterion is needed for the iteration of the nonlinear algebraic equation (25). The convergence criterion of the nonlinear algebraic equations is only based on y_i:

$$\max_{1 \leq j \leq n} \left| \frac{\triangle(y_i)_j}{max(\triangle(y_i)_j, 1)} \right| \leq \epsilon_{tol}$$
$$\triangle(y_i) = (y_i)_{n+1}^{k+1} - (y_i)_{n+1}^{k} \tag{32}$$

where the superscript k represents the kth iterative process and ϵ_{tol} is a tolerance set by the user.

As reported in Ref. [14], for some dynamic problems of multibody systems, the two-loop procedure needs a smaller step size than the step size given by Eq. (31) to achieve a convergence. Here arises the problem of estimating the maximum step size. The method of calculating the maximum step size has been given in [14], but in the actual implementation, it is not convenient to get the maximum time step by using the analytical method. We introduce a convergence judging process to the new two-loop procedure to get the maximum time step size h_{max} just as mentioned in Ref. [14]. If the criterion:

$$\sqrt{\sum_{j=1}^{n} \left(z_{n+1,j}^{k+1} - z_{n+1,j}^{k} \right)^2} \leq \sqrt{\sum_{j=1}^{n} \left(z_{n+1,j}^{k} - z_{n+1,j}^{k-1} \right)^2} \tag{33}$$

is satisfied, the step size h is considered to be smaller than the maximum step size, otherwise the maximum step size is reset as $h_{max} = h$ and the outer loop should be restarted with a smaller step size than h, for example $h/2$. At the beginning of each time step, the maximum step size is set as $h_{max} = 1$. To consider the maximum step size in each time step, the new step size is calculated by:

$$h_{new} = min(h_{max}, h \cdot min(facmax, max(facmin, fac \cdot (1/err)^{1/(q+1)}))) \tag{34}$$

The convergence criteria of the new two-loop procedure includes three judging processes, these judging processes make the specific process of the new two-loop procedure very complex.

The process of the judgement of convergence is:

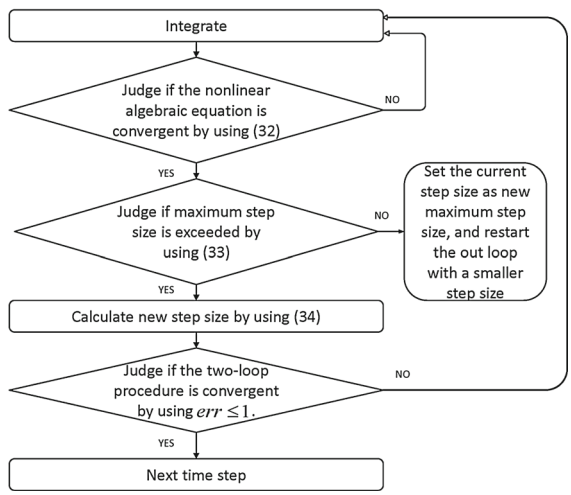


Table 2 Parameters of the four-bar linkage system

m1	L1	m2	L2	m3	L3	L
277.0 kg	0.5 m	221.6 kg	0.4 m	110.8 kg	0.2 m	0.4 m

4 Numerical examples

In this section, we exhibit two examples to show the advantages and disadvantages of the method we proposed.

4.1 Four-bar linkage

Figure 1 is a typical four-bar linkage mechanism. This kind of mechanism is very common in machinery. There is a closed chain in the mechanism, so for most modeling methods, the mathematical model of this mechanism is a set of DAEs.

The dynamic model of this system is based on a set of Cartesian coordinates. Each bar has three coordinates, and the coordinates of each bar are $[x \quad y \quad \theta]$. The three differential equations of the ith bar are:

$$\begin{cases} m_i \ddot{x}_i = Q_x \\ m_i \ddot{y}_i = Q_y \\ J_i \ddot{\theta}_i = M_z \end{cases} \qquad i = 1, 2, 3$$

where m is the mass and J is the moment of the inertia; Q_x and Q_y represent the resultant external force; M_z is the resultant moment of force.

Each bar has constraints on its two ends, and there are four joints in this system which introduce eight constraints. Two constraint equations of point A are:

$$L_1 \sin(\theta_1) + L_2 \sin(\theta_2) + L_3 \cos(\theta_3) = 0 \tag{35}$$

$$L_1 \cos(\theta_1) + L_2 \cos(\theta_2) + L_3 \cos(\theta_3) - L = 0 \tag{36}$$

The mathematical model for the dynamics of the system is a set of DAEs with nine ordinary differential equations and eight algebraic equations. There are nine coordinates in the equations while the degree of freedom of the system is 1, and it means that there are eight dependent coordinates in the inner loop to be determined by the only independent coordinate. We use this example to test if the new two-loop procedure can handle the constraint part of the DAEs.

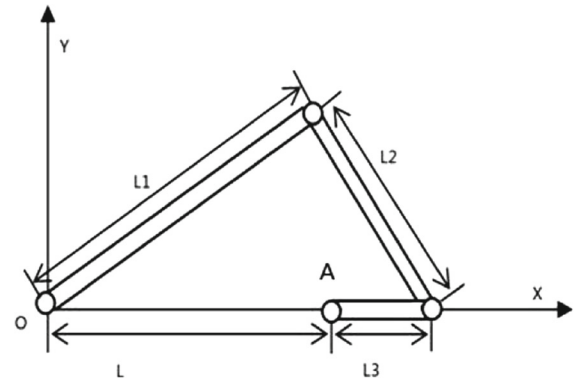


Fig. 1 Four-bar linkage

The parameters of the systems can be found in Table 2. The only external force is gravity, so the system is conservative. At the beginning of the simulation, the position of the third bar is horizontal as shown in Fig. 1. To this conservative system, the conservation of energy is a criterion, and it means that the vertical position of the center of the third bar will reach and not pass the line $y = 0$.

We use two kinds of the two-loop procedures to solve the DAE: the two-loop procedure with the Newmark method, and the new two-loop procedure with implicit symplectic Runge–Kutta method. For the two-loop with the Newmark method, the parameters used in the convergence criteria can be found in Ref. [14] and the parameters in Eq. (12) are $\beta = 0.5$ and $\gamma = 0.5$; for the two-loop procedure with implicit Rung–Kutta method, the parameters used in the convergence criteria are: $Atol = 0.00001$, $Rtol = 0.00001$, $fac = 0.9$, $facmax = 5.0$, $facmin = 0.5$. The tolerance of the error of both the inner loop and the outer loop of these two two-loop procedures is 10^{-9}. The result of the commercial software ADAMS with solver Gstiff is also shown to demonstrate the advantages of the new two-loop procedure. The two two-loop procedures both use variable step size while the Gstiff solver uses fixed step size.

Figures 2, 3, 4, and 5 are the results of the y position of the center of the third bar. Figure 2 shows the results of these methods in short time; Figs. 3, 4, and 5 show the results of a long-time (100 s) simulation of these methods. The step size of these two two-loop procedures is shown in Fig. 6.

When the step size of Gstiff solver is $h = 0.02$ s which is smaller than the step size of the two-loop pro-

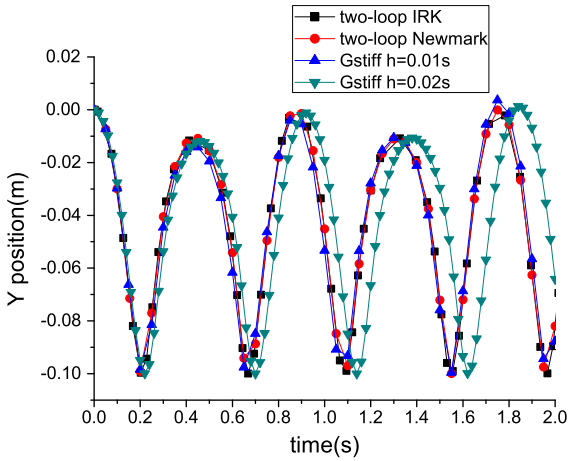


Fig. 2 Y position of the center of the third bar (0–2 s), calculated by different methods

cedure with implicit Runge–Kutta method in most time as shown in Fig. 6, the results of Gstiff solver are different from the results obtained by using the two different two-loop procedures. If the step size is small enough, the Gstiff solver gives reasonable result in the short-time simulation. We can find that, in short-time simulation, the results of the two two-loop procedures and the Gstiff solver with $h = 0.01$ s coincide well. From the result of long-time simulation, we find that the curves of the solution of the two two-loop procedures reach and do not pass the line $y = 0$, while the Gstiff solver with $h = 0.01$ s gives obviously wrong result in long-time simulation.

The tolerances of the error of the inner loop err_{in} and the outer loop err_{out} affects the computation time and step size of the two two-loop procedures, but the effect is different when the integration method is different. The computation time of the two-loop procedure with the Newmark method is mainly affected by the tolerance of the outer loop, and the tolerance of the inner loop makes little influence on the computation time. For the new two-loop procedure with implicit Runge–Kutta method, the tolerance of the inner loop is the main factor that influences the computation time. When the tolerance of the outer loop is set as 10^{-3}, the two-loop procedure with the Newmark method cannot give reasonable result in the long-

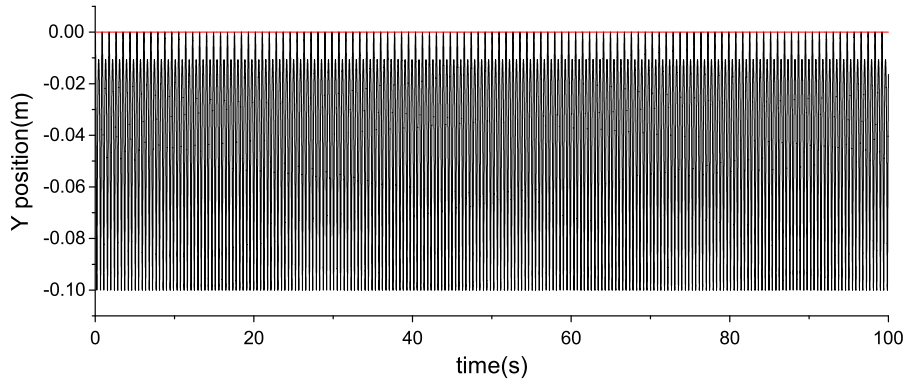


Fig. 3 Y position of the center of the third bar (0–100 s), calculated by the two-loop procedure with the Newmark method

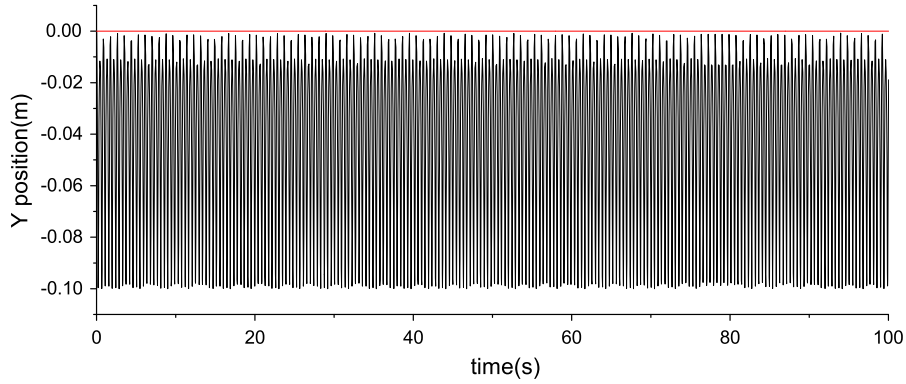


Fig. 4 Y position of the center of the third bar (0–100 s), calculated by the two-loop procedure with implicit Runge–Kutta method

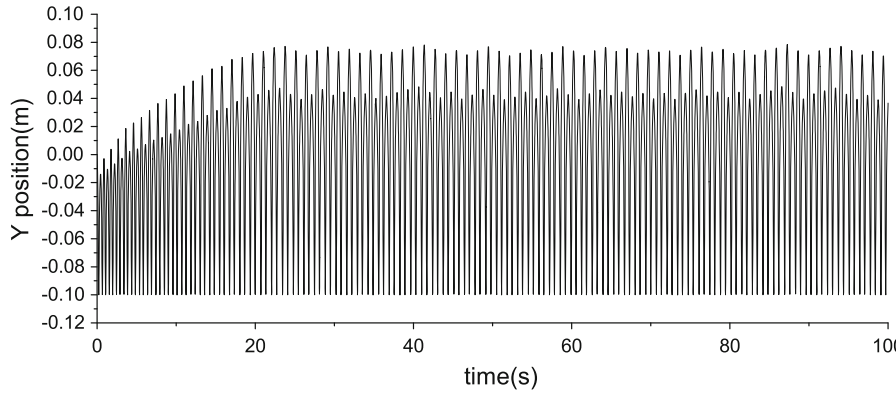


Fig. 5 Y position of the center of the third bar (0–100 s), calculated by ADAMS solver Gstiff ($h = 0.01$ s)

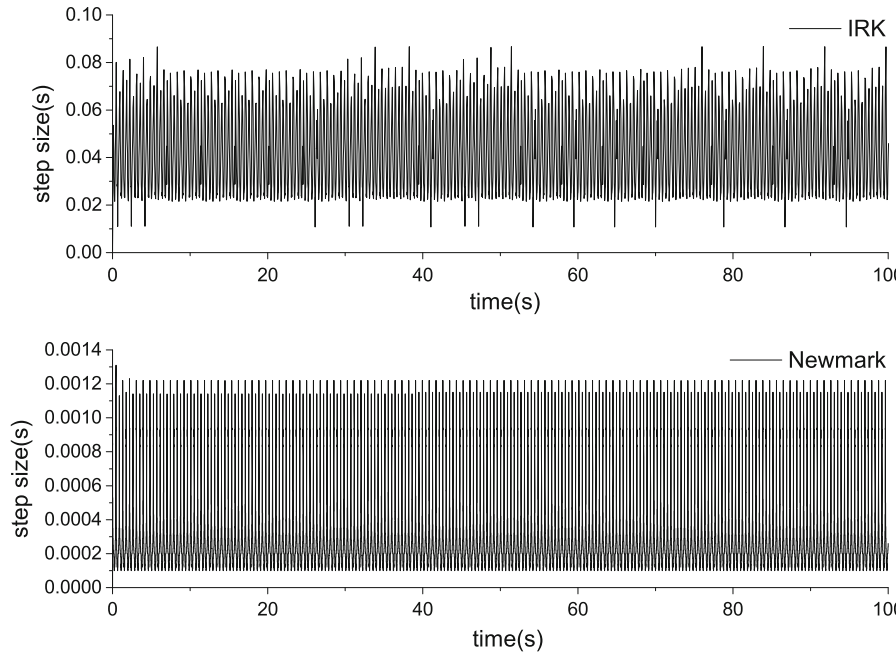


Fig. 6 Change of step size with time, comparison of the two-loop procedure with different integration methods: the Newmark method and implicit Runge–Kutta (IRK) method

time simulation while the new two-loop procedure with implicit Runge–Kutta method can. The computation time of different tolerances is shown in Table 3. When the tolerance of the error of both inner loop and outer loop of these two two-loop procedures is 10^{-9}, the computation time of the two-loop procedure with the Newmark method is 50.93 s and the time of the two-loop procedure with implicit Runge–Kutta method is 3.60 s.

Figure 7 shows the violation of Eqs. (35) and (36) in the new two-loop procedure. The result shows that the new two-loop procedure gives result with very small violation of the nonlinear constraints, and the amplitude does not increase as time increasing.

4.2 Flexible beam modeled by absolute nodal coordinate formulation (ANCF)

We use absolute nodal coordinate formulation (ANCF) to model a flexible beam which is connected to a moving base through a revolute joint, as shown in Fig. 8. ANCF is a kind of non-incremental finite element method, and it can model very flexible bodies in multibody systems [19].

The element which we use here is the two dimensional shear element, and the coordinates of each node is

$$e = \left[\boldsymbol{r}^T, \frac{\partial \boldsymbol{r}}{\partial x}^T, \frac{\partial \boldsymbol{r}}{\partial y}^T \right]^T$$

Table 3 The computation time of the two two-loop procedures with different tolerance of inner loop and outer loop (Unit: second)

		err_{out}							
		10^{-3}		10^{-5}		10^{-7}		10^{-9}	
err_{in}	10^{-3}	1.95	$\star$	1.98	2.71	2.00	12.62	2.08	49.73
	10^{-5}	2.12	$\star$	2.23	2.74	2.31	12.36	2.31	49.68
	10^{-7}	2.64	$\star$	2.85	3.10	2.85	12.68	2.99	49.66
	10^{-9}	3.37	$\star$	3.46	3.09	3.55	13.40	3.60	50.93

Left:the two-loop procedure with IRK

Right:the two-loop procedure with the Newmark method

$\star$:no reasonable result in long time simulation

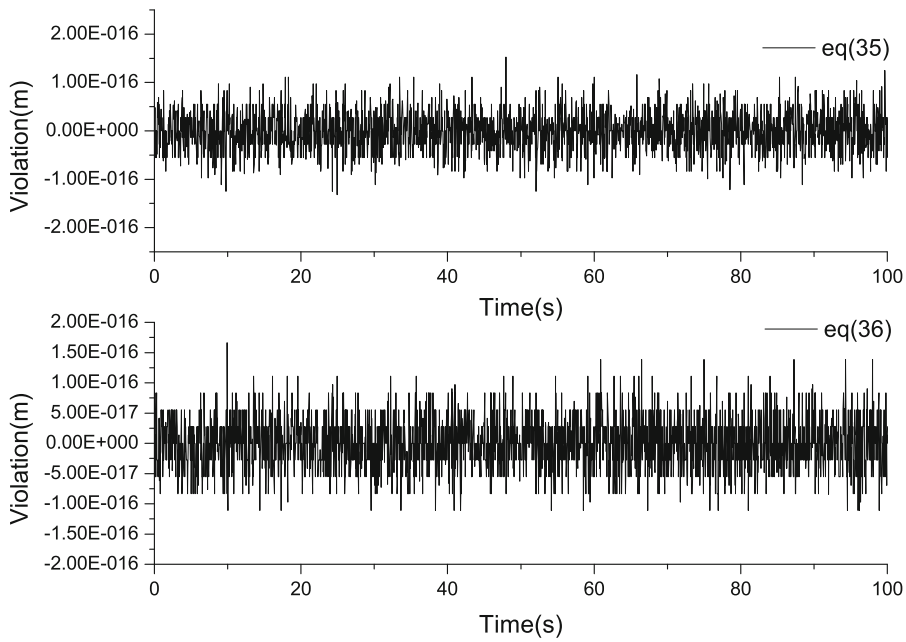


Fig. 7 Violation of the last two position constraint equations, result of the two-loop procedure with implicit Runge–Kutta method

where $\boldsymbol{r}$ is the global positions of the point on the central line of the beam. The elastic force of the beam is calculated by using the continuum mechanics approach.

Fig. 8 Flexible beam

The Green–Lagrange strain tensor $\boldsymbol{\epsilon}$ can be expressed in terms of the nodal coordinates. The strain vector can be written as

$$\boldsymbol{\epsilon} = [\epsilon_{11} \quad \epsilon_{22} \quad 2\epsilon_{12}]^T$$

where ϵ_{22} is the deformation of the cross section of the beam, and this deformation is not included in other beam models.

In the continuum mechanics approach, the elastic force is computed by:

$$Q = -\int_V \boldsymbol{\epsilon} E \frac{\partial \boldsymbol{\epsilon}}{\partial e} \mathrm{d}V$$

where E is the matrix of the elastic constants of the material:

$$E = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{bmatrix}$$

where λ and μ are Lame coefficients. This integral is calculated by Gauss quadrature formula.

Because in the continuum mechanics approach, the deformation of the cross section of the beam introduces high-frequency components and the differential equations that come from ANCF are very stiff [14]. One of the features of ANCF is that the computational efficiency of this method is very low. How to improve the computational efficiency is always the main concern in the study of ANCF since it has been proposed. No matter whether the mathematical model is ODE or DAE, an implicit integration method is needed. The example we used here can be found in [20]. We use this example to test if the new two-loop procedure can be applied to the stiff problem.

The base of the beam is subjected to a known motion which is defined by:

$$X = X_0 \sin(\omega t)$$

So the position constraint equations of the left end point of the beam are:

$$C = \begin{bmatrix} q(1) - X_0 \sin(\omega t) \\ q(2) \end{bmatrix} = 0$$

where $q(1)$ and $q(2)$ represent the positions of the left end point of the beam.

First, we apply the new two-loop procedure to the very flexible structure example. In this example, the beam has a square cross section with side dimension of 0.02 m. The length of the beam is assumed to be 1 m, the mass density is assumed to be 7200 kg/m^3, the modulus of elasticity is assumed to be 2×10^6 N/m^2, and the Poisson ratio is assumed to be 0.3. $X_0 = -0.02$ m, $\omega = 10.0\pi$. The beam is meshed into four elements as in Ref. [20]. The result of tip vertical position, the transverse deformation of the midpoint of the beam, and the step size are shown from Figs.9, 10, and 11,

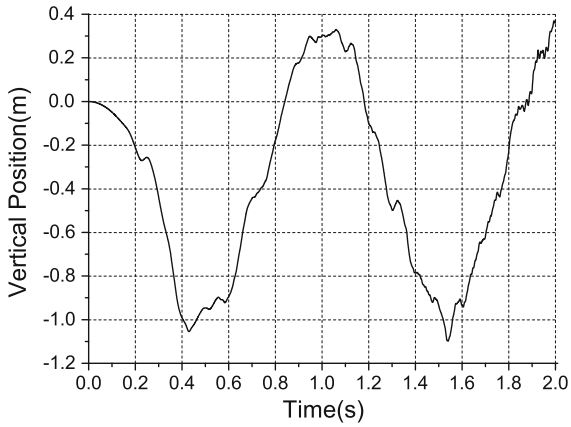


Fig. 9 Tip vertical position of the very flexible structure example

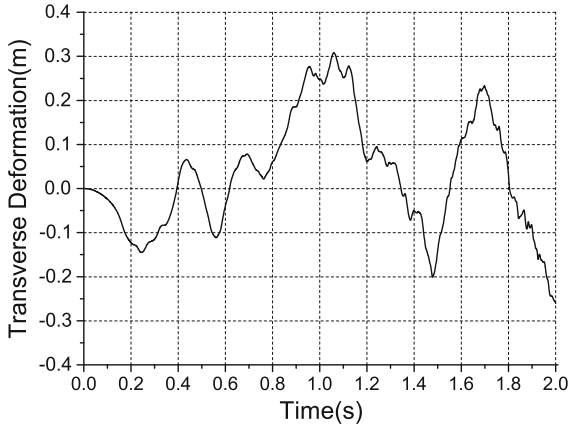


Fig. 10 Transverse deformation of the midpoint of the beam of the very flexible structure example

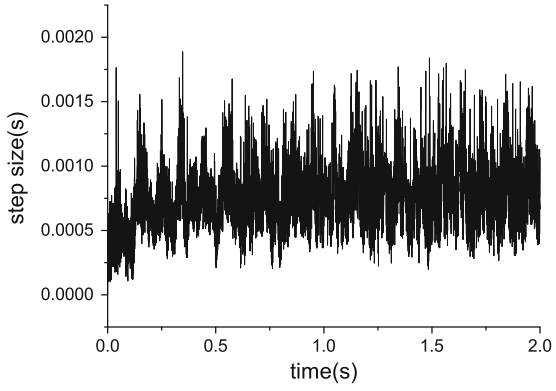


Fig. 11 Step size of the very flexible structure example

respectively. The total computation time is 56.27 s. The magnitude order of the step size is 10^{-3}.

Secondly, we apply the new two-loop procedure to the moderately stiff structure example. In this example,

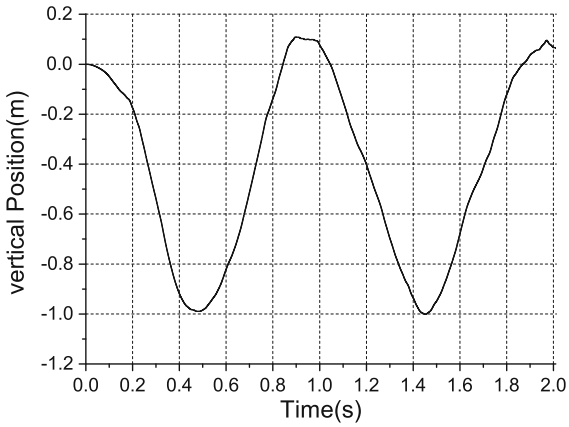


Fig. 12 Tip vertical position of the moderately stiff structure example

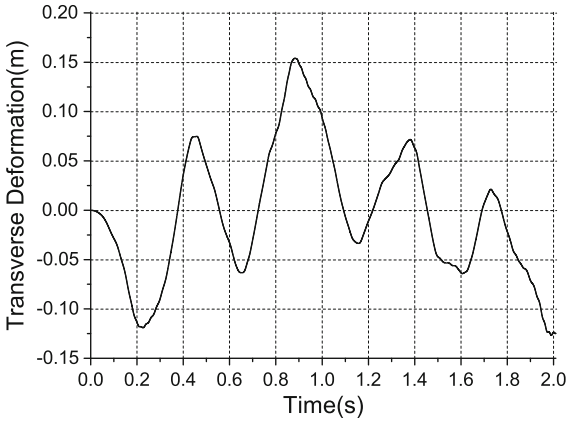


Fig. 13 Transverse deformation of the midpoint of the beam of the moderately stiff structure example

$X_0 = -0.01$, $\omega = 2.0\pi$, and the modulus of elasticity is assumed to be $2 \times 10^8\,\mathrm{N/m^2}$. This example is much stiffer than the previous example. The result of the tip vertical position, the transverse deformation of the midpoint of the beam, and the step size are shown from Figs. 12, 13, and 14, respectively. The total computation time is 6278.54 s. The magnitude order of the step size is 10^{-5}.

The correctness of the result of the new two-loop procedure for this example can be demonstrated by comparing the result here with the result presented in Ref. [20]. For the moderately stiff situation, the results in this paper consist with the results in Ref. [20].For the very flexible situation, from 0 to 1.2 s, the results in this paper consist with the results in Ref. [20]. After 1.2 s,

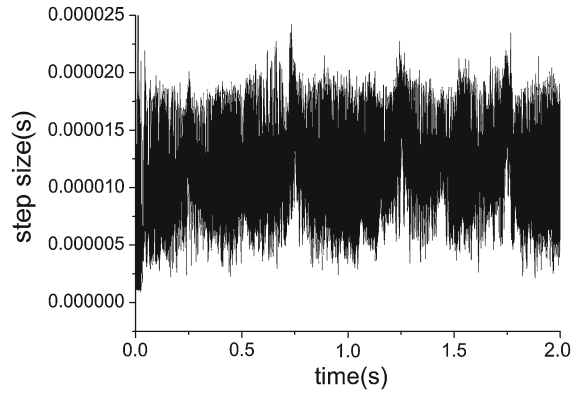


Fig. 14 Step size of the moderately stiff structure example

the results in this paper are different from the results in Ref. [20], but the results are still reasonable.

For both very flexible and moderately stiff structure examples, the new two-loop procedure works, but when the examples become stiffer, the computation time increases a lot.

5 Conclusion and discussion

The two-loop procedure with implicit symplectic Runge–Kutta method as integration method can solve the DAEs of constrained mechanic systems. The fixed-point iterative strategy applied to the implicit Runge–Kutta method works, so the extra Jacobian matrix is no more needed. In the process, only the numerical result rather than the analytic formulas of the mass matrix, generalized force array, and the Jacobian matrix of position constraint equations are needed. This fact makes the method good in generality. Meanwhile, the constraints can be satisfied at position level, velocity level, and acceleration level at the same time. The violation of the constraints is controlled at a very low level. This new two-loop procedure works well over a long-time simulation. The computation time is reduced because the order of numerical accuracy of the implicit Runge–Kutta method is higher than the order of the Newmark method.

The new two-loop procedure with position and velocity as basic unknowns is designed for any integration method that works for the first-order ODE (16), and the implicit Runge–Kutta method is just a particular case. Any implicit integration method that gives nonlinear algebraic equations (21) can be used as the

integration method in this new two-loop procedure. By choosing different variables as basic unknowns, there may be different two-loop procedures. The two-loop procedures proposed in this paper and Ref. [13] are just two particular cases. The basic unknowns of the two-loop procedures can be position or velocity or acceleration or the combination of different variables, so any integration method which can be used to solve the second-order ODE (10) can be used in the two-loop procedures as the integration method.

For a particular integration method, there may exist several different two-loop procedures with different variables as basic unknowns. The two-loop procedures with the same integration method but different variables as the basic unknown have different structures. The optimal structure of the two-loop procedure based on a particular integration method should make it convenient to introduce the fixed-point iterative method and the variable step size strategy for the integration method in the outer loop. For example, the Newmark method can be used in the two-loop procedure with acceleration or velocity or position as basic unknown. The optimal structure of the two-loop procedure with the Newmark method as integration method is the two-loop procedure proposed in Ref. [13], because it is convenient to construct the fixed-point iterative method and the variable step size strategy by using acceleration as the basic unknown.

The idea of applying backward differentiation formulas (BDF) to the two-loop procedure has been mentioned in Ref. [14]. Like implicit Runge–Kutta method, BDF can be used to solve the first-order ODEs (16), and the nonlinear algebraic equations of BDF have the form of Eq. (21). So the two-loop procedure based on BDF with position and velocity as basic unknowns is similar to the two-loop procedure presented in this paper. Besides the combination of position and velocity, the nonlinear algebraic equations of BDF can use other variables as the basic unknowns. So how to use BDF as the integration method in the two-loop procedure and what is the optimal structure of the two-loop procedure based on BDF should be further studied in another paper.

For non-holonomic systems, besides the holonomic constraints, there are non-holonomic constraints in the systems. The non-holonomic constraints constrain the velocities directly, and the corresponding position constraint equations do not exist. The number of the position constraint equations is less than the number of the dependent positions in non-holonomic systems. The set of velocity constraint equations of the non-holonomic systems is the combination of the differential of the holonomic position constraint equations and the non-holonomic constraint equations. When the two-loop procedure is applied to non-holonomic systems, the coordinate partitioning technique is still available to identify the independent coordinates because the Jacobian matrix C_q still exists. In the two-loop procedure, the velocity constraints and the position constraints are handled separately. Once the integration method gives the independent velocities and positions, the dependent velocities can be obtained by solving the velocity constraint equations. As to the positions, it is impossible to obtain all the positions by solving the position constraint equations. The dependent positions need to be determined by other methods, for example, integrating the velocities, so that the two-loop procedure may work.

Acknowledgments This research is supported by the National Nature Science Foundations of China (11272155,11132007), the 333 project of Jiangsu Province in China (BRA2011172), and the Fundamental Research Funds for Central Universities (30920130112009).

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Human and animals participants This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Shabana, A.A.: Dynamics of Multibody Systems. Cambridge University Press, New York (2005)
2. Ascher, U.M., Petzold, L.R.: Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations. Siam, Philadelphia (1998)
3. Petzold, L.R.: Numerical solution of differential–algebraic equations in mechanical systems simulation. Physica D **60**, 269–279 (1992)
4. Blajer, W.: Methods for constraint violation suppression in the numerical simulation of constrained multibody systems—a comparative study. Comput. Methods Appl. Mech. Eng. **200**, 1568–1576 (2011)
5. Negrut, D., Ottarsson, G., Rampalli, R., Sajdak, A.: On an implementation of the Hilber–Hughes–Taylor method in the context of index 3 differential-algebraic equations of multi-

body dynamics (DETC2005-85096). J. Comput. Nonlinear Dyn. **2**, 73–85 (2007)
6. Hussein, B., Negrut, D., Shabana, A.A.: Implicit and explicit integration in the solution of the absolute nodal coordinate differential/algebraic equations. Nonlinear Dyn. **54**, 283–296 (2008)
7. Haug, E.J., Negrut, D., Engstler, C.: Implicit Runge–Kutta integration of the equations of multibody dynamics in descriptor form. J. Struct. Mech. **27**, 337–364 (1999)
8. Negrut, D., Haug, E.J., Horatiu, C.G.: An implicit Runge–Kutta method for integration of differential algebraic equations of multibody dynamics. Multibody Syst. Dyn. **9**, 121–142 (2003)
9. Sandu, A., Haug, E.J., Potra, F.A., Sandu, C.: A Rosenbrock-Nystrom state space implicit approach for the dynamic analysis of mechanical systems: I—theoretical formulation. Proc. Inst. Mech. Eng. Part K J. Multi-body Dyn. **217**, 263–271 (2003)
10. McGrath, J.F., Rampalli, R.: Implicit integration with coordinate partitioning. Appl. Math. Comput. **111**, 7–31 (2000)
11. Fisette, P., Vaneghem, B.: Numerical integration of multibody system dynamic equations using the coordinate partitioning method in an implicit Newmark scheme. Comput. Methods Appl. Mech. Eng. **135**, 85–105 (1996)
12. Haug, E.J., Negrut, D., Iancu, M.: A state-space-based implicit integration algorithm for differential–algebraic equations of multibody dynamics. J. Struct. Mech. **25**, 311–334 (1997)
13. Shabana, A.A., Hussein, B.: A two-loop sparse matrix numerical integration procedure for the solution of differential/algebraic equations: application to multibody systems. J. Sound Vib. **327**, 557–563 (2009)
14. Hussein, B., Shabana, A.A.: Sparse matrix implicit numerical integration of the Stiff differential/algebraic equations: implementation. Nonlinear Dyn. **65**, 369–382 (2011)
15. Shabana, A.A.: Computational Dynamics. Wiley, Chichester (2009)
16. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II: Stiff and Differential–Algebraic Problems. Springer, Berlin (1996)
17. Hairer, E., Nørsett, S.P., Wanner, G.: Solving Ordinary Differential EquationsI: Nonstiff problems. Springer, Berlin (1993)
18. Sun, G.: Construction of high-order symplectic Runge–Kutta methods. J. Comput. Math. **11**, 250–260 (1993)
19. Marcello, C., Berzeri, M., Shabana, A.A.: Performance of the incremental and non-incremental finite element formulations in flexible multibody problems. J. Mech. Des. **122**, 498–507 (2000)
20. Hussein, B., Sugiyama, H., Shabana, A.A.: Coupled deformation modes in the large deformation finite-element analysis: problem definition. J. Comput. Nonlinear Dyn. **2**, 146–154 (2007)