COMPUTER SCIENCE
TECHNICAL REPORT

Rosenbrock-Nystrom Integrators for SSODE of Multibody Dynamics

A. Sandu, D. Negruţ, E.J. Haug,
F.A. Potra and C. Sandu

CSTR-01-06
September 2001

Michigan Technological University, Houghton, MI 49931

ROSENBROCK-NYSTROM INTEGRATORS FOR SSODE OF MULTIBODY DYNAMICS *

A. SANDU†, D. NEGRUŢ‡, E. J. HAUG §, F. A. POTRA¶, AND C. SANDU ||

Abstract. When performing dynamic analysis of a constrained mechanical system, a set of index 3 Differential-Algebraic Equations (DAE) describes the time evolution of the system model. The paper presents a state-space based method for the numerical solution of the resulting DAE. A subset of so called independent generalized coordinates, equal in number to the number of degrees of freedom of the mechanical system, is used to express the time evolution of the mechanical system. The second order state-space ordinary differential equations (SSODE) that describe the time variation of independent coordinates are numerically integrated using a Rosenbrock type formula. For stiff mechanical systems, the proposed algorithm is shown to significantly reduce simulation times when compared to state of the art existent algorithms. The better efficiency is due to the use of an L-stable integrator and a rigorous and general approach to providing analytical derivatives required by it.

Key words. Multibody dynamics, differential-algebraic equations, state space form, Rosenbrock methods.

AMS subject classifications. 65L06

1. Introduction. In this paper, the state of a multibody system at the position level is represented by an array $\mathbf{q} = [q_1, \ldots, q_n]^T$ of generalized coordinates. The velocity of the system is described by the array of generalized velocities $\dot{\mathbf{q}} = [\dot{q_1}, \ldots, \dot{q_n}]^T$. Given the quantities $\mathbf{q}$ and $\dot{\mathbf{q}}$, the position and velocity of each body in the system is uniquely determined.

There is a multitude of ways in which the set of generalized coordinates and velocities can be selected [5, 9, 6]. The generalized coordinates used in this paper are Cartesian coordinates for position and Euler parameters for orientation of body centroidal reference frames. Thus, for each body i the position of the body is described by the vector $\mathbf{p}_i = [x_i, y_i, z_i]^T$, while the orientation is given by the array of Euler parameters [9], $\mathbf{e}_i = [e_{i0}, e_{i1}, e_{i2}, e_{i3}]^T$. Consequently, for a mechanical system containing nb bodies,

$$\mathbf{q} = \left[\begin{array}{cccccc} \mathbf{p}_1^T & \mathbf{e}_1^T & \ldots & \mathbf{p}_{nb}^T & \mathbf{e}_{nb}^T \end{array} \right]^T \in \mathbf{R}^{7nb}. \tag{1}$$

When compared with the alternative of using a set of relative generalized coordinates, the coordinates considered are convenient because of the rather complex formalism employed to obtain the Jacobian information required for implicit integration. Note that since the algorithm proposed in this paper is based on a Rosenbrock formula, it becomes essential to provide accurate integration Jacobians [7].

In any constrained mechanical system, joints connecting bodies restrict their relative motion and impose constraints on the generalized coordinates. To simplify the presentation, only holonomic and scleronomic constraints are considered. Kinematic

* This research was supported in part by the US Army Tank-Automotive Research, Development, and Engineering Center (DoD contract number DAAE07-94-R094), a multiuniversity Center led by the University of Michigan

† Departments of Computer Science, Michigan Technological University, Houghton, MI 49931

‡ Mechanical Dynamics, Inc., Ann Arbor, MI48105

§ Department of Mechanical Engineering, The University of Iowa, Iowa City, IA 52242

¶ Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD 21250

|| Department of Mechanical Engineering, The University of Iowa, Iowa City, IA 52242

constraints are then formulated as algebraic expressions involving generalized coordinates,

$$\mathbf{\Phi(q)} = \left[\begin{array}{ccc} \Phi_1(\mathbf{q}) & \ldots & \Phi_m(\mathbf{q}) \end{array} \right]^{\mathbf{T}} = \mathbf{0} \tag{2a}$$

where m is the total number of constraint equations that must be satisfied by the generalized coordinates throughout the simulation. It is assumed here that the m constraint equations are independent. The number of degrees of freedom $ndof$ is thus the difference between the number of generalized coordinates and the number of constraints $ndof = n - m$.
Differentiating Eq.(2a) with respect to time leads to the velocity kinematic constraint equation

$$\mathbf{\Phi_q(q)\,\dot{q}} = 0 \ , \tag{2b}$$

where the over dot denotes differentiation with respect to time and the subscript denotes partial differentiation, $\mathbf{\Phi_q} = \partial\left(\Phi_1 \cdots \Phi_m\right)/\partial\left(q_1 \cdots q_n\right)$. The acceleration kinematic constraint equations are obtained by differentiating Eq.(2b) with respect to time,

$$\mathbf{\Phi_q(q)\,\ddot{q}} = -\left(\mathbf{\Phi_q(q)\dot{q}}\right)_{\mathbf{q}} \mathbf{\dot{q}} \equiv \tau(\mathbf{q}, \mathbf{\dot{q}}) \ . \tag{2c}$$

Equations (2a)–(2c) characterize the admissible motion of the mechanical system.

The state of the mechanical system changes in time under the effect of applied forces. The time evolution of the system is governed by the Lagrange multiplier form of the constrained equations of motion [9],

$$\mathbf{M(q)\,\ddot{q}} + \mathbf{\Phi_q^T(q)}\,\lambda = \mathbf{Q^A(q, \dot{q}, t)} \tag{2d}$$

where $\mathbf{M(q)} \in \mathbf{R}^{n \times n}$ is the symmetric system mass matrix, $\lambda \in \mathbf{R^m}$ is the array of Lagrange multipliers that account for workless constraint forces, and $\mathbf{Q}^A(\mathbf{q}, \dot{\mathbf{q}}, t) \in \mathbf{R}^n$ represents the generalized applied force that may depend on the generalized coordinates, their first time derivatives, and time.

Equations (2a)–(2d) comprise a system of differential-algebraic equations (DAE). It is known that differential-algebraic equations are not ordinary differential equations [13]. Analytical solutions of Eqs.(2a) and (2d) automatically satisfy Eqs.(2b) and (2c), but this is no longer true for numerical solutions. In general, the task of obtaining a numerical solution of the DAE of Eqs.(2a)–(2d) is substantially more difficult and prone to intense numerical computation than that of solving ordinary differential equations (ODE). For a review of the literature on numerical integration methods for solution of the DAE of multibody dynamics the reader is referred to [14, 2].

Equations (2c)–(2d) are expressed in matrix form as

$$\left[\begin{array}{cc} \mathbf{M(q)} & \mathbf{\Phi_q(q)}^T \\ \mathbf{\Phi_q(q)} & \mathbf{0} \end{array} \right] \left[\begin{array}{c} \mathbf{\ddot{q}} \\ \lambda \end{array} \right] = \left[\begin{array}{c} \mathbf{Q}^A(\mathbf{q}, \dot{\mathbf{q}}, \mathbf{t}) \\ \tau(\mathbf{q}, \mathbf{\dot{q}}) \end{array} \right] \ . \tag{3}$$

Equations (2a), (2b), and (3) must be satisfied by the numerical solution to be constructed. This set of index 3 DAE [7] is first reduced to a set of state-space ordinary differential equations (SSODE), following the approach proposed by [19]. First an independent subset of the generalized coordinates $\mathbf{q}$ is determined. The partitioning of

the generalized coordinate vector $\mathbf{q} \in \mathbf{R}^n$ into independent and dependent coordinate vectors $\mathbf{v} \in \mathbf{R}^{ndof}$ and $\mathbf{u} \in \mathbf{R}^{n-ndof}$,

$$(4)\ \mathbf{v}[i] = \mathbf{q}[\nu(i)]\ , \quad 1 \leq i \leq ndof\ , \quad \text{and} \quad \mathbf{u}[j] = \mathbf{q}[\mu(j)]\ , \quad 1 \leq j \leq n - ndof\ ,$$

is done such that the sub-Jacobian of the constraints with respect to $\mathbf{u}$ is nonsingular

$$\det\left(\mathbf{\Phi_u}(\mathbf{q})\right) \neq 0\ . \tag{5}$$

Such a partition can be found starting with a set of consistent generalized coordinates $\mathbf{q}_0$; i.e., which satisfy Eq.(2a). The constraint Jacobian matrix $\mathbf{\Phi_q}$ is evaluated and numerically factored, using the Gauss-Jordan algorithm with full pivoting [3],

$$\mathbf{\Phi_q}(\mathbf{q}_0) \mapsto (Gauss - Jordan) \mapsto [\mathbf{\Phi_u}(\mathbf{q}_0)|\mathbf{\Phi_v}(\mathbf{q}_0)] \tag{6}$$

Column pivoting during Gauss-Jordan factorization is done such that Eq.(5) holds at all times. Likewise, the functions ν and μ introduced in Eq.(4) are defined as a byproduct of the Gauss-Jordan factorization. Thus, if $\nu : \{1, 2, \ldots, ndof\} \rightarrow S_{indep}$ and $\mu : \{1, 2, \ldots, m\} \rightarrow S_{dep}$ the sets S_{indep} and S_{dep} are such that $S_{indep} \cup S_{dep} = \{1, 2, \ldots, n\}$ and $S_{indep} \cap S_{dep} = \emptyset$. This generalized coordinate partitioning strategy is possible as long as the constraint equations of Eq.(2a) are independent [9]; i.e., as long as the constraint Jacobian $\mathbf{\Phi_q}$ has full row rank.

Based on this partitioning, Eqs.(2a)–(2c) can be rewritten in the associated partitioned form [9]

$$\mathbf{M^{vv}}(\mathbf{u}, \mathbf{v})\ddot{\mathbf{v}} + \mathbf{M^{vu}}(\mathbf{u}, \mathbf{v})\ddot{\mathbf{u}} + \mathbf{\Phi_v^T}(\mathbf{u}, \mathbf{v})\lambda = \mathbf{Q^v}(\mathbf{u}, \mathbf{v}, \dot{\mathbf{u}}, \dot{\mathbf{v}}) \tag{7a}$$

$$\mathbf{M^{uv}}(\mathbf{u}, \mathbf{v})\ddot{\mathbf{v}} + \mathbf{M^{uu}}(\mathbf{u}, \mathbf{v})\ddot{\mathbf{u}} + \mathbf{\Phi_u^T}(\mathbf{u}, \mathbf{v})\lambda = \mathbf{Q^u}(\mathbf{u}, \mathbf{v}, \dot{\mathbf{u}}, \dot{\mathbf{v}}) \tag{7b}$$

$$\mathbf{\Phi}(\mathbf{u}, \mathbf{v}) = \mathbf{0} \tag{7c}$$

$$\mathbf{\Phi_u}(\mathbf{u}, \mathbf{v})\dot{\mathbf{u}} + \mathbf{\Phi_v}(\mathbf{u}, \mathbf{v})\dot{\mathbf{v}} = \mathbf{0} \tag{7d}$$

$$\mathbf{\Phi_u}(\mathbf{u}, \mathbf{v})\ddot{\mathbf{u}} + \mathbf{\Phi_v}(\mathbf{u}, \mathbf{v})\ddot{\mathbf{v}} = \tau(\mathbf{u}, \mathbf{v}, \dot{\mathbf{u}}, \dot{\mathbf{v}}) \tag{7e}$$

The partitioning of Eqs.(7a)–(7e) is induced by the partitioning of the generalized coordinates in Eq.(4). For example, $\mathbf{M^{vu}}[i, j] = \mathbf{M}[\nu(i), \mu(j)]$, for $1 \leq i \leq ndof, 1 \leq j \leq n - ndof$, while $\mathbf{Q^u}[j] = \mathbf{Q}[\mu(j)]$, for $1 \leq j \leq n - ndof$. Likewise, a partial with respect to the dependent set of coordinates $\mathbf{u}$ is obtained by gathering the columns $\mu(1)$ through $\mu(n - ndof)$ of the derivative with respect to the generalized coordinates $\mathbf{q}$. The remaining columns provide the partial with respect to the independent coordinates $\mathbf{v}$.

The condition of Eq.(5) and the implicit function theorem [4] guarantee that Eq.(7c) can be solved for $\mathbf{u}$ as a function of $\mathbf{v}$,

$$\mathbf{u} = \mathbf{g}(\mathbf{v}) \tag{8}$$

where the function $\mathbf{g}(\mathbf{v})$ has as many continuous derivatives as does the constraint function $\mathbf{\Phi}(\mathbf{q})$. For all but the most simple mechanical systems, an analytical expression for the function $\mathbf{g}(\mathbf{v})$ can not be determined. However, for any consistent

configuration $\mathbf{q}_0 = (\mathbf{u}_0, \mathbf{v}_0)$, a value $\bar{\mathbf{u}}$ can be found for each $\bar{\mathbf{v}}$ in a small enough neighborhood of $\mathbf{v}_0$. The value $\bar{\mathbf{u}}$ is computed by keeping $\mathbf{v} = \bar{\mathbf{v}}$ in Eq.(7c) constant and solving for $\mathbf{u} = \bar{\mathbf{u}}$. Because of the form the joint constraint equations assume when used in conjunction with a Cartesian representation, $\bar{\mathbf{u}}$ is always the solution of a system of non-linear equations.

The system of DAE in Eqs.(7a), (7b), and (7e) is reduced to an SSODE, through a sequence of steps that use information provided by Eqs.(7c) and (7d). First, since the coefficient matrix of $\dot{\mathbf{u}}$ in Eq.(7d) is nonsingular, $\dot{\mathbf{u}}$ can be determined as a function of $\dot{\mathbf{v}}$ and $\mathbf{v}$, where Eq.(8) is used to eliminate explicit dependence on $\mathbf{u}$. Next, Eq.(7e) uniquely determines $\ddot{\mathbf{u}}$ as a function of $\mathbf{v}$, $\dot{\mathbf{v}}$, and $\ddot{\mathbf{v}}$, where results from Eqs.(7d) and (8) are substituted. Since the coefficient matrix of λ in Eq.(7b) is nonsingular, λ can be determined uniquely as a function of $\mathbf{v}$, $\dot{\mathbf{v}}$, and $\ddot{\mathbf{v}}$, using previously derived results. Finally, each of the preceding results is substituted into Eq.(7a) to obtain the SSODE in the independent generalized coordinates $\mathbf{v}$ [19],

$$\widehat{\mathbf{M}}(\mathbf{v})\,\ddot{\mathbf{v}} = \widehat{\mathbf{Q}}(\mathbf{t}, \mathbf{v}, \dot{\mathbf{v}}) \tag{9}$$

where

$$\widehat{\mathbf{M}} = \mathbf{M^{vv}} - \mathbf{M^{vu}}\mathbf{\Phi_u^{-1}}\mathbf{\Phi_v} - \mathbf{\Phi_v^T}\mathbf{\Phi_u^{-T}}\left[\mathbf{M^{uv}} - \mathbf{M^{uu}}\mathbf{\Phi_u^{-1}}\mathbf{\Phi_v}\right] , \tag{10a}$$

$$\widehat{\mathbf{Q}} = \mathbf{Q^v} - \mathbf{M^{vu}}\mathbf{\Phi_u^{-1}}\tau - \mathbf{\Phi_v^T}\mathbf{\Phi_u^{-T}}\left[\mathbf{Q^u} - \mathbf{M^{uu}}\mathbf{\Phi_u^{-1}}\tau\right] . \tag{10b}$$

The SSODE (9) is well defined, due to the following property [9].

LEMMA 1.1. *For any* $\mathbf{v} \in \mathbf{R}^{ndof}$, *the matrix* $\widehat{\mathbf{M}}(\mathbf{v})$ *of Eq.(10a) is positive definite.*

Proof. Given $\mathbf{v} \in \mathbf{R}^{ndof}$, the position configuration of the mechanical system is uniquely determined with all remaining dependent coordinates provided by Eq.(8). For any $\dot{\mathbf{v}} \in \mathbf{R}^{ndof}$, define $\dot{\mathbf{u}} = -\mathbf{\Phi_u^{-1}}(\mathbf{u}, \mathbf{v})\mathbf{\Phi_v}(\mathbf{u}, \mathbf{v})\dot{\mathbf{v}}$. Then the pair $\dot{\mathbf{v}}$, $\dot{\mathbf{u}}$ determines a consistent set of generalized velocities, as it satisfies the velocity kinematic constraint equations of Eq.(7d). If the generalized velocity corresponding to the pair $\dot{\mathbf{v}} \neq 0$, $\dot{\mathbf{u}}$ is denoted by $\dot{\mathbf{q}}$, the kinetic energy $K = 1/2\,\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} > 0$. Since $\dot{\mathbf{q}}^T\mathbf{M}\dot{\mathbf{q}} = \dot{\mathbf{v}}^{\mathbf{T}}\widehat{\mathbf{M}}\dot{\mathbf{v}}$, it follows that $\dot{\mathbf{v}}^T\widehat{\mathbf{M}}\dot{\mathbf{v}} > \mathbf{0}$. □

2. Rosenbrock integration formulas for second order systems. For the Initial Value Problem (IVP), $y' = f(t, y)$, $y(t_0) = y_0$, an s-stage Rosenbrock method is defined as [7]

$$y_{n+1} = y_n + \textstyle\sum_{i=1}^{s} b_i k_i , \tag{11a}$$

$$k_i = hf\left(t_n + \alpha_i h, y_n + \textstyle\sum_{j=1}^{i-1} \alpha_{ij} k_j\right) + \gamma_i h^2 \frac{\partial f}{\partial t}(t_n, y_n) + hJ \textstyle\sum_{j=1}^{i} \gamma_{ij} k_j , \tag{11b}$$

where the number of stages s and the coefficients b_i, α_{ij}, and γ_{ij} are chosen to obtain a desired order of consistency and stability, $J = f_y(t_n, y_n)$, $\alpha_i = \sum_{j=1}^{i-1} \alpha_{ij}$, and $\gamma_i = \sum_{j=1}^{i} \gamma_{ij}$. For reasons of computational efficiency the coefficients γ_{ii} are identical for all stages; i.e., $\gamma_{ii} = \gamma$ for all $i = 1, \ldots, s$. Note that formally $\alpha_{ii} = 0$, $1 \leq i \leq s$. For the purpose of error control in the the generic Rosenbrock method a second approximation of the solution at the current time step is used to produce an estimate of the local error. This second approximation $\hat{y}_{n+1}$ is usually of lower order and it recycles the stage values k_i of Eq.(11b), but this time employing a different set of

coefficients $\hat{b}_i$,

$$\hat{y}_{n+1} = y_n + \sum_{i=1}^{s} \hat{b}_i k_i \tag{12}$$

The approximation $|y_{n+1} - \hat{y}_{n+1}|$ of the local error depends on the size of the integration step-size, and the latter is increased or decreased to keep the local error right below a user prescribed absolute and/or relative tolerance. Thus, for the vector case $\mathbf{y} \in \mathbf{R}^m$, at time step $n+1$ the error in component i is kept smaller than a composite error tolerance sc_i

$$|y^i_{n+1} - \hat{y}^i_{n+1}| < sc_i, \quad sc_i = Atol_i + \max(|y^i_n|, |y^i_{n+1}|) \cdot Rtol_i \tag{13}$$

where $Atol_i$ and $Rtol_i$ are the user prescribed absolute and relative integration tolerances for component i. The value

$$err = \left(\frac{1}{m} \sum_{i=1}^{m} \frac{(y^i_{n+1} - \hat{y}^i_{n+1})^2}{sc_i^2} \right)^{1/2} \tag{14}$$

is considered as a measure of local error. If the order of the proper and embedded formulas used is p and $\hat{p}$ respectively, asymptotically $err \approx Ch^{q+1}$, where C is a constant dependening on the choice of formulas and $q = \min(p, \hat{p})$. Optimally, $err = 1$ and therefore $1 \approx Ch_{opt}^{q+1}$. The optimal step-size is computed then as

$$h_{opt} = h \left(\frac{1}{err} \right)^{\frac{1}{q+1}} \tag{15}$$

A safety factor fac multiplies h_{opt} to decrease the chance of a costly rejected step-size, which happens whenever $err > 1$. Further, the step-size is not allowed to increase or decrease too fast. This is achieved by two control parameters $facmin$ and $facmax$,

$$h_{new} = h \min \left(facmax, \max \left(facmin, fac \cdot (1/err)^{1/(q+1)} \right) \right) \tag{16}$$

To apply a generic Rosenbrock formula for the solution of the SSODE of Multibody Dynamics of Eq.(9), first notice that since $\widehat{\mathbf{M}}$ is positive definite, by multiplying from the left with $\widehat{\mathbf{M}}^{-1}$, the second order SSODE of Eq.(9) theoretically can be locally reduced to the form

$$y'' = f(t, y, y') \tag{17a}$$

Note that for the purpose of introducing the Rosenbrock-Nystrom approach and without any loss of generality in the formula above the vector $\mathbf{v}$ was replaced with a scalar quantity y. As the interest is only in determining the coefficients of the Rosenbrock-Nystrom method, the fact that this formula is used to find a scalar or vector numerical solution of an ODE problem is irrelevant.

The previous second order system of ODE is transformed into a standard first order ODE problem,

$$\begin{bmatrix} y \\ y' \end{bmatrix}' = \begin{bmatrix} y' \\ f(t, y, y') \end{bmatrix} \tag{17b}$$

Applying the generic method of Eqs.(11a– 11b) to this first order ODE system yields

$$\text{(18a)} \quad \begin{bmatrix} y_{n+1} \\ y'_{n+1} \end{bmatrix} = \begin{bmatrix} y_n \\ y'_n \end{bmatrix} + \sum_{i=1}^{s} b_i \begin{bmatrix} k_i \\ \ell_i \end{bmatrix} ,$$

$$\text{(18b)} \quad \begin{bmatrix} k_i \\ \ell_i \end{bmatrix} = h \begin{bmatrix} y'_n + \sum_{j=1}^{i-1} \alpha_{ij} \ell_j \\ f(t_n + \alpha_i h, y_n + \sum_{j=1}^{i-1} \alpha_{ij} k_j, y'_n + \sum_{j=1}^{i-1} \alpha_{ij} \ell_j) \end{bmatrix} + \gamma_i h^2 \begin{bmatrix} 0 \\ \frac{\partial f}{\partial t}(t_n, y_n, y'_n) \end{bmatrix} + h \begin{bmatrix} 0 & I \\ J_1 & J_2 \end{bmatrix} \sum_{j=1}^{i} \gamma_{ij} \begin{bmatrix} k_j \\ \ell_j, \end{bmatrix} ,$$

where

$$\text{(19)} \qquad J_1 = \frac{\partial f}{\partial y}(t_n, y_n, y'_n) \qquad \text{and} \qquad J_2 = \frac{\partial f}{\partial y'}(t_n, y_n, y'_n) \ .$$

In order to obtain a numerical method to directly integrate Eq.(17a), the "y-stages" k_i are eliminated to express the formula only in terms of the "y'-stages" ℓ_i. Defining $\beta_{ij} = \alpha_{ij} + \gamma_{ij}$, the first row of (18b) is

$$\text{(20)} \qquad k_i = h y'_n + h \sum_{j=1}^{i} \beta_{ij} \ell_j \ , \quad i = 1, \ldots, s$$

In the second row of Eq.(18b), the sum $\sum_{j=1}^{i-1} \alpha_{ij} k_j$ comes in as the second argument of $f(\cdot, \cdot, \cdot)$. Defining $\delta_{ij} = \sum_{m=j}^{i-1} \alpha_{im} \beta_{mj}$ and using Eq.(20) and the summation interchange procedure, this sum is expressed in terms of ℓ_j as

$$\begin{aligned} \sum_{j=1}^{i-1} \alpha_{ij} k_j &= \sum_{j=1}^{i-1} \alpha_{ij} \left(h y'_n + h \sum_{m=1}^{j} \beta_{jm} \ell_m \right) \\ &= h \alpha_i y'_n + h \sum_{j=1}^{i-1} \sum_{m=1}^{j} \alpha_{ij} \beta_{jm} \ell_m \\ &= h \alpha_i y'_n + h \sum_{m=1}^{i-1} \sum_{j=m}^{i-1} \alpha_{ij} \beta_{jm} \ell_m \\ &= h \alpha_i y'_n + h \sum_{j=1}^{i-1} \delta_{ij} \ell_j \end{aligned}$$

With the above substitution, the second row of Eq.(18b) becomes

$$\text{(21)} \qquad \begin{aligned} \ell_i &= h f \left(t_n + \alpha_i h, y_n + h \alpha_i y'_n + h \sum_{j=1}^{i-1} \delta_{ij} \ell_j, \ y'_n + \sum_{j=1}^{i-1} \alpha_{ij} \ell_j \right) \\ &\quad + \gamma_i h^2 \frac{\partial f}{\partial t}(t_n, y_n, y'_n) \\ &\quad + h J_1 \sum_{j=1}^{i} \gamma_{ij} k_j + h J_2 \sum_{j=1}^{i} \gamma_{ij} \ell_j \end{aligned}$$

Defining $\theta_{ij} = \sum_{m=j}^{i} \gamma_{im}\beta_{mj}$ and substituting the whole sum in k's by a sum in ℓ's

$$\sum_{j=1}^{i} \gamma_{ij} k_j \quad = \quad h\gamma_i y'_n + h\sum_{j=1}^{i} \theta_{ij}\ell_j$$

Eq.(21) becomes the stage relation

$$\begin{aligned} (22) \qquad \ell_i \quad = \quad & hf\left(t_n + \alpha_i h, y_n + h\alpha_i y'_n + h\sum_{j=1}^{i-1}\delta_{ij}\ell_j,\ y'_n + \sum_{j=1}^{i-1}\alpha_{ij}\ell_j\right) \\ & +\gamma_i h^2\left(\frac{\partial f}{\partial t}(t_n, y_n, y'_n) + J_1 y'_n\right) \\ & +h^2 J_1 \sum_{j=1}^{i}\theta_{ij}\ell_j + hJ_2\sum_{j=1}^{i}\gamma_{ij}\ell_j \end{aligned}$$

¿From a computational point of view Eq.(22) has the following interpretation: at stage i $(1 \le i \le s)$ the quantity ℓ_i is to be found as the solution of a linear system

$$S_i \ell_i = RHS$$

where $S_i = I - h\gamma_{ii}J_2 - h^2\theta_{ii}J_1$. Since $\alpha_{ii} = 0,\ 1 \le i \le s$, it follows that $\theta_{ii} = \gamma_{ii}\beta_{ii} = \gamma_{ii}(\alpha_{ii} + \gamma_{ii}) = \gamma^2$. Thus, the linear systems to be solved at each stage have the same matrix,

$$S_i = S = I - h\gamma J_2 - h^2\gamma^2 J_1$$

Substituting k's by ℓ's in Eq.(18a) and denoting $\mu_i = \sum_{j=i}^{s} b_j\beta_{ji}$ leads to

$$(23a) \qquad y_{n+1} \quad = \quad y_n + hy'_n + h\sum_{i=1}^{s}\mu_i\ell_i$$

$$(23b) \qquad y'_{n+1} \quad = \quad y'_n + \sum_{i=1}^{s} b_i\ell_i$$

Using matrix notation for the coefficients (e.g., (α_{ij}) is the matrix whose entries are the α-coefficients of the method) δ, θ, and μ are expressed as

$$(\delta_{ij}) = (\alpha_{ij}) \cdot (\beta_{ij}) \ , \quad (\theta_{ij}) = (\gamma_{ij}) \cdot (\beta_{ij}) \ , \quad (\mu_i) = (b_i) \cdot (\beta_{ij}) \ .$$

To summarize, the following linearly implicit method for the second order system (17a) is defined:

$$(24a) \qquad y_{n+1} \quad = \quad y_n + hy'_n + h\sum_{i=1}^{s}\mu_i\ell_i$$

$$(24b) \qquad y'_{n+1} \quad = \quad y'_n + \sum_{i=1}^{s} b_i\ell_i$$

$$Y_i = y_n + h\alpha_i y'_n + h\sum_{j=1}^{i-1}\delta_{ij}\ell_j \tag{24c}$$

$$Y'_i = y'_n + \sum_{j=1}^{i-1}\alpha_{ij}\ell_j \tag{24d}$$

$$\begin{aligned}\ell_i &= hf\left(t_n+\alpha_i h, Y_i, Y'_i\right)\\ &\quad +\gamma_i h^2\left(\frac{\partial f}{\partial t}(t_n,y_n,y'_n)+J_1 y'_n\right)\\ &\quad +h^2 J_1\sum_{j=1}^{i}\theta_{ij}\ell_j + hJ_2\sum_{j=1}^{i}\gamma_{ij}\ell_j\end{aligned} \tag{24e}$$

It can be seen in Eq.(22) that matrix-vector multiplications are needed. Because of the presence of both J_1 and J_2, the classical transformation removes only the multiplications with one of the J's. Substituting $z_i = \sum_{j=1}^{i}\gamma_{ij}\ell_j$ into the method, Eqs.(24a–24e) leads to the following.

THEOREM 2.1. *Let (α_{ij}), (γ_{ij}), (b_i), and $(\hat{b}_i)$, be the coefficients of an s-stage embedded Rosenbrock method given by Eqs.(11a–11b). The associated Rosenbrock-Nystrom method is defined as*

$$y_{n+1} = y_n + hy'_n + h\sum_{i=1}^{s}\mu_i z_i\ , \quad \hat{y}_{n+1} = y_n + hy'_n + h\sum_{i=1}^{s}\hat{\mu}_i z_i \tag{25a}$$

$$y'_{n+1} = y'_n + \sum_{i=1}^{s} m_i z_i\ , \quad \hat{y}'_{n+1} = y'_n + \sum_{i=1}^{s}\hat{m}_i z_i \tag{25b}$$

$$Y_i = y_n + h\alpha_i y'_n + h\sum_{j=1}^{i-1}\theta_{ij}z_j \tag{25c}$$

$$Y'_i = y'_n + \sum_{j=1}^{i-1}a_{ij}z_j \tag{25d}$$

$$S = I - h\gamma J_2 - h^2\gamma^2 J_1 \tag{25e}$$

$$\begin{aligned}S\cdot z_i &= h\gamma f\left(t_n+\alpha_i h, Y_i, Y'_i\right) + h^2\gamma\gamma_i\left(\frac{\partial f}{\partial t}(t_n,y_n,y'_n)+J_1 y'_n\right)\\ &\quad +\gamma\sum_{j=1}^{i-1}c_{ij}z_j + h^2\gamma J_1\sum_{j=1}^{i-1}\delta_{ij}z_j\end{aligned} \tag{25f}$$

where the new coefficients are

$$\begin{aligned}(a_{ij}) &= (\alpha_{ij})\cdot(\gamma_{ij})^{-1}\\ (c_{ij}) &= \gamma^{-1}I - (\gamma_{ij})^{-1}\\ (\delta_{ij}) &= (\gamma_{ij}) + (\gamma_{ij})\cdot(\alpha_{ij})\cdot(\gamma_{ij})^{-1}\\ (\theta_{ij}) &= (\alpha_{ij}) + (\alpha_{ij})^2\cdot(\gamma_{ij})^{-1}\\ (m_i) &= (b_i)\cdot(\gamma_{ij})^{-1}\\ (\hat{m}_i) &= \left(\hat{b}_i\right)\cdot(\gamma_{ij})^{-1}\end{aligned}$$

$$
\begin{aligned}
(\mu_i) &= (b_i) + (b_i) \cdot (\alpha_{ij}) \cdot (\gamma_{ij})^{-1} \\
(\hat{\mu}_i) &= \left(\hat{b}_i\right) + \left(\hat{b}_i\right) \cdot (\alpha_{ij}) \cdot (\gamma_{ij})^{-1} \\
(\gamma_i) &= (1, \ldots, 1) \cdot (\gamma_{ij})^T \\
(\alpha_i) &= (1, \ldots, 1) \cdot (\alpha_{ij})^T
\end{aligned}
$$

Proof. By direct substitution. □

Note that in Eqs.(25a) and (25b), values for a second, and typically lower order, approximation is provided at time t_{n+1} for both the solution $\hat{y}_{n+1}$ and its first derivative $\hat{y}'_{n+1}$. These values are used for step-size control, as indicated in Eqs.(12) through (16).

3. Providing analytical integration Jacobians. A successful implementation of the Rosenbrock family of formulas introduced above depends upon the ability to provide exact derivative information. Equation (19) indicates the derivatives that must be computed. The numerical solution of a dynamic analysis problem carried out in the proposed framework of a Rosenbrock implicit integrator requires exact computation of the derivatives of independent acceleration with respect to independent positions and velocities,

$$
\mathbf{J}_1 = \ddot{\mathbf{v}}_{\mathbf{v}} \ , \qquad \mathbf{J}_2 = \ddot{\mathbf{v}}_{\dot{\mathbf{v}}} \ . \tag{26}
$$

To obtain these derivatives, first differentiating Eq.(7a) with respect to $\mathbf{v}$ yields

$$
\begin{aligned}
\mathbf{M^{vv}J}_1 &+ \left(\mathbf{M^{vv}}\ddot{\mathbf{v}}\right)_{\mathbf{v}} + \left(\mathbf{M^{vv}}\ddot{\mathbf{v}}\right)_{\mathbf{u}}\mathbf{u}_{\mathbf{v}} + \mathbf{M^{vu}}\ddot{\mathbf{u}}_{\mathbf{v}} + \left(\mathbf{M^{vu}}\ddot{\mathbf{u}}\right)_{\mathbf{v}} + \left(\mathbf{M^{vu}}\ddot{\mathbf{u}}\right)_{\mathbf{u}}\mathbf{u}_{\mathbf{v}} \\
+\mathbf{\Phi}_{\mathbf{v}}^T\lambda_{\mathbf{v}} &+ \left(\mathbf{\Phi}_{\mathbf{v}}^T\lambda\right)_{\mathbf{v}} + \left(\mathbf{\Phi}_{\mathbf{v}}^T\lambda\right)_{\mathbf{u}}\mathbf{u}_{\mathbf{v}} = \mathbf{Q}_{\mathbf{v}}^{\mathbf{v}} + \mathbf{Q}_{\mathbf{u}}^{\mathbf{v}}\mathbf{u}_{\mathbf{v}} + \mathbf{Q}_{\dot{\mathbf{u}}}^{\mathbf{v}}\dot{\mathbf{u}}_{\mathbf{v}}
\end{aligned} \tag{27}
$$

The partial differential notation is explained in the Appendix. The quantities $\mathbf{u}_{\mathbf{v}}$, $\dot{\mathbf{u}}_{\mathbf{v}}$, $\ddot{\mathbf{u}}_{\mathbf{v}}$, and $\lambda_{\mathbf{v}}$ are obtained by taking partials with respect to $\mathbf{v}$ of Eqs.(7c), (7d), (7e), and (7b), respectively. This is an exercise in chain rule differentiation that yields [10, 11]

$$
\mathbf{u}_{\mathbf{v}} = -\mathbf{\Phi}_{\mathbf{u}}^{-1}\mathbf{\Phi}_{\mathbf{v}} \equiv \mathbf{H} \tag{28a}
$$

$$
\dot{\mathbf{u}}_{\mathbf{v}} = -\mathbf{\Phi}_{\mathbf{u}}^{-1}\left[\left(\mathbf{\Phi}_{\mathbf{q}}\dot{\mathbf{q}}\right)_{\mathbf{v}} + \left(\mathbf{\Phi}_{\mathbf{q}}\dot{\mathbf{q}}\right)_{\mathbf{u}}\mathbf{H}\right] \equiv \mathbf{J} \tag{28b}
$$

$$
\ddot{\mathbf{u}}_{\mathbf{v}} = \mathbf{H}\mathbf{J}_1 + \mathbf{L} \tag{28c}
$$

$$
\lambda_{\mathbf{v}} = -\mathbf{\Phi}_{\mathbf{u}}^{-T}\left[\mathbf{R} + \left(\mathbf{M^{uv}} + \mathbf{M^{uu}}\mathbf{H}\right)\mathbf{J_1}\right] \tag{28d}
$$

where

$$
\mathbf{L} = \mathbf{\Phi}_{\mathbf{u}}^{-1}\left[\left[\tau_{\mathbf{u}} - \left(\mathbf{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}}\right)_{\mathbf{u}}\right]\mathbf{H} + \tau_{\mathbf{v}} + \tau_{\dot{\mathbf{u}}}\mathbf{J} - \left(\mathbf{\Phi}_{\mathbf{q}}\dot{\mathbf{q}}\right)_{\mathbf{v}}\right] \ , \tag{29}
$$

$$
\begin{aligned}
\mathbf{R} = {} & \left[\left(\mathbf{\Phi}_{\mathbf{u}}^T\lambda\right)_{\mathbf{u}} + \left(\mathbf{M^u}\ddot{\mathbf{q}}\right)_{\mathbf{u}} - \mathbf{Q}_{\mathbf{u}}^{\mathbf{u}}\right]\mathbf{H} - \mathbf{Q}_{\mathbf{v}}^{\mathbf{u}} - \mathbf{Q}_{\dot{\mathbf{u}}}^{\mathbf{u}}\mathbf{J} \\
& + \left(\mathbf{\Phi}_{\mathbf{u}}^T\lambda\right)_{\mathbf{v}} + \left(\mathbf{M^u}\ddot{\mathbf{q}}\right)_{\mathbf{v}} + \mathbf{M^{uu}}\mathbf{L} \ , \quad \text{with } \mathbf{M^u} = \left[\mathbf{M^{uv}}, \mathbf{M^{uu}}\right] \ .
\end{aligned} \tag{30}
$$

Substituting the expressions for $\mathbf{u_v}$, $\dot{\mathbf{u}}_\mathbf{v}$, $\ddot{\mathbf{u}}_\mathbf{v}$, and $\lambda_\mathbf{v}$ into Eq.(28) and denoting $\mathbf{M^v} = [\mathbf{M^{vv}}, \mathbf{M^{vu}}]$ yields

$$(31)\ \widehat{\mathbf{M}}\mathbf{J}_1 = \mathbf{Q}^\mathbf{v}_\mathbf{v} + \mathbf{Q}^\mathbf{v}_\mathbf{u}\mathbf{H} + \mathbf{Q}^\mathbf{v}_{\dot{\mathbf{u}}}\mathbf{J} - \left[\mathbf{M^{vu}}\mathbf{L} + \mathbf{H^T}\mathbf{R} + \left(\mathbf{\Phi}^\mathbf{T}_\mathbf{v}\lambda\right)_\mathbf{u}\mathbf{H} + \left(\mathbf{\Phi}^\mathbf{T}_\mathbf{v}\lambda\right)_\mathbf{v} + \left(\mathbf{M^v}\ddot{\mathbf{q}}\right)_\mathbf{v} + \left(\mathbf{M^v}\ddot{\mathbf{q}}\right)_\mathbf{u}\mathbf{H}\right]$$

According to Lemma 1.1, the coefficient matrix in this multiple right side linear system is positive definite. Therefore, Eq.(32) properly defines the derivative $\mathbf{J}_1$.

Computation of $\mathbf{J}_2 = \ddot{\mathbf{v}}_{\dot{\mathbf{v}}}$ follows the steps taken for the computation of $\mathbf{J}_1$. Taking the derivative of Eq.(7a) with respect to $\dot{\mathbf{v}}$ yields

$$\mathbf{M^{vv}}\mathbf{J}_2 + \mathbf{M^{vu}}\ddot{\mathbf{u}}_{\dot{\mathbf{v}}} + \mathbf{\Phi}^T_\mathbf{v}\lambda_{\dot{\mathbf{v}}} = \mathbf{Q}^\mathbf{v}_{\dot{\mathbf{u}}}\dot{\mathbf{u}}_{\dot{\mathbf{v}}} + \mathbf{Q}^\mathbf{v}_{\dot{\mathbf{v}}} \tag{32}$$

All derivatives in this expression are available, except the quantities $\mathbf{J}_2$, $\dot{\mathbf{u}}_{\dot{\mathbf{v}}}$, $\ddot{\mathbf{u}}_{\dot{\mathbf{v}}}$, and $\lambda_{\dot{\mathbf{v}}}$. The last three derivatives are obtained by taking partial derivatives with respect to the independent velocity $\dot{\mathbf{v}}$ of Eqs.(7d), (7e), and (7b). By repeatedly applying the chain rule of differentiation these derivatives are obtained as

$$\dot{\mathbf{u}}_{\dot{\mathbf{v}}} = \mathbf{H} \tag{33a}$$

$$\ddot{\mathbf{u}}_{\dot{\mathbf{v}}} = \mathbf{N} + \mathbf{H}\mathbf{J}_2 \tag{33b}$$

$$\lambda_{\dot{\mathbf{v}}} = \mathbf{\Phi}^{-T}_\mathbf{u}\left[\mathbf{Q}^\mathbf{u}_{\dot{\mathbf{u}}}\mathbf{H} + \mathbf{Q}^\mathbf{u}_{\dot{\mathbf{v}}} - \mathbf{M^{uu}}\mathbf{N} - \left(\mathbf{M^{uv}} + \mathbf{M^{uu}}\mathbf{H}\right)\mathbf{J}_2\right] \tag{33c}$$

where

$$\mathbf{N} = \mathbf{\Phi}^{-1}_\mathbf{u}\left(\tau_{\dot{\mathbf{u}}}\mathbf{H} + \tau_{\dot{\mathbf{v}}}\right) \tag{34}$$

Substituting these results into Eq.(32), the derivative of independent accelerations with respect to independent velocities is obtained as the solution of the multiple right side system of linear equations,

$$\widehat{\mathbf{M}}\mathbf{J}_2 = \mathbf{W} - \mathbf{H}^T\mathbf{X} \tag{35}$$

where

$$\mathbf{W} = \mathbf{Q}^\mathbf{v}_{\dot{\mathbf{u}}}\mathbf{H} + \mathbf{Q}^\mathbf{v}_{\dot{\mathbf{v}}} - \mathbf{M^{vu}}\mathbf{N}\ ,$$
$$\mathbf{X} = -\left(\mathbf{Q}^\mathbf{u}_{\dot{\mathbf{u}}}\mathbf{H} + \mathbf{Q}^\mathbf{u}_{\dot{\mathbf{v}}} - \mathbf{M^{uu}}\mathbf{N}\right)\ .$$

With $\widehat{\mathbf{M}}$ positive definite, Eq.(35) properly defines the derivative $\mathbf{J}_2$.

A general framework is provided in this Section to analytically express the integration Jacobian required by the Rosenbrock family of integration formulas. Although rather involved in form, these derivatives are obtained in a straightforward way. Furthermore, they are generic, in the sense that they apply to any mechanical system simulation. It remains to provide all the ingredients that explicitly or implicitly enter the right side of Eqs.(32) and (35). These derivatives change according to the modeling elements used to represent a mechanical system. What makes the approach viable is the fact that even these derivatives can be generated in a completely generic way. This is solely a mechanical system modeling task that hinges upon the fact that, in multibody dynamics, all modeling elements are broken down into primitives. Providing required derivative information for these primitives in Cartesian coordinates is tractable. Derivative information for primitives is then combined to produce derivatives for complex modeling entities. To illustrate this, consider the joints used to connect bodies in a mechanical system model. The vast majority of them can be obtained starting from four simple constraint primitives [9]:

$\mathbf{\Phi}^{d1}$ Dot-1 constraint primitive, imposes that two body-fixed non-zero vectors $\mathbf{a}_i$ and $\mathbf{a}_j$ belonging to bodies i and j respectively, should be perpendicular at all times; i.e., $\mathbf{\Phi}^{d1}(\mathbf{a}_i, \mathbf{a}_j) = \mathbf{a}_i{}^T\mathbf{a}_j = 0$

$\mathbf{\Phi}^{d2}$ Dot-2 constraint primitive, imposes that one body-fixed vector $\mathbf{a}_i$ and a vector $\mathbf{d}_{ij}$ defined by two body-fixed points P_i and P_j should be perpendicular at all times; i.e., $\mathbf{\Phi}^{d2}(\mathbf{a}_i, \mathbf{d}_{ij}) = \mathbf{a}_i{}^T\mathbf{d}_{ij} = 0$

$\mathbf{\Phi}^{s}$ Point Constraint Primitive, imposes that two body-fixed points P_i and P_j belonging to bodies i and j respectively, should coincide at all times; i.e., $\mathbf{\Phi}^{S}(P_i, P_j) = P_i \equiv P_j$

$\mathbf{\Phi}^{dist}$ Distance Constraint Primitive, imposes that the distance between two body-fixed points P_i and P_j belonging to bodies i and j should stay constant and equal to $C > 0$ at all times; i.e., $\mathbf{\Phi}^{dist}(P_i, P_j, C) = dist(P_i, P_j) = C$

In this context, a universal joint that allows two relative degrees of freedom between the constrained bodies is defined by requiring that a Dot-1 and a Point Constraint Primitive be simultaneously satisfied throughout the simulation. Likewise, a spherical joint that allows three degrees of freedom (rotational) between the constrained bodies is simply a Point Constraint Primitive, while a revolute joint is the assembly of two Dot-1 and one Point Constraint primitives.

Analyzing the order of derivatives used to compute the required derivative information $\mathbf{J}_1$ and $\mathbf{J}_2$, it can be seen that the highest order is 3, and it appears as a result of taking partials of the right side of the acceleration kinematic constraint equation. Consequently, derivatives of the four constraint primitives introduced above should be implemented up to order 3. Deriving and coding expressions for all derivatives for the modeling primitives up to order 3 is a one time effort. For details about how these derivatives are obtained for constraint primitive, inertia elements, and forces the interested reader is reffered to [16]. For the scope of the present paper, it suffices to assume that the derivatives required to analytically compute the Rosenbrock integration Jacobian are readily available.

4. Proposed algorithm. For multibody dynamic analysis problems, a low to medium accuracy method with very good stability properties is desirable. Formulas with few function and Jacobian evaluations are favored, since obtaining accelerations and the integration Jacobian are costly operations. A method that is L-stable allows for robust integration of very stiff problems, which enables efficient handling of bushing elements and flexible components used in modeling mechanical systems. In this context the focus is on a L-stable order 4 Rosenbrock-Nystrom method with 4 stages. This allows for room in determining a second set of coefficients that efficiently provide an order 3 embedded formula for step-size control. Furthermore, following an idea of [7], the number of function evaluations for the 4 stage method is kept to 3; i.e., one function evaluation is saved. In terms of function evaluations, this makes the proposed Rosenbrock-Nystrom method competitive with the trapezoidal method, whenever the latter requires 3 or more iterations for convergence. Moreover, the trapezoidal method is of order 2 and only weakly A-stable.

With $\beta'_i = \sum_{j=1}^{i-1} \beta_{ij}$, the defining coefficients α_{ij}, γ_{ij}, and b_i of an order 4 Rosenbrock method of Eqs.(11a– 11b) are subject to the following order conditions [7]:

$$b_1 + b_2 + b_3 + b_4 = 1 \tag{36a}$$

$$b_2\beta'_2 + b_3\beta'_3 + b_4\beta'_4 = 1/2 - \gamma \tag{36b}$$

$$b_2\alpha_2^2 + b_3\alpha_3^2 + b_4\alpha_4^2 = 1/3 \tag{36c}$$

$$b_3\beta_{32}\beta'_2 + b_4\left(\beta_{42}\beta'_2 + \beta_{43}\beta'_3\right) = 1/6 - \gamma + \gamma^2 \tag{36d}$$

$$b_2\alpha_2^3 + b_3\alpha_3^3 + b_4\alpha_4^3 = 1/4 \tag{36e}$$

$$b3\alpha_3\alpha_{32}\beta'_2 + b_4\alpha_4\left(\alpha_{42}\beta'_2 + \alpha_{43}\beta'_3\right) = 1/8 - \gamma/3 \tag{36f}$$

$$b_3\beta_{32}\alpha_2^2 + b_4\left(\beta_{42}\alpha_2^2 + \beta_{43}\alpha_3^2\right) = 1/12 - \gamma/3 \tag{36g}$$

$$b_4\beta 43\beta_{32}\beta'_2 = 1/24 - \gamma/2 + 1.5\gamma^2 - \gamma^3 \tag{36h}$$

The stage values k_i of the order 4 method are recycled, for the purpose of automatic step-size control, to provide an embedded formula of order 3 of the form $\hat{y}_1 = y_0 + \sum_{i=1}^{s} \hat{b}_i k_i$. The order conditions for the order 3 algorithm are as indicated as Eqs.(36a– 36d). These conditions lead to the system

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & \beta'_2 & \beta'_3 & \beta'_4 \\ 0 & \alpha_2^2 & \alpha_3^2 & \alpha_4^2 \\ 0 & 0 & \beta_{32}\beta'_2 & \beta_{42}\beta'_2 + \beta_{43}\beta'_3 \end{bmatrix} \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ \hat{b}_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 - \gamma \\ 1/3 \\ 1/6 - \gamma + \gamma^2 \end{bmatrix} \tag{37}$$

If the coefficient matrix in Eq.(37) is non-singular, uniqueness of the solution of this linear system implies $b_1 = \hat{b}_i$. To prevent this, one additional condition is considered to obtain a distinct order 3 embedded formula. It requires the coefficient matrix in Eq.(37) to be singular, which results in the condition

$$\beta_{32}\beta'_2(\beta'_2\alpha_4^2 - \beta'_4\alpha_2^2) = (\beta'_2\alpha_3^2 - \beta'_3\alpha_2^2)(\beta_{42}\beta'_2 + \beta_{43}\beta'_3) \tag{38}$$

The number of coefficients that must be determined is 17; the diagonal coefficient γ, six coefficients γ_{ij}, six coefficients α_{ij}, and four weights b_i. The number of conditions that these coefficients have to satisfy is nine. There are eight degrees of freedom in the choice of coefficients and some of these are used to construct a method with one less function evaluation. Thus, if

$$\begin{aligned} \alpha_{41} &= \alpha_{31} \\ \alpha_{42} &= \alpha_{32} \\ \alpha_{43} &= 0 \end{aligned} \tag{39}$$

stage 4 of the algorithm saves one function evaluation. Finally, the free parameters can be determined such that several order 5 conditions of the otherwise order 4 proposed formula are satisfied. When the conditions of Eq.(39) hold, one of the nine order 5 conditions associated with a Rosenbrock type formula leads to

$$\alpha_3 = \frac{1/5 - \alpha_2/4}{1/4 - \alpha_2/3} \tag{40}$$

A second order 5 condition is satisfied by imposing the condition

$$b_4\beta_{43}\alpha_3^2(\alpha_3 - \alpha_2) = 1/20 - \gamma/4 - \alpha_2\left(1/12 - \gamma/3\right) \tag{41}$$

Next, two conditions are chosen as

$$\begin{aligned} b_3 &= 0 \\ \alpha_2 &= 2\gamma \end{aligned} \tag{42}$$

to make the task of finding the defining coefficients α_{ij}, γ_{ij}, and b_i more tractable. Finally, the last condition regards the choice of the diagonal element γ. The value

of this parameter determines the stability properties of the Rosenbrock method. In this context, the diagonal entry of the Rosenbrock formula is suggested in [7] as $\gamma = 0.57281606$, which is adopted for the proposed algorithm. With this, there is a set of 17 equations, some of them non-linear, in 17 unknowns. The solution of this system, accurate up to 25 digits, is provided below, along with the coefficients $\hat{b}_i$ of the order 3 embedded formula.

$\gamma = 0.57281606$					
α_{21}	=	1.14563212	γ_{21}	=	2.3419931271120139491705 20
α_{31}	=	0.5209207891306290293285 16	γ_{31}	=	−0.0273337465434898361965 05
α_{32}	=	0.1342941868425048001492 32	γ_{32}	=	0.2138116508366996898674 72
α_{41}	=	0.5209207891306290293285 16	γ_{41}	=	−0.2590838377855102221126 41
α_{42}	=	0.1342941868425048001492 32	γ_{42}	=	−0.1905958077323117516163 58
α_{43}	=	0.0	γ_{43}	=	−0.2280310359731338294777 44
b_1	=	0.3245347078917345134741 96	$\hat{b}_1$	=	0.5209207891306290293285 16
b_2	=	0.0490865447875233086846 33	$\hat{b}_2$	=	0.1445497146653645995846 81
b_3	=	0.0	$\hat{b}_3$	=	0.1245596864147020497748 97
b_4	=	0.6263787473207421778411 71	$\hat{b}_4$	=	0.2099698097893043213119 06

Once the coefficients of the underlying Rosenbrock formula are available, the coefficients of the Rosenbrock-Nystrom formula defined in Theorem 2.1 are easily computed. The full set of coefficients for the order 4, L-stable formula is provided below.

θ_{21}	=	1.14563212	a_{21}	=	0.2000000000000000000 00
θ_{31}	=	0.7895091628156386296269 80	a_{31}	=	1.8679481494982371323447 6
θ_{32}	=	0.1342941868425048001492 32	a_{32}	=	0.2344455685172388500232 2
θ_{41}	=	0.7895091628156386296269 80	a_{41}	=	1.8679481494982371323447 6
θ_{42}	=	0.1342941868425048001492 32	a_{42}	=	0.2344455685172388500232 2
θ_{43}	=	0.0	a_{43}	=	0.0
c_{21}	=	−7.1376499433499798303692 60	δ_{21}	=	−1.1963610071120139491705 20
c_{31}	=	2.5809236665096577144880 50	δ_{31}	=	1.4702802544097807146338 70
c_{32}	=	0.6516298873020320233874 17	δ_{32}	=	0.3481058376792044900167 04
c_{41}	=	−2.1371152665066191168063 70	δ_{41}	=	0.0037650943555561657989 74
c_{42}	=	−0.3214695313399510707692 41	δ_{42}	=	−0.1097624867581032556753 98
c_{43}	=	−0.6949660492824452251573 29	δ_{43}	=	−0.2280310359731338294777 44
m_1	=	2.2555662286045652437288 40	$\hat{m}_1$	=	2.0683991605275837342586 70
m_2	=	0.2870550631941576076626 30	$\hat{m}_2$	=	0.2386813520675327979564 93
m_3	=	0.4353119633797983213402 707	$\hat{m}_3$	=	0.3633733454353917082617 47
m_4	=	1.0935076564032478032148 20	$\hat{m}_4$	=	0.3665571279361551443091 63
μ_1	=	1.5927508194095853420749 00	$\hat{\mu}_1$	=	1.4349039718482094726271 00
μ_2	=	0.1959382663102506096933 29	$\hat{\mu}_2$	=	0.2229786725886983690451 53
μ_3	=	0.0	$\hat{\mu}_3$	=	0.1245596864147020497748 97
μ_4	=	0.6263787473207421778411 71	$\hat{\mu}_4$	=	0.2099698097893043213119 06
γ_2	=	−1.7691770671120139491705 20	a_2	=	1.145632120
γ_3	=	0.7592939642932098536709 67	a_3	=	0.6552149759731338294777 48
γ_4	=	−0.1048946214909558032067 43	a_4	=	0.6552149759731338294777 48

It should be recalled that any Rosenbrock type formula requires an exact Jacobian for the numerical solution to maintain its stability and accuracy properties. Sometimes this might be a very challenging requirement. Consider for example the situation when complex tire models are present in a model, or for a general purpose solver the case when user defined external routines are employed for the computation of active

forces such as aerodynamic forces. Providing an exact Jacobian for these situations is very unlikely. Verwer et al. 1997, proposed a second order W-method [8], which is a Rosenbrock type method in the sense that it does not necessitate the solution of a non-linear system, but which does not require an exact Jacobian. The defining coefficients for this method are provided in the table below.

$\gamma = 1.70710678118650$					
α_1	=	0.00000000000000	γ_1	=	1.70710678118650
α_2	=	1.00000000000000	γ_2	=	-1.70710678118650
a_{21}	=	0.58578643762690	c_{21}	=	-1.17157287525380
δ_{11}	=	1.70710678118650	δ_{22}	=	1.70710678118650
δ_{21}	=	-2.41421356237310			
θ_{21}	=	1.00000000000000			
m_1	=	0.87867965644040	$\hat{m}_1$	=	1.17157287525380
m_2	=	0.29289321881340	$\hat{m}_2$	=	0.58578643762690
μ_1	=	0.79289321881340	$\hat{\mu}_1$	=	0.58578643762690
μ_2	=	0.50000000000000	$\hat{\mu}_2$	=	1.00000000000000

This paper is concerned with the implementation of a Rosenbrock-Nystrom based method. **Algorithm 1** based on the 4 stage, order 4 L-stable Rosenbrock formula introduced is presented below. The W-method can be similarly implemented by replacing the corresponding coefficients of the Rosenbrock-Nystrom formula with the coefficients provided in the previous table. Details of this implementation and the performance of such a method are not the presented here.

Algorithm 1

1. Initialize Simulation
2. Set Integration Tolerance
3. While (time < time-end) do
4. Set Macro-step
5. Get Integration Jacobian
6. Factor Integration Jacobian
7. Get Time Derivative
8. Resolve Stage 1
9. Resolve Stage 2
10. Resolve Stage 3
11 Resolve Stage 4
12. Get Solution. Check Accuracy. Determine New Step-size
13. Recover Dependent Generalized Coordinates
14. Check Partition
15. End do

Step 1 initializes the simulation. Based on user provided values at time t_0, a consistent set of initial conditions $(\mathbf{u}_0, \mathbf{v}_0, \dot{\mathbf{u}}_0, \dot{\mathbf{v}}_0)$; i.e., satisfying Eqs.(7c) and (7d), is determined and simulation starting and ending times are defined. User defined integration tolerances $Atol_i$, and $Rtol_i$ of Eq.(13) are read in and set during Step 2. These tolerances are used to control error in both independent position $\mathbf{v}$ and velocity $\dot{\mathbf{v}}$. Step 4 backs up the system configuration to be used upon a rejected time step. During Step 5, the integration Jacobian is evaluated. Since obtaining $\mathbf{J}_1$ and $\mathbf{J}_2$ is a costly operation, Eqs.(32) and (35) suggest that $\mathbf{z}_i$ is more efficiently computed if the stage linear system of Eq.(25f) is replaced with an equivalent one obtained by formally

multiplying the original linear system with the positive definite matrix $(1/\gamma)\widehat{\mathbf{M}}$. The new linear system assumes the form

$$\mathbf{\Pi}\mathbf{z}_i = \mathbf{r}_i \tag{43}$$

where, with $\widehat{\mathbf{M}}$ and $\widehat{\mathbf{Q}}$ defined in Eqs.(10a) and (10b),

$$\mathbf{\Pi} = \frac{1}{\gamma^2}\widehat{\mathbf{M}}\cdot\mathbf{S} = \frac{1}{\gamma^2}\widehat{\mathbf{M}} - \frac{h}{\gamma}\widehat{\mathbf{M}}\mathbf{J}_2 - h^2\widehat{\mathbf{M}}\mathbf{J}_1 \tag{44}$$

$$\begin{aligned}\mathbf{r}_i \;=\; & \frac{h}{\gamma}\widehat{\mathbf{Q}}\left(t_n+\alpha_i h, \mathbf{V}_i, \dot{\mathbf{V}}_i\right) + h^2\frac{\gamma_i}{\gamma}\left(\widehat{\mathbf{M}}\frac{\partial\ddot{\mathbf{v}}}{\partial t}(t_n,\mathbf{v}_n,\dot{\mathbf{v}}_n) + \widehat{\mathbf{M}}\mathbf{J}_1\dot{\mathbf{v}}_n\right) \\ & + \frac{1}{\gamma}\widehat{\mathbf{M}}\sum_{j=1}^{i-1} c_{ij}\mathbf{z}_j + \frac{h^2}{\gamma}\widehat{\mathbf{M}}\mathbf{J}_1\sum_{j=1}^{i-1}\delta_{ij}\mathbf{z}_j\end{aligned} \tag{45}$$

Thus, the linear systems in Eqs.(32) and (35) need not be solved for $\mathbf{J}_1$ and $\mathbf{J}_2$. Finding only the right side of these two linear systems suffices to compute the coefficient matrix $\mathbf{\Pi}$ and right side $\mathbf{r}_i$ at each integration stage. In [12] it is showed that the matrix $\mathbf{\Pi}$ of Eq.(43)is the integration Jacobian that also appears in the context of multistep implicit integration of the DAE of multibody dynamics. This observation allows for a unitary implementation of implicit methods, based on either singly diagonal implicit Runge-Kutta formulas or on multistep BDF-type formulas.

The matrix Π is factored during Step 6. The dimension of this matrix is equal to the number of degrees of freedom of the mechanical system model, and is typically small. Consequently, here and for that matter everywhere else the proposed algorithm uses LAPACK and level 1 and 2 BLAS dense linear algebra routines [1].

During Step 7, the quantity $\widehat{\mathbf{M}}\partial\ddot{\mathbf{v}}/\partial t$ needed to compute the stage right side $\mathbf{r}_i$ is evaluated in the consistent configuration $(\mathbf{u}_n, \mathbf{v}_n, \dot{\mathbf{u}}_n, \dot{\mathbf{v}}_n)$ from the beginning of each macro-step. As the position kinematic constraints in Eq.(2a) are assumed time independent, $\widehat{\mathbf{M}}$ does not depended on time. Therefore, using Eq.(9) and Eq.(10b),

$$\widehat{\mathbf{M}}\frac{\partial\ddot{\mathbf{v}}}{\partial t} = \frac{\partial}{\partial t}\widehat{\mathbf{Q}}(t,\mathbf{v}_n,\dot{\mathbf{v}}_n) = \mathbf{Q}_t^{\mathbf{v}} + \mathbf{H}^T\mathbf{Q}_t^{\mathbf{u}} \tag{46}$$

The simplifying assumptions made in computing $\widehat{\mathbf{M}}\partial\ddot{\mathbf{v}}/\partial t$ in Eq.(46) are that the kinematic constraint equations are time independent and holonomic. The first assumption is to quantitatively simplify the presentation. Otherwise, the algorithm would step by step follow the derivation for the time independent case, with the caveat that terms of the form $\mathbf{\Phi}_t$, $\mathbf{\Phi}_{\mathbf{u}t}$, $\mathbf{\Phi}_{tt}$, etc., would have to be accounted for. As a result, all derivatives would be more complicated. The second assumption is made because covering the non-holonomic constraint case is a qualitatively different process, which is not targeted by this paper. For a thorough account of the non-holonomic scenario, the reader is referred to [15]. Finally, note that in the case of scleronomic mechanical systems, $\widehat{\mathbf{M}}\partial\ddot{\mathbf{v}}/\partial t = 0$.

The next four steps compute the stage variables $\mathbf{z}_i$ of Eq.(25f). During each of the four stages of the Rosenbrock-Nystrom algorithm, some or all of the following steps are taken:

a) Obtain consistent configuration at position and velocity levels.

b) Compute stage generalized forces $\widehat{\mathbf{Q}}_i$

c) Compute stage right side $\mathbf{r}_i$ as in Eq.(46)
d) Solve stage linear system of Eq.(43) to obtain $\mathbf{z}_i$

With regard to sub-step (a)according to Eq.(25c), defining $\mathbf{V}_i = \mathbf{v}_n + h\alpha_i\dot{\mathbf{v}}_n + h\sum_{j=1}^{i-1}\theta_{ij}\mathbf{z}_j$, the stage dependent generalized coordinates $\mathbf{U}^i$ are the solution of the non-linear system $\mathbf{\Phi}(\mathbf{U}^i, \mathbf{V}^i) = \mathbf{0}$. The matrix $\mathbf{\Phi_u}$, along with its factorization, are at the cornerstone of the algorithm, and Eq.(5) guarantees that $\mathbf{U}^i$ is properly defined. Likewise, denoting the stage independent velocities as $\dot{\mathbf{V}}^i = \dot{\mathbf{v}}_n + \sum_{j=1}^{i-1}\alpha_{ij}\mathbf{z}_j$, the stage dependent velocities $\dot{\mathbf{U}}_i$ are, as in Eq.(25d), the solution of the linear system $\mathbf{\Phi_u}(\mathbf{U}^i, \mathbf{V}^i)\dot{\mathbf{U}}^i = -\mathbf{\Phi_v}(\mathbf{U}^i, \mathbf{V}^i)\dot{\mathbf{V}}^i$. Finally, note that the stage forces $\widehat{\mathbf{Q}}_i = \widehat{\mathbf{Q}}\left(t_n + \alpha_i h, \mathbf{V}_i, \dot{\mathbf{V}}_i\right)$ are computed during sub-step (b), as in Eq.(10b), using the factorization of the matrix $\mathbf{\Phi_u}(\mathbf{U}^i, \mathbf{V}^i)$ available at the end of sub-step (a).

Due to the particular choice of coefficients defining the Rosenbrock-Nystrom formula and the way in which the code was implemented, each of the four stages has its own particularities. Thus,

- Stage 1; i.e., Step 8 of the algorithm, marks the beginning of a new integration step, or equivalently the end of the prior one. Therefore the system is in an assembled configuration and sub-step (a) above is skipped. During this stage, the matrix $\mathbf{\Pi}$ of Eq.(43) is evaluated, and generalized accelerations are obtained as a by product of the process. Thus, to obtain $\mathbf{\Pi}$, the matrix $\widehat{\mathbf{M}}$ is computed and the dependent constraint sub-Jacobian $\mathbf{\Phi_u}$ is factored. The latter is then used to obtain the matrices $\mathbf{H}$, $\mathbf{J}$, $\mathbf{L}$, and $\mathbf{N}$ of Section 3. Notice that, due to the choice of a constant diagonal element γ for the original Rosenbrock method, the matrix $\mathbf{\Pi}$ is constant and needs to be factored only once. The remaining 3 stages reuse this factorization.
- Stages 2 and 3 of the Rosenbrock-Nystrom formula follow exactly the sub-steps (a) through (d) outlined above.
- Stage 4; i.e., Step 11, bypasses the stage generalized force computation of $\widehat{\mathbf{Q}}_i$ in Eq.(46), because of the special choice of formula coefficients (see Eq.(39)). Since no force computation is required, there is no need to provide consistent position and velocity configurations, consequently sub-step (a) is skipped. It remains to compute the right side $\mathbf{r}_4$ and, with the coefficient matrix already factored to do a forward/backward substitution, to obtain $\mathbf{z}_4$.

During Step 12, the position level independent generalized coordinates at time-step $n+1$ are computed according to Eq.(25a) as $\mathbf{v}_{n+1} = \mathbf{v}_n + h\dot{\mathbf{v}}_n + h\sum_{i=1}^{s}\mu_i\mathbf{z}_i$. Likewise, according to Eq.(25b), the velocity level independent generalized coordinates are computed as $\dot{\mathbf{v}}_{n+1} = \dot{\mathbf{v}}_n + \sum_{i=1}^{s} b_i\mathbf{z}_i$. The accuracy of the solution is verified using a second approximation of the solution at time $n+1$. The less accurate solution is provided by the embedded order 3 formula $\hat{\mathbf{v}}_{n+1} = \mathbf{v}_n + h\dot{\mathbf{v}}_n + h\sum_{i=1}^{s}\hat{\mu}_i\mathbf{z}_i$ and $\hat{\dot{\mathbf{v}}}_{n+1} = \dot{\mathbf{v}}_n + \sum_{i=1}^{s}\hat{b}_i\mathbf{z}_i$, and it is used in Eqs.(14) and (16) for the purpose of step-size control.

Step 13 computes dependent position level generalized coordinates $\mathbf{u}_{n+1}$ such that they satisfy $||\mathbf{\Phi}(\mathbf{u}_{n+1}, \mathbf{v}_{n+1})||_\infty < tol$. Dependent velocities are obtained as the solution of the linear system $\mathbf{\Phi_u}(\mathbf{u}_{n+1}, \mathbf{v}_{n+1})\dot{\mathbf{u}}_{n+1} = -\mathbf{\Phi_v}(\mathbf{u}_{n+1}, \mathbf{v}_{n+1})\dot{\mathbf{v}}_{n+1}$. Note that at the end of this step a factorization of the dependent constraint sub-Jacobian $\mathbf{\Phi_u}(\mathbf{u}_{n+1}, \mathbf{v}_{n+1})$ is available, since it was used to compute the dependent generalized positions.

Along with the factorization of the dependent constraint sub-Jacobian, its condition number is monitored and it is used during Step 14 of the algorithm to check the

partitioning of the generalized coordinates. The current partitioning is reused as long as the condition number of the current dependent constraint sub-Jacobian does not exceed by 25% the value of the condition number produced by the most recent partitioning. This value was determined after carrying out numerical experiments with different values.

5. Numerical Experiments. A set of numerical experiments is first carried out to validate the proposed algorithm. Then a comparison with an explicit integrator is performed to assess the efficiency of the proposed algorithm for numerical integration of a larger stiff mechanical system.

5.1. Validation of Proposed Algorithm. Validation is carried out using the double pendulum mechanism shown in Fig.1. Stiffness is induced by means of two rotational spring-damper-actuators (RSDA). The masses of the two pendulums are $m_1 = 3$ and $m_2 = 0.3$, the dimension of the pendulums are $L_1 = 1$ and $L_2 = 1.5$, the stiffness coefficients are $k_1 = 400$ and $k_2 = 3.E5$, and the damping coefficients are $C_1 = 15$ and $C_2 = 5.E4$. All units are SI. Note that the zero-tension angles for the two RSDA elements are $\alpha_1^0 = 3\pi/2$ and $\alpha_2^0 = 0$. In its initial configuration, the two degree of freedom dynamic system has a dominant eigenvalue with a small imaginary part and a real part of the order -10E5. Since the two pendulums are connected through two parallel revolute joints the problem is planar. In terms of initial conditions, the centers of mass (CM) of bodies 1 and 2 are located at $x_1^{CM} = 1$, $y_1^{CM} = 0$, and $x_2^{CM} = 3.4488887$, $y_2^{CM} = -0.388228$. In the initial configuration, the centroidal principal reference frame of body 1 is parallel with the global reference frame, while the centroidal principal reference frame of body 2 is rotated with $\theta_2 = 23\pi/12$ around an axis perpendicular on the plane of motion. For body 1, $\dot{x}_1^{CM} = \dot{y}_1^{CM} = \dot{\theta}_1^{CM} = 0$, while for body 2, $\dot{x}_2^{CM} = 3.8822857$, $\dot{y}_2^{CM} = 14.4888887$, and $\dot{\theta}_2^{CM} = 10$. Note that all initial conditions are in SI units and are consistent with the kinematic constraint equations at position and velocity levels (Eqs.(2a) and (2b)).

The first set of numerical experiments focuses on assessing the reliability of the step size control mechanism. The goal is to verify that user imposed levels of absolute and relative error are met by the simulation results. For this, a so called *reference simulation* is first run using a very small constant integration step-size. Other simulations, run with different combinations of absolute and relative tolerances, are compared to the *reference simulation* to find the infinity norm of the error, the time at which this largest error occurred, and average error per time step. In this context, suppose that n time steps are taken during the current simulation and that the variable whose accuracy is analyzed is denoted by e. The grid points of the current simulation are denoted by $t_{init} = t_1 < t_2 < \ldots < t_n = t_{end}$. If N is the number of time steps taken during the *reference simulation*; i.e., $T_{init} = T_1 < T_2 < \ldots < T_N = T_{end}$, assume that for the quantity of interest the computed reference values are E_j, for $1 \leq j \leq N$. For each $1 \leq i \leq n$, an integer $\mathbf{r}(i)$ is defined such that $T_{\mathbf{r}(i)} \leq t_i \leq T_{\mathbf{r}(i)+1}$. Using the reference values $E_{\mathbf{r}(i)-1}$, $E_{\mathbf{r}(i)}$, $E_{\mathbf{r}(i)+1}$, and $E_{\mathbf{r}(i)+2}$, spline cubic interpolation is used to generate an interpolated value E_i^* at time t_i. If $\mathbf{r}(i) - 1 \leq 0$, the first four reference points are considered for interpolation, while if $\mathbf{r}(i) + 2 \geq N$, the last four reference points are used for interpolation. The error at time step i is then defined as $\Delta_i = \mid E_i^* - e_i \mid$. For each tolerance set k, accuracy is measured by both the maximum ($\Delta^{(k)}$) and the average ($\overline{\Delta}^{(k)}$) trajectory errors, as well as by the percentage relative

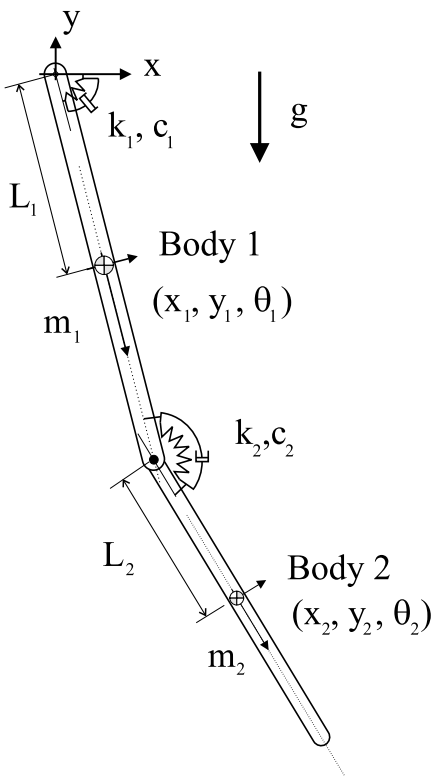

FIG. 1. *Double pendulum problem*

error

$$\Delta^{(k)} = \max_{1\le i\le n}(\Delta_i)\ , \qquad \overline{\Delta}^{(k)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\Delta_i^2}\ , \qquad \text{RelErr}[\%] = \frac{\Delta^{(k)}}{E^*}\times 100\ ,$$

Simulations are run for tolerances between 10^{-2} and 10^{-5}, a range that typically covers mechanical engineering accuracy requirements. The length of the simulation is 2 seconds. The time variation of the angle θ_1 is presented on the left of Fig.2. Notice that body 1 eventually stabilizes in the configuration $\theta_1 = 3\pi/2$, which is the zero-tension angle for the RSDA.

Table 1 contains error analysis information for angle θ_1. The first column contains the value of the tolerance with which the simulation is run. Relative and absolute tolerances ($Rtol_i$ and $Atol_i$ of Eqs.(13)) are set to 10^k, and they are applied for both position and velocity error control. The second column contains the time t^* at which the largest error $\Delta^{(k)}$ occurred. The third column contains the values of $\Delta^{(k)}$. Column four contains the relative error, and the last column shows the average trajectory error.

The most relevant information for step-size control validation is $\Delta^{(k)}$. If, for example, $k = -3$; i.e., accuracy of the order 10^{-3} is demanded, $\Delta^{(-3)}$ should have this order of magnitude. It can be seen from the results in Table 1 that this is the case for all tolerances. Notice that these results are obtained with a non-zero relative tolerance. According to Eq.(13), depending on the magnitude of the variable being analyzed, the

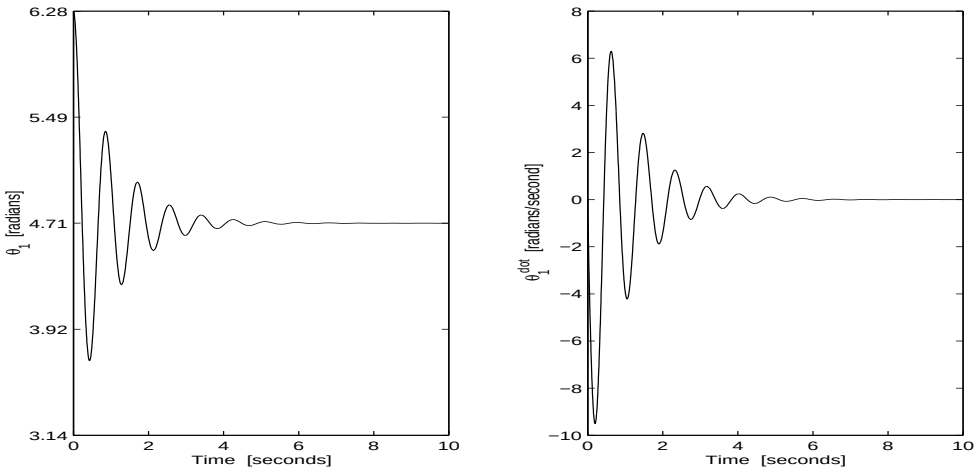

FIG. 2. *Time Variation of orientation θ_1 (Left) and of angular velocity $\dot{\theta_1}$ (Right) for Body 1.*

relative tolerance loosens or tightens the step-size control. Based on results shown on the left of Fig.2, the relative tolerance is multiplied by a value that oscillates between 4.0 and 6.0. Consequently, the actual upper bound of accuracy imposed on θ_1 fluctuates and reaches values up to $7 \cdot 10^{-k}$. Thus, the step-size controller is slightly conservative. For an explanation of this stiffness induced order reduction, the reader is referred to [7]. To remedy this, in [18], a scaling of the truncation error that enters in Eq.(13) the computation of the new step-size is recommended. In this context, the quantity $(\hat{y}_1 - y_1)$ is replaced by the scaled value $\delta = (I - h\gamma \partial f/\partial y)^{-1}(\hat{y}_1 - y_1)$. This step-size control strategy remains to be investigated.

TABLE 1
Position Error Analysis for the Double Pendulum Problem.

k	t^*	$\Delta^{(k)}$	RelErr[%]	$\overline{\Delta}^{(k)}$
-2	0.592127	5.223e-2	0.12126	3.234e-3
-3	0.599954	4.198e-3	0.00964	2.631e-4
-4	0.626135	4.916e-4	0.00108	2.946e-5
-5	1.065146	1.902e-5	0.00039	9.868e-6

TABLE 2
Velocity Error Analysis for the Double Pendulum Problem.

k	t^*	$\Delta^{(k)}$	RelErr[%]	$\Delta^{(k)}$
-2	0.795548	4.061e-2	1.84434	2.348e-2
-3	0.373114	3.792e-3	0.12340	2.181e-3
-4	0.217757	8.652e-4	0.00922	3.445e-4
-5	0.186183	2.343e-4	0.00246	9.357e-5

Error analysis is also performed at the velocity level. The time variation of angular velocity $\dot{\theta}_1$ is shown on the right of Fig.2. The angular velocity of body 1 fluctuates between -10 and 7 rad/s. As a result, values of $\Delta^{(k)}$ of up to the order 10^{k+1} are

considered very good. Error analysis results for $\dot{\theta}_1$ are presented in Table 2. The step-size controller performs well, slightly on the conservative side.

The error analysis results presented suggest that the step-size controller employed is reliable. The step-size control mechanism used indicates that using an embedded formula for local error estimation is suitable, since for the test problem considered accuracy requirements are met or exceeded by the numerical results. In order to avoid unjustified CPU penalties, the algorithm may be improved for extremely stiff mechanical systems by adopting the step-size controller proposed in [18].

5.2. Performance Comparison with Explicit Integrator. In order to compare the performance of the proposed implicit algorithm with a state of the art SSODE explicit integrator, a model of the US Army High Mobility Multipurpose Wheeled Vehicle (HMMWV) is considered for dynamic analysis. The HMMWV shown in Fig.3 is modeled using 14 bodies, as shown in Fig.4. In this figure, vertices represent bodies, while edges represent joints connecting the bodies of the system. Thus, vertex number 1 is the chassis, 2 and 5 are the right and left front upper control arms, 3 and 6 are the right and left front lower control arms, 9 and 12 are the right and left rear lower control arms, and 8 and 11 are the right and left rear upper control arms. Bodies 4, 7, 10, and 13 are the wheel spindles, and body 14 is the steering rack. Spherical joints are denoted by S, revolute joints by R, distance constraints by D, and translational joints by T. This set of joints imposes 79 constraint equations. One additional constraint equation is imposed on the steering system, such that the steering angle is zero; i.e., the vehicle drives straight. A total of 98 generalized coordinates are used to model the vehicle, which renders 18 degrees of freedom to the model.

Stiffness is induced in the model through means of four translational spring-damper actuators (TSDA). These TSDAs act between the front/rear and right/left upper control arms and the chassis. The stiffness coefficient of each TSDA is 2E07 N/m, while the damping coefficient is $2E06N \cdot s/m$. For the purpose of this numerical experiment, the tires of the vehicle are modeled as vertical TSDA elements with stiffness coefficient 296325 N/m and damping coefficient $3502N \cdot s/m$. Finally, the dominant eigenvalue of the corresponding SSODE of Eq.(9) has a real component of approximately $-2.6E5$, and a small imaginary part.

Dynamic analysis of the model is carried out for the vehicle driving straight at 10mph over a bump. The shape of the bump is a half-cylinder of diameter 0.1m. Figure 4 shows the time variation of the vehicle chassis height. The front wheels hit the bump at time 0.5 seconds, and the rear wheels hit the bump at time 1.2 seconds. The length of the simulation in this plot is 5 seconds. Toward the end of the simulation (after 4 seconds), due to over-damping the, chassis height stabilizes at approximately $z_1 = 0.71$m.

The test problem is first run with an explicit integrator based on the code DEABM of Shampine and Watts [18]. **Algorithm 2** outlines the explicit integration approach used for SSODE integration of the equations of motion for the HMMWV model.

FIG. 3. *HMMWV*

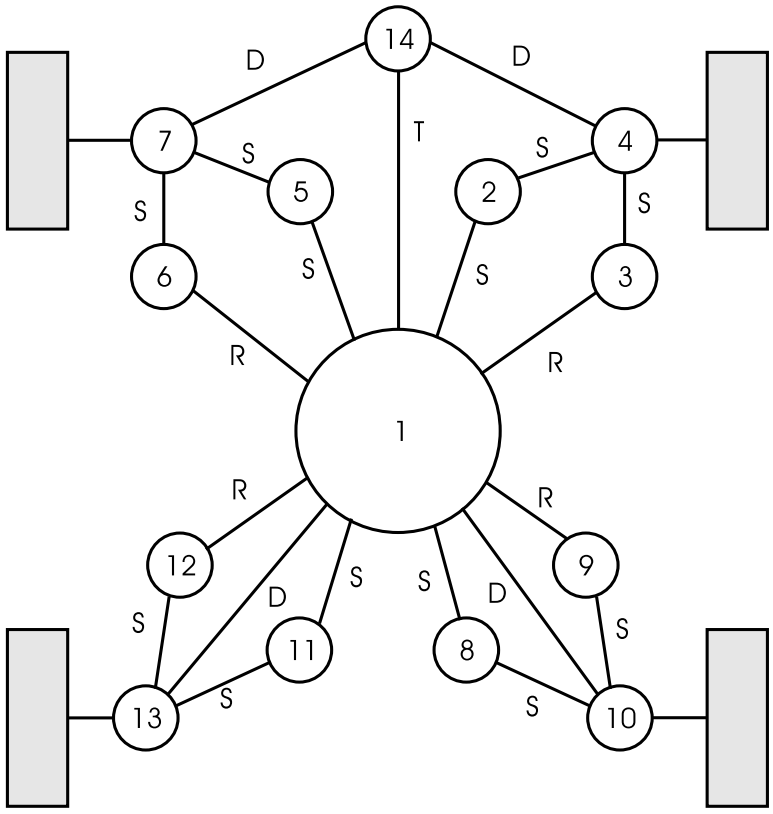

FIG. 4. *HMMWV model representation*

Algorithm 2
1. Initialize Simulation
2. Set Integration Tolerance
3. While (time < time-end) do
4. Get Acceleration
5. Apply Integration Step.
   Check Accuracy. Determine New Step-size
6. Recover Dependent Generalized Coordinates
7. Check Partition
8. End do

The first 3 steps are identical to the ones in **Algorithm 1**. Step 4 computes the acceleration $\ddot{\mathbf{q}}$, by solving the linear system of Eq.(3). A topology-based approach [17], that takes into account the sparsity of the coefficient matrix is used to solve for the generalized accelerations $\ddot{\mathbf{q}}$. The DDEABM integrator of Shampine and Watts is then used to integrate for independent velocities $\dot{\mathbf{v}}_n$, and independent positions $\mathbf{v}_n$. The integrator is also used to integrate for the dependent coordinates $\mathbf{u}_n$, with the sole purpose of providing a good starting point during Step 6 for the iterative solver. At each time step t_n it computes $\mathbf{u}_n$ by ensuring that the kinematic position constraint equations are satisfied; i.e., solving $\mathbf{\Phi}(\mathbf{v}_n, \mathbf{u}_n) = \mathbf{0}$. Likewise, dependent velocities $\dot{\mathbf{u}}_n$ are the solution of the linear system $\mathbf{\Phi_u}(\mathbf{u}_n, \mathbf{v}_n)\dot{\mathbf{u}}_n = -\mathbf{\Phi_v}(\mathbf{u}_n, \mathbf{v}_n)\dot{\mathbf{v}}_n$, which thus guarantees that the generalized velocities satisfy the kinematic velocity constraint equations. The dependent/independent partitioning of the generalized coordinates is checked during Step 7 and with this the algorithm concludes one integration step and proceeds to the next one.

Timing results reported are obtained on a SGI Onyx computer with an R10000 processor. Computer times required by **Algorithm 2** are listed in Table 3. Results for the Rosenbrock Nystrom algorithm are presented in Table 4.

TABLE 3
Explicit Integrator Timing Results for the HMMWV Problem.

Tol	10^{-2}	10^{-3}	10^{-4}	10^{-5}
1 sec	3618	3641	3667	3663
2 sec	7276	7348	7287	7276
3 sec	10865	11122	10949	10965
4 sec	14480	14771	14630	14592

TABLE 4
Implicit Integrator Timing Results for the HMMWV Problem.

Tol	10^{-2}	10^{-3}	10^{-4}	10^{-5}
1 sec	5.6	13.2	40.7	172
2 sec	12.6	32.6	95	405
3 sec	13	36.3	105	422
4 sec	13.3	37	106	428

Results in Table 3 are typical for the situation in which an explicit integrator is used for the numerical solution of a stiff IVP. For the stiff test problem considered, the performance limiting factor is stability of the explicit code. For any tolerance in

the range 1E-2 through 1E-5 and any given simulation length, CPU times are almost identical. The average explicit integration step-size turns out to be between 1E-5 and 1E-6, and it is not affected by accuracy requirements. The code is compelled to select very small step-sizes to assure stability of the integration process, and this is the criteria for step-size selection for a broad spectrum of tolerances. Only when extremely sever accuracy constraints are imposed on integration, does the step-size become limited by accuracy considerations. In this context, note that the results in Table 4 indicate that stability is of no concern for the proposed algorithm, and solution accuracy solely determines the duration of the simulation. As expected, more accurate results demand longer CPU times. In this situation, the integration step-size is automatically adjusted to keep integration error within the prescribed limits. Figure 5 shows the time variation for the integration step-size when the absolute and relative errors at position and velocity levels are set to 10^{-3}. The y-axis for the step-size is provided at the right of Fig.5, on a logarithmic scale. In the lower half of the same figure, relative to the left y-axis is provided the time variation of the chassis height. Note that when the vehicle hits the bump; i.e., when in Fig.5 the z coordinate of the chassis increases suddenly, the step-size is simultaneously decreased to preserve accuracy of the numerical solution. On the other hand, for the region in which the road becomes flat; i.e., toward the end of the simulation, the integrator is capable of taking larger integration steps, thus decreasing simulation time.

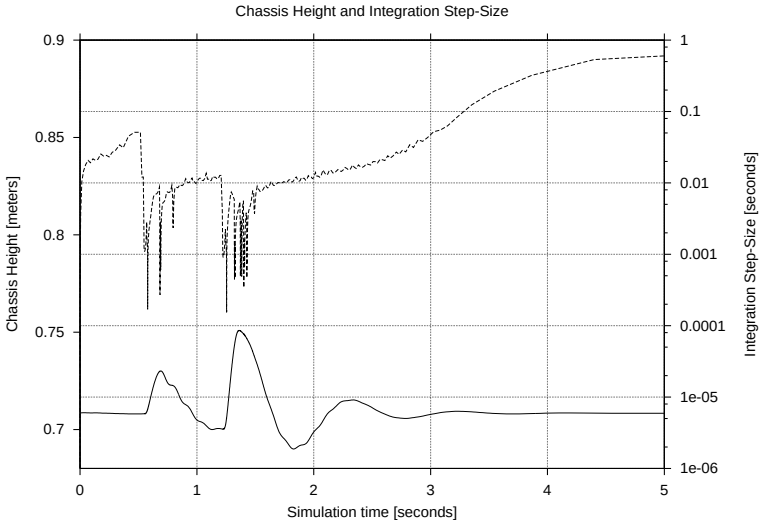

FIG. 5. *Chassis height and integration step-size*

6. Conclusions. A generalized coordinate partitioning based state-space implicit integration method is presented for dynamic analysis of multibody systems. The method is based on a Rosenbrock type formula with 4 stages that does not require the solution of a non-linear system for the stage values. The order 4 formula is L-stable and has an embedded order 3 formula for error control. For a 14 body 18 degree of freedom vehicle, the proposed algorithm is almost two orders of magnitude faster than an explicit integrator based method. There is room for improvement as far as the step-size controller is concerned, as it is seen to be conservative when very stiff models are run under stringent accuracy requirements. The most restrictive condition

imposed on the user by the Rosenbrock formula employed is the requirement of an exact integration Jacobian. A formalism is presented for computing the analytical integration Jacobian required. When providing an exact Jacobian is not possible, a lower order W-method is suggested as an alternative.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. LAPACK User's Guide, second edition. 1995.

[2] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial & Applied Mathematics, 1998.

[3] K. E. Atkinson. *An Introduction to Numerical Analysis*. John Wiley & Sons, New York, 1989.

[4] L. J. Corwin and R. H. Szczarba. *Multivariable Calculus*. Marcel Dekker, New York, 1982.

[5] R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *International Journal of Robotics Research*, 2(1):13–30, 1983.

[6] Javier Garcia de Jalon and Eduardo Bayo. *Kinematic and dynamic simulation of multibody systems : The real-time challenge*. Springer-Verlag New York, Mechanical Engineering Series, 1994.

[7] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems*. Springer-Verlag, Berlin Heidelberg New York, 1996.

[8] E. Hairer, Norsett S. P., and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Springer-Verlag, Berlin Heidelberg New York, 1993.

[9] E. J. Haug. *Computer-Aided Kinematics and Dynamics of Mechanical Systems*. Allyn and Bacon, Boston, London, Sydney, Toronto, 1989.

[10] Haug E. J., D. Negrut, and M. Iancu. A State-Space Based Implicit Integration Algorithm for Differential-Algebraic Equations of Multibody Dynamics. *Mech. Struct.&Mach.*, 25(3):311–334, 1997.

[11] Haug E. J., D. Negrut, and M. Iancu. *Implicit Integration of the Equations of Multibody Dynamics*, volume 161. NATO ASI Series: Springer-Verlag, J. Angeles and E. Zakhariev editors, 1997.

[12] D. Negrut. On the implicit integration of differential-algebraic equations of multibody dynamics. *Ph.D. Thesis*, The University of Iowa, 1998.

[13] L. R. Petzold. Differential-Algebraic Equations are not ODE's. *SIAM J. Sci., Stat. Comput.*, 3(3):367–384, 1982.

[14] F. A. Potra. Implementation of linear multistep methods for solving constrained equations of motion. *SIAM. Numer. Anal.*, 30(3):474–489, 1993.

[15] P. Rabier and W.C. Rheinboldt. *Nonholonomic Motion of Rigid Mechanical Systems from a DAE Viewpoint*. Society for Industrial & Applied Mathematics, 2000.

[16] R. Serban. Dynamic and Sensitivity Analysis of Multibody Systems. *Ph.D. Thesis*, The University of Iowa, 1998.

[17] R. Serban, D. Negrut, E. J. Haug, and F. A. Potra. A Topology Based Approach for Exploiting Sparsity in Multibody Dynamics in Cartesian Formulation. *Mech. Struct.&Mach.*, 25(3):379–396, 1997.

[18] L. F. Shampine and H. A. Watts. The art of writing a Runge-Kutta code.II. *Appl. Math. Comput.*, 5:93–121, 1979.

[19] R. A. Wehage and E. J. Haug. Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems. *J. Mech. Design*, 104:247–255, 1982.