Efficient and robust implementation of the TLISMNI method

Ahmed K. Aboubakr
Ahmed A. Shabana

Department of Mechanical and Industrial Engineering,
University of Illinois at Chicago, 842 West Taylor Street,
Chicago, Illinois 60607.

ABSTRACT

The dynamics of large scale and complex multibody systems (MBS) that include flexible bodies and contact/impact pairs is governed by stiff equations. Because explicit integration methods can be very inefficient and often fail in the case of stiff problems,the use of implicit numerical integration methods is recommended in this case. This paper presents a new and efficient implementation of the two-loop implicit sparse matrix numerical integration (TLISMNI) method proposed for the solution of constrained rigid and flexible MBS differential and algebraic equations. The TLISMNI method has desirable features that include avoiding numerical differentiation of the forces, allowing for an efficient sparse matrix implementation, and ensuring that the kinematic constraint equations are satisfied at the position, velocity and acceleration levels. In this method, a sparse Lagrangian augmented form of the equations of motion that ensures that the constraints are satisfied at the acceleration level is used to solve for all the accelerations and Lagrange multipliers. The generalized coordinate partitioning or recursive methods can be used to satisfy the constraint equations at the position and velocity levels. In order to improve the efficiency and robustness of the TLISMNI method, the simple iteration and the Jacobian-Free Newton-Krylov approaches are used in this investigation. The new implementation is tested using several low order formulas that include Hilber–Hughes–Taylor (HHT), L- stable Park, A-stable Trapezoidal, and A-stable BDF methods. The HHT method allow for including numerical damping. Discussion on which method is more appropriate to use for a certain application is provided. The paper also discusses TLISMNI implementation issues including the step size selection, the convergence criteria, the error control, and the effect of the numerical damping. The use of the computer algorithm described in this paper is demonstrated by solving complex rigid and flexible tracked vehicle models, railroad vehicle models, and very stiff structure problems. The results, obtained using these low order formulas, are compared with the results obtained using the explicit Adams-Bashforth predictor-corrector method. Using the TLISMNI method, which does not require numerical differentiation of the forces and allows for an efficient sparse matrix implementation, for solving complex and stiff structure problems leads to significant computational cost saving as demonstrated in this paper. In some problems, it was found that the new TLISMNI implementation is 35 times faster than the explicit Adams-Bashforth method.

Keywords: TLISMNI implicit numerical integration; multibody system differential /algebraic equations; sparse matrix implementation; Jacobian-free Newton-Krylov, stiff equations.

1. Introduction

The numerical solution of constrained MBS problems requires the numerical integration of differential and algebraic equation (DAE) systems. Potra [1] and Negrut [2] discussed several techniques for the numerical solution of the DAE system in MBS dynamics. Large scale and complex MBS applications that include flexible bodies and contact/impact elements lead to stiff problems. Because explicit integration methods can be very inefficient and often fail in the case of stiff problems, the use of implicit numerical integration methods is recommended. A good understanding of the process of converting the DAE system to a second order ordinary differential equations (ODE) system, along with the ability to efficiently generate the needed derivative information for the implicit integration, allowed developing robust implicit numerical methods [3, 4]. Nonetheless, existing implicit numerical integration methods used for the solution of MBS applications have several drawbacks. First, most of these methods do not ensure that the nonlinear algebraic constraint equations are satisfied at all levels (position, velocity, and acceleration). Second, existing implicit methods require the use of numerical differentiation of the force vectors in order to solve a nonlinear system of algebraic equations using a Newton–Raphson algorithm; analytical differentiation cannot in general be used with general MBS algorithms that allow for the use of tabulated and Spline data to define the forces and constraints. Third, many of the existing implicit MBS integration methods are not suited for an efficient sparse matrix implementation because they lead to dense matrices. In order to address these limitations, the two-loop implicit sparse matrix numerical integration (TLISMNI) method was proposed [5]. The TLISMNI method ensures that the constraint equations are satisfied at the position, velocity, and acceleration levels, does not require the use of numerical or analytical differentiation of the forces, and allows for efficient sparse matrix implementation.

In the case of constrained MBS dynamics, the algebraic kinematic constraint equations are used to describe mechanical joints and specified motion trajectories that restrict relative and absolute motion of the MBS components. Different techniques have been proposed in the literature for the numerical integration of DAE systems. These techniques include the direct integration method, the generalized coordinates partitioning method, and the constraint stabilization method [3, 6]. The direct integration method employs a numerical integration method of ordinary differential equations to integrate the differential algebraic equations without any modification of the integration algorithm or dynamic equations. This integration method, which integrates the second time derivative of the generalized coordinates without considering their dependency, is simple, easy to implement, and computationally fast, but it suffers from a lack of error control on the constraints and may lead to erroneous results [7]. In the generalized coordinates partitioning method, proposed by Wehage and Haug [8], the system generalized coordinates are partitioned into dependent and independent coordinates, and only the independent accelerations are integrated to define the state of the system at the next time step. Dependent coordinates are obtained by solving the nonlinear algebraic constraint equations using a Newton-Raphson algorithm. This method has the advantage of ensuring that the constraint equations are satisfied to within a user specified tolerance. A good prediction of the dependent generalized coordinates can improve the efficiency of the method, since such a prediction can satisfy the constraint equations without the need for using Newton-Raphson iterations. Numerical experimentation has shown that, with a good prediction of the dependent variables, Newton-Raphson iterations are needed only for a few time steps during the dynamic simulation. Baumgarte [9] proposed a different approach based on a constraint stabilization method. In this approach, the constraint equations and their derivatives are used with the second derivative of the

constraint equations to form a penalty term. Using this penalty term, the constraint stabilization integration algorithm requires direct integration of all the accelerations. The disadvantage of this method, however, is that there is no general and uniformly accepted method for selecting the coefficients of the constraint equations and their derivatives in the penalty expression. An improper selection of these coefficients can lead to wrong solutions and make the method less robust.

As previously mentioned, the implicit integration methods can be more efficient than explicit integration methods in solving stiff DAE systems [10]. Implicit methods require the solution of a system of nonlinear algebraic equations at each time step. Newton's method or modified Newton's method appears to be the most widely used approach for iteratively solving the resulting implicit method nonlinear algebraic equations. For large problems, most of the calculations required for the implicit integration are associated with the computation of the Jacobian and the solution of the resulting large system of nonlinear equations. This is in addition to the need for a relatively large core memory for the storage of the coefficient matrix and its decomposition. Gear and Saad [11] proposed the use of Krylov-subspace projection method known as the *incomplete orthogonalization method* (IOM) and *Arnoldi's algorithm*, which are iterative methods for the solution of linear systems [12]. Arnlodi's algorithm and IOM do not require the factorization of the coefficient matrix in any form, and therefore, less storage as compared to the direct methods is needed. Brown and Hindmarsh [13] referred to the combined stiff ODE method and Krylov method as a matrix-free method and discussed the theoretical and computational aspects of the combined algorithm. Brown and Hindmarsh [13] viewed the combined Newton-IOM and Newton-Arnoldi method as an inexact Newton method, which belongs to a class of methods in which the linear system of Newton iteration is solved

approximately. The implementations of the Arnoldi's algorithm and IOM for solving linear system of equations are discussed in detail by Saad [12].

This paper discusses TLISMNI implementation issues, including the step size selection, the error control, and the effect of the numerical damping and proposes a new efficient and robust TLISMNI-based solution procedure for constrained MBS equations. Furthermore, the paper examines the use of the generalized coordinate partitioning and recursive methods in the TLISMNI framework solution; both approaches are used in order to satisfy the constraint equations at the position and velocity levels. In both cases, the constraints are also satisfied at the acceleration level by using the augmented Lagrangian form to solve for the accelerations and Lagrange multipliers. The use of the computer implementation described in this paper is demonstrated by solving complex rigid and flexible body tracked vehicle models, railroad vehicle models, and very stiff structural problems. Several second order formulas such as Hilber–Hughes–Taylor (HHT) method, including numerical damping, L- stable Park formula, A-stable Trapezoidal formula, and A-stable BDF formula are used in this investigation as the integration formulas, and recommendations are made with regard to the appropriateness of each of these integration formula for a particular MBS application. The efficient integration of the Krylov subspace projection method with these integration formulas within the TLISMNI framework solution procedure is one of the main contributions of this investigation. The results, obtained using these integration methods, are compared with the results obtained using the explicit Adams predictor-corrector method. The TLISMNI method does not require numerical differentiation of the forces, allows for an efficient sparse matrix implementation for solving complex and very stiff structure problems, and ensures that the constraint equations are satisfied at the position, velocity, and acceleration levels. The use of this method significantly improves

the computational efficiency as demonstrated in this paper using several examples. In some stiff problems, it was found that the new TLISMNI implementation is 35 times faster than the explicit Adams-Bashforth method.

2. Background

In this section, the constrained MBS equations of motion and the solution procedures used to solve these equations are briefly discussed. The definitions provided in this section will be used repeatedly in this paper.

2.1 MBS equations of motion

In MBS dynamics, the constraint relationships are used with the differential equations of motion to solve for the unknown accelerations and constraint forces. While this approach leads to a sparse matrix structure, it has the drawback of increasing the problem dimensionality and requiring more sophisticated numerical algorithms to solve the resulting DAE system. Using the generalized absolute Cartesian coordinates, the equations of motion of a body i can be written as [7,14]

$$\mathbf{M}^i\ddot{\mathbf{q}}^i = \mathbf{Q}_e^i + \mathbf{Q}_c^i + \mathbf{Q}_v^i \tag{1}$$

where $\mathbf{M}^i$ is the mass matrix of the body, $\ddot{\mathbf{q}}^i = \begin{bmatrix} \ddot{\mathbf{R}}^{iT} & \ddot{\boldsymbol{\theta}}^{iT} \end{bmatrix}^T$ is the vector of the accelerations of the body, $\mathbf{R}^i$ defines the body translation and $\boldsymbol{\theta}^i$ is a set of parameters that define the body orientation, $\mathbf{Q}_e^i$ is the vector of external forces, $\mathbf{Q}_c^i$ is the vector of the constraint forces, which can be written in terms of Lagrange multipliers $\boldsymbol{\lambda}$ as $\mathbf{Q}_c^i = -\mathbf{C}_{\mathbf{q}^i}^T\boldsymbol{\lambda}$, $\mathbf{C}_{\mathbf{q}^i}$ is the constraint Jacobian

matrix associated with the coordinates of body i, and $\mathbf{Q}_{\upsilon}^{i}$ is the vector of the inertia forces that absorb terms that are quadratic in the velocities. The constraint equations at the acceleration level can be written as $\mathbf{C}_{\mathbf{q}^i}\ddot{\mathbf{q}}^i = \mathbf{Q}_d^i$, where $\mathbf{Q}_d^i$ is a vector that absorbs first derivatives of the coordinates. Using Eq. (1) with the constraint equations at the acceleration level, one obtains the system equations of motion written in the augmented form as

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_{\mathbf{q}}^T \\ \mathbf{C}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_e + \mathbf{Q}_{\upsilon} \\ \mathbf{Q}_d \end{bmatrix} \tag{2}$$

The symbols that appear in this equation (without the superscript i) refer to system vectors and matrices obtained by standard MBS assembly of body vector and matrices. The preceding matrix equation, which ensures that the constraint equations are satisfied at the acceleration level, can be solved for the accelerations and Lagrange multipliers. Lagrange multipliers, on the other hand, can be used to determine the constraint forces. For a given joint k, the generalized constraint forces acting on body i, connected by this joint, can be obtained from the equation

$$\left(\mathbf{Q}_c^i\right)_k = -\left(\mathbf{C}_k\right)_{\mathbf{q}^i}^T \boldsymbol{\lambda}_k = \begin{bmatrix} \mathbf{F}_k^{iT} & \mathbf{T}_k^{iT} \end{bmatrix}^T \tag{3}$$

As previously mentioned, $\mathbf{F}_k^i$ and $\mathbf{T}_k^i$ are the generalized joint forces associated, respectively, with the translation and orientation coordinates of body i. Using the results of Eq. (3), the reaction forces at the joint definition points can be determined using the concept of the equipollent systems of forces.

2.2 Generalized coordinate partitioning

In order to ensure that the algebraic kinematic constraint equations are satisfied at the position and velocity levels, the independent accelerations $\ddot{\mathbf{q}}_i$ are identified and integrated forward in time in order to determine the independent velocities $\dot{\mathbf{q}}_i$ and independent coordinates $\mathbf{q}_i$. Knowing the independent coordinates from the numerical integration, the dependent coordinates $\mathbf{q}_d$ can be determined from the nonlinear constraint equations using an iterative Newton-Raphson algorithm that requires the solution of the system $\mathbf{C}_{\mathbf{q}_d} \Delta\mathbf{q}_d = -\mathbf{C}$, where $\Delta\mathbf{q}_d$ is the vector of Newton differences, and $\mathbf{C}_{\mathbf{q}_d}$ is the constraint Jacobian matrix associated with the dependent coordinates. Knowing the system coordinates and the independent velocities, the dependent velocities $\dot{\mathbf{q}}_d$ can be determined by solving a linear system of algebraic equations that represents the constraint equations at the velocity level. This linear system of equations in the velocities can be written as $\mathbf{C}_{\mathbf{q}_d} \dot{\mathbf{q}}_d = -\mathbf{C}_{\mathbf{q}_i} \dot{\mathbf{q}}_i - \mathbf{C}_t$; where $\mathbf{C}_{\mathbf{q}_i}$ is the constraint Jacobian matrix associated with the independent coordinates, and $\mathbf{C}_t = \partial\mathbf{C}/\partial t$ is the partial derivative of the constraint functions with respect to time. The selection of the set of independent coordinates is an important step in the numerical solution using the generalized coordinate partitioning. This selection can have a significant effect on the stability of the solution and also on reducing the accumulation of the numerical errors when the algebraic constraint equations are solved for the dependent variables. Furthermore, numerical problems can be encountered when using implicit integrations with the generalized coordinate partitioning, particularly when a large time step is selected by the integrator. In this case, the iterative Newton-Raphson algorithm may fail to converge. On the other hand, the generalized coordinate partitioning technique is suited for the

sparse matrix implementation and can be used for MBS applications that include rigid and flexible bodies and systems that suffer from singularity problems, such as closed chains.

2.3 Algorithm and sparse matrix implementation

In order to ensure that the constraint equations are satisfied at the position level, the dependent coordinates are determined by solving the nonlinear algebraic constraint equations using the iterative Newton-Raphson procedure. In the sparse matrix implementation one can use the following system of algebraic equations in Newton iterations [7, 8]:

$$\begin{bmatrix} \mathbf{C_q} \\ \mathbf{I}_d \end{bmatrix} \Delta\mathbf{q} = \begin{bmatrix} \mathbf{-C} \\ \mathbf{0} \end{bmatrix} \tag{4}$$

In this equation, $\mathbf{C_q}$ is the constraint Jacobian matrix, $\Delta\mathbf{q}$ is the vector of Newton differences, and $\mathbf{I}_d$ is a Boolean matrix that has zeroes and ones only; with the ones in the locations corresponding to the independent coordinates in order to ensure that $\Delta\mathbf{q}_i = \mathbf{0}$. The square coefficient matrix in Eq. (4) is sparse, and therefore, sparse matrix techniques can be used to efficiently solve the preceding system of equations for the dependent coordinates. Once the dependent coordinates are determined, the dependent velocities can be obtained using the following linear sparse system that defines the constraint equations at the velocity level:

$$\begin{bmatrix} \mathbf{C_q} \\ \mathbf{I_d} \end{bmatrix} \dot{\mathbf{q}} = \begin{bmatrix} \mathbf{-C_t} \\ \dot{\mathbf{q}}_\mathbf{i} \end{bmatrix} \tag{5}$$

The right hand side of Eq. (5) is assumed to be known since $\dot{\mathbf{q}}_i$ is determined from the numerical integration, and $\mathbf{C}_t$ depends on time and the coordinates that are assumed to be known from the

position analysis. One advantage of the structure of Eqs. 4 and 5 is that if the set of independent coordinates change during the simulation time one has only to change the locations of the nonzero entries of the matrix $\mathbf{I}_d$, while the structure of the Jacobian matrix $\mathbf{C}_\mathbf{q}$ remains the same. Once the generalized coordinates and velocities are determined, the augmented form of Eq. (2) can be constructed and solved for the acceleration $\ddot{\mathbf{q}}$, and Lagrange multipliers $\boldsymbol{\lambda}$ as previously mentioned. The main steps for a numerical algorithm using the generalized coordinate partitioning can then be summarized as follows:

1. An estimate of the initial conditions that define the MBS initial configuration is made. The initial conditions that represent the initial coordinate and velocities must be a good approximation of the initial configuration of the system.
2. Using the coordinates, the constraint Jacobian matrix $\mathbf{C}_\mathbf{q}$ can be constructed, and an LU factorization algorithm can be used to identify a set of independent coordinates.
3. Using the values of the independent coordinates, the constraint equations $\mathbf{C}(\mathbf{q},t)=\mathbf{0}$ can be considered as a nonlinear system of algebraic equations in the dependent coordinates. This system can be solved iteratively using Eq. (4) and a Newton-Raphson algorithm that employs sparse matrix techniques.
4. Assuming that the independent coordinates and velocities are known from the numerical integration and the dependent coordinates are determined from the previous step, one can construct the constraint equations at the velocity level as shown in Eq. (5). This linear system of algebraic equations can be solved for the dependent velocities.

5. Having determined the generalized coordinates and velocities, Eq. (2) can be constructed and solved for the accelerations and Lagrange multipliers. The vector of Lagrange multipliers can be used to determine the generalized reaction forces.
6. The independent accelerations can be identified and used to define the state space equations which can be integrated forward in time. The numerical solution of the state equations defines the independent coordinates and velocities.
7. If the end of the simulation is not reached, the algorithm returns to Step 2.

In the following section, it is shown how the steps of this generalized coordinate partitioning solution procedure can be modified for the TLISMNI implementation discussed in this paper.

3. TLISMNI algorithm

The TLISMNI method was recently proposed for the solution of the MBS differential/algebraic equations [15, 16]. As mentioned before, this method ensures that the algebraic constraint equations are satisfied at the position, velocity, and acceleration levels; does not require numerical or analytical differentiation of the forces, and allows for efficient sparse matrix implementation. In this section, the solution steps for a modified TLISMNI algorithm that allows for the implementation of different low order integration formulas are presented. The performance of this algorithm will be tested in a later section of the paper using complex flexible and rigid body vehicle models. The two-loop implicit procedure proposed in this study can be designed to have an iterative outer loop that involves equations which are linear in the accelerations and Lagrange multipliers and nonlinear in the coordinates and velocities. The use of the generalized coordinate partitioning leads to another iterative inner Newton–Raphson loop

that involves the kinematic constraint equations which are nonlinear functions in the dependent coordinates, in addition to a system of linear equations that can be solved for the dependent velocity. In the case of using the recursive approach instead of the generalized coordinates partitioning, the inner Newton–Raphson loop is avoided and the procedure has only the iterative outer loop that involves equations which are nonlinear in the independent coordinates and velocities and linear in accelerations and Lagrange multipliers. The TLISMNI computational algorithm that employs the generalized coordinate partitioning can be summarized as follows:

1. Assuming that the state of the system is known at time t_n, the constraint Jacobian matrix $\mathbf{C}_\mathbf{q}$ can be constructed, and used with Gaussian elimination procedure to determine a set of system independent coordinates $\mathbf{q}_i$. Therefore, the vector of system generalized coordinates $\mathbf{q}$ can be partitioned to dependent and independent coordinates as $\mathbf{q} = \begin{bmatrix} \mathbf{q}_i^T & \mathbf{q}_d^T \end{bmatrix}^T$, where $\mathbf{q}_d$ is the set of system dependent coordinates.
2. A prediction of the independent coordinates $\mathbf{q}_{i,n+1}^{(k)}$ at time t_{n+1}, using an explicit integration formula can be made and used to predict the independent velocity $\dot{\mathbf{q}}_{i,n+1}^{(k)}$ using an implicit integration formula such as HHT, BDF, Park, or Trapezoidal method.
3. Knowing the independent coordinates $\mathbf{q}_{i,n+1}^{(k)}$, the constraint equations $\mathbf{C}(\mathbf{q},t) = \mathbf{0}$ can be considered as a nonlinear system of algebraic equations in the dependent coordinates $\mathbf{q}_{d,n+1}^{(k)}$. This system can be solved iteratively for the dependent coordinates $\mathbf{q}_{d,n+1}^{(k)}$, using Eq. (4) and a Newton-Raphson algorithm that employs sparse matrix techniques,.

4. Using the predicted independent coordinates $\mathbf{q}_{i,n+1}^{(k)}$ and velocities $\dot{\mathbf{q}}_{i,n+1}^{(k)}$ from Step 2 and the dependent coordinates $\mathbf{q}_{d,n+1}^{(k)}$ determined from the previous step, one can construct the constraint equations at the velocity level (Eq. (5)). The solution of this system of linear equations defines the dependent velocities $\dot{\mathbf{q}}_{d,n+1}^{(k)}$.
5. Having determined all the coordinates and velocities at time t_{n+1}, Eq. (2) can be constructed and solved using sparse matrix techniques for the accelerations $\ddot{\mathbf{q}}_{n+1}^{(k)}$ and Lagrange multipliers $\boldsymbol{\lambda}_{n+1}^{(k)}$.
6. The independent accelerations can be identified and used to define the state space equations which can be integrated forward in time using lower order formulas such as the HHT, BDF, Park, or the Trapezoidal method to obtain $\mathbf{q}_{i,n+1}^{(k+1)}$, and $\dot{\mathbf{q}}_{i,n+1}^{(k+1)}$.
7. Using the convergence and error criteria introduced in a later section of this paper, one can judge whether or not the integration is successful. If the solution satisfies the convergence and error criteria, the coordinates, velocities, accelerations, and Lagrange multipliers are accepted. In this case, one needs to calculate the new step size in order to advance the integration. If, on the other hand, convergence is not achieved or the error exceeds the specified error tolerance, then more iterations are required using a time step size that achieves convergence. In this case, the algorithm returns to Step 2 and the process continues until the convergence and error criteria are satisfied.
8. This process continues until the desired end of the simulation time is reached.

It is clear from the computational algorithms presented in this section that the TLISMNI method uses sparse coefficient matrices that require minimum storage. Furthermore, no numerical

differentiation of the external or inertia forces with respect to the coordinates and velocities is required. The algorithm outlined above is equivalent to performing fixed point iteration for the solution of nonlinear system of equations for $\mathbf{q}_{n+1}$, and $\dot{\mathbf{q}}_{n+1}$ using an implicit integration formula [17, 18]. The TLISMNI algorithm employs the implicit integration formulas which can be more efficient than the explicit formulas in many applications, particularly in the case of stiff problems.

4. Krylov subspace and inexact Newton method

One of the main contributions of this paper is to implement the Krylov subspace projection method in the TLISMNI algorithm. In the TLISMNI algorithm described in this paper, one can consider the outer loop as a set of ODE initial value problem that can be written as $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$ where $\mathbf{y}(t_0) = \mathbf{y}_0$, and $\mathbf{y} = \begin{bmatrix} \mathbf{q}_i^T & \dot{\mathbf{q}}_i^T \end{bmatrix}^T$. Assuming these ODE's are stiff and there is a need to use an implicit integration formula to obtain a solution, one can write the general formula for implicit integration methods as

$$\mathbf{y}_{n+1} = \boldsymbol{\psi} + h\eta \mathbf{f}\left(\mathbf{y}_{n+1}, t_{n+1}\right), \qquad h\eta > 0 \tag{6}$$

where $\boldsymbol{\psi}$ is a vector that has variables previously computed, h is the time step size, η is a constant that depends on the implicit formula used, and $\mathbf{f}(\mathbf{y}_{n+1}, t_{n+1})$ is the right hand side of the system of differential equations. In this paper, an equivalent form of Eq. (6) will be used. This form can be written in terms of

$$\mathbf{x}_{n+1} = h\dot{\mathbf{y}}(\mathbf{y}_{n+1}, t_{n+1}) = (\mathbf{y}_{n+1} - \boldsymbol{\psi}) / \eta \tag{7}$$

Typically, Newton and modified Newton algorithms are employed to solve Eq. (7) for $\mathbf{x}_{n+1}$, and this generates a set of linear systems to be solved at each time step, where Eq. (7) can be written as $\mathbf{F}(\mathbf{x}_{n+1}) = \mathbf{0}$. There are standard methods to solve Eq. (7), one method is to approximate the Jacobian matrix $\mathbf{J} = \partial \mathbf{F}^k / \partial \mathbf{y}_{n+1}^k$ at time t_{n+1} and then solve the following iterative equation:

$$\left(\mathbf{I} - h\eta\mathbf{J}\right)\Delta\mathbf{x}_{n+1} = -\mathbf{F}\left(\mathbf{y}_{n+1}^k, t_{n+1}\right) \tag{8}$$

If we consider $\mathbf{J} = \mathbf{0}$, the iterative procedure will be simple and is referred to in this case as *simple iterations*, and it will be equivalent to performing fixed point iteration for the solution of the nonlinear equations for $\mathbf{y}_{n+1}$ as follows:

$$\mathbf{y}_{n+1}^{k+1} = \boldsymbol{\psi} + h\eta\mathbf{f}\left(\mathbf{y}_{n+1}^k, t_{n+1}\right) \tag{9}$$

Equation (9) represents the outer loop for the algorithm described in Section 3. In case $\mathbf{J} \neq \mathbf{0}$ expensive computations are required for the numerical calculations or approximations of $\mathbf{J}$. Furthermore, matrix decomposition, and large storage space will be required, especially when solving large scale problems. In addition, the numerical differentiation required for a general MBS implementation can be a source of numerical errors that may lead to non-accurate solutions as the problem dimensionality increases. Therefore, for such complex and large scale problems, the use of a method that can approximately solve Eq. (8) at reasonable cost and also reduces the core memory required is desirable. The *Incomplete orthogonalization method* (IOM) and Arnoldi's algorithm are examples of such methods that will be discussed in a following subsection. IOM and Arnoldi's algorithm are methods for the approximate solution of a linear system $\mathbf{Ax} = \mathbf{b}$ in $\mathbf{R}^N$ [12]. The use of such methods in the solution of the nonlinear system

$\mathbf{F}(\mathbf{x}_{n+1}) = \mathbf{0}$ by Newton's method gives rise to what is called the *inexact Newton method*. An inexact Newton method has the following form [13]:

Set $\mathbf{x}(0)$ as an initial guess

For $m = 0, 1, 2.....$ until convergence

Find in some unspecific manner a vector $\mathbf{s}(m)$ satisfying

$\mathbf{F}'(\mathbf{x}(m))\mathbf{s}(m) = -\mathbf{F}(\mathbf{x}(m)) + \mathbf{r}(m)$

Set $\mathbf{x}(m+1) = \mathbf{x}(m) + \mathbf{s}(m)$

---

where the residual $\mathbf{r}(m)$ represents the amount by which the vector $\mathbf{s}(m)$ fails to satisfy the Newton equation (Eq. (8)). The theoretical foundation and convergence of the inexact Newton method is discussed by Brown and Hindmarsh [13].

4.1 Krylov subspace projection method

In this subsection, the iterative Arnoldi's algorithm for the solution of linear system $\mathbf{Ax} = \mathbf{b}$ is briefly reviewed [12]. For the nonlinear problem given in Eq. (7), which can be written as $\mathbf{F}(\mathbf{x}_m) = \mathbf{0}$, the vector $\mathbf{b}$ represents $-\mathbf{F}(\mathbf{x}_m)$, the matrix $\mathbf{A}$ represents $\mathbf{F}'(\mathbf{x}_m)$, and the vector $\mathbf{x}$ represents the increment $\mathbf{s}_{m+1} = \mathbf{x}_{m+1} - \mathbf{x}_m$ .Given the initial value $\mathbf{x}_0$ to the original linear system, one considers an orthogonal projection method which takes $\boldsymbol{\kappa} = \boldsymbol{\kappa}_m(\mathbf{A}, \mathbf{r}_0)$ with

$$\boldsymbol{\kappa}_m(\mathbf{A}, \mathbf{r}_0) = \text{span}\left\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, ..., \mathbf{A}^{m-1}\mathbf{r}_0 \right\} \tag{10}$$

where $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. The objective of this method is to obtain an approximate solution $\mathbf{x}_m$ from the affine subspace $\mathbf{x}_0 + \boldsymbol{\kappa}_m$ of dimension m by imposing the Galerkin condition that $\mathbf{b} - \mathbf{A}\mathbf{x}_m$ is orthogonal to $\boldsymbol{\kappa}_m$. More details about the method can be found in [12, 13]. The Arnoldi's algorithm can be described as follows [12]:

---

1. Compute $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\beta := \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 := \mathbf{r}_0/\beta$
2. Define $\mathbf{H}_m = \{h_{ij}\}_{i,j=1,\ldots,m}$, set $\mathbf{H}_m = \mathbf{0}$
3. for $j = 1,\ldots,m,$ Do
4. Compute $\mathbf{w}_j := \mathbf{A}\mathbf{v}_j$
5. for $i = 1,\ldots, j,$ Do
6. $h_{ij} = (\mathbf{w}_j, \mathbf{v}_i)$
7. $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$
8. End Do
9. Compute $h_{j+1,j} = \|\mathbf{w}_j\|_2$, if $h_{j+1,j} = 0$ set $m := j$ and go to 12
10. Compute $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$
11. End Do
12. Compute $\mathbf{y}_m = \mathbf{H}_m^{-1}(\beta\mathbf{e}_1)$, and set $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m\mathbf{y}_m$

Here m is the dimension of the Krylov subspace as previously mentioned, $(.,.)$ is the Euclidean inner product, $\|.\|_2$ is the Euclidean norm, and $\mathbf{e}_1 = [1,0,0,....,0]^T \in R^m$. $[\mathbf{v}_1,....,\mathbf{v}_m]$ is an orthonormal basis for $\boldsymbol{\kappa}_m$ and the matrix $\mathbf{V}_m^T\mathbf{A}\mathbf{V}_m$ is the upper Hessenberg matrix $\mathbf{H}_m$ whose nonzero elements are the h_{ij} defined in the above algorithm. In the previous algorithm as m becomes large, a considerable amount of the work involved is in making the vector $\mathbf{v}_{m+1}$ orthogonal to all the previous vectors, $\mathbf{v}_1.....\mathbf{v}_m$. Gear and Saad [11] proposed a modification of Arnoldi's algorithm in which the vector $\mathbf{v}_{m+1}$ is only required to be orthogonal to the p vectors $\mathbf{v}_{m-p+1}.....\mathbf{v}_m$. This leads to an algorithm called *incomplete orthogonalizaton method* (IOM). IOM differs from Arnoldi's algorithm in the modified Gram-Schmidt orthogonalization process in Step 5; instead of starting with $i = 1$, one starts with $i = i_0$ where $i_0 = \max(1, m - p + 1)$. All the features and properties of Alrnoldi's algorithm still hold for IOM [12].

The Hessenberg matrix obtained in the above algorithm has a band structure with a bandwidth $m+1$. Due to the structure of the upper Hessenberg matrix $\mathbf{H}_m$, there is a convenient way to obtain the LU factorization of $\mathbf{H}_m$ by using the LU factors of $\mathbf{H}_{m-1}$ ($m>1$). Therefore the approximate solution can be given as

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{V}_m \mathbf{U}_m^{-1} \mathbf{L}_m^{-1} (\beta \mathbf{e}_1) \tag{11}$$

where $\mathbf{H}_m = \mathbf{L}_m \mathbf{U}_m$, and $\mathbf{L}_m$ is a unit lower bidiagonal matrix, and $\mathbf{U}_m$ is an upper triangular matrix. Defining $\mathbf{P}_m = \mathbf{V}_m \mathbf{U}_m^{-1}$, and $\mathbf{z}_m = \mathbf{L}_m^{-1}(\beta \mathbf{e}_1)$, Eq. (11) can be written as

$$\mathbf{x}_m = \mathbf{x}_0 + \mathbf{P}_m \mathbf{z}_m \tag{12}$$

Due to the structure of $\mathbf{U}_m$, the vector $\mathbf{p}_m$ can be calculated using the previous $\mathbf{p}_i$'s and $\mathbf{v}_m$ as follows:

$$\mathbf{p}_m = \frac{1}{u_{mm}} \left[\mathbf{v}_m - \sum_{i=1}^{m-1} u_{im} \mathbf{p}_i \right] \tag{13}$$

Similarly, because of the structure of $\mathbf{L}_m$, the vector $\mathbf{z}_m$ can be updated using $\mathbf{z}_{m-1}$ as

$$\mathbf{z}_m = \begin{bmatrix} \mathbf{z}_{m-1} \\ \xi_m \end{bmatrix} \tag{14}$$

where $\xi_m = -l_{m,m} \xi_{m-1}$. The approximate solution can be updated, at each iteration, as

$$\mathbf{x}_m = \mathbf{x}_{m-1} + \xi_m \mathbf{p}_m \tag{15}$$

One of the important practical considerations of the previous algorithm is the choice of m, which amounts to a stopping criterion. An interesting feature of the algorithm is that one does not have to obtain $\mathbf{x}_m$ in order to compute $\|\mathbf{b} - \mathbf{A}\mathbf{x}_m\|_2$ and it is easy to show that [12]

$$\left\|\mathbf{b}-\mathbf{A}\mathbf{x}_m\right\|_2 = h_{m+1,m}\left|\frac{\xi_m}{u_{mm}}\right| \tag{16}$$

The Arnoldi algorithm with stopping criteria can be efficiently implemented as follows:

---

1. Choose $\mathbf{x}_0$, and compute $\mathbf{r}_0 = \mathbf{b}-\mathbf{A}\mathbf{x}_0$, $\beta := \left\|\mathbf{r}_0\right\|_2$, and $\mathbf{v}_1 := \mathbf{r}_0/\beta$
2. For $m = 1,.....,$ until convergence
   Compute $h_{im}, i = 1....m$, and $\mathbf{v}_{m+1}$ as previously described.
3. Update LU factorization of $\mathbf{H}_m$, and compute ξ_m ($\xi_m = \beta$ for $m = 1$) otherwise $\xi_m = -l_{m,m}\xi_{m-1}$
4. Update $\mathbf{p}_m$ using Eq. (13), and $\mathbf{x}_m$ using Eq. (15)
5. Compute $\rho_m = \left\|\mathbf{b}-\mathbf{A}\mathbf{x}_m\right\|_2 = h_{m+1,m}\left|\frac{\xi_m}{u_{mm}}\right|$, if $\rho_m \le \delta$ go to 6 .Otherwise go to 2
6. End Do

Jacobian-Free Newton-Krylov methods

An extensive literature about the Jacobian-free Newton-Krylov method can be found in [19]. It can be shown for the algorithms described in the preceding subsection that the matrix $\mathbf{A}$ is not needed explicitly. One needs to compute the matrix-vector product $\mathbf{A}\mathbf{v}$ only. Since for the stiff ODE, it is assumed that $\mathbf{A} = \mathbf{F}'(\bar{\mathbf{x}}) = \mathbf{I} - h\eta\mathbf{J}$, where $\bar{\mathbf{x}}$ is an approximation to the root of $\mathbf{F}(\mathbf{x}) = \mathbf{0}$, then the matrix-vector product $\mathbf{A}\mathbf{v}$ in the above algorithm can be replaced by different quotients of the form

$$\mathbf{F}'(\bar{\mathbf{x}})\mathbf{v} = \left[\mathbf{F}(\bar{\mathbf{x}}+\sigma\mathbf{v}) - \mathbf{F}(\bar{\mathbf{x}})\right]/\sigma \tag{17}$$

where σ is a scalar. The resulting algorithm can be referred to as a *finite- difference projection method* or Jacobian-free Newton Krylov Method. The choice of σ is important, if σ is too large the derivatives are poorly approximated, and if it is too small the results of the finite difference are contaminated by floating point round off error. Approaches for choosing σ are

discussed by Knoll, and Keyes [19]. Brown and Hindmarsh [13] presented the convergence theory for the combined inexact-Newton/finite-difference projection methods. The Arnoldi's or IOM algorithm using the finite difference method (Jacobian-free) is presented as follows:

---

1. Choose $\mathbf{x}_0$, and compute $\mathbf{q}_0 = \left[\mathbf{F}(\bar{\mathbf{x}} + \sigma_0 \mathbf{x}_0) - \mathbf{F}(\bar{\mathbf{x}})\right]/\sigma_0$
2. Set $\mathbf{r}_0 = \mathbf{b} - \mathbf{q}_0$, $\beta := \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 := \mathbf{r}_0/\beta$, and $\mathbf{q}_1 := \mathbf{v}_1$
3. For $m = 1, ...,$ until convergence
   Compute $\mathbf{A}\mathbf{v}_m \approx \mathbf{q}_m \approx \left[\mathbf{F}(\bar{\mathbf{x}} + \sigma_m \mathbf{v}_m) - \mathbf{F}(\bar{\mathbf{x}})\right]/\sigma_m$
   Compute $h_{im}, i = 1....m$, and $\mathbf{v}_{m+1}$ as previously described.
4. Update LU factorization of $\mathbf{H}_m$, and compute ξ_m ($\xi_m = \beta$ for $m = 1$) otherwise $\xi_m = -l_{m,m}\xi_{m-1}$
5. Update $\mathbf{p}_m$ using Eq. (13), and $\mathbf{x}_m$ using Eq. (15)
6. Compute $\rho_m = \|\mathbf{b} - \mathbf{A}\mathbf{x}_m\|_2 = h_{m+1,m}\left|\frac{\xi_m}{u_{mm}}\right|$, if $\rho_m \le \delta$ go to 6 .Otherwise go to 3
7. End Do

4.2 TLISMNI Newton-Krylov algorithm implementation

In this section, the use of Jacobian-free Newton-Krylov approach for solving stiff DAE is presented. The proposed algorithm will employ the scaled Arnoldi's or IOM method [13], where the weight associated with component y_i of $\mathbf{y}$ during the iteration is

$$w_i = RTOL\left|y_i^{n-1}\right| + ATOL \tag{18}$$

where $RTOL$ and $ATOL$ are the relative and absolute error tolerances, respectively. In order to avoid a bias when dealing with similar ODE systems of different sizes, we use the root mean square (RMS) norm instead of the Euclidean norm. Given a diagonal matrix $\mathbf{D} = diag(d_1, ..., d_N)$, where $d_i = w_i\sqrt{N}$, and N is the length of a vector $\mathbf{y}$, the weighted RMS of a vector $\mathbf{y}$ approximating $\mathbf{y}_n$ can be written as $\|\mathbf{y}\|_{RMS} = \left\|\mathbf{D}^{-1}\mathbf{y}\right\|_2$. Based on the scaling in the

Arnoldi's algorithm or IOM algorithm, the TLISMNI-Newton-Krylov algorithm can be described as follows:

1. Assuming that the state of the system is known at time t_n, the constraint Jacobian matrix $\mathbf{C}_\mathbf{q}$ can be constructed, and used with Gaussian elimination procedure to determine the set of system independent coordinates $\mathbf{q}_i$. Therefore, the vector of system generalized coordinates $\mathbf{q}$ can be partitioned to dependent and independent coordinates as $\mathbf{q} = \begin{bmatrix} \mathbf{q}_i^T & \mathbf{q}_d^T \end{bmatrix}^T$, where $\mathbf{q}_d$ is the set of dependent coordinates.
2. A prediction of the independent coordinates $\mathbf{q}_{i,n+1}^{(k)}$ at time t_{n+1}, using an explicit integration formula can be obtained. Using an implicit integration formula such as HHT, BDF, Park, or Trapezoidal formula, one can predict the independent velocity $\dot{\mathbf{q}}_{i,n+1}^{(k)}$.
3. Using the known independent coordinates $\mathbf{q}_{i,n+1}^{(k)}$, the constraint equations $\mathbf{C}(\mathbf{q},t) = \mathbf{0}$ can be considered as a nonlinear system of algebraic equations in the dependent coordinates $\mathbf{q}_{d,n+1}^{(k)}$. This system can be solved iteratively for the dependent coordinates $\mathbf{q}_{d,n+1}^{(k)}$ using Eq. (4), sparse matrix techniques, and a Newton-Raphson algorithm .
4. Using the predicted independent coordinates $\mathbf{q}_{i,n+1}^{(k)}$ and velocities $\dot{\mathbf{q}}_{i,n+1}^{(k)}$ from step 2 and the dependent coordinates $\mathbf{q}_{d,n+1}^{(k)}$ determined from the previous step, one can construct the constraint equations at the velocity level (Eq. (5)). The solution of this system of linear equations defines the dependent velocities $\dot{\mathbf{q}}_{d,n+1}^{(k)}$.

5. Knowing all the coordinates and velocities at time t_{n+1} , Eq. (2) can be constructed and solved using sparse matrix techniques for the accelerations $\ddot{\mathbf{q}}_{n+1}^{(k)}$ and Lagrange multipliers $\boldsymbol{\lambda}_{n+1}^{(k)}$.
6. The independent accelerations can be identified and used to define the state space equations which can be integrated forward in time using the Jacobian-free Newton-Krylov algorithm described before. If convergence is achieved, go to Step 7, otherwise update to obtain $\mathbf{q}_{i,n+1}^{(k+1)}$, and $\dot{\mathbf{q}}_{i,n+1}^{(k+1)}$ and go to Step 3. In case the convergence is not achieved within a specific number of iterations, the time step is reduced and the algorithm is restarted.
7. If the convergence criteria proposed in the following section is satisfied and the error is less than the user specified tolerance, the coordinates, velocities, and the solution for the acceleration and Lagrange multipliers are accepted. In this case, one needs to update the history, and calculate the new step size in order to advance the integration. If convergence is not achieved or the error exceeds the specified error tolerance, then the time step should be reduced and the algorithm is restarted. In this case, the algorithm goes to step 2 again until the convergence and error criteria are satisfied.
8. This process continues until the desired end of the simulation time is reached.

The proposed algorithm takes advantage of the Jacobian-Free Newton-Krylov algorithm, and still exploits the sparse matrix structure to achieve minimum storage. Furthermore, the constraints are satisfied at the position, velocity, and acceleration levels, and no numerical differentiation of the external or inertia forces with respect to the coordinates and velocities is required to obtain the Jacobian matrix.

5. Convergence criteria

The TLISMNI method can be designed to have an iterative outer loop to solve for the coordinates, velocities, accelerations, and Lagrange multipliers. The steps of the TLISMNI outer loop can be summarized as follows:

1- Define the system coordinates $\mathbf{q}_{n+1}^{0}$, and velocities $\dot{\mathbf{q}}_{n+1}^{0}$.

2- Knowing $\mathbf{q}_{n+1}^{0}$, and $\dot{\mathbf{q}}_{n+1}^{0}$, one can construct the augmented form in Eq. (2) and solve for accelerations $\ddot{\mathbf{q}}_{n+1}^{0}$, and Lagrange multipliers $\boldsymbol{\lambda}_{n+1}^{0}$.

3- Using simple iteration algorithm(TLSMNI), or the Jacobian-Free Newton-Krylov an implicit integration formula can be used to obtain $\mathbf{q}_{n+1}^{1}$, and $\dot{\mathbf{q}}_{n+1}^{1}$.

4- Check if the solution converges, for example, one can check if $\left|\dfrac{\mathbf{q}_{n+1}^{k+1}-\mathbf{q}_{n+1}^{k}}{\mathbf{q}_{n+1}^{k+1}}\right|$ less than the specified tolerance.

5- If the convergence is achieved, then go to the next step otherwise go to step 3 for more iterations $k=1,2,.....n$.

The algorithm outlined above is equivalent to performing fixed point iteration for the solution of nonlinear equations for $\mathbf{q}_{n+1}$, and $\dot{\mathbf{q}}_{n+1}$ using an implicit integration formula [17, 18]. It follows that the condition $h\eta\|\mathbf{J}\|<1$,where $\|\mathbf{J}\|$ is the maximum norm of the Jacobian must hold for successive convergence. Fulfilling this condition guarantees the convergence of the solution. In other words, the TLISMNI's outer loop has a linear convergence rate and to guarantee the convergence of iteration $\mathbf{x}_m$ to the exact solution $\mathbf{x}^*$ of the equations $\mathbf{F}(\mathbf{x}_m)=\mathbf{0}$, one must have $\left\|\mathbf{x}_{m+1}-\mathbf{x}^*\right\|/\left\|\mathbf{x}_m-\mathbf{x}^*\right\|\to C$, where $0<C<1$, and $\|.\|$ denotes an R^N norm. Because $\mathbf{x}^*$ is

unknown, one can assume a linear convergence with a convergence rate estimated as $C_m = \|\mathbf{x}_{m+1} - \mathbf{x}_m\| / \|\mathbf{x}_m - \mathbf{x}_{m-1}\|$. Therefore, one can assume convergence is achieved for $\mathbf{x}_m$ satisfying $\|\mathbf{x}_m - \mathbf{x}^*\| \le \varepsilon$, which is approximately satisfied if $\bar{C}\|\mathbf{x}_m - \mathbf{x}_{m-1}\| \le \varepsilon$, where $\bar{C} = C/(1-\mathrm{C})$. More details on the linear convergence analysis can be found in [13]. The condition used in this section implies that the iteration converges in a sufficient small ball around the root. It can be shown that TLISMNI's iteration is much less expensive than the Newton iterations and allows for much more rapid variation of the step size in order to achieve convergence. In order to ensure rapid convergence in a practical implementation, the convergence rate C is selected to be much smaller than 1, for example 0.3. The goal is to have an algorithm that takes less than four iterations to converge, particularly in the case of large scale problems. On the other hand, this restriction can slow TLISMNI simple iteration method and the implicit formula can lose its properties, such as numerical damping in case of HHT [15]. In such a case, it is recommended to use the Jacobian-Free Newton-Krylov method [18] instead of the TLISMNI simple iteration method. The convergence procedure for both the TLISMNI simple iteration method and the TLISMNI Newton-Krylov method can be described as [13]:

---

1. At the beginning of the iterations (time step t_n) set $C = 0.5$
2. After number of iterations $m \ge 1$, compute $C_m = \|\mathbf{x}_{m+1} - \mathbf{x}_m\| / \|\mathbf{x}_m - \mathbf{x}_{m-1}\|$
3. Update $C = \max(0.2C, C_m)$
4. Check if $\|\mathbf{x}_{m+1} - \mathbf{x}_m\| \min(1, 1.5\,\mathrm{C}) \le \text{constant}$, then convergence achieved. Otherwise reduce the time step and restart the iterations.

The constant in Step 4 depends only on the order of the implicit integration formula used.

6. Low order integration formulas

As previously mentioned, implicit integration methods transform the MBS differential equations of motion into a set of nonlinear algebraic equations. These nonlinear algebraic equations can be solved iteratively for the required solution. Explicit methods can be very inefficient or fail to solve stiff problems which are characterized by widely separated eigenvalues because of high frequency contents. Higher order integration formulas cannot be used effectively for the solution of such stiff problems. For these problems, low order A-stable integration formulas can be more efficient. Using A-stable low order integration formula, the restriction on the size of the time step to maintain absolute stability is no longer required. In this section, different low order integration formulas will be discussed and recommendations are made on the appropriateness of each method for a particular problem.

6.1 Park method

The first integration formula considered in this section is the Park method, which has order 2 and was proposed by Park [20] as an improved stiffly stable method for direct integration of nonlinear structural dynamic equations. By combining the Gear's two-step and three-step method, a superior stiffly stable method was developed [20, 21]. Park method can be applied to both stiff and non-stiff problems. The results indicate that Park method is second best after the trapezoidal rule for non-stiff problems and appears to be stable for stiff problems that include frictional contact/impact phenomena, as will be demonstrated in the numerical results section. Pogorelov [22] proposed the use of Park method for the solution of stiff constrained MBS applications. Park method does not require any history derivative information, which can cause numerical instability in nonlinear dynamics problems even though the methods are

unconditionally stable. The equations that define the generalized coordinates and velocities at time t_{n+1} in Park method are given, respectively, by

$$\mathbf{q}_{n+1} = \frac{15}{10}\mathbf{q}_n - \frac{6}{10}\mathbf{q}_{n-1} + \frac{1}{10}\mathbf{q}_{n-2} + \frac{6}{10}h\dot{\mathbf{q}}_{n+1} \tag{19}$$

and

$$\dot{\mathbf{q}}_{n+1} = \frac{15}{10}\dot{\mathbf{q}}_n - \frac{6}{10}\dot{\mathbf{q}}_{n-1} + \frac{1}{10}\dot{\mathbf{q}}_{n-2} + \frac{6}{10}h\ddot{\mathbf{q}}_{n+1} \tag{20}$$

where h is the time step, and subscript n refers to vectors at time t_n. The local truncation error of Park method in terms of the accelerations is

$$\boldsymbol{\delta}_{n+1} = \frac{1}{10}h^2\left(\ddot{\mathbf{q}}_{n+1} - \ddot{\mathbf{q}}_n\right) \tag{21}$$

BDF method

The second integration method considered in this investigation is the second order backward differentiation formula (BDF2) method. This method was proposed by Gear [23]. The BDF2 method is a stiffly A-stable method that has been widely used for the solution of stiff problems due to their good stability properties for such problems. The BDF2 equations for the generalized coordinates and velocities are

$$\mathbf{q}_{n+1} = \frac{4}{3}\mathbf{q}_n - \frac{1}{3}\mathbf{q}_{n-1} + \frac{2}{3}h\dot{\mathbf{q}}_{n+1} \tag{22}$$

and

$$\dot{\mathbf{q}}_{n+1} = \frac{4}{3}\dot{\mathbf{q}}_n - \frac{1}{3}\dot{\mathbf{q}}_{n-1} + \frac{2}{3}h\ddot{\mathbf{q}}_{n+1} \tag{23}$$

As is clear from these equations, the BDF2 method does not require any history derivative information; therefore, it will be stable with stiff problems as is the case with Park method [20]. The local truncation error for the BDF2 method is

$$\boldsymbol{\delta}_{n+1} = \frac{2}{9} h^2 \left(\ddot{\mathbf{q}}_{n+1} - \ddot{\mathbf{q}}_n \right) \tag{24}$$

Comparing the BDF2 method's truncation error and the Park method's truncation error, one can see that Park method can achieve the same accuracy as the BDF2 method for a larger time step size by a factor of approximately 1.5.

Trapezoidal method

The third integration method that will be considered in this investigation is the trapezoidal method, which is considered the most accurate second order A-stable methods. The method does not damp out any frequency content from the system and it is unconditionally stable for linear problems, and conditionally stable for nonlinear systems [17, 24]. More details about the stability and the accuracy of the trapezoidal method can be found in [25]. The trapezoidal method equations do require history derivative information; therefore the method suffers from stability problems with stiff problems. The trapezoidal algebraic equations that represent the generalized coordinates and velocities are given, respectively, as

$$\mathbf{q}_{n+1} = \frac{1}{2}\mathbf{q}_n - \frac{h}{2}(\dot{\mathbf{q}}_n + \dot{\mathbf{q}}_{n+1}) \tag{25}$$

and

$$\dot{\mathbf{q}}_{n+1} = \frac{1}{2}\dot{\mathbf{q}}_n - \frac{h}{2}(\ddot{\mathbf{q}}_n + \ddot{\mathbf{q}}_{n+1}) \tag{26}$$

The truncation error using the trapezoidal method can be estimated as

$$\boldsymbol{\delta}_{n+1} = \frac{1}{10} h^2 \left(\ddot{\mathbf{q}}_{n+1} - \ddot{\mathbf{q}}_n \right) \tag{27}$$

HHT/Newmark method

Solving numerically stiff ODE obtained using finite element discretization, requires the use of numerical methods with good stability properties and controlled numerical dissipation such as Hilber–Hughes–Taylor (HHT) method. The foundation of the HHT method is the Newmark method which was proposed by Newmark [26]. The Newmark equations can be used to define, respectively, the generalized coordinates $\mathbf{q}_{n+1}$ and the generalized velocities $\dot{\mathbf{q}}_{n+1}$ as follows:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\dot{\mathbf{q}}_n + \frac{h^2}{2}\left(\left(1-2\beta\right)\ddot{\mathbf{q}}_n + 2\beta\ddot{\mathbf{q}}_{n+1} \right) \tag{28}$$

and

$$\dot{\mathbf{q}}_{n+1} = \dot{\mathbf{q}}_n + h\left(\left(1-\gamma\right)\ddot{\mathbf{q}}_n + \gamma\ddot{\mathbf{q}}_{n+1} \right) \tag{29}$$

where β and γ are constants. The differential equations of motion and the constraint equations can be written in the form

$$\mathbf{M}\left(\mathbf{q}\right)\ddot{\mathbf{q}} + \mathbf{C}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{Q}\left(t, \mathbf{q}, \dot{\mathbf{q}}\right) \tag{30}$$

The Nemark method is a first order method that produces a second order accurate method when $\gamma = 1/2$, and $\beta = 1/4$ only, this choice leads to the trapezoidal method. The Nemark method is unconditionally stable when $0.5 \le \gamma < 2\beta$, where the high frequency dissipation is obtained when $\beta = \left(\gamma + 0.5\right)^2 / 4$ [26].

The HHT method introduces a numerical damping parameter α to the equations of motion in order to allow for energy dissipation and retain the order and stability condition of the method. The modified equations of motion can be written as

$$\frac{1}{1+\alpha}\left(\mathbf{M}\ddot{\mathbf{q}}\right)_{n+1}+\left(\mathbf{C}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda}-\mathbf{Q}\right)_{n+1}-\frac{\alpha}{1+\alpha}\left(\mathbf{C}_{\mathbf{q}}^{\mathrm{T}}\boldsymbol{\lambda}-\mathbf{Q}\right)_{n}=\mathbf{0} \tag{31}$$

In order to obtain a stable solution using the implicit HHT method, the following relations should be satisfied $-0.3\le\alpha\le 0$, $\gamma=0.5-\alpha$ and $\beta=\left(1-\alpha\right)^{2}/4$ .The local truncation error using HHT method is

$$\boldsymbol{\delta}_{n+1}=\left(\beta-\frac{1}{6(1+\alpha)}\right)h^{2}\left(\ddot{\mathbf{q}}_{n+1}-\ddot{\mathbf{q}}_{n}\right) \tag{32}$$

7. Error control and time step selection

The error criteria and the time step-size selection for the proposed TLISMNI algorithm are discussed in this section. Using the estimate of the truncation errors $\boldsymbol{\delta}_{n+1}$ given in the preceding section for each integration formula, the following maximum norm for the vector of truncation error can be used as an estimate of the error at the current time step t_{n+1}:

$$e=\left\|\frac{\boldsymbol{\delta}_{n+1}}{\mathbf{Y}}\right\| \tag{33}$$

where $\mathbf{Y}$ is a weighted vector such that $\mathrm{Y}_{i}=\max\left(1,\left|q_{i}\right|\right)$. In order to accept the solution at the time step t_{n+1} the error must be less than the user specified tolerance ε, such that $e\le\varepsilon$.

The time step-size selection is an important issue in the implementation of efficient numerical integration algorithms. A very small time step-size leads to unnecessary calculations that may exceed the user required accuracy and at the same time negatively impact the computational efficiency. On the other hand, a large time step leads to large number of iterations for the iterative method to achieve convergence and at the same time can have a negative impact on the accuracy. The new time step size is selected such that the truncation error associated with each integration method is within the user specified tolerance. This selection is made according to

$$h_{new} = \begin{cases} sh\left|\dfrac{e}{0.5\varepsilon}\right|^{-0.5} & \mathrm{e} \le \varepsilon \\ sh\left|\dfrac{e}{0.5\varepsilon}\right|^{-0.55} & \mathrm{e} > \varepsilon \end{cases} \tag{34}$$

where s is a safety factor and h is the current time step size. In case the time step size does not satisfy the user specified error, $\mathrm{e} > \varepsilon$, then the time step size is reduced and the iteration step is restarted.

8. Explicit Adams method

Adams method is an explicit predictor-corrector method for the numerical solution of first order ordinary differential equations. This method cannot be used to directly solve a system of differential and algebraic equations. The algebraic equations must be first eliminated using the generalized coordinate partitioning technique, which is in principle equivalent to the embedding technique [7, 14, 27]. The embedding technique, that eliminates the reaction forces and the dependent coordinates, leads to a minimum set of differential equations expressed in terms of the

independent coordinates (degrees of freedom) only. Using the generalized coordinate partitioning, one can write the system coordinates as $\mathbf{q} = \begin{bmatrix} \mathbf{q}_i^T & \mathbf{q}_d^T \end{bmatrix}^T$ where $\mathbf{q}_i$ is the vector of independent coordinates or degrees of freedom, and $\mathbf{q}_d$ is the vector of dependent coordinates. One can rewrite the equations of motions in terms of the independent coordinates as follows [7, 14]:

$$\mathbf{B}_{di}^T \mathbf{M} \mathbf{B}_{di} \ddot{\mathbf{q}}_i + \mathbf{B}_{di}^T \mathbf{M} \bar{\mathbf{Q}}_d = \mathbf{B}_{di}^T \mathbf{Q} \tag{35}$$

In this equation, $\mathbf{B}_{di} = \begin{bmatrix} \mathbf{I} & \left(-\mathbf{C}_{\mathbf{q}_d}^{-1} \mathbf{C}_{\mathbf{q}_i} \right)^T \end{bmatrix}^T$ is the velocity transformation matrix, and $\bar{\mathbf{Q}}_d = \begin{bmatrix} \mathbf{0} & \left(\mathbf{C}_{\mathbf{q}_d}^{-1} \mathbf{Q}_d \right)^T \end{bmatrix}^T$, where $\mathbf{Q}_d = -\mathbf{C}_{tt} - 2\mathbf{C}_{\mathbf{q}t} - \left(\mathbf{C}_{\mathbf{q}} \dot{\mathbf{q}} \right)_{\mathbf{q}} \dot{\mathbf{q}}$. Note that the augmented formulation previously discussed in this paper can be used to determine all the accelerations. The independent accelerations can then be identified and integrated forward in time. Therefore, the explicit Adams method can be used with both the augmented formulation and the embedding technique.

In order to use the explicit Adams method, one has to transform the second order ordinary differential equations to a system of first order ordinary differential equations (ODE). This is accomplished by introducing the state vector $\mathbf{y} = \begin{bmatrix} \mathbf{q}_i^T & \dot{\mathbf{q}}_i^T \end{bmatrix}^T$ .Taking the time derivative of this vector and substituting the value of $\ddot{\mathbf{q}}_i$ from the solution of Eq. (30) yields

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{\mathbf{q}}_i \\ \left(\mathbf{B}_{di}^T \mathbf{M} \mathbf{B}_{di} \right)^{-1} \left(\mathbf{B}_{di}^T \mathbf{Q} - \mathbf{B}_{di}^T \mathbf{M} \bar{\mathbf{Q}}_d \right) \end{bmatrix} \tag{36}$$

The right hand side of this equation is a function of $\mathbf{q}_i$, $\dot{\mathbf{q}}_i$ and t. Therefore, the original problem is reformulated as $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}, t)$, which is the standard form of a first order ODE that can

be solved by Adams method. The explicit Adams method used in this study is the Adams predictor-corrector method documented in the book by Shampine and Gordon [26]. In this method, a predicted value $\mathbf{y}_{n+1}^{p}$ at the time step t_{n+1} based on the solution at time step t_n and several previous values of $\mathbf{f}$ is obtained by using the Adams-Bashforth formula:

$$\mathbf{y}_{n+1}^{p} = \mathbf{y}_n + \int_{t_n}^{t_{n+1}} \mathbf{P}_{m,n}(t)\ dt \tag{37}$$

where $\mathbf{P}_{m,n}(t)$ is a Lagrange interpolating polynomial that interpolates previous m values of $\mathbf{f}(\mathbf{y},t)$, that is,

$$\mathbf{P}_{m,n}(t) = \sum_{j=1}^{m} \mathbf{f}_{n+1-j} \left(\prod_{\substack{k=1 \\ k\neq j}}^{m} \frac{t - t_{n+1-k}}{t_{n+1-j} - t_{n+1-k}} \right) \tag{38}$$

with $\mathbf{y}_{n+1}^{p}$ available, one can then find the value of $\mathbf{f}_{n+1}^{p}\left(\mathbf{y}_{n+1}^{p}, t_{n+1}\right)$. The corrector Adams-Moulton formula is then used to find a corrected value of $\mathbf{y}_{n+1}^{p}$, which is denoted by $\mathbf{y}_{n+1}$ and is defined as

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \int_{t_n}^{t_{n+1}} \mathbf{P}_{m,n}^{*}(t)\, dt\ . \tag{39}$$

where $\mathbf{P}_{m,n}^{*}(t)$ is an interpolating polynomial that interpolates the previous m points in addition to the predicted value $\mathbf{f}_{n+1}^{p}$, that is

$$\mathbf{P}_{m,n}^{*}(t) = \mathbf{f}_{n+1}^{p} \left(\prod_{k=1}^{m} \frac{t - t_{n+1-k}}{t_{n+1} - t_{n+1-k}} \right) + \sum_{j=1}^{m} \mathbf{f}_{n+1-j} \left(\prod_{\substack{k=0 \\ k\neq j}}^{m} \frac{t - t_{n+1-k}}{t_{n+1-j} - t_{n+1-k}} \right) \tag{40}$$

One can then use the corrected value $\mathbf{y}_{n+1}$ to evaluate the function $\mathbf{f}_{n+1}$ which is used in the next time step. A full documentation of the procedures used in this explicit method for error control and selection of the order and time step can be found in the literature [27].

9. Numerical examples

In this section, numerical results obtained using a simple pendulum, a tracked vehicle model, and railroad vehicle models, are used to demonstrate the use of the TLISMNI algorithm proposed in this investigation for solving large and complex stiff systems that include flexible bodies and contact/impact forces. The results obtained using the TLISMNI algorithm and the explicit Adams predictor-corrector method, are compared in terms of efficiency and accuracy. In addition, recommendations are made on the appropriateness of each integration formula for a particular problem.

9.1 Pendulum example

The pendulum used in this example is assumed to be initially horizontal and fall under the effect of the gravity forces. The pendulum model is developed using the finite element absolute nodal coordinate formulation (ANCF). The beam in this pendulum is discretized using two-dimensional finite beam elements along its length as shown in Fig. 1 [28]. The pendulum is assumed to have undeformed length 0.4 m, cross sectional area 0.04×0.04 m^2, the mass density 7200 kg/m^3, modulus of elasticity 2×10^{11} N/m^2 , and Poison's ratio 0.3. In order to compare the performance of the TLISMNI algorithm and the explicit Adams method, a dynamic simulation of the stiff flexible pendulum is performed by using the two integration methods. Figure 2 shows that the implicit TLISMNI method damped out some high frequency oscillations and this integration method produces a smoother solution than the one obtained using the explicit Adams method. With regard to the efficiency, the TLISMNI method is 35 times faster than the Adams method for this example. Also it is important to mention that there is no significant difference between the two reference motion solutions, and therefore, the numerical damping of

the TLISMNI method does not have a significant effect on the rigid body motion of the beam, as will be demonstrated using more complex examples.

9.2 Rigid tracked vehicle model

The tracked vehicle model shown in Fig. 3 is a challenging model that represents an armored personnel carrier consisting of a chassis, 2 idlers, 2 sprockets, 10 road-wheels, and 128 track links (64 for each track). Figure 5 shows the engagement of the track links with some of the vehicle components. The vehicle has a suspension system that consists of road arms placed between the road wheels and the chassis as well as shock absorbers connected to each road arm, as shown in Fig. 4. Table 1 shows the stiffness and the damping coefficients of the contact models and the suspension system used for this model. The road arms and the sprockets are connected to the chassis by revolute joints, and the road arms are connected to the road-wheels by revolute joints. The track links are connected to each other using revolute joints. Tensioners are added to the system, each idler is connected to a tensioner with a revolute joint and the tensioner is connected to the chassis with a prismatic joint to ensure only relative translation. The model in this example is subjected to prescribed sprocket angular velocity that increases linearly until it reaches a constant value after 8 seconds. Both the generalized coordinate partitioning approach and the recursive approach are used for the solution of this tracked vehicle model. The number of equations used in the two formulations is 2184 and 648, respectively. Park integration method was found to be more efficient for this example compared to the BDF2, HHT, and the Trapezoidal integration methods. The simulation is carried out for 10 seconds with error tolerance 1×10^{-6} for the explicit Adams method and 1×10^{-7} for Park Integration method.

Figures 5-10 show the results of the model using the recursive approach with the explicit Adams method, and TLISMNI Park method. The results show very good agreement with maximum error difference less than 2% in the acceleration/force results and less than 1% in both the position and velocity results. For this example, the total simulation using the TLISMNI algorithm with Park method was at least five times faster than Adams method. All the simulations were performed on Windows 7, 3.40 GHZ CPU computer. It is important to mention that as the simulation time increases and the sprocket angular velocity increases, the TLISMNI algorithm with Park integration method becomes more efficient compared to the Adams explicit method. In the case of 10 rad/sec sprocket angular velocity, the TLISMNI algorithm with Park was found to be more than 10 time faster than the Adams explicit method. The chassis forward position and velocity are shown in Figs. 5 and 6, respectively. The results presented in these figures show a very good agreement between TLISMNI Park integration method and Adams explicit method. Euler parameter and vertical acceleration of one of the road-wheels are shown in Figs. 7 and 8, respectively, while the vertical position and velocity for one of the track links are shown in Figs. 9 and 10, respectively. Figures 11 and 12 show the forward position and velocity of the tracked vehicle model with 25 rad/s sprocket angular velocity using the generalized coordinate partitioning approach with error tolerance 1×10^{-5} for the explicit Adams, Park, and BDF2 Integration methods, while Fig. 13 shows the angular velocity of one of the road-wheels. The results show a good agreement for the position, velocity, and acceleration with maximum difference less than 3% in the acceleration results. The results obtained also show that the TLISMNIN BDF2 results have a better agreement with Adams results as compared to the TLISMNIN Park results. The CPU time using TLISMNI Park and BDF2 methods was found to be at least six times faster than the explicit Adams method on the same machine. It is also

important to mention that the recursive approach was found to be more efficient than the augmented formulation for such a complex chain problem.

9.3 Flexible tracked vehicle model

In this model, the vehicle model shown in Fig. 3 is used, where a new *compliant continuum-based joint formulation* is used for the joint formulation between the track links. In the numerical investigation presented in this paper, a three-dimensional cable element is used to model the flexibility of the chain links [29, 30]. The use of the new ANCF finite element mesh makes the CPU times of the augmented formulation and the recursive approach approximately the same because the chain revolute joint constraints are eliminated at a preprocessing stage. The most efficient integration method to be used with this model was found to be the HHT integration method that allows filtering out high frequencies. The number of equations for that model is 2192, and the length of the simulation time is 10 sec with error tolerance 1×10^{-4} with Adams method and 1×10^{-7} with TLISMNI HHT method. It was observed that in order to capture correctly the rotational coordinates using the TLISMNI HHT integration method, the HHT error tolerance should be three orders of magnitude tighter than the Adams error tolerance. The results show good agreement with maximum difference less than 0.1% in the acceleration results, while the simulation time significantly reduced using the TLISMNI HHT method. The simulation time using TLISMNI HHT method was at least 8 times faster than that of the explicit Adams. It is important to mention that as the simulation time and the angular velocity of the sprocket increases, the TLISMNI HHT method becomes more efficient. Figures 14 and 15 show the chassis forward position and velocity, while Fig. 16 shows the chassis vertical acceleration. The angular velocity and acceleration of one of the road-wheels are shown in Figs. 17 and 18,

respectively, while Figures 19-22 show the global nodal position and velocity for a certain node in the ANCF finite element mesh. It can be shown from these results that although the TLISMNI HHT algorithm employs numerical damping to filter out the high frequencies, such a method does not damp out any of the frequencies associated with the rigid body modes or any important deformation modes in this particular example. Numerical experimentations show that decreasing the numerical dissipation parameter α increases the numerical dissipation, while at the same time reduces the time step as can be demonstrated from Eq. (32). Therefore, it is important to select the proper value for the numerical dissipation α in order to be able to increase the time step.

In the flexible tracked vehicle example discussed in this section, the ANCF three-dimensional cable element was used to model the flexibility of the chain links. This element, however, does not capture the cross section and sheer deformations. Several simulations were carried out using the fully parameterized three-dimensional beam element to model the flexibility of the chain links. The general continuum mechanics approach and the elastic line approach are used to formulate the structural element elastic forces. The use of the general continuum mechanics approach leads to the ANCF coupled deformation modes including the Poisson modes. These modes couple the cross section deformation, bending, and extension of the structural elements. On the other hand, in the elastic line approach, all the deformation modes are defined along the beam centerline, and the curvature expression is used to define the bending strains. Using the elastic line approach to formulate the elastic forces for the three-dimensional fully parameterized beam element used in modeling the chain of the tracked vehicle example, the HHT TLISMNI method was found to be at least 18 times faster than the explicit Adams method with very good agreement in the results. Using the continuum mechanics approach, it was

difficult to obtain the solution using the explicit Adams method due to the stiffness of the equations, while using the HHT TLISMNI method, the results are obtained efficiently and accurately compared to the elastic line approach.

9.4 Pantograph/ Catenary railroad vehicle example

The pantograph/catenary model, shown in Fig. 23, is integrated with a railroad vehicle model that consists of 14 rigid bodies including the track, four wheelsets, two frames, four equalizers, two bolsters, and the car body. The model has two revolute joints connecting the frames and bolsters, 48 bushing elements connecting the bodies, and eight bearing elements between the wheelsets and the equalizers. The pantograph is modeled after the CX pantograph and is composed of six rigid bodies: a lower arm, upper arm, lower link, upper link, plunger, and a pan-head as shown in Fig. 24. The pantograph system has three revolute joints and four spherical joints connecting its bodies to each other and to the railroad vehicle. More details on the model, including the inertia properties as well as the initial global positions and orientations of the bodies for the pantograph model are reported by Patel et al. [31]. The catenary contact wire is supported by dropper (spring/damper) elements from a fully constrained messenger wire as shown in Fig. 25. The contact wire is modeled using 16 fully parameterized three-dimensional ANCF beam elements with the following properties; density 8960 kg/m^3, modulus of elasticity 1.17×10^{11} N/m^2 , and modulus of rigidity 4.5×10^{10} N/m^2 .The number of equations of that system is 744, while the length of the simulation time is 3.87 seconds. The car body is constrained to move with 20 $\mathrm{m/s}$ over a tangent track, the error tolerance used with Adams method is 1×10^{-5} and with TLISMNI-HHT method is 1×10^{-7}. The results obtained using both Adams and TLISMNI-HHT method show very good agreement with maximum difference 1%

as shown in Figs. 26-28. The angular velocity of one of the wheelsets is shown in Fig. 26, while Figs. 27 and 28 show the forward global position and velocity of one of the nodes on the catenary, respectively. The time required for TLISMNI HHT simulation was found to be about 8 times faster than the time required by Adams method.

10. Summary and conclusions

The objective of this paper is to integrate the Newton-Krylov projection method in a MBS solution algorithm based on two-loop implicit sparse matrix numerical integration (TLISMNI) procedure, with the goal of improving the efficiency and robustness of the TLISMNI method when used for the numerical solution of constrained complex rigid and flexible MBS differential and algebraic equations. The simple iterations and Jacobian-Free Newton-Krylov approaches are used in the TLISMNI implementation. The TLISMNI method does not require numerical differentiation of the forces, allows for an efficient sparse matrix implementation, and ensures that the algebraic constraint equations are satisfied at the position, velocity, and acceleration levels. In the augmented formulation and recursive method used in this investigation, the constraint equations were satisfied at all levels. Different low order integration formulas such as HHT, which includes numerical damping, Park, Trapezoidal, and BDF2 methods were used and recommendations on the appropriateness of each method for a particular problem are made. TLISMNI implementation issues including step size selection, convergence criteria, error control, and the effect of the numerical damping were discussed. Simple pendulum, complex rigid and flexible tracked vehicle, and railroad vehicle models were used to demonstrate the use of the proposed TLISMNI implementation. A comparison between the results obtained using the

TLISMNI algorithm and the explicit Adams predictor-corrector method showed good agreement. On the other hand, using TLISMNI method, which does not require numerical differentiation of the forces and allows for an efficient sparse matrix implementation for solving complex and very stiff structure problems, significantly improves the simulation time. For the rigid body model considered in this investigation, the TLISMNI is at least five times faster than the explicit Adams method. Using the TLISMNI algorithm with integration formulas that employ numerical damping such as HHT in the simulation of flexible body models considered in this study can achieve up to thirty five times faster simulation compared to Adams method. Nonetheless, it is important to mention that there are cases of non-stiff problems in which the use of explicit Adams method can be more efficient than the TLISMNI methods. The use of the Jacobian-Free Newton-Krylov approach instead of the simple iteration approach improves the convergence and accuracy of the TLSMNI method. Preconditioning and parallelization techniques need to be explored in a computational framework based on the TLISMNI Newton-Krylov approach, which improves the convergence for the stiff equations. Also the use of the generalized coordinate partitioning proved to be efficient in the case of differential/algebraic equation problems. More investigations are required in order to develop better criteria for the selection of the best set of independent coordinates and the time to change these coordinates. This is necessary in order to improve the performance of the TLISMNI method and avoid singularity problems arising in applications that include closed chains.

References

1. Porta,F. A., Numerical Methods for Differential-Algebraic Equations with Application to Real-Time Simulation of Mechanical Systems. ZAMM 74(94) (1994) 177-187.
2. Negrut, D., On the Implicit Integration of Differential-Algebraic Equations of Multibody Dynamics, PhD thesis, Departement of Mechanical Engineering, University of Iowa, 1998.
3. Negrut, D., Rampalli, R., Ottarsson, G., Sajdak, A., On the Use of HHT Method in the Context of Index 3 Differential Algebraic Equations of Multibody Dynamics. Journal of Computational and Nonlinear Dynamics 2(1) (2002) 73-85.
4. Pogorelov, D., Differential-algebraic Equations in Multibody System Modeling. Numerical Algorithms 19 (2002) 183-194.
5. Shabana, A. A., Computational Dynamics. Third Edition ,John Wiley and Sons, New York, 2010.
6. Haug, E. J., Computer-Aided Kinematics and Dynamics of Mechanical Systems. Boston, MA, Allyn and Bacon, 1989.
7. Shabana, A.A., Hussein, B.A., A two-loop sparse matrix numerical integration procedure for the solution of differential/ algebraic equations: Application to multibody systems. Journal of Sound and Vibration 327 (2009) 557–563.
8. Wehage, R.A., Haug E. J., Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems. J. Mech. Design 104 (1982) 247-255.
9. Baumgarte, J., Stabilization of constraints and integrals of motion in dynamical Systems. *Comp. Meth. In Appl. Mech. and Eng.* 1 (1972) 1-16.
10. Petzold, L. R., Differential/Algebraic Equations are not ODE's. *SIAM J. Sci., Stat. Comput*. 3 (1982) 367-384.

11. Gear, W., Saad, Y., Iterative Solution of Linear Equations in ODE Codes. SIAM J. SCI. STAT. COMPUT. 4 (1983) 583-601
12. Saad, Y., Iterative Methods for Sparse Linear Systems. Second Edition, SIAM, Philadelphia, 2003 .
13. Brown, P.N, Hindmarsh A.C., Matrix-Free Methods for Stiff Systems of ODE's. SIAM NUMER. ANAL. 23 (1986) 610-638.
14. Roberson, R.E., and Schwertassek, R., Dynamics of Multibody Systems. Springer Verlag, Berlin, 1988.
15. Hussein, B.A., Shabana, A.A., Sparse Matrix Implict Numerical Integration of Stiff Differential/Algebraic Equations: Implementaion. Nonlinear Dyn. 65 (2010) 369–382.
16. Shabana, A.A., Hussein, B.A., A two-loop Sparse matrix numerical integration procedure for the solution of Differential/algebraic equations: Application to multibody systems. Journal of Sound and Vibration, 327(2009) 557-563.
17. Gracia, J., Bayo, E., Kinematic and Dynamic Simulation of Multibody Systems. Springer-Verlag, New York, 1994.
18. Shampine, L.F., Type-Insensitive ODE Codes Based on Implicit A-Stable formulas. Mathematics of Computation, 36(154) (1981) 499-510.
19. Knoll, D.A., Keyes, D.E., Jacobine-free Newton-Krylov methods: a survey of approaches and applications. Journal of Computational Physics 193(2004) 357-397.
20. Park, K.C., An improved Stiffly stable Method for Direct Integration of Nonlinear Structural Dynamics Equations. Journal of Applied Mechanics, (1975) 464-470.
21. Park, K.C., Practical Aspects of Numerical Time Integration. Computers and structures 7(1975) 343-353.

22. Pogorelov, D., Differential–Algebraic Equations in Multibody System Modeling. Numerical Algorithms 19 (1998) 183–194.
23. Gear, C.W., The Simultaneous Numerical Solution of Differential-Algebraic Equations. IEEE Trans. Circuit Theory CT-18 (1971) 89-95.
24. Gourley, A.R., A Note on Trapezoidal Methods for the Solution of Initial Value Problems. Mechanica of Computation, 24 (1970) 629-633.
25. Hughes, T.J.R., The Finite Element Method: Linear Static and Dynamic Analysis. prentice-Hall, 1987.
26. Newmark, N.M., A Method of Computation for Structural Dynamics. Journal of Engineering Mechanics Division **ASCE** (1959) 67-94.
27. Shampine, L., Gordon, M., Computer Solution of ODE: The Initial Value Problem. Freeman, San Francisco, 1975.
28. Hussein, B.A., Negrut, D., Shabana, A.A., Implicit and Explicit Intgeration in the Solution of Absolute Nodal Coordinate Differential/Algebraic Equations. Nonlinear Dyn. 54 (2008) 283–296.
29. Shabana, A. A., Hamed, A. M., Mohamed, A. A., Jayakumar, P., and Letherwood, M. D. Use of B-Spline in the Finite Element Analysis: Comparison with ANCF Geometry. *Journal of Computational and Nonlinear Dynamics* 7 (2012) 81-88.
30. Wallin, M., Aboubakr, A.K., Jayakumar, P., Letherwood,M.D., Gorsich,D. J., Hamed, A.M., Shabana A.A., A Comparative Study of Joint Formulations: Application to Multibody System Tracked Vehicles ", Journal of Nonlinear Dynamics. 74 (2013) 783–800.

31. Pappalardo1, C.M., Patel M.D., Tinsley B., Shabana A.A., Control of the Pantograph/Catenary Contact Forces. Technical Report # MBS2014-8-UIC, Department of Mechanical and Industrial Engineering, University of Illinois at Chicago . 2014.

Table 1: Contact Parameters

Parameters	Sprocket-Track Contact	Roller-Track Contact	Ground-Track Contact
k	2.00×10^{6} N/m	2.00×10^{6} N/m	1.00×10^{6} N/m
c	5.00×10^{3} N·s/m	5.00×10^{3} N·s/m	1.50×10^{4} N·s/m
μ	0.150	0.100	0.300

List of figures

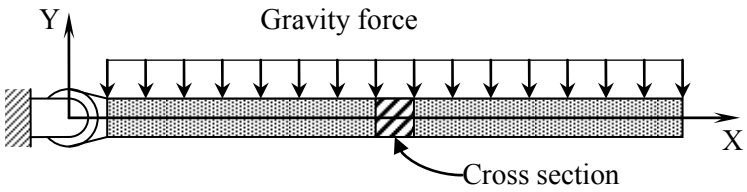

Figure 1. Flexible pendulum initial configuration

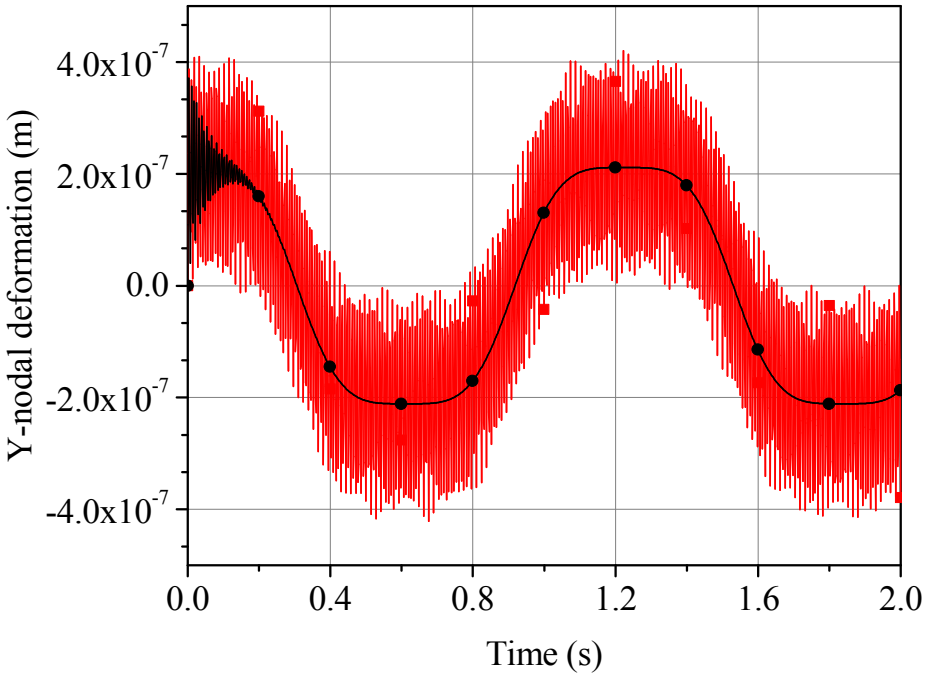

Figure 2: Nodal deformation of the mid-node
(—■— Adams, —●— HHT)

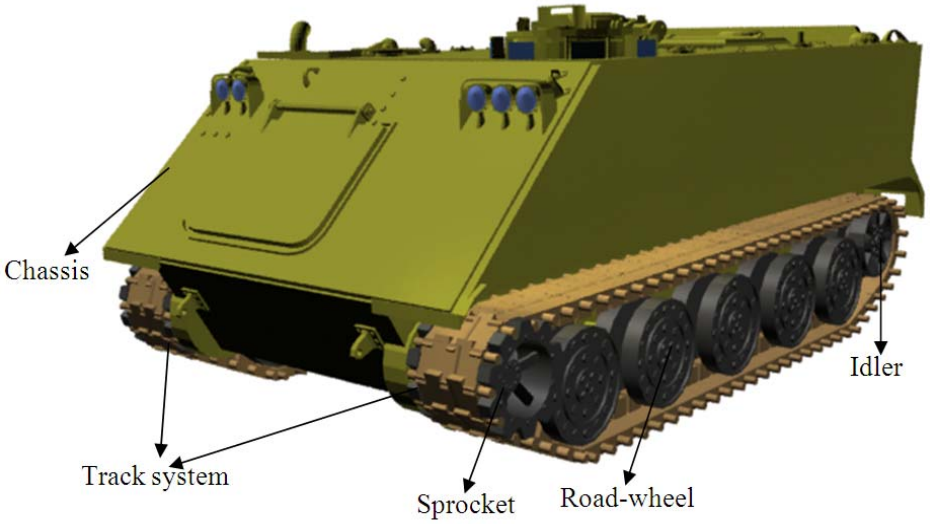

Figure 3: Tracked vehicle model

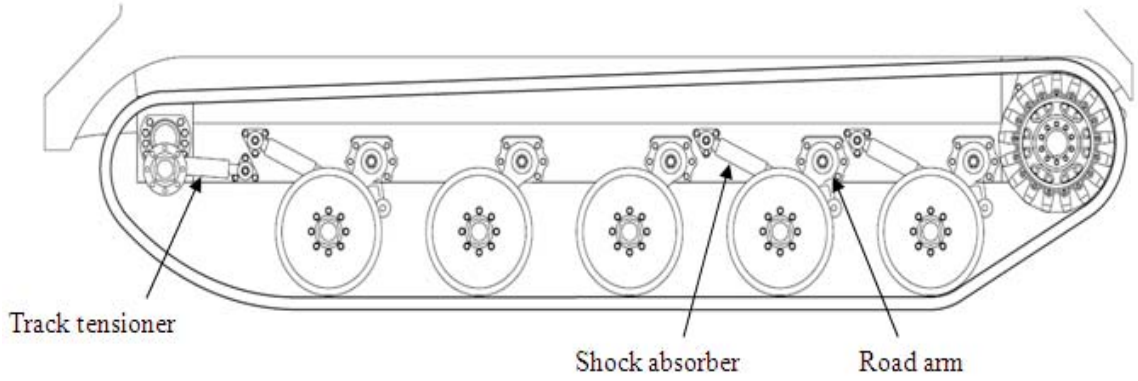

Figure 4: Suspension system layout of the tracked vehicle

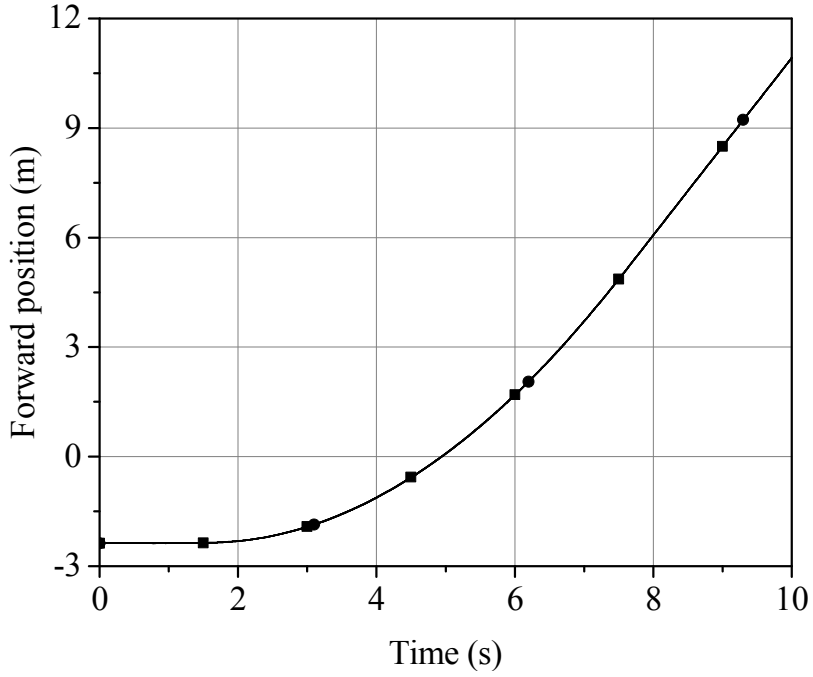

Figure 5: Chassis forward position
(—■— Adams, —●— Park)

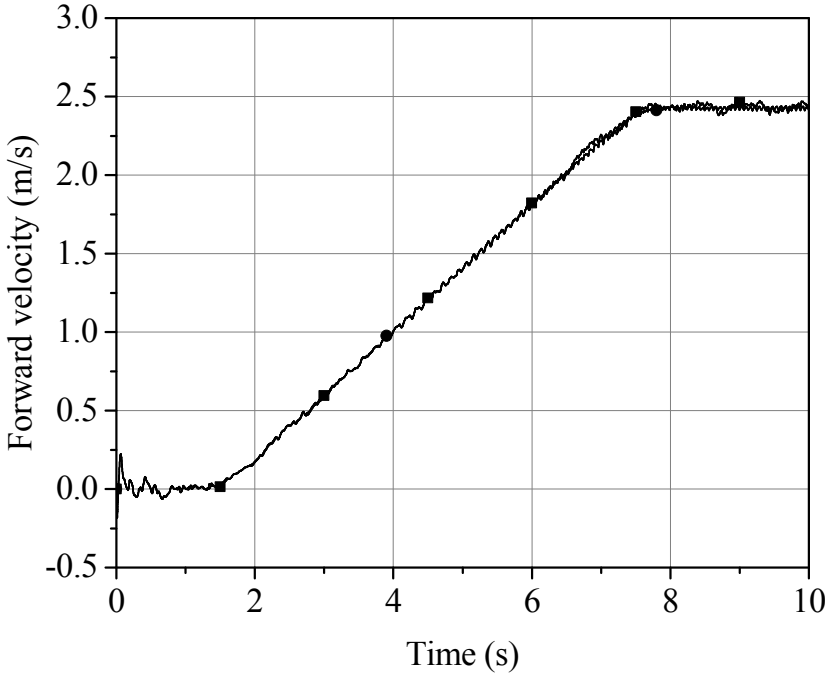

Figure 6: Chassis forward velocity
(—■— Adams, —●— Park)

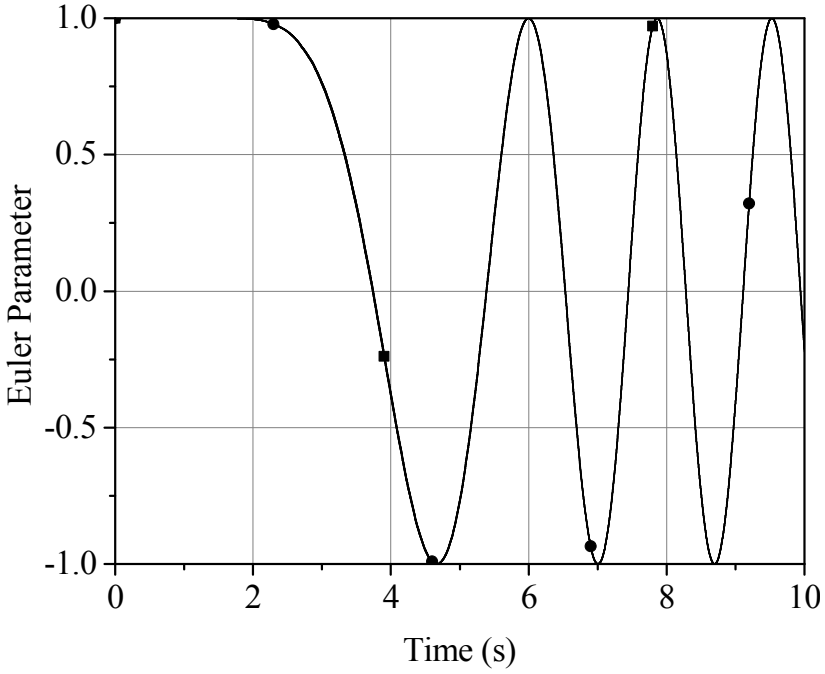

Figure 7: Road-wheel Euler parameter
(—■— Adams, —●— Park)

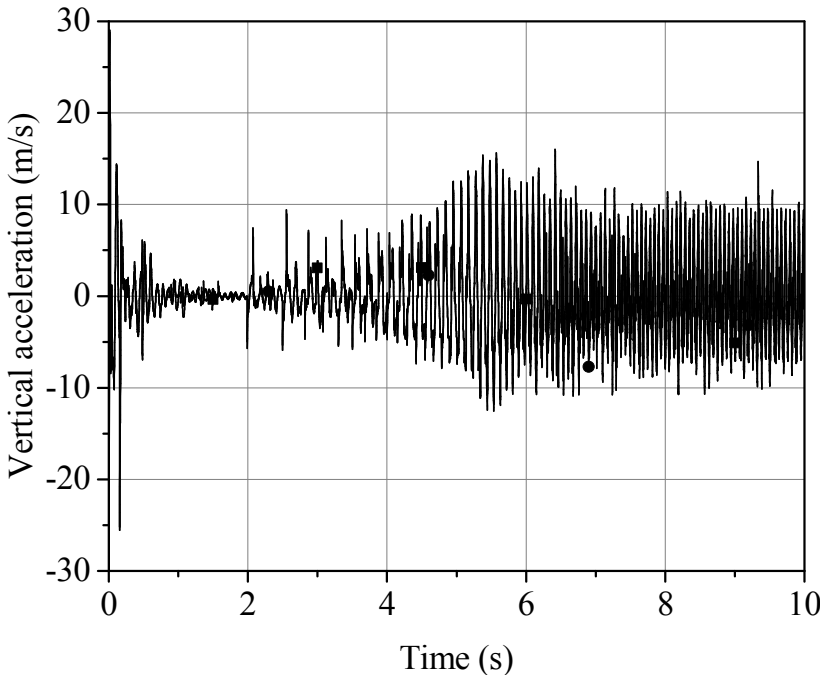

Figure 8: Road-wheel vertical acceleration
(—■— Adams, —●— Park)

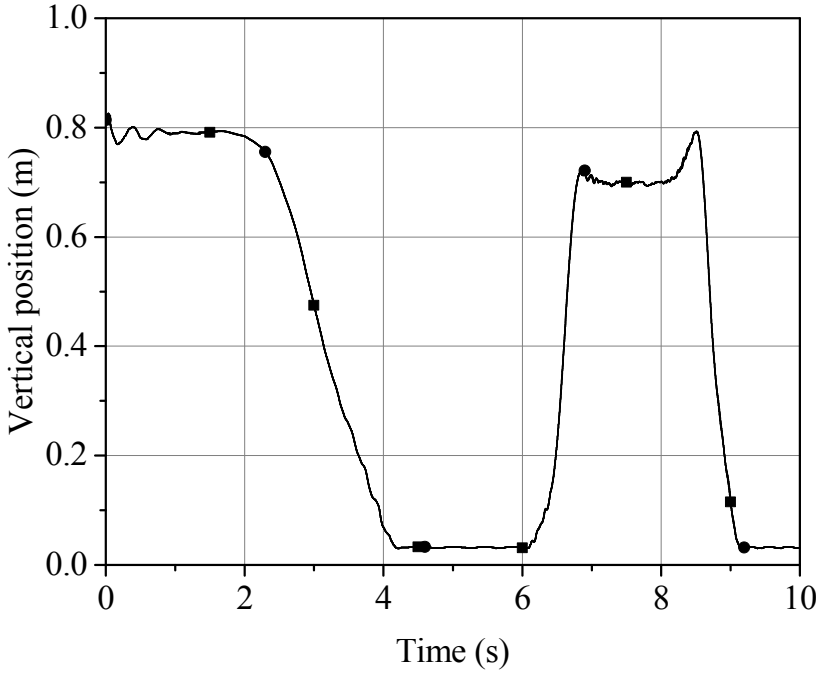

Figure 9: Track link vertical position
(—■— Adams, —●— Park)

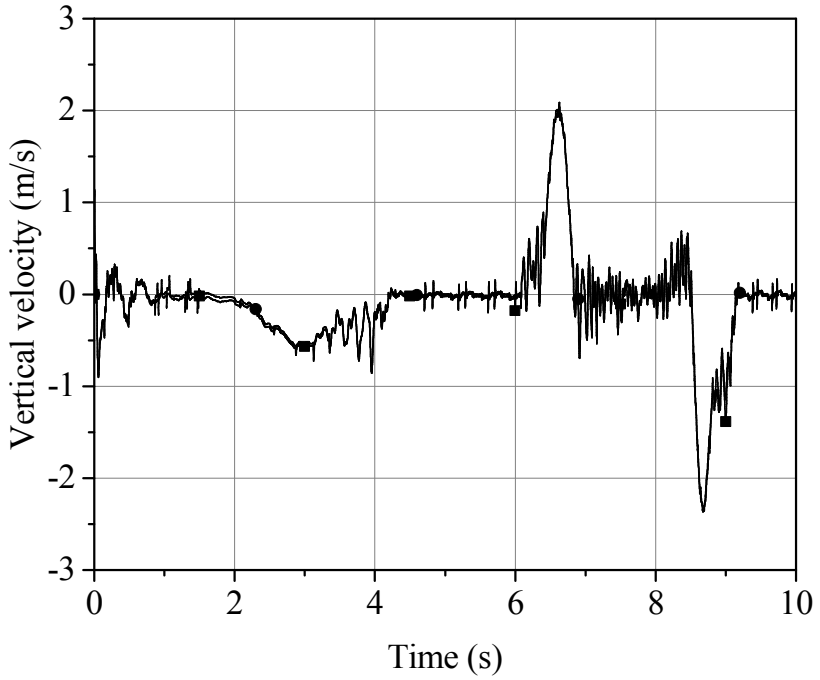

Figure 10: Track link vertical velocity
(—■— Adams, —●— Park)

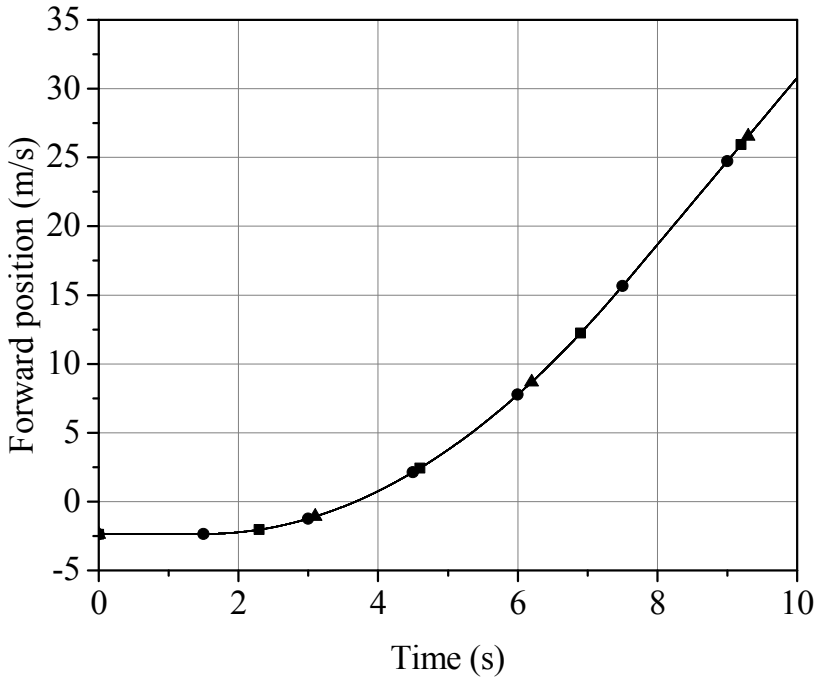

Figure 11: Chassis forward position
(—■— Adams, —●— Park, —▲— BDF2)

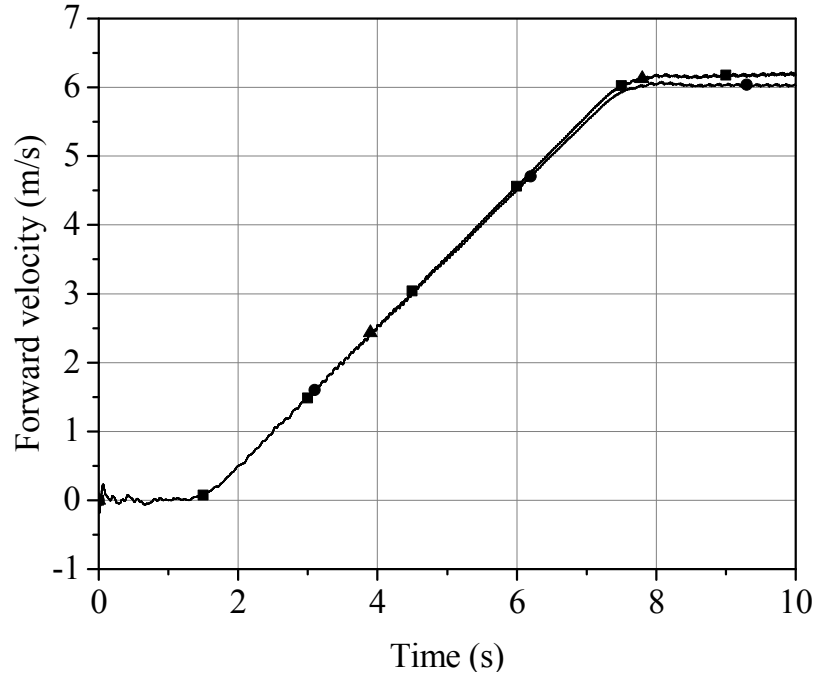

Figure 12: Chassis forward velocity
(—■— Adams, —●— Park, —▲— BDF2)

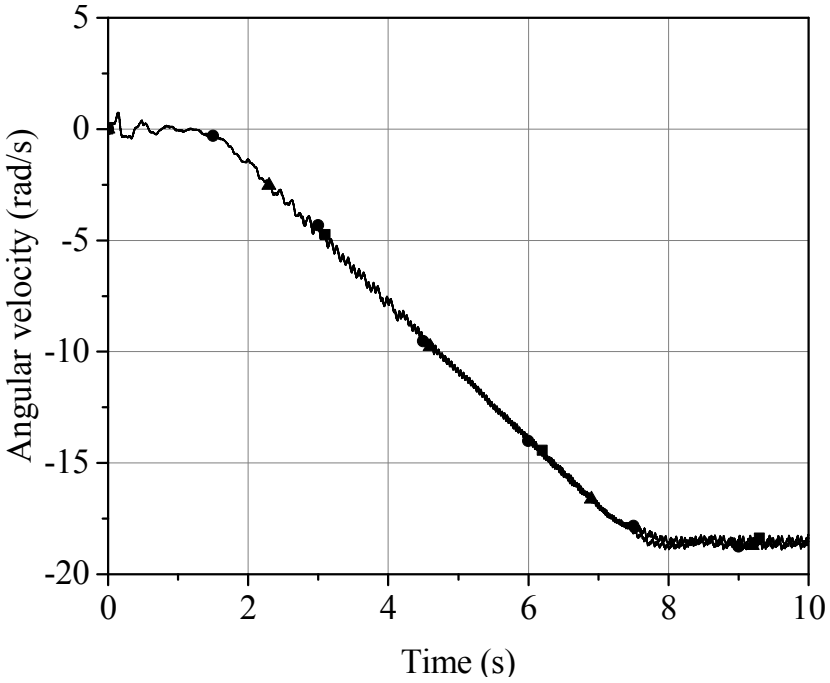

Figure 13: Road-wheel angular velocity
(—■— Adams, —●— Park, —▲— BDF2)

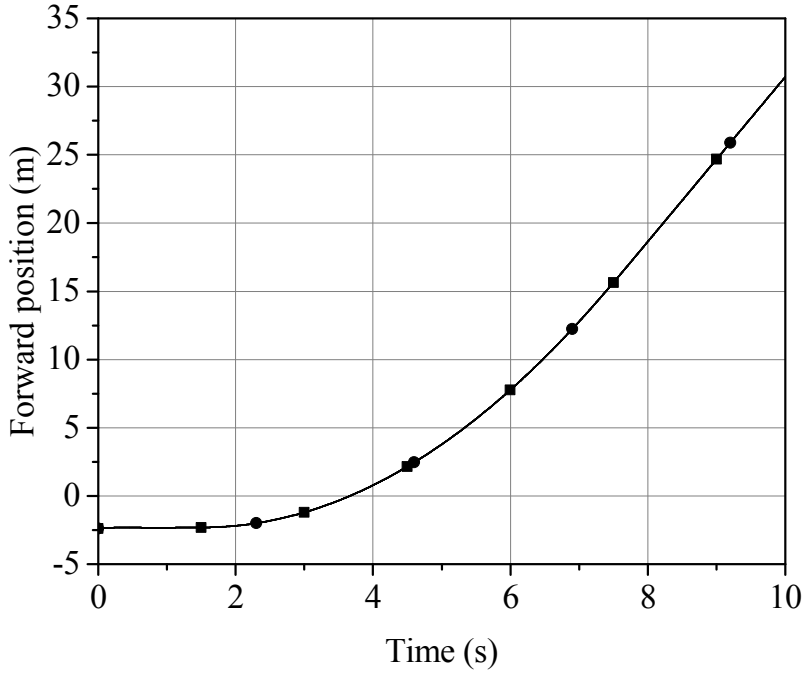

Figure 14: Chassis forward position
(—■— Adams, —●— HHT)

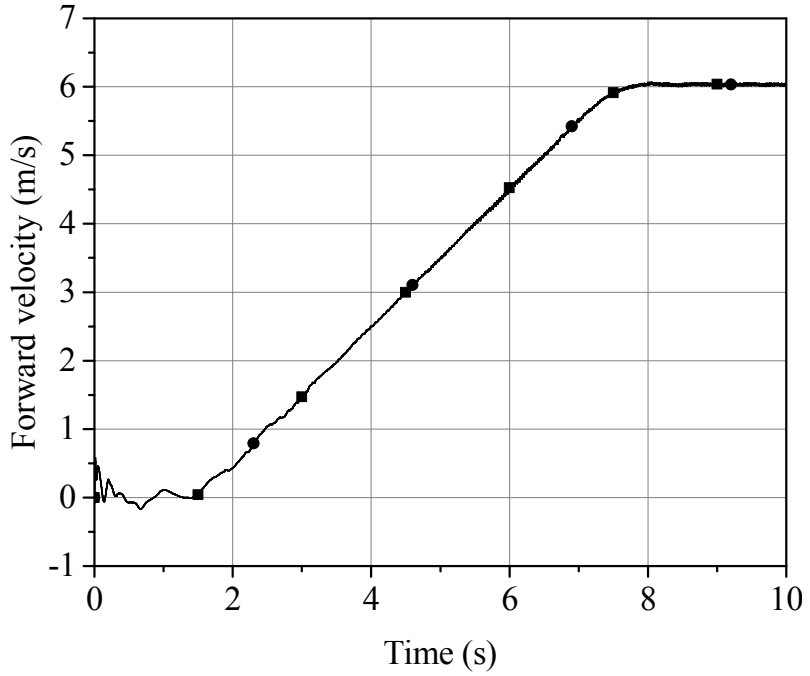

Figure 15: Chassis forward velocity
(—■— Adams, —●— HHT)

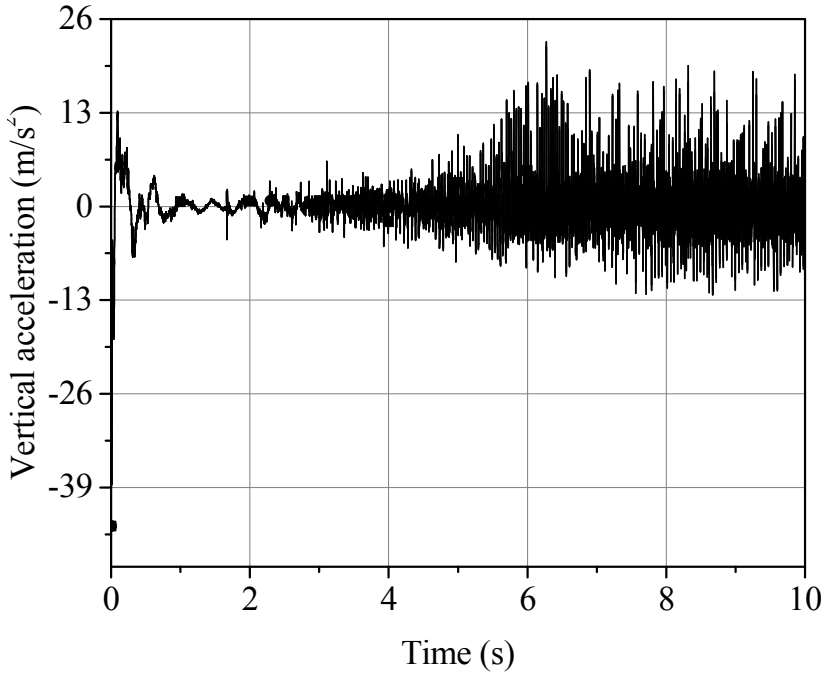

Figure 16: Chassis vertical acceleration
(—■— Adams, —●— HHT)

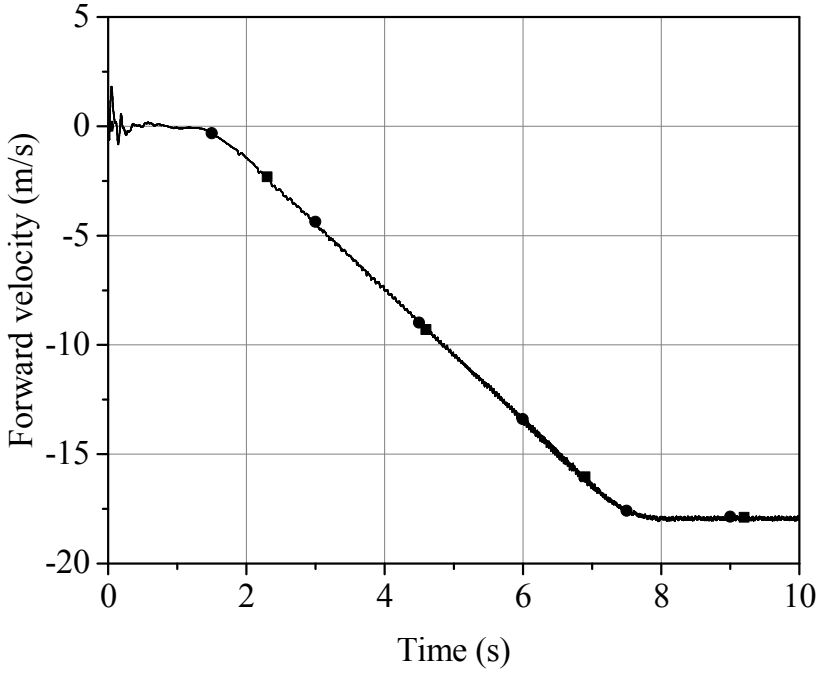

Figure 17: Road-wheel angular velocity
(—■— Adams, —●— HHT)

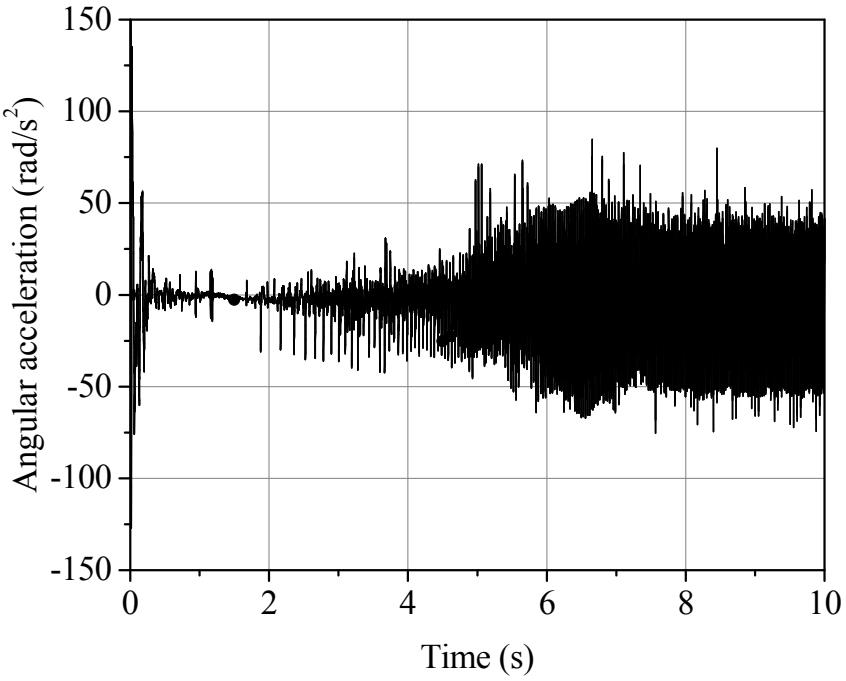

Figure 18: Road-wheel angular acceleration
(—■— Adams, —●— HHT)

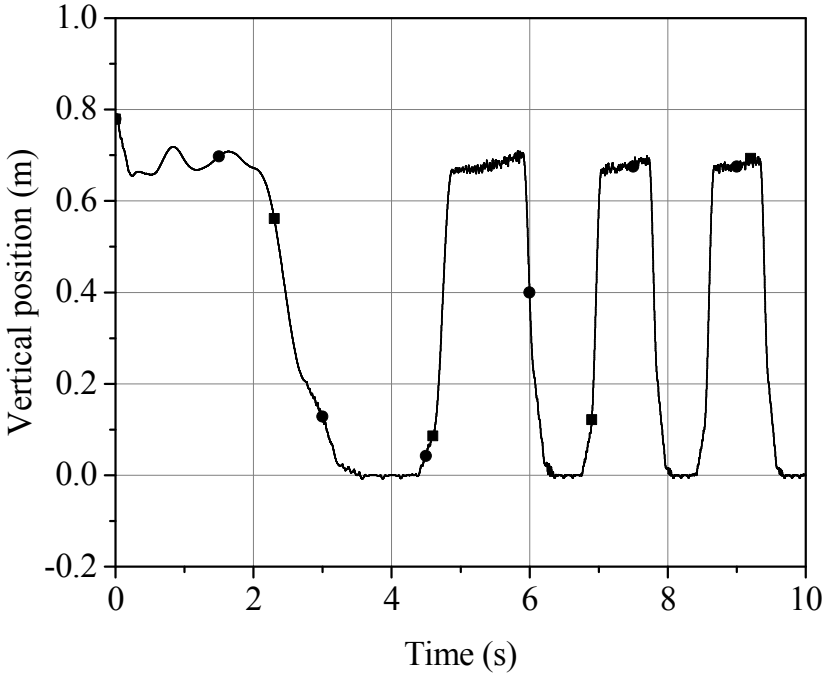

Figure 19: Vertical global nodal position
(—■— Adams, —●— HHT)

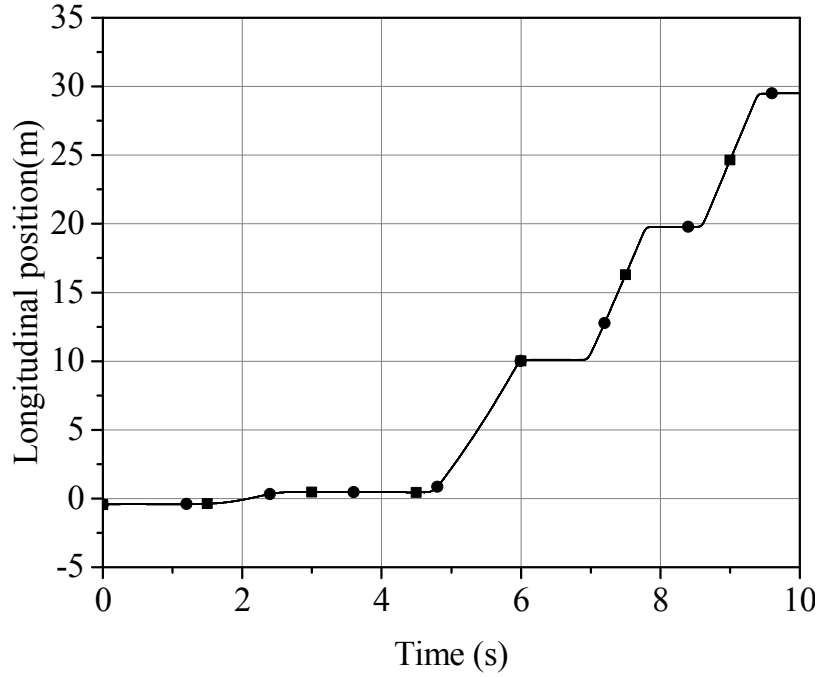

Figure 20: Longitudinal global nodal position
(—■— Adams, —●— HHT)

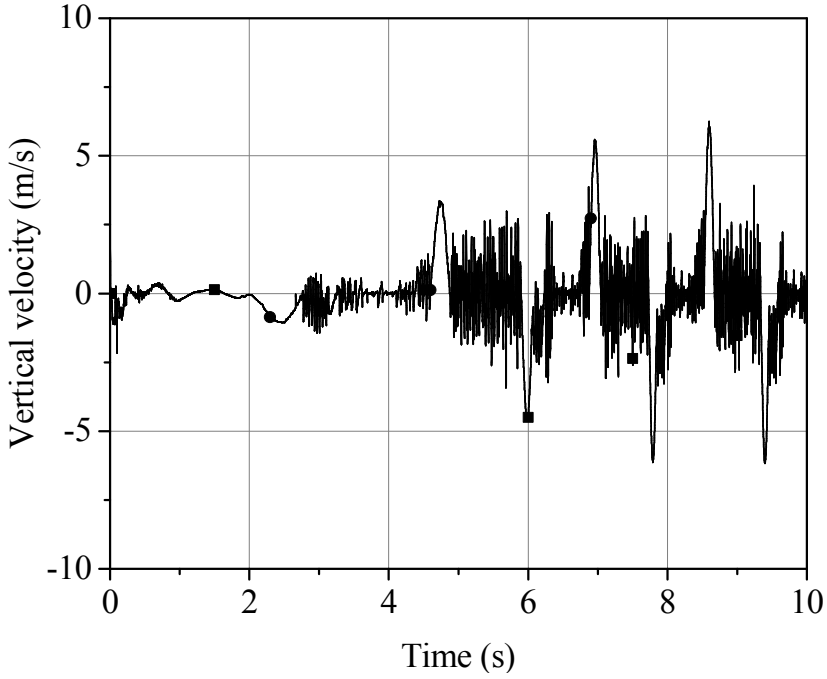

Figure 21: Vertical global nodal velocity
(—■— Adams, —●— HHT)

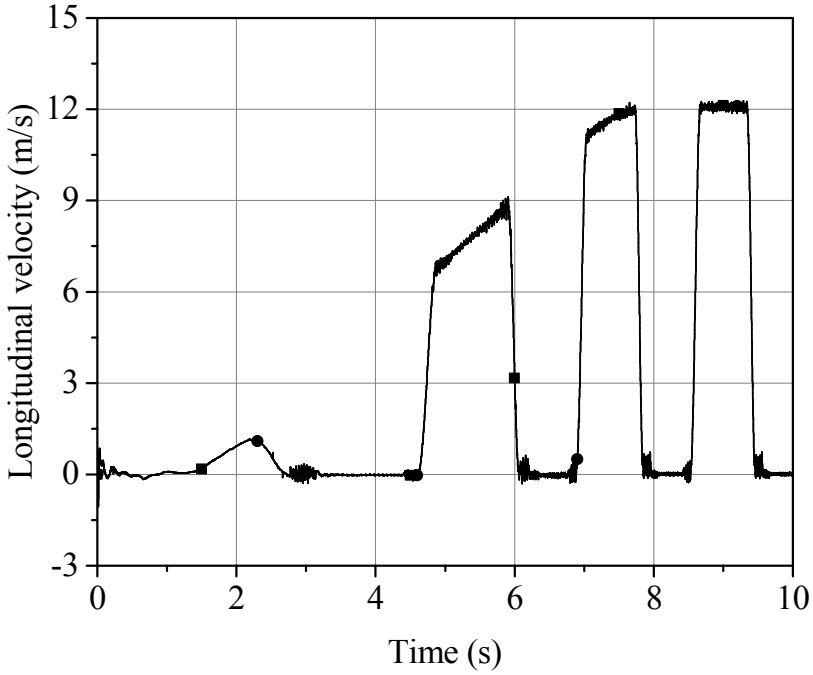

Figure 22: Longitudinal global nodal velocity
(—■— Adams, —●— HHT)

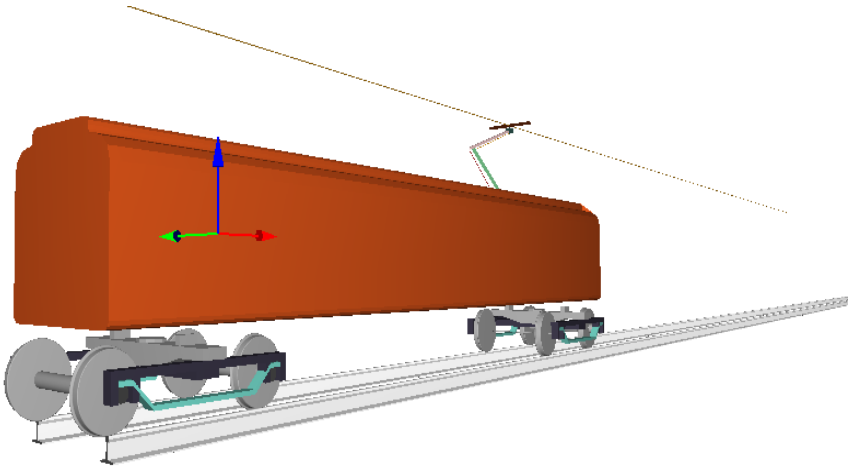

Figure 23: Pantograph/catenary railroad vehicle model

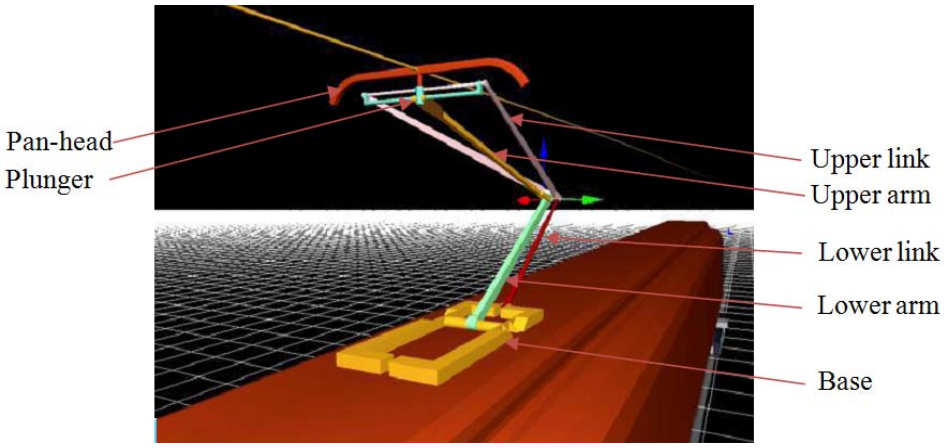

Figure 24: Articulated pantograph System

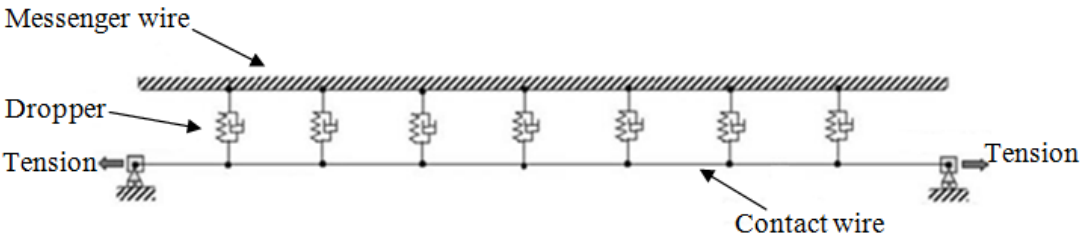

Figure 25: Catenary model

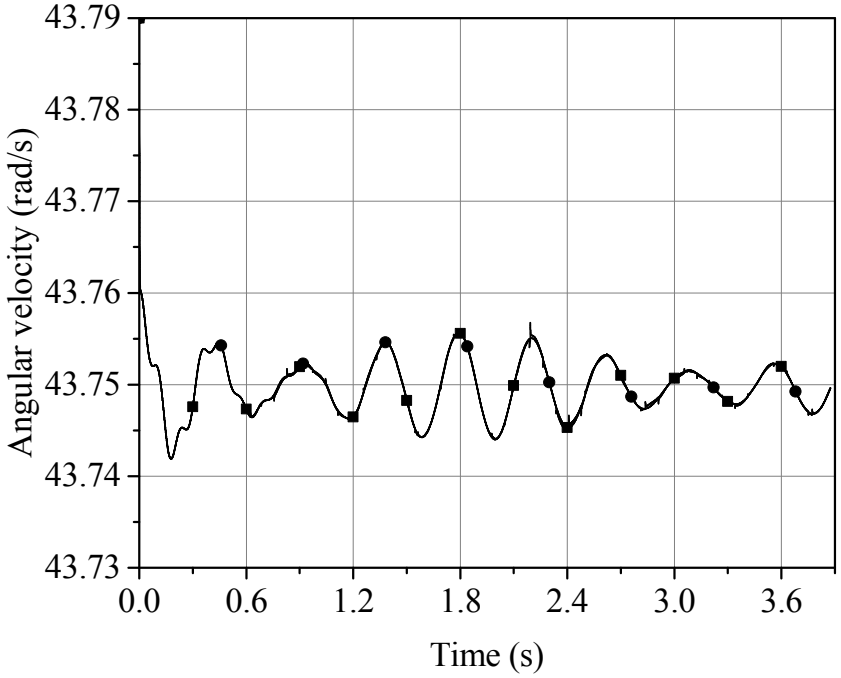

Figure 26: Wheelset angular velocity
(—■— Adams, —●— HHT)

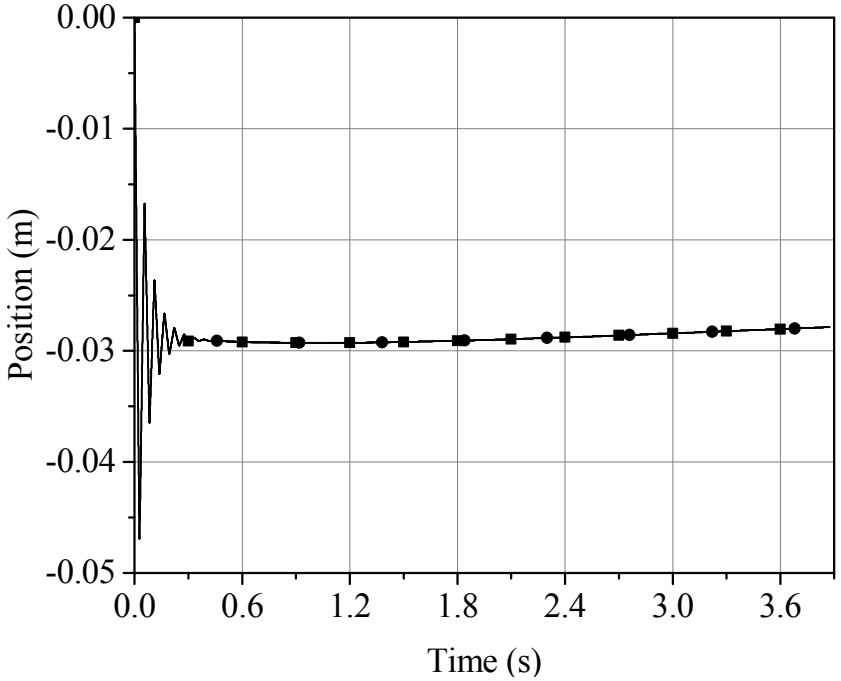

Figure 27: Longitudinal global nodal position
(—■— Adams, —●— HHT)

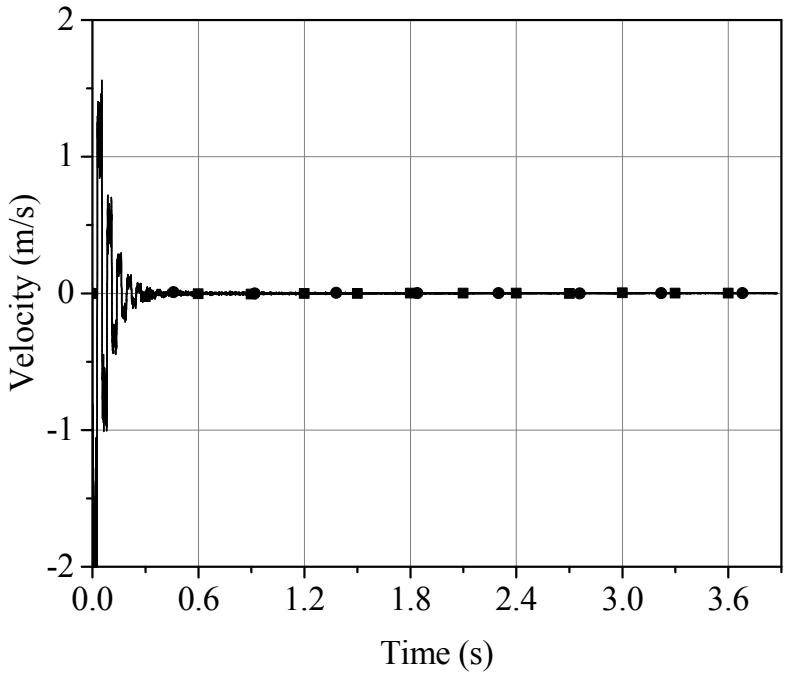

Figure 28: Longitudinal global nodal velocity
(—■— Adams, —●— HHT)