

**ON AN APPROACH FOR THE LINEARIZATION OF THE DIFFERENTIAL  
 ALGEBRAIC EQUATIONS OF MULTIBODY DYNAMICS**

**Dan Negrut\***

Mathematics and Computer Science Division  
 Argonne National Laboratory  
 Argonne, Illinois 60439  
 Email: negrut@mcs.anl.gov

**Jose L. Ortiz**

MSC.Software  
 Ann Arbor, Michigan, 48105  
 Email: jose.ortiz@mscsoftware.com

**ABSTRACT**

*The paper presents an approach to linearize the set of index 3 nonlinear Differential Algebraic Equations (DAE) that govern the dynamics of constrained mechanical systems. The proposed method allows for treatment of heterogeneous systems with flexible bodies, friction, control elements (user-defined differential equations), non-holonomic constraints, etc. While analytically equivalent to a state-space formulation of the dynamics in a minimal set of states, the proposed method inflates the governing equations and then computes a set of sensitivities that provide the linearization of interest. The most attractive attributes associated with the method are (a) handling of large heterogeneous problems, (b) ability to linearize the system in terms of arbitrary user-defined coordinates, and (c) straightforward algorithm implementation in a general-purpose dynamics simulation code such as MSC.ADAMS*

**INTRODUCTION**

The state of a multibody system at the position level is represented in this paper by an array  $\mathbf{q} = [q_1, \dots, q_n]^T$  of generalized coordinates. The velocity of the system is most often described by the array of generalized velocities  $\dot{\mathbf{q}} = [\dot{q}_1, \dots, \dot{q}_n]^T$ . The set of generalized coordinates and velocities can be selected in a multitude of ways [1–4]. The generalized coordinates used here are Cartesian coordinates for position, and Euler angles for orientation of body centroidal reference frames. Thus, for each body  $i$ ,

its position is described by the vector  $\mathbf{r}_i = [x_i, y_i, z_i]^T$ , while its orientation is given by the array of local 3-1-3 Euler angles [5],  $\mathbf{e}_i = [\psi_i, \theta_i, \phi_i]^T$ . Consequently, for a mechanical system containing  $n_b$  bodies,  $\mathbf{q} = [\mathbf{r}_1^T, \mathbf{e}_1^T, \dots, \mathbf{r}_{n_b}^T, \mathbf{e}_{n_b}^T]^T \in \mathbf{R}^p$ ,  $p = 6n_b$ . Note that the set of position generalized coordinates is augmented with deformation modes when flexible bodies are present in the model. For notational simplicity, the assumption is that there are no flexible bodies in the model, although this does not pose a problem with the proposed algorithm as shown by numerical experiments discussed later in the paper.

In any constrained mechanical system, joints connecting bodies restrict their relative motion and impose constraints on the generalized coordinates. Kinematic constraints are then formulated as algebraic expressions involving generalized coordinates,

$$\mathbf{C}(\mathbf{q}, t) = [C_1(\mathbf{q}, t) \dots C_m(\mathbf{q}, t)]^T = \mathbf{0} \quad (1a)$$

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. Although the implementation of the proposed method handles non-holonomic constraints, to keep the presentation simpler, the case of holonomic constraints is assumed in what follows, and short remarks will be provided to indicate how the method would accommodate the non-holonomic case.

Differentiating Eq. (1a) with respect to time leads to the ve-

---

\*Address all correspondence to this author.

locity kinematic constraint equation

$$\mathbf{C}_q(\mathbf{q}, t) \dot{\mathbf{q}} + \mathbf{C}_t(\mathbf{q}, t) = \mathbf{0} \quad (1b)$$

where the over-dot denotes differentiation with respect to time and the subscript denotes partial differentiation,  $[\mathbf{C}_q]_{ij} = \left[ \frac{\partial C_i}{\partial q_j} \right]$ , for  $1 \leq i \leq m$ , and  $1 \leq j \leq n$ . Finally, the acceleration kinematic constraint equations are obtained by differentiating Eq. (1b) with respect to time,

$$\mathbf{C}_q(\mathbf{q}, t) \ddot{\mathbf{q}} + (\mathbf{C}_q(\mathbf{q}, t) \dot{\mathbf{q}})_{\mathbf{q}} \dot{\mathbf{q}} + 2\mathbf{C}_{qt}(\mathbf{q}, t) \dot{\mathbf{q}} + \mathbf{C}_{tt}(\mathbf{q}, t) = \mathbf{0} \quad (1c)$$

Equations (1a)–(1c) 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 [2],

$$\mathbf{M}(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{C}_q^T(\mathbf{q}) \lambda = \mathbf{Q}(\dot{\mathbf{q}}, \mathbf{q}, t) \quad (1d)$$

where  $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{p \times p}$  is the generalized mass, and  $\mathbf{Q}(\dot{\mathbf{q}}, \mathbf{q}, t) \in \mathbb{R}^p$  is the action (as opposed to the reaction  $\mathbf{C}_q^T(\mathbf{q}) \lambda$ ) force acting on the generalized coordinates  $\mathbf{q} \in \mathbb{R}^p$ . These equations are neither linear, nor ordinary differential. Additionally, the solution  $\mathbf{q}(t)$  must also satisfy the kinematic constraint equations in Eq. (1a). These constraint equations lead, in Eq. (1d), to the presence of the reaction force  $\mathbf{C}_q^T(\mathbf{q}) \lambda$ , where  $\lambda \in \mathbb{R}^m$  is the Lagrange multiplier associated with the kinematic constraints.

Equations (1a) and (1d) represent a set of index 3 DAE [6]. While the analytical solution of this problems automatically satisfies the velocity and acceleration constraint equations (Eqs. (1b) and (1c)), this ceases to be the case for the numerical solution of this highly nonlinear problem. There are many ways in which the index 3 problem can be analyzed, be it through a direct approach [7], a state-space reduction [8–10], an index reduction method [11], and a stabilization approach [12], to cite only a few. While these approaches are designed for finding the time evolution of a mechanical system (dynamic analysis), and their advantages and weaknesses are well understood, when it comes to a linear analysis of the index 3 DAE associated with a mechanical system, the picture is not that clear. The approach proposed in this paper draws on a well-established approach for dynamic analysis, namely the state-space reduction (SSR) method. At the cornerstone of SSR lies the observation that if a minimal set of generalized coordinates is chosen, locally the dynamics of the system is represented in terms of a set of second-order ordinary

differential equations (ODE), rather than the index 3 DAE introduced above. The state-space approach is theoretically sound, and the only challenge left is to make it efficient. In this regard it should be pointed out that while efficiency is always important, in linear analysis the weight placed on efficiency is less than in the case when a dynamic analysis is targeted. This is because a linear analysis is carried out once in a given configuration of the system, unlike a dynamic analysis that has to draw on an efficient algorithm, since this is applied at each integration step to advance the simulation time.

Obtaining the state-space ordinary differential equations (SSODE) representation directly through SSR is difficult, as it will be presented shortly. It should be first pointed out, however, that the state-space representation is not unique, and, for example the coordinate partitioning [8], the null space [9], or a QR-based approach [10] are several of the methodologies used in the past. In the coordinate partitioning based SSR, which is the approach employed in the present paper, the starting point is a partitioning of  $\mathbf{q}$  in independent and dependent coordinates  $\mathbf{v} \in \mathbb{R}^s$ ,  $s = n - m$ , and  $\bar{\mathbf{v}} \in \mathbb{R}^m$ . The partitioning is based on two mappings  $\nu : \{1, 2, \dots, s\} \rightarrow S_{indep}$  and  $\mu : \{1, 2, \dots, m\} \rightarrow S_{dep}$ , with  $S_{indep} \cup S_{dep} = \{1, 2, \dots, n\}$  and  $S_{indep} \cap S_{dep} = \emptyset$ .

$$\mathbf{v}[i] = \mathbf{q}[\nu(i)], \quad 1 \leq i \leq s, \quad \text{and} \quad \bar{\mathbf{v}}[j] = \mathbf{q}[\mu(j)], \quad 1 \leq j \leq m \quad (2)$$

The partitioning is such that the sub-Jacobian of the constraints with respect to  $\bar{\mathbf{v}}$  is nonsingular

$$\det(\mathbf{C}_{\bar{\mathbf{v}}}(\mathbf{q})) \neq 0 \quad (3)$$

Such a partition can be found starting with a set of consistent generalized coordinates  $\mathbf{q}_0$ ; i.e., which satisfy Eq. (1a). The constraint Jacobian matrix  $\mathbf{C}_q$  is evaluated and numerically factored, using the Gauss-Jordan algorithm with full pivoting [13],

$$\mathbf{C}_q(\mathbf{q}_0) \mapsto (\text{Gauss} - \text{Jordan}) \mapsto [\mathbf{C}_{\bar{\mathbf{v}}}(\mathbf{q}_0) | \mathbf{C}_v(\mathbf{q}_0)] \quad (4)$$

Based on this partitioning, Eqs.(1a)–(1d) can be rewritten in the associated partitioned form [2]

$$\mathbf{M}^{\nu\nu}(\bar{\mathbf{v}}, \mathbf{v}) \ddot{\mathbf{v}} + \mathbf{M}^{\bar{\nu}\bar{\nu}}(\bar{\mathbf{v}}, \mathbf{v}) \ddot{\bar{\mathbf{v}}} + \mathbf{C}_{\bar{\mathbf{v}}}^T(\bar{\mathbf{v}}, \mathbf{v}) \lambda = \mathbf{Q}^v(\mathbf{t}, \bar{\mathbf{v}}, \mathbf{v}, \dot{\bar{\mathbf{v}}}, \dot{\mathbf{v}}) \quad (5a)$$

$$\mathbf{M}^{\bar{\nu}\nu}(\bar{\mathbf{v}}, \mathbf{v}) \ddot{\mathbf{v}} + \mathbf{M}^{\nu\nu}(\bar{\mathbf{v}}, \mathbf{v}) \ddot{\bar{\mathbf{v}}} + \mathbf{C}_{\nu}^T(\bar{\mathbf{v}}, \mathbf{v}) \lambda = \mathbf{Q}^{\bar{\nu}}(\mathbf{t}, \bar{\mathbf{v}}, \mathbf{v}, \dot{\bar{\mathbf{v}}}, \dot{\mathbf{v}}) \quad (5b)$$

$$\mathbf{C}(\mathbf{t}, \bar{\mathbf{v}}, \mathbf{v}) = \mathbf{0} \quad (5c)$$

$$\mathbf{C}_{\bar{\mathbf{v}}}(t, \bar{\mathbf{v}}, \mathbf{v})\dot{\bar{\mathbf{v}}} + \mathbf{C}_{\mathbf{v}}(t, \bar{\mathbf{v}}, \mathbf{v})\dot{\mathbf{v}} + \mathbf{C}_t(t, \bar{\mathbf{v}}, \mathbf{v}) = \mathbf{0} \quad (5d)$$

$$\mathbf{C}_{\bar{\mathbf{v}}}(t, \bar{\mathbf{v}}, \mathbf{v})\ddot{\bar{\mathbf{v}}} + \mathbf{C}_{\mathbf{v}}(t, \bar{\mathbf{v}}, \mathbf{v})\ddot{\mathbf{v}} = \boldsymbol{\tau}(t, \bar{\mathbf{v}}, \mathbf{v}, \dot{\bar{\mathbf{v}}}, \dot{\mathbf{v}}) \quad (5e)$$

where  $\boldsymbol{\tau}$  is defined as the sum of all the terms in the left side of Eq. (1c) that do not depend on acceleration  $\ddot{\mathbf{q}}$ . The partitioning of Eqs. (5a)–(5e) is induced by the partitioning of the generalized coordinates in Eq. (2). For instance,  $\mathbf{M}^{\bar{\mathbf{v}}\bar{\mathbf{v}}}[i, j] = \mathbf{M}[\mathbf{v}(i), \mu(j)]$ , for  $1 \leq i \leq s, 1 \leq j \leq n-s$ , while  $\mathbf{Q}^{\bar{\mathbf{v}}}[j] = \mathbf{Q}[\mu(j)]$ , for  $1 \leq j \leq n-s$ . Likewise, a partial with respect to the dependent set of coordinates  $\bar{\mathbf{v}}$  is obtained by gathering the columns  $\mu(1)$  through  $\mu(m)$  of the derivative with respect to the generalized coordinates  $\mathbf{q}$ . The remaining  $s$  columns provide the partial with respect to the independent coordinates  $\mathbf{v}$ .

It is showed in [8] that based on the non-singularity condition of Eq. (3), a second-order ODE can be determined for the time evolution of  $\mathbf{v}$ . The expression of this second-order SODE is

$$\widehat{\mathbf{M}}(\mathbf{v})\ddot{\mathbf{v}} = \widehat{\mathbf{Q}}(t, \mathbf{v}, \dot{\mathbf{v}}) \quad (6a)$$

where

$$\widehat{\mathbf{M}} = \mathbf{M}^{\mathbf{v}\mathbf{v}} - \mathbf{M}^{\bar{\mathbf{v}}\bar{\mathbf{v}}}\mathbf{C}_{\bar{\mathbf{v}}}^{-1}\mathbf{C}_{\mathbf{v}} - \mathbf{C}_{\bar{\mathbf{v}}}^T\mathbf{C}_{\bar{\mathbf{v}}}^{-T}[\mathbf{M}^{\bar{\mathbf{v}}\mathbf{v}} - \mathbf{M}^{\bar{\mathbf{v}}\bar{\mathbf{v}}}\mathbf{C}_{\bar{\mathbf{v}}}^{-1}\mathbf{C}_{\mathbf{v}}] \quad (6b)$$

$$\widehat{\mathbf{Q}} = \mathbf{Q}^{\mathbf{v}} - \mathbf{M}^{\bar{\mathbf{v}}\bar{\mathbf{v}}}\mathbf{C}_{\bar{\mathbf{v}}}^{-1}\boldsymbol{\tau} - \mathbf{C}_{\bar{\mathbf{v}}}^T\mathbf{C}_{\bar{\mathbf{v}}}^{-T}[\mathbf{Q}^{\bar{\mathbf{v}}} - \mathbf{M}^{\bar{\mathbf{v}}\bar{\mathbf{v}}}\mathbf{C}_{\bar{\mathbf{v}}}^{-1}\boldsymbol{\tau}] \quad (6c)$$

The SODE (6a) is well defined (since the matrix  $\widehat{\mathbf{M}}(\mathbf{v})$  of Eq. (6b) is positive definite), and it is this set of second-order differential equations that is linearized to understand the local behavior of the mechanism.

Two remarks are in order here. First, unless the mechanical system is in equilibrium, due to the nonlinear nature of the constraints in Eq. (1a), the eigenvalues are going to depend on the choice of coordinates  $\mathbf{v}$ .

Second, due to the extremely contrived form of  $\widehat{\mathbf{M}}(\mathbf{v})$  and  $\widehat{\mathbf{Q}}(t, \mathbf{v}, \dot{\mathbf{v}})$ , linearization in this form is difficult, and a *different but equivalent* approach will be pursued. Linear analysis of Eq. (6a) is difficult due to the level of matrix manipulation required by this approach. Eventually, what makes this direct approach impractical for a general-purpose mechanical simulation package such as MSC.ADAMS [14], is that the index 3 DAE system is usually embellished with additional equations that further complicate the problem. Thus, in addition to the equations of motion and kinematic constraint equations, there are several classes of equations that need to be considered in a general purpose mechanical simulation package:

1. *Ordinary differential equations* that in the most general case are provided in implicit form

$$\mathbf{d}(\dot{\mathbf{X}}, \mathbf{X}, \mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\lambda}, \mathbf{V}, \mathbf{F}, t) = \mathbf{0}_{n_d} \quad (7a)$$

This type of differential equations are encountered for example when controls are active in the system, such as cars with anti-lock braking system (ABS), active suspension control, etc. The state of the controller is  $\mathbf{X}$ , its time derivative is  $\dot{\mathbf{X}}$ , and the assumption is that in its implicit-form Eq. (7a) properly and uniquely defines  $\dot{\mathbf{X}}$  as a function of the state of the system through the user-specified function  $\mathbf{d}$ . If  $n_d$  represents the number of differential states, then  $\mathbf{X}, \mathbf{d} \in \mathbb{R}^{n_d}$ .

2. *User-defined variables*, which can technically be regarded as aliases or definition equations. A set of  $n_v$  user defined variables  $\mathbf{V} \in \mathbb{R}^{n_v}$  is typically specified through an equation of the form

$$\mathbf{V} - \mathbf{v}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{X}, \dot{\mathbf{X}}, \boldsymbol{\lambda}, \mathbf{V}, \mathbf{F}, t) = \mathbf{0}_{n_v} \quad (7b)$$

which during the solution sequence are solved (or rather evaluated) simultaneously with the equations of motion and the kinematic constraint equations. Here,  $\mathbf{v} \in \mathbb{R}^{n_v}$  is a user-defined function that depends on other system states, as indicated in Eq. (7b).

3. *External force definition*,  $\mathbf{F}$ , which allows the user to more conveniently define the set of  $n_f$  applied forces  $\mathbf{F} \in \mathbb{R}^{n_f}$  that act on the system. This is the mechanism through which a complex tire model can be hooked up with a vehicle model, or the avenue through which the user can define his/her own bushing elements, custom nonlinear dampers, friction, etc.

$$\mathbf{F} - \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \mathbf{X}, \dot{\mathbf{X}}, \boldsymbol{\lambda}, \mathbf{V}, \mathbf{F}, t) = \mathbf{0}_{n_f} \quad (7c)$$

The method proposed in this paper and implemented in the simulation package MSC.ADAMS [14], handles simultaneously the set of equations (1) and (7). In doing so, the method augments the governing equations and then relies on the sparse matrix factorization and solution of multiple right side linear systems to compute the quantities of interest. The reasons for embracing this approach are as follows: (a) no need for low-level matrix manipulations that lead to loss of topological structure and sparsity, (b) supports handling of a large variety of legacy modeling elements in MSC.ADAMS, (c) supports the computation of various sensitivities in a unitary framework, (d) straightforward implementation of the method, and (e) formulation favors a multi-threaded approach in calculation of sensitivities.

## LINEARIZING THE EQUATIONS OF MOTION Sensitivities as induced by an algebraic equation

The idea of sensitivity computation is first introduced by means of a simple example. Consider an equation of the form  $\psi(x, y) = 0$ , with  $\psi(x, y) = x^2 - 4y$ . Obviously  $\psi(2, 1) = 0$ , and the question is how does the defining equation  $\psi(x, y) = 0$  marry the change in  $x$  to a small perturbation in  $y$ . For the considered equation, this is equivalent to asking for the sensitivity of  $x$  with respect of  $y$  at a consistent operating point  $(2, 1)$ . Accepting that  $x = x(y)$  and introducing a new function  $\phi(y) = \psi(x(y), y)$ , a partial derivative of  $\phi$  with respect to the "independent" variable  $y$  is taken to produce

$$\frac{\partial \phi}{\partial y} = \frac{\partial \psi}{\partial x} \frac{\partial x}{\partial y} + \frac{\partial \psi}{\partial y} \quad (8)$$

Since  $\frac{\partial \phi}{\partial y} = 0$ , then as long as

$$\frac{\partial \psi}{\partial x} \neq 0, \quad (9)$$

$x_y = -[\Psi_x]^{-1} \Psi_y$ . In other words, the sensitivity of  $x$  with respect to  $y$  at a consistent operation point  $(x^*, y^*)$ ; that is, for which  $\psi(x^*, y^*) = 0$ , can be computed as soon as  $\Psi_x(x^*, y^*) \neq 0$ .

A different sensitivity computation approach is possible, and it is the approach that the proposed linearization method relies on. Thus, the problem of computing the sensitivity of  $x$  with respect to  $y$  in Eq. (8) is equivalent to that of computing the sensitivity of  $x$  with respect to a "virtual" state  $\bar{y}$  that the problem is augmented with (please recall that an over-bar was previously used in to denote dependent coordinates; hereafter the over-bar indicates a virtual state used to augment the original set of equations):

$$\begin{cases} \Psi(x, y) = 0 \\ y - \bar{y} = 0 \end{cases} \quad (10)$$

The goal is in this case to compute  $\frac{\partial x}{\partial \bar{y}}$  and  $\frac{\partial y}{\partial \bar{y}}$ , and immediately  $\frac{\partial y}{\partial \bar{y}} = 1$ . It is easy to see that the quantity  $\frac{\partial x}{\partial \bar{y}}$  is identical to the sensitivity of  $x$  with respect to  $y$  as computed with the direct approach, and that in the end, the first-order sensitivities are the solution of the linear system

$$\begin{bmatrix} \frac{\partial \psi}{\partial x} & \frac{\partial \psi}{\partial y} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial \bar{y}} \\ \frac{\partial y}{\partial \bar{y}} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (11)$$

The determinant of the coefficient matrix in Eq. (11) is  $\frac{\partial \psi}{\partial x}$ , and the same sufficient condition of Eq. (9) renders the augmented approach well defined.

## The proposed augmented approach to linearization

The method proposed was implemented in a general-purpose simulation software, and as such the formulation of the equations of motion was brought to a slightly more general form. First, the level one set of generalized coordinates  $\mathbf{u}$ , the generalized velocities, is not assumed to be the straight time derivative of the position coordinates  $\mathbf{q}$ . Rather,

$$\mathbf{u} = \mathbf{T}(\mathbf{q})\dot{\mathbf{q}} \quad (12)$$

The old case is obtained if  $\mathbf{T}(\mathbf{q}) = \mathbf{I}$ , where  $\mathbf{I}$  is the identity matrix; when Euler angles are used for level zero generalized coordinates and, for instance, the set of angular velocities  $\omega$  is used for level one generalized coordinates leads to a non-constant transformation matrix  $\mathbf{T}(\mathbf{q})$ .

Second, since the approach allows for treatment of non-holonomic constraints, the projection of the Lagrange multipliers onto the equations of motion in the most general case is carried out by means of a projection operator,  $\mathbf{P} = \mathbf{P}(\mathbf{q}, \mathbf{u})$ . However, even if there are non-holonomic constraints present, they are typically linear in velocities, in which case  $\mathbf{P} = \mathbf{P}(\mathbf{q})$ . In the case of exclusively holonomic constraints,  $\mathbf{P} = \mathbf{P}(\mathbf{q}) = \mathbf{C}_q^T(\mathbf{q})$ . With this, the set of equations governing the dynamics of the mechanical system is:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{P}(\mathbf{q}, \mathbf{u})\lambda - \mathbf{Q}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (13a)$$

$$\mathbf{T}(\mathbf{q})\dot{\mathbf{q}} - \mathbf{u} = \mathbf{0} \quad (13b)$$

$$\ddot{\mathbf{C}}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (13c)$$

$$\dot{\mathbf{C}}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (13d)$$

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{0} \quad (13e)$$

$$\mathbf{B}_1 \mathbf{u} - \bar{\mathbf{u}} = \mathbf{0} \quad (13f)$$

$$\mathbf{B}_0 \mathbf{q} - \bar{\mathbf{q}} = \mathbf{0} \quad (13g)$$

*Remarks:* (1). Equations (13c) and (13d) indicate that the set of constraints considered are holonomic. This assumption is not necessary and made in order to keep the presentation simpler. If non-holonomic constraints were present, Eq. (13d) would have been rewritten as  $\tilde{\mathbf{C}}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0}$ , where  $\tilde{\mathbf{C}}$  would have included the expressions of  $\dot{\mathbf{C}}$  plus any strictly non-holonomic constraints. This would carry over into Eq. (13c), which would become  $\dot{\tilde{\mathbf{C}}}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0}$

(2). Another simplification was made in that the set of equations (7) was not added above to keep the presentation simple. These

three sets of equations do not bring anything qualitatively new into the picture.

(3). Note the introduction of the virtual variables  $\bar{\mathbf{q}}$  and  $\bar{\mathbf{u}}$  in Eqs. (13f) and (13g), respectively. This is the augmentation process already discussed for the simple algebraic equation, and in this paper it is based on partitioning of the generalized coordinates  $\mathbf{q}$  and  $\mathbf{u}$  into dependent and independent, at both position and velocity levels. The process is carried out as discussed previously in conjunction with Eqs. (2) through (4). Note that the factorization in Eq. (4) might have to be applied twice, both in the presence of non-holonomic constraints, or when  $\mathbf{T}(\mathbf{q}) \neq \mathbf{I}$ . Based on Eq. (2), the important outcome of the partitioning is the set of two boolean matrices (containing only ones and zeros)  $\mathbf{B}_0 \in \mathbb{R}^{s_0 \times n}$  and  $\mathbf{B}_1 \in \mathbb{R}^{s_1 \times n}$ . Typically  $s_0 = s_1$  (which is not the case in the presence of non-holonomic constraints though) and then the number of degrees of freedom is defined as  $s = s_0$ .

The nonlinear system above can be written as

$$F(\dot{\mathbf{u}}, \dot{\mathbf{q}}, \lambda, \mathbf{u}, \mathbf{q}, \bar{\mathbf{u}}, \bar{\mathbf{q}}, t) = \mathbf{0}$$

The key fact is the inclusion of the first and second derivatives of the constraint equations in order to make the system solvable in terms of  $\bar{\mathbf{u}}$  and  $\bar{\mathbf{q}}$ . Thus, given  $\bar{\mathbf{q}}$ , Eq. (13g) provides the set of independent positions, which in turn are used in Eq. (13e) to retrieve the set of dependent positions (this is possible because locally a condition of type Eq. (3) holds as a requisite of the partitioning process). Similarly, Eq. (13f) provides the set of independent velocities, which in turn are used in Eq. (13d) to retrieve the set of dependent velocities. Once  $\mathbf{u}$  is available,  $\dot{\mathbf{q}}$  is solved based on Eq. (13b), and finally  $\dot{\mathbf{u}}$  and  $\lambda$  are solved for simultaneously, as the solution of the Eq. (13a) and (13c). Since both  $\bar{\mathbf{u}}$  and  $\bar{\mathbf{q}}$  are arbitrary, the following must hold:

$$\frac{\partial F}{\partial \bar{\mathbf{u}}} = \mathbf{0} \quad \frac{\partial F}{\partial \bar{\mathbf{q}}} = \mathbf{0} \quad (14)$$

Using the notation

$$\Upsilon_{\mathbf{u}} = [\mathbf{P}(\mathbf{q}, \mathbf{u})\lambda - \mathbf{Q}(\mathbf{u}, \mathbf{q}, t)]_{\mathbf{u}}$$

$$\Upsilon_{\mathbf{q}} = [\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{P}(\mathbf{q}, \mathbf{u})\lambda - \mathbf{Q}(\mathbf{u}, \mathbf{q}, t)]_{\mathbf{q}}$$

the computation of the sensitivities proceeds by using the chain rule on the conditions of Eq. (14). This yields a multiple right

side system of linear equations that will also provide, among several sensitivities, the quantities of interest, namely  $\dot{\bar{\mathbf{u}}}$ ,  $\dot{\bar{\mathbf{q}}}$ ,  $\dot{\bar{\mathbf{u}}}$ , and  $\dot{\bar{\mathbf{q}}}$ .

$$\begin{bmatrix} \mathbf{M}(\mathbf{q}) & \mathbf{0} & \mathbf{P}(\mathbf{q}, \mathbf{u}) & \Upsilon_{\mathbf{u}} & \Upsilon_{\mathbf{q}} \\ \mathbf{0} & \mathbf{T}(\mathbf{q}) & \mathbf{0} & -\mathbf{I} & [\mathbf{T}(\mathbf{q})\dot{\mathbf{q}}]_{\mathbf{q}} \\ \ddot{\mathbf{C}}_{\bar{\mathbf{u}}} & \mathbf{0} & \mathbf{0} & \ddot{\mathbf{C}}_{\bar{\mathbf{u}}} & \ddot{\mathbf{C}}_{\bar{\mathbf{q}}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dot{\mathbf{C}}_{\bar{\mathbf{u}}} & \dot{\mathbf{C}}_{\bar{\mathbf{q}}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C}_{\bar{\mathbf{q}}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_0 \end{bmatrix} \begin{bmatrix} \dot{\bar{\mathbf{u}}} & \dot{\bar{\mathbf{q}}} \\ \ddot{\bar{\mathbf{u}}} & \ddot{\bar{\mathbf{q}}} \\ \lambda_{\bar{\mathbf{u}}} & \lambda_{\bar{\mathbf{q}}} \\ \mathbf{u}_{\bar{\mathbf{u}}} & \mathbf{u}_{\bar{\mathbf{q}}} \\ \mathbf{q}_{\bar{\mathbf{u}}} & \mathbf{q}_{\bar{\mathbf{q}}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (15)$$

Notice the balance between the number of unknowns and equations. For the more common case when there are no non-holonomic constraints and  $\mathbf{T}(\mathbf{q}) = \mathbf{I}$ , assuming that each body contributes six entries in  $\mathbf{q}$ ,  $\mathbf{u}$ ,  $\dot{\mathbf{q}}$  and  $\dot{\mathbf{u}}$ , the total number of unknowns is  $24n_b + m$ . The total number of equations is obtained as follows:  $6n_b$  equations of motion (Eq. (13a)),  $6n_b$  kinematic relations (Eq. (13b)),  $3m$  constraint equations (Eqs. (13c) to (13e)),  $2s$  equations given by the coordinate partition (Eqs. (13f) and (13g)). Summing up, the number of equations matches the number of unknowns.

Compared with the alternative based on the linearization of Eq. (6a) in conjunction with Eqs. (6b) and (6c), the proposed method (a) is significantly simpler to implement, (b) it is more general, in that it can handle non-holonomic constraints, as well as controls, ordinary differential equations, user-defined variables, etc. as defined in Eq. (7), (c) there is no need to partition the mass or constraint Jacobian, or for that matter the generalized force vector, (d) there is no requirement to explicitly invert any matrix (as  $\mathbf{C}_{\bar{\mathbf{v}}}^{-1}$  (see Eqs. (6b) and (6c)) and subsequently compute its linearization. These attractive features, particularly (a) and (b) above, by far outweigh an increase in the dimension of the problem to be solved. The dimension of the problem has not led in the numerical experiments carried out to any increase in CPU time, a fact credited to the inherent sparsity associated with the augmented approach. Finally, although currently not implemented, note that a multi-threaded approach to solving the multiple right side system is very direct.

## LINEARIZATION IN USER-DEFINED STATES

The model shown in Fig. 1 is a representation of a double pendulum. This simple example is used to make the easily provable point that if a *nonlinear* transformation locally establishes a one-to-one mapping between two sets of states, then in a *non-equilibrium* configuration of the system, the eigenvalues corresponding to the two sets of states are different.

In the model, the first link has a prescribed motion  $\theta = \omega t$  leaving the system with only one degree of freedom. Selecting

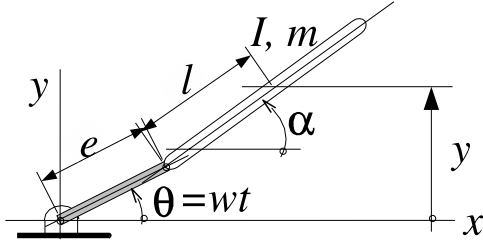


Figure 1. SINGLE-DEGREE OF FREEDOM HELICOPTER BLADE.

$\alpha$  as the coordinate to express the equations of motion yields the following second-order equation:

$$(I + ml^2)\ddot{\alpha} + m e l w^2 \sin(\alpha - wt) = 0 \quad (16)$$

Defining  $\alpha_1 = \dot{\alpha}$ , and  $\alpha_2 = \alpha$  the equations of motion in first-order ODE form are

$$\begin{bmatrix} \dot{\alpha}_1 \\ \dot{\alpha}_2 \end{bmatrix} = \begin{bmatrix} f(\alpha_1, \alpha_2, t) \\ \alpha_1 \end{bmatrix} \quad (17)$$

and the linearized system is

$$\begin{bmatrix} \delta\dot{\alpha}_1 \\ \delta\dot{\alpha}_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial \alpha_1} & \frac{\partial f}{\partial \alpha_2} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \delta\alpha_1 \\ \delta\alpha_2 \end{bmatrix} \quad (18)$$

Equation (18) can be rewritten in matrix form as

$$\delta\dot{\alpha} = \mathbf{A}\delta\alpha \quad (19)$$

where the state vector  $\alpha = [\alpha_1, \alpha_2]^T$ . Selecting the operating point defined by  $t = 0$ ,  $\alpha_1 = \dot{\alpha} = w$  and  $\alpha_2 = \alpha = 0$ ,

$$\mathbf{A} = \begin{bmatrix} 0 & \frac{-m e l w^2}{I + ml^2} \\ 1 & 0 \end{bmatrix}, \quad (20)$$

with eigenvalues:

$$\lambda_{1,2} = \pm i \sqrt{\frac{m e l}{I + ml^2}} w. \quad (21)$$

Assuming that for this problem  $m = 3$ ,  $I = 10$ ,  $e = 2$ ,  $l = 10$  and  $w = 10$ , this leads to eigenvalues  $\lambda_{1,2} = \pm 4.3994 i$ .

In a second formulation, the equations of motion are expressed in terms of the coordinate  $y$  (see Fig. 1). This leads to a different matrix  $\mathbf{A}^*$

$$\mathbf{A}^* = \begin{bmatrix} 0 & \frac{-m e l w^2}{I + ml^2} - w^2 \\ 1 & 0 \end{bmatrix} \quad (22)$$

Using the same operating point, namely,  $t = 0$ ,  $y_1 = \dot{y} = (e + l)w$  and  $y_2 = y = 0$ , the eigenvalues are

$$\lambda_{1,2}^* = \pm i \sqrt{\frac{m e l}{I + ml^2} + 1} w \quad (23)$$

The same numerical values of the physical properties lead to the eigenvalues  $\lambda_{1,2}^* = \pm 10.92496i$ .

Therefore, allowing the user to specify a certain set of generalized coordinates for the linearization to be carried out becomes all important. In what follows the augmented approach introduced to handle the stand-alone equations of motion is extended to handle linearization of the index 3 DAE of multibody dynamics in terms of user defined arbitrary states.

### Sensitivities in the presence of user defined states

Suppose that a general purpose software package formulates a problem in the set of internal states  $x_1$  and  $x_2$ , and the user does not have any control over this selection. For example, in MSC.ADAMS [14], the internal states are the Cartesian positions and 3-1-3 local Euler angles. In the case of a simple pendulum, the user might be interested in linearizing the dynamics in a non-equilibrium configuration using the pendulum equilibrium deviation angle  $\theta$ , which may be a state that the software package does not use for defining the equations of motion. In an augmented framework, this section introduces by means of a simple example the linearization process as carried out in the user (rather than the internal) state space. To this end, consider the set of differential equations

$$\dot{x}_1 = f_1(x_1, x_2) \quad (24)$$

$$\dot{x}_2 = f_2(x_1, x_2) \quad (25)$$

where  $f_1$  and  $f_2$  are nonlinear, and the linearized equations are found to be

$$\begin{bmatrix} \delta \dot{x}_1 \\ \delta \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \end{bmatrix} \quad (26)$$

Suppose that the user indicates by means of two functions  $g_1$  and  $g_2$  a new set of states for the purpose of reformulating the equations of motion (assumed here to be given by Eqs. (24) and (25)) in these new states  $s_1$  and  $s_2$ :

$$s_1 = g_1(x_1, x_2) \quad (27a)$$

$$s_2 = g_2(x_1, x_2) \quad (27b)$$

Locally, the nonlinear mapping above is assumed one-to-one, which is the case as soon as the Jacobian  $[\mathbf{G}]_{ij} = [\frac{\partial g_i}{\partial x_j}]$ ,  $i, j \in \{1, 2\}$  of the transformation is non-singular. For the linearization problem, the interest is in finding  $\frac{\partial \dot{g}_i}{\partial s_j}$ , for  $i, j \in \{1, 2\}$ , or equivalently  $\frac{\partial \dot{s}_i}{\partial s_j}$ , for  $i, j \in \{1, 2\}$ . It is assumed in what follows that the quantities  $\frac{\partial g_i}{\partial x_j}$ , and  $\frac{\partial \dot{g}_i}{\partial x_j}$  for  $i, j \in \{1, 2\}$  are readily available because it is these derivatives with respect to the internal states that can be computed internally in a general-purpose simulation package.

The problem is augmented by the introduction of two virtual states  $\bar{s}_1$  and  $\bar{s}_2$ , to obtain an augmented system of equations:

$$\dot{x}_1 - f_1(x_1, x_2) = 0 \quad (28a)$$

$$\dot{x}_2 - f_2(x_1, x_2) = 0 \quad (28b)$$

$$s_1 - g_1(x_1, x_2) = 0 \quad (28c)$$

$$s_2 - g_2(x_1, x_2) = 0 \quad (28d)$$

$$\dot{s}_1 - \dot{g}_1(x_1, x_2, \dot{x}_1, \dot{x}_2) = 0 \quad (28e)$$

$$\dot{s}_2 - \dot{g}_2(x_1, x_2, \dot{x}_1, \dot{x}_2) = 0 \quad (28f)$$

$$s_1 - \bar{s}_1 = 0 \quad (28g)$$

$$s_2 - \bar{s}_2 = 0 \quad (28h)$$

Using the chain rule to take the partials with respect to  $\bar{s}_1$  and  $\bar{s}_2$  and rearranging leads to the following system of equations with multiple right sides:

$$\begin{bmatrix} -f_{11} & -f_{12} & 1 & 0 & 0 & 0 & 0 & 0 \\ -f_{21} & -f_{22} & 0 & 1 & 0 & 0 & 0 & 0 \\ -g_{11} & -g_{12} & 0 & 0 & 1 & 0 & 0 & 0 \\ -g_{21} & -g_{22} & 0 & 0 & 0 & 1 & 0 & 0 \\ -\dot{g}_{11} & -\dot{g}_{12} & -\dot{g}_{11} & -\dot{g}_{12} & 0 & 0 & 1 & 0 \\ -\dot{g}_{21} & -\dot{g}_{22} & -\dot{g}_{21} & -\dot{g}_{22} & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{1,\bar{s}_1} & x_{1,\bar{s}_2} \\ x_{2,\bar{s}_1} & x_{2,\bar{s}_2} \\ \dot{x}_{1,\bar{s}_1} & \dot{x}_{1,\bar{s}_2} \\ \dot{x}_{2,\bar{s}_1} & \dot{x}_{2,\bar{s}_2} \\ s_{1,\bar{s}_1} & s_{1,\bar{s}_2} \\ s_{2,\bar{s}_1} & s_{2,\bar{s}_2} \\ \dot{s}_{1,\bar{s}_1} & \dot{s}_{1,\bar{s}_2} \\ \dot{s}_{2,\bar{s}_1} & \dot{s}_{2,\bar{s}_2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (29)$$

where the notation  $\frac{\partial f_i}{\partial x_j} = f_{ij}$ , and  $\frac{\partial \dot{g}_i}{\partial x_j} = \dot{g}_{ij}$  was used. Note that solving this system produces the quantities of interest, namely  $\frac{\partial \dot{s}_i}{\partial s_j}$  for  $i, j \in \{1, 2\}$ . The key observation that carries over to the general case of the differential-algebraic multibody dynamics equations of motion is that without low-level manipulations, a new set of variables can be introduced and using an augmented approach compute the linearization in terms of these user-defined states as the solution of a multiple right side linear system of equations.

### The selection of linearization state $\mathbf{S}$

Assume that a set of  $\hat{s}$  user-defined states  $\hat{\mathbf{S}}$

$$\hat{\mathbf{S}} = \hat{\mathbf{s}}(\mathbf{q}, t) \quad (30)$$

are defined and indicated to be the user's choice of coordinates for carrying out the linearization process. For a large system containing for example flexible bodies, it is unreasonable to expect the user to provide a full set of states equal to the number of model degrees of freedom, which it is not known a priori in most of the real-life models. Likewise, it is possible to over prescribe a set of user-states.

Considering the set of  $m + \hat{s}$  equations in the  $n + \hat{s}$  variables  $\mathbf{q}$  and  $\hat{\mathbf{S}}$ ,

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{0} \quad (31)$$

$$\hat{\mathbf{S}} - \hat{\mathbf{s}}(\mathbf{q}, t) = \mathbf{0} \quad (32)$$

the state selection algorithm proceeds by building the Jacobian matrix

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{C}(\mathbf{q}, t)}{\partial \mathbf{q}} & \mathbf{0} \\ -\frac{\partial \hat{\mathbf{s}}(\mathbf{q}, t)}{\partial \mathbf{q}} & \mathbf{I} \end{bmatrix} \quad (33)$$

The selection of the actual linearization states  $\mathbf{S}$  continues by performing an **LU** factorization with full pivoting. At the

end of it, a non-singular sub-matrix will span the columns corresponding to partials with respect to the *dependent coordinates*. The value of  $s$  is subsequently defined as the count of remaining columns in the factored Jacobian, and it is these right-most columns that actually define the set of independent coordinates  $\mathbf{S}$ . Note that each of the  $s$  right-most states may belong to either  $\mathbf{q}$  or  $\hat{\mathbf{S}}$ . On one hand, if too many user-states have been proposed, the ones that don't find their way into the set of independent states are dropped out of the problem. On the other hand, if too few states are selected, some of the internal states  $q_i \in \mathbf{q}$  are elevated to the rank of independent states. Note that in retrospect this is actually what happens for the case when no user-states are specified, and the algorithm ends up selecting  $\mathbf{S}$  amongst the internal states  $\mathbf{q}$ .

Regardless of the number of user-defined coordinates in  $\hat{\mathbf{S}}$ , the algorithm favors the selection of variables  $\mathbf{S}$  out of this set. This is accomplished by means of a weighting strategy in which each row is assigned a weight that biases the factorization algorithm towards preferring a user-defined coordinate over an internal coordinate out of  $\mathbf{q}$ . Moreover, to bias the column pivoting, the identity matrix shown in Eq. (33) is actually replaced by a diagonal matrix of small values.

### The general case of the multibody equations of motion

Along with the selected state  $\mathbf{S} \in \mathbb{R}^s$  goes a set of equations  $\mathbf{S} = \mathbf{s}(\mathbf{q}, t)$  defining these states, and a set of velocity states,  $\mathbf{W} = \dot{\mathbf{S}}$ . The augmented set of equations assumes the form

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{u}} + \mathbf{P}(\mathbf{q}, \mathbf{u})\lambda - \mathbf{Q}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (34a)$$

$$\mathbf{T}(\mathbf{q})\dot{\mathbf{q}} - \mathbf{u} = \mathbf{0} \quad (34b)$$

$$\ddot{\mathbf{C}}(\dot{\mathbf{u}}, \mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (34c)$$

$$\dot{\mathbf{C}}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (34d)$$

$$\mathbf{C}(\mathbf{q}, t) = \mathbf{0} \quad (34e)$$

$$\mathbf{S} - \mathbf{s}(\mathbf{q}, t) = \mathbf{0} \quad (34f)$$

$$\dot{\mathbf{S}} - \mathbf{W} = \mathbf{0} \quad (34g)$$

$$\mathbf{W} - \dot{\mathbf{s}}(\mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (34h)$$

$$\dot{\mathbf{W}} - \ddot{\mathbf{s}}(\dot{\mathbf{u}}, \mathbf{u}, \mathbf{q}, t) = \mathbf{0} \quad (34i)$$

$$\mathbf{W} - \bar{\mathbf{W}} = \mathbf{0} \quad (34j)$$

$$\mathbf{S} - \bar{\mathbf{S}} = \mathbf{0} \quad (34k)$$

In the augmented approach, the dependent states are  $\dot{\mathbf{u}}$ ,  $\dot{\mathbf{q}}$ ,  $\mathbf{W}$ ,  $\dot{\mathbf{S}}$ ,  $\lambda$ ,  $\mathbf{u}$ ,  $\mathbf{q}$ ,  $\mathbf{W}$ , and  $\mathbf{S}$ . The independent states are the virtual states  $\bar{\mathbf{W}}$ , and  $\bar{\mathbf{S}}$ . The independent states can assume arbitrary values, and locally all the dependent coordinates can be uniquely computed. Thus, based on the value of  $\bar{\mathbf{S}}$ , Eq. (34k) is used to compute  $\mathbf{S}$ . Due to the fashion in which  $S$  was selected,

$$\det \begin{vmatrix} \frac{\partial \mathbf{C}(\mathbf{q}, t)}{\partial \mathbf{q}} & \mathbf{0} \\ -\frac{\partial \mathbf{s}(\mathbf{q}, t)}{\partial \mathbf{q}} & \mathbf{I} \end{vmatrix} \neq 0$$

and therefore, based on the implicit function theorem,  $\mathbf{q}$  can be locally computed based on Eqs. (34e) and (34f). Likewise, based on the value of  $\bar{\mathbf{W}}$ , Eq. (34j) is used to compute  $\mathbf{W}$ , which is then used in conjunction with Eqs. (34h) and (34d) to compute  $\mathbf{u}$ . Equation (34b) provides  $\dot{\mathbf{q}}$ , Eqs. (34a) and (34c) are simultaneously solved for  $\lambda$  and  $\dot{\mathbf{u}}$ , and finally Eq. (34i) is used to compute  $\dot{\mathbf{W}}$ . Introducing for the system of equations (34a) through (34k) the notation

$$F(\dot{\mathbf{u}}, \dot{\mathbf{q}}, \dot{\mathbf{W}}, \dot{\mathbf{S}}, \lambda, \mathbf{u}, \mathbf{q}, \mathbf{W}, \mathbf{S}, \bar{\mathbf{W}}, \bar{\mathbf{S}}, t) = \mathbf{0} \quad (35)$$

the quantities of interest, namely  $\frac{\partial \dot{\mathbf{S}}}{\partial \bar{\mathbf{S}}}$ ,  $\frac{\partial \dot{\mathbf{S}}}{\partial \bar{\mathbf{W}}}$ ,  $\frac{\partial \dot{\mathbf{W}}}{\partial \bar{\mathbf{S}}}$ , and  $\frac{\partial \dot{\mathbf{W}}}{\partial \bar{\mathbf{W}}}$  are computed based on the conditions

$$\frac{\partial F}{\partial \bar{\mathbf{W}}} = \mathbf{0} \quad \frac{\partial F}{\partial \bar{\mathbf{S}}} = \mathbf{0} \quad (36)$$

which leads to  $\mathbf{A} \mathbf{X} = \mathbf{B}$ , a multiple right side linear system for which  $\mathbf{A}$ ,  $\mathbf{X}$ , and  $\mathbf{B}$  are provided in Table 1. Note that the presentation relied on the fact that the constraints are holonomic. The method easily accommodates the case of non-holonomic constraints, the caveat is that Eq. (34d) is replaced by a new set of equations that contain  $\dot{\mathbf{C}}$ , but also the strictly non-holonomic constraints. Likewise, the selection of states  $\mathbf{S}$  and  $\mathbf{W}$  should be carried out independently at level 0 and level 1, as  $\mathbf{S} \in \mathbb{R}^{s_0}$ , and  $\mathbf{W} \in \mathbb{R}^{s_1}$ . Furthermore, the concept of number of degrees of freedom becomes somewhat fuzzy, as  $s_1 < s_0$ . However, these technicalities are not taking away from the generality of the augmented approach, which remains a direct and simple way of conducting sensitivity or linear analysis of the nonlinear differential-algebraic set of equations of motion.

### NUMERICAL EXPERIMENTS

The proposed linearization algorithm was validated in three ways. First, to test the cases when no user-states were provided and the linearization was carried out in terms of internal states, a complete suite of examples ranging from simple models to complex models with several hundred states (including flexible bodies) were tested and results were compared with results obtained using the previously available MSC.ADAMS FORTRAN-based solver. This previous implementation is built around an SODE



Table 2. WASHER MODEL RESULTS.

Undmp. Freq.	Damping Ratio	$\Re$	$\Im$
7.316E-3	1.000E0	-7.316E-3	0.000E0
7.404E-3	1.000E0	-7.404E-3	0.000E0
8.440E-3	1.000E0	-8.440E-3	0.000E0
8.508E-3	1.000E0	-8.508E-3	0.000E0
3.216E-1	1.000E0	-3.216E-1	0.000E0
3.447E-1	1.000E0	-3.447E-1	0.000E0
3.449E-1	1.000E0	-3.449E-1	0.000E0
1.395E+1	1.000E0	-1.395E01	0.000E0
1.398E+1	1.000E0	-1.398E01	0.000E0
1.439E+1	1.000E0	-1.439E01	0.000E0
1.702E+1	1.000E0	-1.702E01	0.000E0
2.706E+1	1.000E0	-2.706E01	0.000E0
1.020E+2	1.000E0	-1.020E02	0.000E0
1.323E+2	1.000E0	-1.323E02	0.000E0
7.233E-1	3.744E-2	-2.708E-2	7.228E-1
7.237E-1	3.753E-2	-2.716E-2	7.232E-1
2.712E0	1.021E-1	-2.771E-1	2.698E0

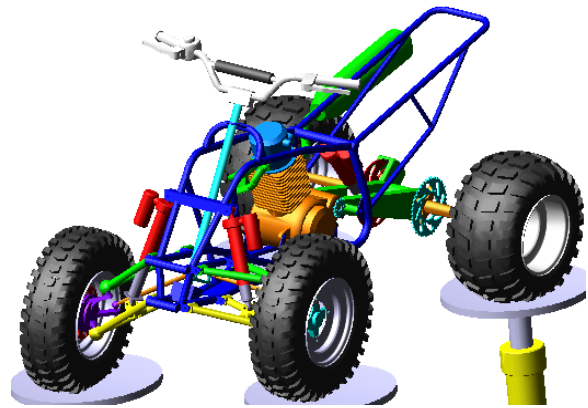


Figure 3. AN ALL TERRAIN VEHICLE (ATV).

Table 3. ATV EIGENSOLUTION FOR SELECTED VALUES. PROPOSED METHOD RESULTS AND % ERROR.

$\Re$	$\Im$	ERR- $\Re$ [%]	ERR- $\Im$ [%]
-1.426E0	5.145E1	17.80	1.13
-1.244E0	6.331E1	21.98	2.78
-4.272E1	3.966E2	0.20	0.00
-4.915E1	4.236E2	1.11	0.80
-8.835E2	1.083E3	0.86	0.04
-6.279E3	2.265E3	0.08	0.09
-6.279E3	2.266E3	0.00	0.06
-3.827E3	3.234E3	0.05	0.15
-4.338E3	4.221E3	0.04	0.04
-8.848E3	5.021E3	0.00	0.06

former MSC.ADAMS linearization solver. Typically the results agree well, with the exception of two sets of eigenvalues. The source of these difference is (a) a series of simplifying assumptions embedded in the previous MSC.ADAMS solver, and (b) the fact that the linearization was performed at the end of a dynamic simulation. The dynamic simulation was first carried out with the legacy MSC.ADAMS FORTRAN-based solver, and then with a new object-oriented C++ solver available in the 2005 version of MSC.ADAMS in which the proposed linearization method is implemented. Given the complexity of the model, the system configuration at the end of the dynamic analysis is slightly different in the two solvers and as indicated in (b) this contributes to the difference between the two sets of eigenvalues. Overall though, good agreement is noticed between the old and new implementations. Notice that errors in the high frequency range are negligible.

### Front suspension system

The model in Fig. 4 represents a vehicle suspension. It has nine rigid bodies, one flexible body, three revolute joints, six

spherical joints, two translational joints, one universal joint, two fixed joints, two prescribed motions, 27 degrees of freedom, and a total of 88 coordinates describing the system (34 modal coordinates were used to model the flexible body).

Table 4 shows the resulting eigenvalues (28 values with positive imaginary part) after running the model using the proposed method. Again, no user-defined coordinates were used in order to match the states used by the old solver. When compared with results obtained with the old MSC.ADAMS FORTRAN-based solver the error differences were negligible.

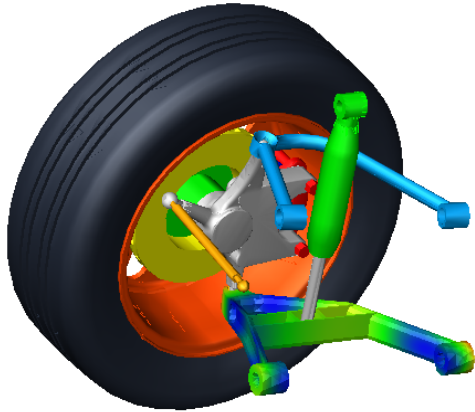


Figure 4. A SUSPENSION MODEL.

## CONCLUSIONS

The paper presents an approach for computing the linearization around an operating point of the index 3 DAEs associated with a constrained mechanical system. The method proposed is equivalent to a state-space reduction followed by linearization of the set of second order state space ordinary differential equations (SSODE) of motion. However, the approach does not reduce the problem to SSODE form and instead it augments the set of equations of motion and computes the quantities of interest as the solution of multiple right side linear systems. Thus, the reduction process required to bring the DAEs of motion to SSODE form is deferred to the sparse linear solver. The proposed method is simple and versatile; it handles user-defined differential equations coming out of controls, arbitrary user-defined holonomic/non-holonomic constraints, arbitrary user forces and variables, etc. A second attractive attribute of the method is its ability to allow the user to define a specific set of coordinates (user-states) in which the linearization will be carried out. If the user-states specified are in deficit, the method completes this set with internal coordinates. If specified in excess, an appropriate number of user-states are dropped by the algorithm. Lastly, the method is straightforward to implement and it is now the method of choice in the MSC.ADAMS simulation package. Three numerical experiments with real-life models validate the accuracy of the proposed method.

## ACKNOWLEDGMENT

The authors would like to thank Anthony Sajdak of MSC.Software for providing the ATV model presented in this paper.

## REFERENCES

- [1] Featherstone, R., 1983. "The calculation of robot dynamics using articulated-body inertias". *International Journal of Robotics Research*(1), pp. 13–30.
- [2] Haug, E. J., 1989. *Computer-Aided Kinematics and Dynamics of Mechanical Systems*. Prentice-Hall, Englewood Cliffs, New Jersey.
- [3] de Jalon, J. G., and Bayo, E., 1994. *Kinematic and Dynamic Simulation of Multi-body Systems. The real time challenge*. Springer-Verlag, Berlin.
- [4] Anderson, K., and Duan, S., 2000. "Highly parallelizable low order dynamics algorithm for complex multi-rigid-body systems". *AIAA Journal of Guidance, Control and Dynamics*, **23**(2).
- [5] Shabana, A. A., 1994. *Computational Dynamics*. John Wiley & Sons.
- [6] Brennan, K. E., Campbell, S. L., and Petzold, L. R., 1989. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. North-Holland, New York.
- [7] Orlandea, N., Chace, M. A., and Calahan, D. A., 1977. "A sparsity-oriented approach to the dynamic analysis and design of mechanical systems – part I". *Transactions of the ASME Journal of Engineering for Industry*, pp. 773–779.
- [8] Wehage, R. A., and Haug, E. J., 1982. "Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems". *J. Mech. Design*, **104**, pp. 247–255.
- [9] Liang, C. D., and Lance, G. M., 1987. "A differentiable null-space method for constrained dynamic analysis". *ASME Journal of Mechanism, Transmission, and Automation in Design*, **109**, pp. 405–410.
- [10] Kim, S. S., and Vanderploeg, M. J. "QR decomposition for state space representation of constrained mechanical dynamic systems". *ASME Journal of Mechanism, Transmission, and Automation in Design*, **108**, pp. 183–188.
- [11] Gear, C. W., Gupta, G., and Leimkuhler, B., 1985. "Automatic integration of the eulerlagrange equations with constraints". *J. Comp. Appl. Math.*, **12**, pp. 77–90.
- [12] Baumgarte, J., 1972. "Stabilization of constraints and integrals of motion in dynamical systems". *Computer Methods in Applied Mechanics and Engineering*, **1**, pp. 1–16.
- [13] Atkinson, K. E., 1989. *An introduction to numerical analysis*, second ed. John Wiley & Sons Inc., New York.
- [14] MSCsoftware, 2005. ADAMS User Manual.
- [15] Sohoni, V. N., and Whitesell, J., 1986. "Automatic linearization of constrained dynamical models". *ASME Journal of Mechanisms, Transmissions and Automation in Design*, **108**(8), pp. 300–304.

Table 4. SUSPENSION MODEL RESULTS.

Undmp. Freq.	Damping Ratio	$\Re$	$\Im$
2.260E3	1.000E00	-2.260E3	0.000E0
5.009E3	1.000E00	-5.009E3	0.000E0
4.262E1	2.288E-2	-0.975E0	4.261E1
1.500E2	5.906E-2	-8.863E0	1.497E2
1.324E3	9.799E-1	-1.297E3	2.640E2
3.015E2	1.401E-1	-4.225E1	2.985E2
3.378E2	9.887E-2	-3.340E1	3.361E2
2.507E3	9.905E-1	-2.484E3	3.443E2
2.743E3	9.907E-1	-2.718E3	3.717E2
3.845E2	9.910E-2	-3.811E1	3.826E2
3.441E3	9.932E-1	-3.418E3	3.981E2
3.785E3	9.927E-1	-3.758E3	4.554E2
1.315E3	9.354E-1	-1.231E3	4.651E2
9.233E2	8.238E-1	-7.607E2	5.233E2
1.986E3	9.642E-1	-1.915E3	5.262E2
3.215E3	9.850E-1	-3.167E3	5.536E2
4.930E3	9.933E-1	-4.897E3	5.685E2
4.054E3	9.898E-1	-4.013E3	5.754E2
6.360E2	2.555E-1	-1.625E2	6.149E2
1.323E3	8.824E-1	-1.168E3	6.228E2
7.528E2	2.043E-1	-1.538E2	7.370E2
6.219E3	9.902E-1	-6.158E3	8.670E2
6.029E3	9.867E-1	-5.949E3	9.793E2
2.592E3	8.589E-1	-2.226E3	1.327E3
9.678E3	9.868E-1	-9.551E3	1.564E3
3.474E3	8.519E-1	-2.960E3	1.819E3
4.165E3	8.958E-1	-3.731E3	1.850E3
2.309E4	9.451E-1	-2.182E4	7.546E3