



# Analytical Fully-Recursive Sensitivity Analysis for Multibody Dynamic Chain Systems

KURT S. ANDERSON and YUHUNG HSU

*Department of Mechanical Engineering, Aeronautical Engineering, and Mechanics, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, U.S.A.; E-mail: anderk5@rpi.edu*

(Received: 7 April 2000; accepted in revised form: 8 February 2001)

**Abstract.** This paper presents a novel fully recursive method, a direct differentiation based approach, which facilitates first-order sensitivity analysis in optimal design problems involving multibody dynamic systems. A state space  $O(n)$  dynamic analysis algorithm based on a velocity space projection method, as promoted by Kane [18], forms the foundation of the underlying formulation. This algorithm can significantly reduce the massive number of mathematical and associated computational operations involved in explicitly generating and solving the sensitivity equations. This benefit is particularly evident for systems involving a combination of many state variables and design parameters. The development presented in this paper focuses on chain systems to illustrate the recursive nature of the algorithm. The computational efficiency and solution accuracy of the presented algorithm are investigated through the procedures application to the simulation and design sensitivity determination of spatial chain systems involving 2, 4, 6, . . . , 24 degrees of freedom, as well as a simple planar double pendulum.

**Key words:** multibody optimization, sensitivity analysis, recursive algorithm, order- $n$ .

## 1. Introduction

Computational multibody dynamics as a design and analysis tool has become essential in many applications. Evidence of this is seen in the design of practical systems which find broad application in modern society, ranging from simple mechanisms, to automobiles, to complex spacecraft. The design of multibody systems may therefore be thought of as a keystone of modern technology and links various engineering disciplines.

For multibody design problems where analysis and computation are expensive, a robust optimization strategy with rapid solution convergence characteristics is highly desirable. Optimization strategies employing sensitivity (derivative) information have been shown to be more efficient in many cases than non-derivative methods, and thus are well-suited for current design purposes. Sensitivity analysis for dynamic systems requires the derivatives of performance measures (objective and constraint functions) and the state variables (generalized coordinates and their associated time derivatives) with respect to the design variables [9, 14]. These latter

derivatives are often needed because performance measures are generally functions of both design and state variables.

In the literature concerning multibody design problems, direct differentiation [9, 24, 26] and adjoint variable methods [5, 14, 19, 25] are two major analytical techniques on which the sensitivity analysis has been based. From a mathematical perspective, the direct differentiation methods may be *conceptually* the most straightforward techniques because the system governing equations are explicitly and directly differentiated with respect to design variables to yield key quantities of state sensitivities [9]. By comparison, the adjoint variable method, based on variation principles, avoids these direct calculations through the introduction of a set of so-called adjoint variables [4, 15]. Manipulating these adjoint variables, variations of the system equations, and the variations of design criteria, produces the required adjoint relationships. Solving this sequence of adjoint relationships yields the design sensitivity vector which directly corresponds to the variation (sensitivity) of design criteria in terms of design variable variations [14].

The works of many individuals [3–7, 9–11, 14–17, 19, 20, 24–26] have established sensitivity analysis as a viable tool in dynamic system design. Unfortunately, many of these works do not provide detailed information regarding the overall computational cost associated with forming and solving the first-order sensitivity equations. This cost can be significant, and may become prohibitive. For a dynamic system design problem involving  $p$  design variables,  $n$  generalized coordinates, and  $m$  independent algebraic constraint equations, the simultaneous solution of  $(n + m)(p + 1)$  and  $(n + m + p)$  differential algebraic equations are required for the direct differentiation method and the adjoint variable method, respectively [24]. If a direct solutions scheme is used, then  $O(n^3)$  operations are usually needed for the decomposition and solve aspects of the problem solution. This implies that the overall computational requirements of both methods could be a prohibitive high order of  $n^4$ .

Since there is a one way coupling of the sensitivity analysis to the forward dynamics analysis, many different computation schemes used in dynamic analysis can be employed to derive an effective design sensitivity method. Within this context, direct differentiation methods are well-suited due to their procedural similarity to the underlying dynamic analysis, their ease of implementation, and solution accuracy. The first-order sensitivity analysis algorithm presented in this paper has its roots in a recursive state space dynamic formulation [1, 21]. Combining the recursive formulation with direct differentiation methods permits a systematic and effective strategy for generating and solving sensitivity equations simultaneously. The method presented will greatly reduce computational expense of a specialize direct method in handling large sets of sensitivity equations.

## 2. A Sensitivity Analysis Overview

Several methods exist for performing the important task of determining the derivative quantities required in sensitivity-based optimization strategies. The first and possibly most broadly applied method is divided-difference approximations. Divided-difference estimates the sensitivity by analyzing perturbed design points without explicitly differentiating the equations. Due to the additional set of function evaluations required, this approach may be numerically expensive [7]. The accuracy of this approach is also affected by the magnitude of perturbation as well as the shape of the true design hyperplane at this design point, and is subject to truncation and possible round-off errors.

The use of the adjoint variable method to compute first-order sensitivity as it relates to multibody dynamic systems was initiated in the late 1970s [15] and has made significant progress since then [4, 5, 10]. A set of adjoint equations is introduced to represent the variations of state, initial and final times, and Lagrange multipliers. This approach can yield sensitivity equations with either Differential-Algebraic Equations (DAE) [14] or Ordinary Differential Equations (ODE) [4]. The adjoint variable method reduces the amount of computations associated with direct differentiation by reducing the number of equations to be solved [16, 24]. Specifically, the adjoint variable method requires the solution of  $(n + m + p)$  differential algebraic equations, compared to the solution of  $(n + m)(p + 1)$  equations for the direct differentiation method. While this can dramatically reduce the amount of computational operations for large system, this method can still be prohibitively expensive and has the disadvantage of not yielding the potentially useful state derivatives explicitly [24].

Additionally, the adjoint variable method may result in highly complex formulations, and for a variety of reasons can be inconvenient to implement [9, 20].

Firstly, a massive volume of information may be determined during the forward integration of the system equations, and must be stored for subsequent use in the backward integration. This increases the number of input/output operations necessary, greatly slowing the simulation/analysis process, and the amount of data storage space required may be quite large. Second, because efficient integration schemes usually use an adaptive stepsize control, the timesteps of forward and backward integration may not coincide [10]. An interpolation model must thus be employed. Whenever the derivatives at intermediate timesteps are required, their values are evaluated from the interpolation model. This procedure may be a source of inaccuracy causing the solutions of adjoint method to be rather sensitive to the interpolation error.

Potential advantages of using a direct differentiation method are that sensitivity equations can be derived in a very straightforward manner by directly differentiating the system equations of motion without introducing additional complex numerical schemes. Additionally, the set of sensitivity equations associated with

each design variable can be generated/integrated separately from, and concurrently with, those associated with the other design variables.

The resulting analytical expressions of sensitivity are free from truncation errors and the derivatives are accurate up to integration precision. These methods result in initial value problems which can be integrated forward in time simultaneously with the equations of motion and performance measures. Another potential advantage of direct differentiation methods is the explicit determination of state derivatives with respect to design parameters, which may be desired. However, a general direct differentiation procedure becomes cumbersome due to its production of large quantities of derivative terms, and the size of the set of sensitivity equation is considerably greater than those produced by adjoint methods.

Another method that is applicable for yielding derivatives is the automatic differentiation technique (AD) [3, 7]. Simply speaking, AD is a collection of computer science techniques, which systematically implement the chain rule of differentiation. AD can be proceed by either *forward* or *reverse* modes. The difference in these modes is that the implementation of the chain rule of differentiation starting from the independent (design) variables results in the forward mode, while differentiation which begins from the dependent (performance measure) variables is the reverse mode. If the number of independent variables is larger than the number of dependent variables, then the reverse mode will require fewer operations than the forward mode, but is quite complicated to implement. Recently, AD codes have become available for the production of sensitivity-enhanced versions of computer programs. Many cases have been reported of the superior performance of AD relative to divided-difference approaches in speed and accuracy [3, 6]. Additionally, significantly less computer memory usage tends to be necessary for AD, than is required using other symbolic formulation manipulation programs (e.g., MAPLE) [8].

In general, AD tools can reduce the engineer effort in hand-coding and are recommended over finite difference for moderately sized computer programs [3]. However, some researchers have pointed out that one must be careful in the application of AD tools to arbitrary numerical algorithms. Eberhard [10] indicates that including expert knowledge about the problem structure within the algorithm is extremely hard to achieve with AD tools. Without implementation precautions, a pure syntactical analysis may not be sufficient and could possibly yield wrong results.

Additionally, recursive state-space formulations are often used for the derivation of complicated system equations. The operation cost for such a recursive computation must be accounted for when an AD technique is employed and the AD procedure used must ensure convergence of the process [3]. For more advanced dynamic formulation algorithms which manipulate equations very differently from conventional (more traditional) methods, care must also be taken in the augmentation of the original codes for the purpose of producing sensitivity information.

This is particularly so for cases involving algorithms which decompose and solve the equations of motion as they are being formed.

### 3. Dynamic and Sensitivity Analysis Preliminaries

Design optimization problems of multibody dynamic systems begin with the mathematical formulation of the system equations of motion, configuration and design constraint conditions, and the objective functions. For a nominal design problem, the performance measure functions can be expressed as

$$\begin{aligned} \min. \quad J &= g(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}, t; \underline{P}) + \int_{t_0}^{t_1} f(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}, t; \underline{P}) dt \\ \text{st.} \quad G_i(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}, t; \underline{P}) &\leq 0 \\ H_j(\underline{q}, \underline{\dot{q}}, \underline{\ddot{q}}, t; \underline{P}) &= 0. \end{aligned} \quad (1)$$

In this expression,  $\underline{q}$  is the vector of system's  $n$  generalized coordinates;  $\underline{\dot{q}}$  and  $\underline{\ddot{q}}$  are the first and second time derivative of  $\underline{q}$ , respectively;  $t$  denotes time; and  $\underline{P}$  is a vector of all  $p$  design variables. Sensitivity-base optimization methods require the derivatives of Equation (1) with respect to each of the design variable  $p_j$  in  $\underline{P}$ . To this end, applying a direct differentiation approach to the objective functions  $J$  yields

$$\frac{dJ}{dp_j} = J_{,p_j} + J_{,q_i} \frac{dq_i}{dp_j} + J_{,\dot{q}_i} \frac{d\dot{q}_i}{dp_j} + J_{,\ddot{q}_i} \frac{d\ddot{q}_i}{dp_j}, \quad (2)$$

where the derivatives of the design constraints  $G$  and  $H$  in Equation (1) yield the same form as Equation (2). In expression (1), and all subsequent mathematical expressions, the following indicial notation is employed

$$\frac{\partial A_i}{\partial B_j} = A_{i,B_j}, \quad (3)$$

where the subscripts  $i$  and  $j$  indicate the indices/components of  $A$  and  $B$ , respectively. Any subscript character placed after the symbol ',' denotes the partial derivative of the preceding term with respect to this quantity, and summation over all repeated indices within each term is implied.

The quantity  $J_{,p_j}$  appearing in Equation (2) can be calculated without much difficulty because it only involves terms of  $J$  containing  $p_j$  explicitly. The terms  $J_{,q_i}$ ,  $J_{,\dot{q}_i}$ , and  $J_{,\ddot{q}_i}$  in Equation (2) can also be determined in a similar manner. However, due to the implicit relationships between  $\underline{q}$  and  $p_j$ , the contribution to  $dJ/dp_j$  from  $dq_i/dp_j$ ,  $d\dot{q}_i/dp_j$ , and  $d\ddot{q}_i/dp_j$  is considerably more involved. Fortunately, these derivatives need not be performed independently, since the relations

$$\frac{dq_i}{dp_j} = \left[ \int_0^\tau \frac{d\dot{q}_i}{dp_j} dt + \frac{dq_i}{dp_j} \Big|_{t=0} \right] \Big|_{\underline{P}}, \quad (4)$$

and

$$\frac{d\dot{q}_i}{dp_j} = \left[ \int_0^\tau \frac{d\ddot{q}_i}{dp_j} dt + \frac{d\dot{q}_i}{dp_j} \Big|_{t=0} \right] \Big|_{\underline{P}}, \quad (5)$$

exist.

Equation (5) enables the total derivatives of  $dq_i/dp_j$  and  $d\dot{q}_i/dp_j$  to be obtained from the temporal integration of  $d\ddot{q}_i/dp_j$  evaluated at the current design point  $\underline{P}$ . The determination of derivatives for use in the sensitivity analysis is now reduced to the problem of determining values of  $d\ddot{q}_i/dp_j$  for the entire time interval of interest. Traditionally, these key elements are computed by considering a general form of equations of motion [9]

$$\underline{\mathcal{M}}(\underline{q}, t; \underline{P})\ddot{\underline{q}} = \underline{\mathcal{K}}(\underline{q}, \dot{\underline{q}}, t; \underline{P}), \quad (6)$$

where  $\underline{\mathcal{M}}(\underline{q}, t; \underline{P})$  is the  $n \times n$  system mass matrix and  $\underline{\mathcal{K}}(\underline{q}, \dot{\underline{q}}, t; \underline{P})$  is an  $n$  term vector containing the applied forces, as well as the centripetal and Coriolis acceleration contributions to the inertia forces.

Taking the first-order total derivative of Equation (6) with respect to  $p_j$  yields

$$\frac{d}{dp_j}(\underline{\mathcal{M}}\ddot{\underline{q}}) = \frac{d}{dp_j}\underline{\mathcal{K}}, \quad (7)$$

$$\left( \underline{\mathcal{M}}_{,p_j} + \underline{\mathcal{M}}_{,q_r} \frac{dq_r}{dp_j} \right) \ddot{\underline{q}} + \underline{\mathcal{M}} \frac{d\ddot{\underline{q}}}{dp_j} = \underline{\mathcal{K}}_{,p_j} + \underline{\mathcal{K}}_{,q_r} \frac{dq_r}{dp_j} + \underline{\mathcal{K}}_{,\dot{q}_r} \frac{d\dot{q}_r}{dp_j}, \quad (8)$$

$$\frac{d\ddot{\underline{q}}}{dp_j} = \underline{\mathcal{M}}^{-1} \left[ \underline{\mathcal{K}}_{,p_j} + \underline{\mathcal{K}}_{,q_r} \frac{dq_r}{dp_j} + \underline{\mathcal{K}}_{,\dot{q}_r} \frac{d\dot{q}_r}{dp_j} - \left( \underline{\mathcal{M}}_{,p_j} + \underline{\mathcal{M}}_{,q_r} \frac{dq_r}{dp_j} \right) \ddot{\underline{q}} \right], \quad (9)$$

where  $r = 1, \dots, n$ . Equation (9) is the set of governing equations yielding  $d\ddot{q}_r/dp_j$  in first-order sensitivity analysis. It represents the sensitivity for an open-loop dynamic system written in state space form associated with the design variable  $p_j$ . At this point the analyst is left with the still considerable task of producing the individual terms appearing on the right-hand side of Equation (9), and then performing the indicated matrix operations to complete the first-order sensitivity analysis.

### 3.1. UNDERLYING DYNAMIC FORMULATION

A velocity space projection method as presented by Kane and Levinson in [18] is the underlying formulation used in this paper to produce the terms appearing in Equation (9). The concepts of generalized speeds, partial angular velocities, and partial velocities are the key to this method. The angular velocity of body  $k$  with

respect to an inertial reference frame  $N$ ,  ${}^N\boldsymbol{\omega}^k$ , and the velocity of the mass center of body  $k$  in  $N$ ,  ${}^N\mathbf{v}^k$ , can be expressed as

$${}^N\boldsymbol{\omega}^k = \sum_{r=1}^n {}^N\boldsymbol{\omega}_r^k u_r + {}^N\boldsymbol{\omega}_t^k, \quad (10)$$

$${}^N\mathbf{v}^k = \sum_{r=1}^n {}^N\mathbf{v}_r^k u_r + {}^N\mathbf{v}_t^k. \quad (11)$$

In general, the generalized speeds  $u_r$  ( $r = 1, \dots, n$ ), appearing in Equation (10–11) are analyst specified invertible combinations of the generalized coordinate first time derivatives, and characterize the motion of the system. The vector quantities  ${}^N\boldsymbol{\omega}_r^k$  and  ${}^N\mathbf{v}_r^k$  are the  $r$ th partial angular velocity of body  $k$  in  $N$ , and the  $r$ th partial velocity of the center of mass of body  $k$  in  $N$ , respectively. These quantities form the basis vectors in the holonomic velocity space associate with the system and are the key to forming the generalized active forces and generalized inertia forces. Finally, the quantities  ${}^N\boldsymbol{\omega}_t^k$  and  ${}^N\mathbf{v}_t^k$  are respectively referred to as the angular velocity and velocity *remainder terms*, and result from specified/prescribed motions.

Differentiating  ${}^N\boldsymbol{\omega}^k$  and  ${}^N\mathbf{v}^k$  with respect to time yields the angular acceleration  ${}^N\boldsymbol{\alpha}^k$ , and the center of mass acceleration  ${}^N\mathbf{a}^k$  of body  $k$  in  $N$

$${}^N\boldsymbol{\alpha}^k \equiv \dot{{}^N\boldsymbol{\omega}^k} = \sum_{r=1}^n {}^N\boldsymbol{\omega}_r^k \dot{u}_r + \left( \sum_{r=1}^n \dot{{}^N\boldsymbol{\omega}_r^k} u_r + \dot{{}^N\boldsymbol{\omega}_t^k} \right), \quad (12)$$

$${}^N\mathbf{a}^k \equiv \dot{{}^N\mathbf{v}^k} = \sum_{r=1}^n {}^N\mathbf{v}_r^k \dot{u}_r + \left( \sum_{r=1}^n \dot{{}^N\mathbf{v}_r^k} u_r + \dot{{}^N\mathbf{v}_t^k} \right). \quad (13)$$

Kane's dynamical equations [18] can be written as

$$F_r + F_r^* = 0 \quad (r = 1, 2, \dots, n), \quad (14)$$

with the  $r$ th *generalized active force*  $F_r$  and the  $r$ th *generalized inertia force*  $F_r^*$  defined as

$$F_r \triangleq \sum_{k=1}^{\nu} ({}^N\boldsymbol{\omega}_r^k \cdot {}^N\mathbf{T}^k + {}^N\mathbf{v}_r^k \cdot {}^N\mathbf{R}^k), \quad (15)$$

$$F_r^* \triangleq \sum_{k=1}^{\nu} ({}^N\boldsymbol{\omega}_r^k \cdot {}^N\mathbf{T}^{k*} + {}^N\mathbf{v}_r^k \cdot {}^N\mathbf{R}^{k*}), \quad (16)$$

where  $\nu$  is the number of bodies which comprise the system. Quantities  ${}^N\mathbf{T}^k$  and  ${}^N\mathbf{R}^k$  represent a resultant force-torque system acting through the mass center of body  $k$  which is equivalent to the set of all contact and distance forces acting on

$k$ . Similarly,  ${}^N\mathbf{R}^{k*}$  is the inertia force associated with acceleration of the center of mass of body  $k$  in  $N$ , and  ${}^N\mathbf{T}^{k*}$  is the central inertia torque associated with angular acceleration of the body  $k$  in  $N$ . Equation (14) can thus be written as

$$\sum_{i=1}^n \underline{\mathcal{M}}_{ri} \dot{u}_i - \underline{\mathcal{K}}_r = 0 \quad (r = 1, 2, \dots, n) \quad (17)$$

with the elements of  $\underline{\mathcal{M}}$  and  $\underline{\mathcal{K}}$  matrices obtained from

$$\underline{\mathcal{M}}_{ri} = \sum_{k=1}^v [{}^N\boldsymbol{\omega}_r^k \cdot \mathbf{I}^{k/k*} \cdot {}^N\boldsymbol{\omega}_i^k + {}^N\mathbf{v}_r^k \cdot \mathbf{M}^k \cdot {}^N\mathbf{v}_i^k], \quad (18)$$

$$\underline{\mathcal{K}}_r = \sum_{k=1}^v [{}^N\boldsymbol{\omega}_r^k \cdot \mathbf{I}^{k/k*} \cdot {}^N\boldsymbol{\omega}_i^k + {}^N\mathbf{v}_r^k \cdot \mathbf{M}^k \cdot {}^N\mathbf{v}_i^k], \quad (19)$$

where  $\mathbf{M}^k$  and  $\mathbf{I}^{k/k*}$  are the central mass and rotational inertial dyadics of body  $k$ .

If one defines scalar matrices for: the partial velocities (or *free mode of motion* [23])  $\underline{\mathcal{P}}_r^k$ ; the local central inertia matrix for the body  $\underline{\mathcal{I}}_1^k$ ; and the total applied load  $\underline{\mathcal{F}}_1^k$ , each associated with the body  $k$  and the local body  $k$  basis, as

$$\underline{\mathcal{P}}_r^k = \begin{bmatrix} {}^N\boldsymbol{\omega}_r^k \\ {}^N\mathbf{v}_r^k \end{bmatrix}_{6 \times 1}, \quad (20)$$

$$\underline{\mathcal{I}}_1^k = \begin{bmatrix} \mathbf{I}^{k/k*} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^k \end{bmatrix}_{6 \times 1}, \quad (21)$$

$$\underline{\mathcal{F}}_1^k = \begin{bmatrix} \underline{\mathcal{T}}_1^{k*} + \underline{\mathcal{T}}_1^k \\ \underline{\mathcal{R}}_1^{k*} + \underline{\mathcal{R}}_1^k \end{bmatrix}_{6 \times 1}, \quad (22)$$

with  $\underline{\mathcal{T}}_1^{k*}$  and  $\underline{\mathcal{R}}_1^{k*}$  expressed as

$$\underline{\mathcal{T}}_1^{k*} = -\underline{\mathcal{M}}^k \underline{\mathcal{a}}_1^k, \quad (23)$$

$$\underline{\mathcal{R}}_1^{k*} = -\underline{\mathcal{I}}^{k/k*} \underline{\mathcal{a}}_1^k - ({}^N\boldsymbol{\omega}_r^k \times) \underline{\mathcal{I}}^{k/k*} \underline{\mathcal{a}}_1^k, \quad (24)$$

where the quantity  $(\boldsymbol{\gamma} \times)$  is a matrix form of the vector cross product operator  $\boldsymbol{\gamma} \times$ ;  $\underline{\mathcal{M}}_{ri}$  and  $\underline{\mathcal{K}}_r$  can then be expressed as

$$\underline{\mathcal{M}}_{ri} = \sum_{k=1}^v (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{I}}_1^k \underline{\mathcal{P}}_i^k, \quad (25)$$

and

$$\underline{\mathcal{K}}_r = \sum_{k=1}^v (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{F}}_1^k. \quad (26)$$

Differentiating Equation (25) with respect to  $p_j$ , and generalized coordinates  $q_r$  yields

$$\underline{\mathcal{M}}_{r,p_j} = \sum_{k=1}^v [(\underline{\mathcal{P}}_r^k)^T_{,p_j} \underline{\mathcal{J}}_1^k \underline{\mathcal{P}}_i^k + (\underline{\mathcal{P}}_r^k)^T_{1,p_j} \underline{\mathcal{J}}_{1,p_j}^k \underline{\mathcal{P}}_i^k + (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{J}}_1^k \underline{\mathcal{P}}_{i,p_j}^k], \quad (27)$$

$$\underline{\mathcal{M}}_{r,q_r} = \sum_{k=1}^v [(\underline{\mathcal{P}}_r^k)^T_{,q_r} \underline{\mathcal{J}}_1^k \underline{\mathcal{P}}_i^k + (\underline{\mathcal{P}}_r^k)^T_{1,q_r} \underline{\mathcal{J}}_{1,q_r}^k \underline{\mathcal{P}}_i^k + (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{J}}_1^k \underline{\mathcal{P}}_{i,q_r}^k], \quad (28)$$

while the partial derivatives of Equation (26) with respect to  $p_j$ ,  $q_r$ , and  $\dot{q}_r$  produce

$$\underline{\mathcal{K}}_{r,p_j} = \sum_{k=1}^v [(\underline{\mathcal{P}}_r^k)^T_{,p_j} \underline{\mathcal{F}}_1^k + (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{F}}_{1,p_j}^k], \quad (29)$$

$$\underline{\mathcal{K}}_{r,q_r} = \sum_{k=1}^v [(\underline{\mathcal{P}}_r^k)^T_{,q_r} \underline{\mathcal{F}}_1^k + (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{F}}_{1,q_r}^k], \quad (30)$$

$$\underline{\mathcal{K}}_{r,\dot{q}_r} = \sum_{k=1}^v [(\underline{\mathcal{P}}_r^k)^T_{,\dot{q}_r} \underline{\mathcal{F}}_1^k + (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{F}}_{1,\dot{q}_r}^k]. \quad (31)$$

Equations (27–31) can now be substituted into Equation (9) to yield the first-order sensitivity information. However, this brute force direct approach suffers from the major shortcoming of quickly becoming prohibitively costly and cumbersome for systems which involve large  $n$  and/or  $p$ . The production, and subsequent system of equations decomposition associated with  $\underline{\mathcal{M}}^{-1}$  appearing in (9), requires  $O(n^3)$  operations overall by direct methods. Worse still, the matrices  $\underline{\mathcal{M}}_{,p_j}$  and  $\underline{\mathcal{M}}_{,q_r}$  ( $j = 1, \dots, p$ ;  $r = 1, \dots, n$ ) appearing in (9) and determined from Equations (27) and (28), may require  $O(p * n^3)$  and  $O(n * n^3)$  operations, respectively, to be fully determined.

Therefore, an overall  $O(n^4)$  cost is inevitable for the determination of the solutions  $d\dot{q}/dp_j$  from Equation (9). This arguably high cost is an outgrowth of the manner in which the equations of motion are constructed and the blind application of the chain rule of differentiation. More specifically, each element in  $\underline{\mathcal{M}}$  or  $\underline{\mathcal{K}}$  is explicitly assembled and subsequently differentiated, often without regard of the fact that these quantities can be more efficiently determined by making full use of previously determined quantities, or need not be explicitly found. This explicit calculation and differentiation of often recurring quantities, a phenomenon generally inherent in more traditional state-space dynamics formulations, can result in many more operations being performed than is truly needed. To achieve a significant reduction in computational cost through the elimination of many explicit, repeated calculations and differentiations, incorporating recursive relations offers a viable solution.

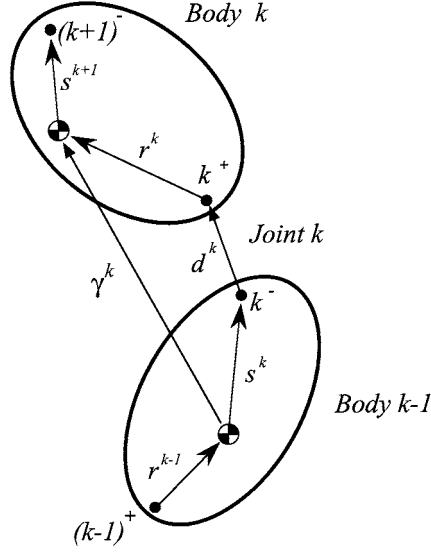


Figure 1. Schematic of adjacent bodies within chain system.

#### 4. Recursive First-Order Sensitivity Analysis

The use of relative coordinates and recursive relations has been shown to have some desirable aspects for producing system equations of motion when using a state space (particularly recursive) formulation. Based on a velocity space projection method [18], the algorithms developed in [1, 2, 21] have demonstrated that generalized acceleration can be determined in  $O(n)$  operation overall for general multibody dynamic systems in sequential implementations. These recursive  $O(n)$  algorithm form the basis for the derivations presented here.

##### 4.1. RECURSIVE DYNAMICS FORMULATIONS

Consider a chain system of rigid bodies, of which an arbitrary pair of adjacent bodies are shown in Figure 1. For this system, parent body  $k - 1$  is connected to its child body  $k$  through joint  $k$ , via joint points  $k^-$  and  $k^+$  which reside in bodies  $k - 1$  and  $k$ , respectively. The position vector  $\mathbf{s}^k$  locates joint  $k$  relative to the mass center of body  $k - 1$ , while the position vector  $\mathbf{r}^k$  locates the mass center of body  $k$  with respect to the outboard end of this same joint. It will also prove convenient to describe the position of child mass center  $k$  relative to parent mass center  $k - 1$  by the vector  $\boldsymbol{\gamma}^k$ . Finally, the generalized speeds  $u_i$  to be used in the recursive holonomic relations are

$$u_k = \dot{q}_k \quad (k = 1, 2, \dots, n). \quad (32)$$

With the various system parameter and variables so defined, the  $O(n)$  algorithms developed by Anderson [1, 2] and Rosenthal [21], give the kinematic relationships between bodies  $k - 1$  and  $k$  as

$${}^N\boldsymbol{\omega}^k = {}^N\boldsymbol{\omega}^{k-1} + {}^{k-1}\boldsymbol{\omega}_r^k \dot{q}_r^k + {}^{k-1}\boldsymbol{\omega}_t^k, \quad (33)$$

$$\begin{aligned} {}^N\mathbf{v}^k &= {}^N\mathbf{v}^{(k-1)} + {}^N\boldsymbol{\omega}^{k-1} \times \mathbf{r}^k \\ &\quad + ({}^{k-1}\boldsymbol{\omega}_r^k \dot{q}_r^k + {}^{k-1}\boldsymbol{\omega}_t^k) \times \mathbf{s}^k + {}^{k-}\mathbf{v}_r^{k+} \dot{q}_r^k + {}^{k-}\mathbf{v}_t^{k+}, \end{aligned} \quad (34)$$

$${}^N\boldsymbol{\alpha}^k = {}^N\boldsymbol{\alpha}^{k-1} + {}^{k-1}\boldsymbol{\omega}_r^k \ddot{q}_r^k + {}^N\boldsymbol{\omega}^{k-1} \times ({}^{k-1}\boldsymbol{\omega}_r^k \dot{q}_r^k + {}^{k-1}\boldsymbol{\omega}_t^k), \quad (35)$$

$$\begin{aligned} {}^N\mathbf{a}^k &= {}^N\mathbf{a}^{(k-1)} + {}^N\boldsymbol{\alpha}^{k-1} \times \boldsymbol{\gamma}^k + {}^N\boldsymbol{\omega}^{k-1} \times ({}^N\boldsymbol{\omega}^{k-1} \times \mathbf{s}^k) \\ &\quad + {}^N\boldsymbol{\omega}^k \times ({}^N\boldsymbol{\omega}^k \times \mathbf{r}^k) \\ &\quad + [{}^{k-1}\boldsymbol{\omega}_r^k \ddot{q}_r^k + {}^N\boldsymbol{\omega}^{k-1} \times ({}^{k-1}\boldsymbol{\omega}_r^k \dot{q}_r^k + {}^{k-1}\boldsymbol{\omega}_t^k)] \times \mathbf{r}^k \\ &\quad + 2 {}^N\boldsymbol{\omega}^{k-1} \times ({}^{k-}\mathbf{v}_r^{k+} \dot{q}_r^k + {}^{k-}\mathbf{v}_t^{k+}) + {}^{k-}\mathbf{v}_r^{k+} \ddot{q}_r^k. \end{aligned} \quad (36)$$

In Equations (33–36), the vectors  ${}^{k-1}\boldsymbol{\omega}_r^k$  and  ${}^{k-}\mathbf{v}_r^{k+}$  represent the  $r$ th partial angular velocity and partial velocity, respectively, associated with joint  $k$ . By comparison, the terms  ${}^{k-1}\boldsymbol{\omega}_t^k$  and  ${}^{k-}\mathbf{v}_t^{k+}$  which also appear in these expressions, are associated with the specified (prescribed) rotational and translational motions, respectively, of this joint. In addition, the free mode of motion  $\underline{\mathcal{P}}^k$  of body  $k$  due to joint  $k - 1$  can be shown [1] to be

$$\begin{aligned} \underline{\mathcal{P}}_{k-1}^k &\equiv \begin{bmatrix} {}^N\boldsymbol{\omega}_{k-1}^k \\ {}^N\mathbf{v}_{k-1}^k \end{bmatrix} = \begin{bmatrix} \underline{U} & \underline{0} \\ \boldsymbol{\gamma}^k \times & \underline{U} \end{bmatrix} \begin{bmatrix} {}^N\boldsymbol{\omega}_{k-1}^{k-1} \\ {}^N\mathbf{v}_{k-1}^{k-1} \end{bmatrix} \\ &= (\underline{\mathcal{J}}^k)^T \begin{bmatrix} {}^N\boldsymbol{\omega}_{k-1}^{k-1} \\ {}^N\mathbf{v}_{k-1}^{k-1} \end{bmatrix} = (\underline{\mathcal{J}}^k)^T \underline{\mathcal{P}}_{k-1}^{k-1}, \end{aligned} \quad (37)$$

where the elements of  $\underline{\mathcal{J}}^k$  and  $\underline{\mathcal{P}}_{k-1}^{k-1}$  are each associated with the local basis vectors fixed in body  $k - 1$ . The *shifting* matrices  $\underline{\mathcal{J}}^k$  is defined as

$$\underline{\mathcal{J}}^k \triangleq \begin{bmatrix} \underline{U} & \boldsymbol{\gamma}^k \times \\ \underline{0} & \underline{U} \end{bmatrix}_{6 \times 6}, \quad (38)$$

where  $\underline{U}$  is simply an identity matrix, and  $\boldsymbol{\gamma}^k \times$  is a skew symmetric matrix representation of the vector cross product operator. One may apply the kinematic relations Equations (33–36) recursively working outward from the system base to its terminal body, producing the generalized velocities together with expressions for the generalized accelerations associated with each body.

When the procedure reaches the system terminal body, a recursive inward procedure takes place to implicitly assemble  $\underline{\mathcal{M}}$  and  $\underline{\mathcal{K}}$ . At the terminal body  $k = n$ , the summation indicated in Equation (25) contains only a single term. This is due to the use of relative coordinates for describing the location of all child bodies relative to their parents, and yields

$$\underline{\mathcal{M}}_{ni} = (\underline{\mathcal{P}}_n^n)^T \underline{\mathcal{I}}_1^n \underline{\mathcal{P}}_i^n. \quad (39)$$

Continuing this process by working inward to body  $n$ 's proximal/parent body  $n-1$ , Equation (25) yields the summation over bodies  $n-1$  and  $n$ , namely

$$\underline{\mathcal{M}}_{(n-1)i} = (\underline{\mathcal{P}}_{n-1}^{n-1})^T \underline{\mathcal{I}}_1^{n-1} \underline{\mathcal{P}}_i^{n-1} + (\underline{\mathcal{P}}_{n-1}^n)^T \underline{\mathcal{I}}_1^n \underline{\mathcal{P}}_i^n. \quad (40)$$

From Equation (37),  $(\underline{\mathcal{P}}_{n-1}^1)^T$  can be written as

$$\begin{aligned} (\underline{\mathcal{P}}_{n-1}^1)^T &= [{}^n \underline{\mathcal{C}}^{n-1} (\underline{\mathcal{J}}^n)^T \underline{\mathcal{P}}_{n-1}^{n-1}]^T \\ &= [(\underline{\mathcal{J}}^n)^T \underline{\mathcal{P}}_{n-1}^{n-1}]^T. \end{aligned} \quad (41)$$

The special matrix  $\overleftarrow{\underline{\mathcal{J}}}^k$  used in Equation (41) provides a basis consistent shifting operation. This matrix is defined as

$$\overleftarrow{\underline{\mathcal{J}}}^k \triangleq \underline{\mathcal{J}}^k \underline{\mathcal{C}}^k, \quad (42)$$

where  $\underline{\mathcal{C}}^k$  is the direction cosine matrix relating basis  $k$  and that of its parent body  $k-1$ . Equation (40) now becomes

$$\begin{aligned} \underline{\mathcal{M}}_{(n-1)i} &= (\underline{\mathcal{P}}_{n-1}^{n-1})^T \underline{\mathcal{I}}_1^{n-1} \underline{\mathcal{P}}_i^{n-1} + (\underline{\mathcal{P}}_{n-1}^n)^T \overleftarrow{\underline{\mathcal{J}}}^n \underline{\mathcal{I}}_1^n \underline{\mathcal{P}}_i^n \\ &= (\underline{\mathcal{P}}_{n-1}^{n-1})^T [\underline{\mathcal{I}}_1^{n-1} \underline{\mathcal{P}}_i^{n-1} + \overleftarrow{\underline{\mathcal{J}}}^n \underline{\mathcal{I}}_1^n \underline{\mathcal{P}}_i^n] \\ &= (\underline{\mathcal{P}}_{n-1}^{n-1})^T (\underline{\mathcal{R}}_{1i}^{n-1} + \overleftarrow{\underline{\mathcal{J}}}^n \underline{\mathcal{R}}_{2i}^n), \end{aligned} \quad (43)$$

or, for a general element  $\underline{\mathcal{M}}_{ki}$ ,

$$\underline{\mathcal{M}}_{ki} = (\underline{\mathcal{P}}_k^k)^T (\underline{\mathcal{R}}_{1i}^k + \overleftarrow{\underline{\mathcal{J}}}^{k+1} \underline{\mathcal{R}}_{2i}^{k+1}) \quad (44)$$

$$= (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{R}}_{2i}^k, \quad (45)$$

where

$$\underline{\mathcal{R}}_{1i}^k = \underline{\mathcal{I}}_1^k \underline{\mathcal{P}}_i^k, \quad (46)$$

$$\underline{\mathcal{R}}_{2i}^k = \underline{\mathcal{R}}_{1i}^k + \overleftarrow{\underline{\mathcal{J}}}^{k+1} \underline{\mathcal{R}}_{2i}^{k+1}. \quad (47)$$

In a manner directly analogous to the recursive procedures demonstrated above, the matrix  $\underline{\mathcal{K}}$  can also be determined. With the shifting matrices so defined, the

system of forces and moments whose line of action passes through mass center of body  $k$  can be transformed to an equivalent systems of forces and moments whose line of action passes through a point of body  $k$  that is instantaneously coincident with the mass center of parent body  $k - 1$ . Thus, one obtains

$$\begin{aligned}\underline{\mathcal{K}}_{n-1} &= (\underline{\mathcal{P}}_{n-1}^{n-1})^T \underline{\mathcal{F}}_1^{n-1} + (\underline{\mathcal{P}}_{n-1}^n) \underline{\mathcal{F}}_1^n \\ &= (\underline{\mathcal{P}}_{n-1}^{n-1})^T (\underline{\mathcal{P}}_1^{n-1} + \underline{\mathcal{J}}^n \underline{\mathcal{F}}_1^n).\end{aligned}\quad (48)$$

The general expression for  $\underline{\mathcal{K}}_k$  is

$$\underline{\mathcal{K}}_k = (\underline{\mathcal{P}}_k^k)^T (\underline{\mathcal{F}}_1^k + \underline{\mathcal{J}}^n \underline{\mathcal{F}}_1^{k+1}) \quad (49)$$

$$= (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{F}}_2^k, \quad (50)$$

with  $\underline{\mathcal{F}}_2^k$  defined as

$$\underline{\mathcal{F}}_2^k = \underline{\mathcal{F}}_1^k + \underline{\mathcal{J}}^{k+1} \underline{\mathcal{F}}_1^{k+1}. \quad (51)$$

The procedures indicated by Equations (45) and (50) offer considerable computational savings over those of Equations (25) and (26). Equations (45) and (50) indicate that producing individual element of  $\underline{\mathcal{M}}_{r,i}$  and  $\underline{\mathcal{K}}_r$  each requires four  $3 \times 3$  sub-matrix multiplications. By comparison, Equation (25) requires  $4n$  multiplications to compute  $\underline{\mathcal{M}}_{r,i}$  while  $2n$  multiplications for  $\underline{\mathcal{K}}_r$  are needed in Equation (26). Thus, employing the recursive scheme permits  $\underline{\mathcal{M}}$  to be filled in  $O(n^2)$  operations, and  $\underline{\mathcal{K}}$  may be filled in  $O(n)$ . Likewise, direct differentiating elements of  $\underline{\mathcal{M}}_{r,i}$  and  $\underline{\mathcal{K}}_r$  can benefit from the kinematic recursive relations to yield derivatives in a more efficient manner.

#### 4.2. FULLY RECURSIVE SOLUTION SCHEME

If one chooses to simply proceed along the lines indicated by Equations (40–51), then one is left with what the authors term a *partial recursive* sensitivity scheme, and can achieve  $O(n^2)$  computational performance in producing the essential partial derivatives associated with matrices  $\underline{\mathcal{M}}$  and  $\underline{\mathcal{K}}$ . This procedure is presented in Appendix A and does not take full advantage of all useful recursive relationships and operations available. The result is a formulation which is similar, but not identical, to that derived by Tak [26]. Due to their similarities, both approaches suffer from the same shortcomings. Specifically, in both cases the form of sensitivity equation (9) has been maintained, so one is still left with the requirement (and expense) of explicit procedures for matrix multiplications and decomposition. These matrix manipulations result in  $O(n^3)$  computational requirements per sensitivity value, which may still be too expensive for application to large systems. As a means of circumventing these shortcomings, a *fully recursive* procedure is proposed. The central aspect of the fully recursive approach to be presented is the formulation

of equations and solution for first-order sensitivity values without ever having to explicitly form  $\underline{\mathcal{M}}$ ,  $\underline{\mathcal{K}}$ , or their derivatives.

Featherstone [12], and others have demonstrated that the generalized acceleration  $\underline{\ddot{q}}$  can be solved for in  $O(n)$  operation without explicitly forming  $\underline{\mathcal{M}}$  and  $\underline{\mathcal{K}}$ . Such a philosophy can significantly reduce the cost associated with modeling large systems. The solution scheme is to recursively perform the implicit inward triangularization associated with the decomposition of the system of equations and the outward back-substitution for obtaining  $\underline{\ddot{q}}_r$ . Following this concept, the procedure begins from with the terminal body once the recursive outward sweep given by Equations (33–36) to calculate kinematic quantities has been performed.

At the terminal body, the scalar representation of the generalized acceleration matrix can be written as

$$\underline{\bar{\mathcal{A}}}^n = (\underline{\mathcal{J}}^n)^T \underline{\bar{\mathcal{A}}}^{n-1} + \underline{\mathcal{P}}_n \ddot{q}_n, \quad (52)$$

where  $\underline{\mathcal{A}} = \underline{\bar{\mathcal{A}}} + \underline{\mathcal{A}}_r$ , and  $\underline{\bar{\mathcal{A}}}$  contains only those accelerations terms which are explicit in the unknown  $\underline{\ddot{q}}$ 's. Equation (52) allows one to isolate  $\underline{\ddot{q}}_n$  for recursive substitution in the remaining development. Kane's dynamical equations [18] associated with this terminal degree of freedom may be written as

$$\begin{aligned} F_n + F_n^* = 0 &= (\underline{\mathcal{P}}_n^T)^T [\underline{\mathcal{J}}_1^n \underline{\mathcal{A}}^n - \underline{\mathcal{F}}_1^n] \\ &= (\underline{\mathcal{P}}_n^T)^T [\underline{\mathcal{J}}_1^n (\underline{\mathcal{J}}^n)^T \underline{\bar{\mathcal{A}}}^{n-1} - \underline{\mathcal{F}}_1^n] + \mathcal{M}_{nn} \ddot{q}_n \end{aligned} \quad (53)$$

from which the generalized acceleration  $\underline{\ddot{q}}_n$  is acquired as

$$\ddot{q}_n = \frac{(\underline{\mathcal{P}}_n^T)^T}{\mathcal{M}_{nn}} [\underline{\mathcal{F}}_1^n - \underline{\mathcal{J}}_1^n (\underline{\mathcal{J}}^n)^T \underline{\bar{\mathcal{A}}}^{n-1}]. \quad (54)$$

Proceeding inward to  $n$ 's parent body  $n - 1$ , one has

$$\begin{aligned} F_{n-1} + F_{n-1}^* &= 0 \\ &= (\underline{\mathcal{P}}_{n-1}^T)^T [\underline{\mathcal{J}}_1^{n-1} \underline{\bar{\mathcal{A}}}^{n-1} - \underline{\mathcal{F}}_1^{n-1}] + (\underline{\mathcal{P}}_{n-1}^T)^T [\underline{\mathcal{J}}_1^n \underline{\bar{\mathcal{A}}}^n - \underline{\mathcal{F}}_1^n] \\ &= (\underline{\mathcal{P}}_{n-1}^T)^T \left( [\underline{\mathcal{J}}_1^{n-1} + \underline{\mathcal{J}}^n \underline{\mathcal{J}}_1^n (\underline{\mathcal{J}}^n)^T - \frac{1}{\mathcal{M}_{nn}} \underline{\mathcal{J}}^n \underline{\mathcal{J}}_1^n \underline{\mathcal{P}}_n^T (\underline{\mathcal{J}}^n \underline{\mathcal{J}}_1^n \underline{\mathcal{P}}_n^T)^T] \underline{\bar{\mathcal{A}}}^{n-1} \right. \\ &\quad \left. - \left[ \underline{\mathcal{F}}_1^{n-1} + \underline{\mathcal{J}}^n \underline{\mathcal{F}}_1^n - \frac{1}{\mathcal{M}_{nn}} \underline{\mathcal{J}}^n \underline{\mathcal{J}}_1^n \underline{\mathcal{P}}_n^T (\underline{\mathcal{P}}_n^T)^T \underline{\mathcal{F}}_1^n \right] \right) \\ &= (\underline{\mathcal{P}}_{n-1}^T)^T (\underline{\mathcal{J}}_3^{n-1} \underline{\bar{\mathcal{A}}}^{n-1} - \underline{\mathcal{F}}_3^{n-1}), \end{aligned} \quad (55)$$

or for a general body  $k$

$$F_k + F_k^* = (\underline{\mathcal{P}}_k^T)^T (\underline{\mathcal{J}}_3^k \underline{\bar{\mathcal{A}}}^k - \underline{\mathcal{F}}_3^k) = 0, \quad (56)$$

where the matrices  $\underline{\mathcal{J}}_3^k$ ,  $\underline{\mathcal{F}}_3^k$ , and  $\mathcal{M}_{kk}$  are recursively defined as

$$\begin{aligned} \underline{\mathcal{J}}_3^k &= \underline{\mathcal{J}}_1^k + (\underline{\mathcal{J}}^{k+1} \underline{\mathcal{J}}_3^{k+1} (\underline{\mathcal{J}}^{k+1})^T \\ &\quad - \frac{1}{\mathcal{M}_{k+1k+1}} [\underline{\mathcal{J}}^{k+1} \underline{\mathcal{J}}_3^{k+1} \underline{\mathcal{P}}_{k+1}^{k+1} (\underline{\mathcal{J}}^{k+1} \underline{\mathcal{J}}_3^{k+1} \underline{\mathcal{P}}_{k+1}^{k+1})^T]), \end{aligned} \quad (57)$$

$$\begin{aligned} \underline{\mathcal{F}}_3^k &= \underline{\mathcal{F}}_1^k + (\underline{\mathcal{J}}^{k+1} \underline{\mathcal{F}}_3^{k+1} \\ &\quad - \frac{1}{\mathcal{M}_{k+1k+1}} [\underline{\mathcal{J}}^{k+1} \underline{\mathcal{J}}_3^{k+1} \underline{\mathcal{P}}_{k+1}^{k+1} (\underline{\mathcal{P}}_{k+1}^{k+1})^T \underline{\mathcal{F}}_3^{k+1}]) \end{aligned} \quad (58)$$

$$\mathcal{M}_{kk} = (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{J}}_3^k \underline{\mathcal{P}}_k^k. \quad (59)$$

Proceeding in this manner the equations of motion are effectively put in lower triangular form as they are being built. Once the base body is reached, information associated with the entire set of outboard bodies has been accumulated and is implicitly available such that the equation  $\underline{\mathcal{M}}_{11} \ddot{q}_1 = (\underline{\mathcal{P}}_1^1)^T \underline{\mathcal{F}}_3^1$  can be isolated and solved for  $\ddot{q}_1$ . At this point the method reverses direction and again works outward from the system base body to the terminal, recursively performing a backsubstitution via the relations

$$\ddot{q}_k = \frac{(\underline{\mathcal{P}}_k^k)^T}{\mathcal{M}_{kk}} [\underline{\mathcal{F}}_3^k - \underline{\mathcal{J}}_3^k (\underline{\mathcal{J}}^k)^T \underline{\mathcal{A}}^{k-1}] \quad (60)$$

and

$$\underline{\mathcal{A}}^k = (\underline{\mathcal{J}}^k)^T \underline{\mathcal{A}}^{k-1} + \underline{\mathcal{P}}_k^k \ddot{q}_k. \quad (61)$$

Equations (60) and (61) clearly indicate that solving for each  $\ddot{q}_i$  ( $i = 1, \dots, n$ ) can be performed at a fixed cost due to the recursive manipulations. Such a solution scheme does not require forming the entire system matrix  $\underline{\mathcal{M}}$  explicitly. Instead, each off diagonal element of row  $k$  for the triangularized form of this matrix is implicitly manipulated as the diagonal element  $\mathcal{M}_{kk}$  is formed, eliminating many operations.

This recursive solution scheme can be extended to the determination of first-order sensitivity values equally well. Specifically, the triangularization process is conducted first to express each of  $d\ddot{q}_i/dp_j$  in terms of  $d\ddot{q}_{i-1}/dp_j$ . When the procedure reaches the base body,  $d\ddot{q}_1/dp_j$  is solved for explicitly, and is then used to begin the recursive back-substitution which follows to yield all  $d\ddot{q}_i/dp_j$ .

At the terminal body  $k = n$ , the articulated body inertia mass matrix  ${}^n \underline{\mathcal{J}}_3^n$  and the articulated body generalized forces matrix  ${}^n \underline{\mathcal{F}}_3^n$  [12] involve only one term so their total derivatives yield

$$\frac{d}{dp_j} \underline{\mathcal{J}}_3^n = \frac{d}{dp_j} \underline{\mathcal{J}}_2^n = \frac{d}{dp_j} \underline{\mathcal{J}}_1^n \quad \text{and} \quad \frac{d}{dp_j} \underline{\mathcal{F}}_3^n = \frac{d}{dp_j} \underline{\mathcal{F}}_1^n. \quad (62)$$

The derivative of  $d^n \underline{\mathcal{J}}_3^n / dp_j$  allows the calculation of  $d\mathcal{M}_{nn} / dp_j$  from Equation (59), specifically,

$$\begin{aligned} \frac{d}{dp_j} \mathcal{M}_{nn} &= \left[ \frac{d}{dp_j} (\underline{\mathcal{P}}_n)^T \right] \underline{\mathcal{J}}_3^n \underline{\mathcal{P}}_n + (\underline{\mathcal{P}}_n)^T \left[ \frac{d}{dp_j} \underline{\mathcal{J}}_3^n \right] \underline{\mathcal{P}}_n \\ &\quad + (\underline{\mathcal{P}}_n)^T \underline{\mathcal{J}}_3^n \left[ \frac{d}{dp_j} \underline{\mathcal{P}}_n \right]. \end{aligned} \quad (63)$$

Working inward to body  $n - 1$  yields the total derivatives of  $d\underline{\mathcal{J}}_3^{n-1} / dp_j$  and  $d\underline{\mathcal{F}}_3^{n-1} / dp_j$ , namely

$$\begin{aligned} \frac{d}{dp_j} \underline{\mathcal{J}}_3^{n-1} &= \frac{d}{dp_j} \underline{\mathcal{J}}_1^{n-1} + \frac{d}{dp_j} \left( \underline{\mathcal{J}}_3^n \underline{\mathcal{J}}_3^n (\underline{\mathcal{J}}_3^n)^T \right. \\ &\quad \left. - \frac{1}{\mathcal{M}_{nn}} [\underline{\mathcal{J}}_3^n \underline{\mathcal{J}}_3^n \underline{\mathcal{P}}_n (\underline{\mathcal{J}}_3^n \underline{\mathcal{J}}_3^n \underline{\mathcal{P}}_n)^T] \right), \end{aligned} \quad (64)$$

$$\begin{aligned} \frac{d}{dp_j} \underline{\mathcal{F}}_3^{n-1} &= \frac{d}{dp_j} \underline{\mathcal{F}}_1^{n-1} + \frac{d}{dp_j} \left( \underline{\mathcal{J}}_3^n \underline{\mathcal{F}}_3^n \right. \\ &\quad \left. - \frac{1}{\mathcal{M}_{nn}} [\underline{\mathcal{J}}_3^n \underline{\mathcal{J}}_3^n \underline{\mathcal{P}}_n (\underline{\mathcal{P}}_n)^T \underline{\mathcal{F}}_3^n] \right). \end{aligned} \quad (65)$$

So  $d\mathcal{M}_{n-1n-1} / dp_j$  is now available. Examination of Equations (64) and (65) indicate that a constant operation count for differentiation is achievable due to the fixed dimension of each matrix involved. Similarly, the derivatives of each quantity with respect to the generalized coordinates and velocities can be performed at a fixed cost due to the local differentiation property. Ideally, the derivative operation performed on each body is of  $O(1)$  which then leads to an  $O(n)$  overall cost for the triangularization process.

The triangularization procedures are performed recursively inward until the base body is reached, leaving a single function of  $\ddot{q}_1$

$$\ddot{q}_1 = \frac{(\underline{\mathcal{P}}_1^1)^T \underline{\mathcal{F}}_1^1}{\mathcal{M}_{11}}, \quad (66)$$

from which one is able to compute the derivative of  $\ddot{q}_1$  with respect to  $p_j$

$$\frac{d}{dp_j} \ddot{q}_1 = \frac{d}{dp_j} \left[ \frac{(\underline{\mathcal{P}}_1^1)^T \underline{\mathcal{F}}_1^1}{\mathcal{M}_{11}} \right]. \quad (67)$$

At this point in the implementation (as with Equation (60)) the procedure changes direction and moves outward from base body to terminal body to perform the sensitivity back-substitution. At body  $k = 2$ , the information of  $d\underline{\mathcal{A}}_1^{-1} / dp_j$  has

to be computed in order to obtain the solution of  $d\ddot{q}_2/dp_j$ . From Equation (52), one obtains

$$\frac{d}{dp_j} \bar{\mathcal{A}}^1 = \left[ \frac{d}{dp_j} \mathcal{P}_1^1 \right] \ddot{q}_1 + \mathcal{P}_1^1 \left[ \frac{d}{dp_j} \ddot{q}_1 \right]. \quad (68)$$

Substituting the result obtained from Equation (68) into the differentiated form of Equation (60) yields the desired  $d\ddot{q}_2/dp_j$ . The generalized procedure for the sensitivity back-substitution is carried out in an orderly manner through the two equations

$$\frac{d}{dp_j} \ddot{q}_k = \frac{d}{dp_j} \left[ \frac{(\mathcal{P}_k^k)^T}{\mathcal{M}_{kk}} (\mathcal{F}_3^k - \mathcal{L}_3^k (\bar{\mathcal{G}}^k)^T \bar{\mathcal{A}}^{k-1}) \right], \quad (69)$$

$$\frac{d}{dp_j} \bar{\mathcal{A}}^k = \frac{d}{dp_j} \left[ (\bar{\mathcal{G}}^k)^T \bar{\mathcal{A}}^{k-1} + \mathcal{P}_k^k \ddot{q}_k \right]. \quad (70)$$

To this end, the key quantities  $d\ddot{q}_k/dp_j$  in the first-order sensitivity analysis are determined in a fully recursive manner. Inspecting the back-substitution procedure as indicated in Equations (69) and (70) one observes that neither the number of terms nor the required number of operation increase as the procedures proceed. Consequently, the triangularization and back-substitution processes yields an  $O(n)$  operation performance overall for one design variable. For an entire set of  $p$  design variables, the proposed solution scheme is able to achieve an  $O(pn)$  operation performance overall for obtaining all solutions associated with the  $p$  sets of sensitivity coefficient equations.

## 5. Numerical Examples

To demonstrate the characteristics of the presented multibody dynamics systems first-order sensitivity algorithm, this section focuses on its fundamental performance and compares its solution accuracy with that obtained by more traditional direct differentiation approaches. The dynamic analysis and simulation software AUTOLEV [22], as well as the symbolic manipulation package MAPLE were employed to perform equation formulations and differentiation operations. For these numerical comparisons, the same coding strategies, integration procedures, very limited I/O, etc., were employed so that the resulting principal difference in the simulation/analysis codes was in the underlying formulations used. Thus, these code's raw performances can be better compared.

A spatial chain system is used to compare the computational performance of the presented algorithm with a more traditional approaches based on a more conventional  $O(n^3)$  dynamics algorithm. As illustrated in Figure 2, the required CPU-time for the presented algorithm to perform sensitivity analysis increases linearly as the number of system degrees of freedom  $n$  increase. This result directly indicates that the algorithm is indeed an  $O(n)$  method. When  $n$  is small ( $\sim \leq 4$  for the

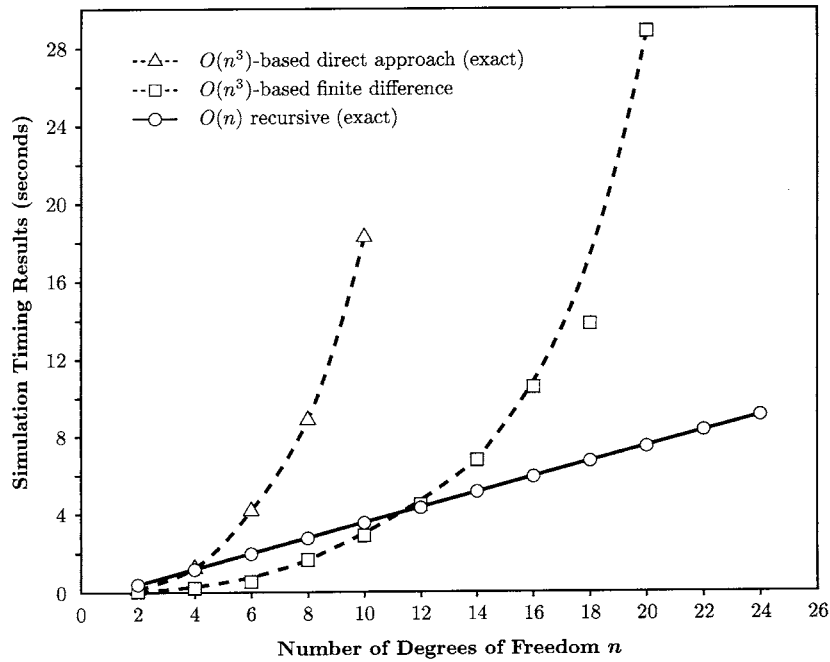


Figure 2. Comparison of experimentally obtained CPU times for sensitivity analysis.

chain systems investigated), superior (faster) performance for the determination of sensitivity information will likely be achieved using the  $O(n^3)$  based methodologies, which actually perform as  $O(n^4)$  overall per sensitivity value. This is due to the relatively small coefficient these methods have associated with their  $n^4$  cost term, which results in comparatively low overall costs for small values of  $n$ . However, as  $n$  becomes large, the approaches based upon the  $O(n)$  formulation become significantly faster than the  $O(n^3)$  based approaches for performing the first-order sensitivity analysis. This relative performance gain arises from the fact that the computational costs associated with the  $O(n)$  based methods increase much less rapidly with  $n$ , than for the  $O(n^3)$  based  $O(n^4)$  methods. Consequently, one can conclude that the presented  $O(n)$  sensitivity algorithm is superior than the traditional  $O(n^3)$  based approaches when dealing with large dynamic systems.

A double pendulum, similar to the single pendulum example presented in [24], is provided as another example to demonstrate the validity of the presented method. The double pendulum shown in Figure 3 is modeled as a two degrees of freedom system and is restricted to the planar motion. The system properties are defined as in Figure 3. The system is acted upon by gravity  $g$ , but is free from all other applied or friction forces.

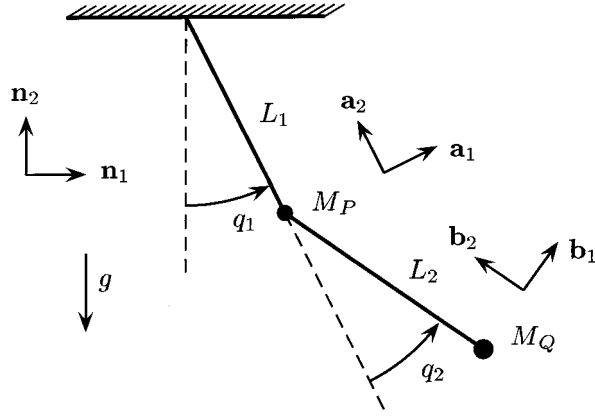


Figure 3. A planar double pendulum.

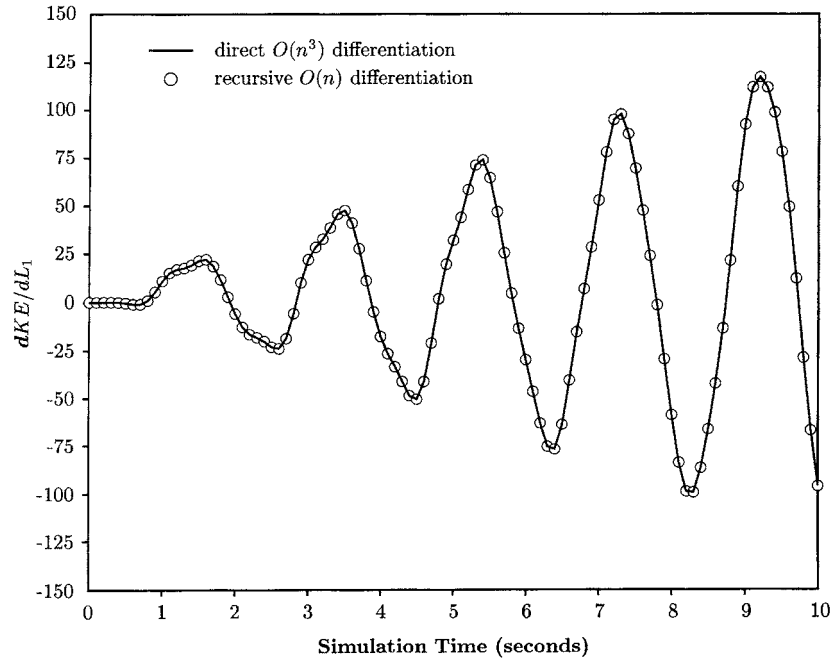


Figure 4. Sensitivity time history for  $\delta L_1$ .

The objective of this example is to compute the sensitivity coefficients of the system kinetic energy to perturbations of the set of variables  $[L_1, L_2, M_P, M_Q]$ . The system kinetic energy is expressed as

$$\text{K.E.} = \frac{1}{2}(M_P {}^N\mathbf{V}_{M_P} \cdot {}^N\mathbf{V}_{M_P} + M_Q {}^N\mathbf{V}_{M_Q} \cdot {}^N\mathbf{V}_{M_Q}). \quad (71)$$

The first-order sensitivity information associated with each variable is then calculated using: a traditional  $O(pn^4)$  method as described in Section 3.1; a straight

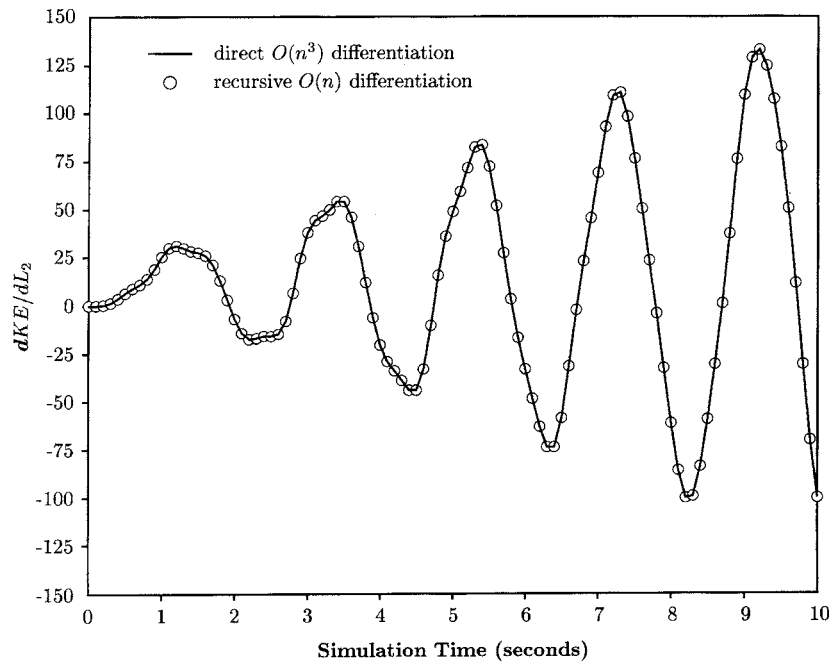


Figure 5. Sensitivity time history for  $\delta L_2$ .

finite difference approximation; and finally using the ‘fully recursive’ procedure presented in Section 4.2. The magnitude of perturbation is  $1.0 \times 10^{-5}$  for variables  $L_1$ ,  $L_2$ , and  $M_Q$  where  $5.0 \times 10^{-4}$  is used for  $M_p$ . Figures 4–7 plot the time history of the computed first-order sensitivity with respect to each design variable. These figures clearly indicated a good solution agreement between the presented  $O(n)$  method and the other two approaches. Additionally, the presented  $O(n)$  method does not exhibit numerical instability during an entire 20 seconds simulation and being analytically exact, does not suffer from sensitivity to the magnitude of the parameter variations, integration scheme characteristics, or system stiffness which the finite difference approaches commonly exhibit. Both the solution accuracy and numerical stability provide evidence for the proposed method to be a practical, viable sensitivity analysis tool.

## 6. Conclusions

A fully recursive sensitivity analysis method is presented for the determination of first-order sensitivity information for the design of multibody dynamic systems. The use of recursive relationships coupled with the local representations of dynamic quantities significantly eliminates many matrix manipulations and derivative operations exhibited with traditional direct differentiation approaches involving more conventional  $O(n^3)$  state space formulations. The provided numerical exam-

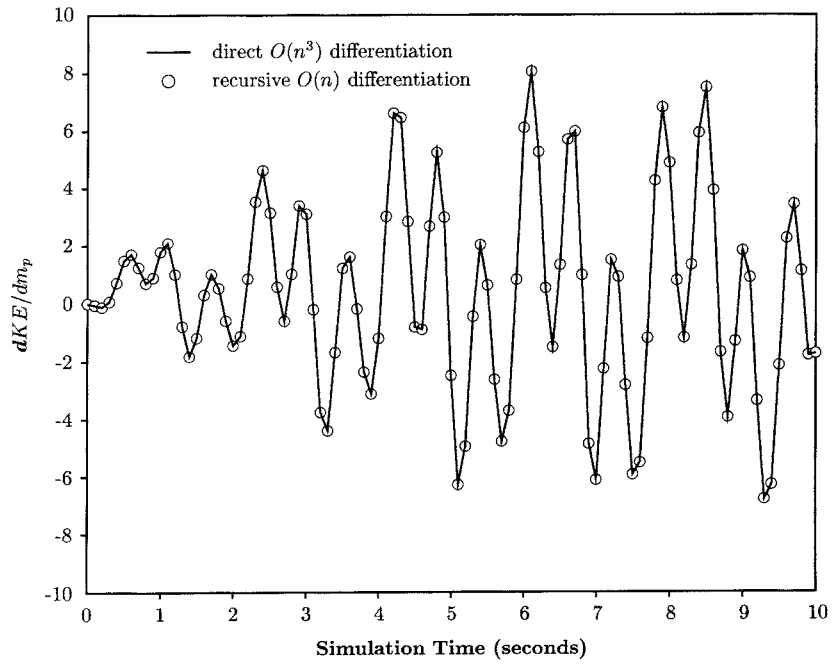


Figure 6. Sensitivity time history for  $\delta M_p$ .

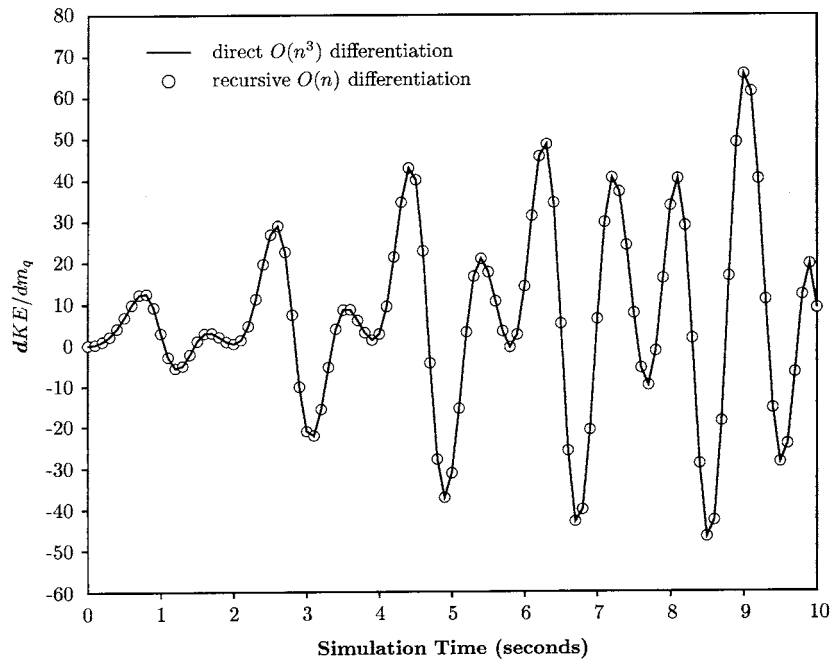


Figure 7. Sensitivity time history for  $\delta M_q$ .

ples demonstrate the solution accuracy, method validity and stability. Indeed, the  $O(n)$  based approach presented obtains all sensitivity terms associated with  $p$  design variables and  $n$  generalized coordinates in  $O(pn)$  operations overall, compared to the  $O(pn^4)$  performance which is realized by the straight direct differentiation approach applied to a conventional  $O(n^3)$  state-space dynamics formulation.

### Appendix A: Recursive Sensitivity Formulations

Recursive sensitivity formulations procedures, which are logical extensions of the relationships presented in Section 3, do not necessarily utilize all potentially beneficial recursive relationships available. Indeed, the Fully Recursive Sensitivity Algorithm presented in Section 4.2 differs markedly from other ‘Recursive Sensitivity Algorithms’ which predated it, in their recursive steps. As with the fully recursive procedure, the first set of outward calculations recursively works from the system base body out to the terminal body, determining kinematic quantities and their derivatives. To this point, the presented  $O(n)$  first-order sensitivity method and the recursive method as presented by Tak [26], are highly similar. However, from this juncture the methods deviate greatly. Instead of triangularizing the sensitivity equations as they are being formed, one may instead work inward recursively generating partial derivatives of kinetic quantities and assemble matrix elements.

According to Equation (37), the following partial derivative relationships exist

$$\underline{\mathcal{P}}_{k-1,p_j}^k = (\underline{\mathcal{J}}^k)_{,p_j}^T \underline{\mathcal{P}}_{k-1}^{k-1} + (\underline{\mathcal{J}}^k)^T (\underline{\mathcal{P}}_{k-1}^{k-1})_{,p_j}^T, \quad (72)$$

$$\underline{\mathcal{P}}_{k-1,q_r}^k = (\underline{\mathcal{J}}^k)_{,q_r}^T \underline{\mathcal{P}}_{k-1,q_r}^{k-1} + (\underline{\mathcal{J}}^k)^T (\underline{\mathcal{P}}_{k-1,q_r}^{k-1})_{,q_r}^T, \quad (73)$$

$$\underline{\mathcal{P}}_{k-1,\dot{q}_r}^k = (\underline{\mathcal{J}}^k)_{,\dot{q}_r}^T \underline{\mathcal{P}}_{k-1,\dot{q}_r}^{k-1} + (\underline{\mathcal{J}}^k)^T (\underline{\mathcal{P}}_{k-1,\dot{q}_r}^{k-1})_{,\dot{q}_r}^T. \quad (74)$$

The procedure then proceeds recursively inward from terminal body to base body in order to formulate each element in the matrices  $\underline{\mathcal{M}}_{,p_j}$ ,  $\underline{\mathcal{M}}_{,q_r}$ ,  $\underline{\mathcal{K}}_{,p_j}$ ,  $\underline{\mathcal{K}}_{,q_r}$ , and  $\underline{\mathcal{K}}_{,\dot{q}_r}$ . At the terminal body  $k = n$ , partial derivative of Equation (39) with respect to design variable  $p_j$  yields

$$\underline{\mathcal{M}}_{ns,p_j} = (\underline{\mathcal{P}}_n^T)_{,p_j} \underline{\mathcal{J}}_1^n \underline{\mathcal{P}}_s^n + (\underline{\mathcal{P}}_n^T) \underline{\mathcal{J}}_{1,p_j}^n \underline{\mathcal{P}}_s^n + \underline{\mathcal{P}}_n^n \underline{\mathcal{J}}_1^n \underline{\mathcal{P}}_{s,p_j}^n. \quad (75)$$

Since  $\underline{\mathcal{P}}_{i,p_j}^n$  ( $i = 1, \dots, n$ ) have all been determined previously from the outward recursive procedures,  $\underline{\mathcal{J}}_{1,p_j}^n$  is the only term which still needs to be computed. The expression of  $\underline{\mathcal{J}}_{1,p_j}^n$  can be simply written as

$$\underline{\mathcal{J}}_{1,p_j}^n \equiv \frac{\partial}{\partial p_j} \underline{\mathcal{J}}_1^n = \begin{bmatrix} \underline{\mathcal{J}}_{,p_j}^{n/n*} & \underline{\mathbf{0}} \\ \underline{\mathbf{0}} & \underline{\mathcal{M}}_{,p_j}^n \end{bmatrix}. \quad (76)$$

The local differentiation ensures that the term  $\underline{\mathcal{J}}_{1,p_j}^n$  will vanish if  $p_j$  is neither a geometric quantity manifest in an inertia scalar, nor a mass property. Additionally,

both  $\underline{\mathbf{l}}_{1,q_r}^n$  and  $\underline{\mathbf{l}}_{1,\dot{q}_r}^n$  are equal to zero, since  $\underline{\mathbf{l}}_1^n$  is not a function of  $q_r$ , or  $\dot{q}_r$ . Therefore, the computational order associated with the derivatives of inertia mass matrices reduces to  $O(1)$ . By working inward to body  $n - 1$  one obtains

$$\begin{aligned}
\underline{\mathcal{M}}_{(n-1)s,p_j} &= [(\underline{\mathcal{P}}_{n-1}^{n-1})^T_{,p_j} \underline{\mathbf{l}}_1^{n-1} \underline{\mathcal{P}}_s^{n-1} + (\underline{\mathcal{P}}_{n-1}^{n-1})^T \underline{\mathbf{l}}_{1,p_j}^{n-1} \underline{\mathcal{P}}_s^{n-1} \\
&\quad + (\underline{\mathcal{P}}_{n-1}^{n-1})^T \underline{\mathbf{l}}_1^{n-1} \underline{\mathcal{P}}_{s,p_j}^{n-1}] + [(\underline{\mathcal{P}}_{n-1}^n)^T_{,p_j} \underline{\mathbf{l}}_1^n \underline{\mathcal{P}}_s^n \\
&\quad + (\underline{\mathcal{P}}_{n-1}^n)^T \underline{\mathbf{l}}_{1,p_j}^n \underline{\mathcal{P}}_s^n + (\underline{\mathcal{P}}_{n-1}^n)^T \underline{\mathbf{l}}_1^n \underline{\mathcal{P}}_{s,p_j}^n], \\
&= (\underline{\mathcal{P}}_{n-1}^{n-1})^T_{,p_j} (\underline{\mathbf{l}}_1^{n-1} \underline{\mathcal{P}}_s^{n-1} + \overleftarrow{\underline{\mathbf{g}}}^n \underline{\mathbf{l}}_1^n \underline{\mathcal{P}}_s^n) \\
&\quad + (\underline{\mathcal{P}}_{n-1}^{n-1})^T (\underline{\mathbf{l}}_{1,p_j}^{n-1} \underline{\mathcal{P}}_s^{n-1} + \overleftarrow{\underline{\mathbf{g}}}^n \underline{\mathbf{l}}_{1,p_j}^n \underline{\mathcal{P}}_s^n) \\
&\quad + (\underline{\mathcal{P}}_{n-1}^{n-1})^T (\underline{\mathbf{l}}_1^{n-1} \underline{\mathcal{P}}_{s,p_j}^{n-1} + \overleftarrow{\underline{\mathbf{g}}}^n \underline{\mathbf{l}}_1^n \underline{\mathcal{P}}_{s,p_j}^n) \\
&\quad + (\underline{\mathcal{P}}_{n-1}^{n-1})^T \overleftarrow{\underline{\mathbf{g}}}^n_{,p_j} \underline{\mathbf{l}}_1^n \underline{\mathcal{P}}_s^n. \tag{77}
\end{aligned}$$

Therefore, recursive relations yield a concise expression for the derivative of a general element  $\underline{\mathcal{M}}_{ki}$  with respect to  $p_j$ , specifically

$$\begin{aligned}
\underline{\mathcal{M}}_{ki,p_j} &= (\underline{\mathcal{M}}_k^k)^T_{,p_j} [\underline{\mathcal{R}}_{1i}^k + \overleftarrow{\underline{\mathbf{g}}}^{k+1} \underline{\mathcal{R}}_{2i}^{k+1}] \\
&\quad + (\underline{\mathcal{P}}_k^k)^T [\underline{\mathcal{R}} \underline{\mathbf{l}}_{1i}^k + \overleftarrow{\underline{\mathbf{g}}}^{k+1} \underline{\mathcal{R}} \underline{\mathbf{l}}_{2i}^{k+1}] \\
&\quad + (\underline{\mathcal{P}}_k^k)^T [\underline{\mathcal{R}} \underline{\mathcal{P}}_{1i}^k + \overleftarrow{\underline{\mathbf{g}}}^{k+1} \underline{\mathcal{R}} \underline{\mathcal{P}}_{2i}^{k+1}] \\
&\quad + (\underline{\mathcal{P}}_k^k)^T \left[ \sum_{j=k+1}^n \left( \prod_{\alpha=k+1}^j \overleftarrow{\underline{\mathbf{g}}}_{,p_j}^\alpha \right) (\underline{\mathcal{R}}_1^j + \overleftarrow{\underline{\mathbf{g}}}^{j+1} \underline{\mathcal{R}}_2^{j+1}) \right] \tag{78}
\end{aligned}$$

$$\begin{aligned}
&= (\underline{\mathcal{P}}_k^k)^T_{,p_j} \underline{\mathcal{R}}_{2i}^k + (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{R}} \underline{\mathbf{l}}_{2i}^k \\
&\quad + (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{R}} \underline{\mathcal{P}}_{2i}^k + (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{R}} \underline{\mathcal{R}} \underline{\mathcal{P}}_2^k, \tag{79}
\end{aligned}$$

where

$$\begin{cases} \underline{\mathcal{R}}_{1i}^k = \underline{\mathbf{l}}_1^k \underline{\mathcal{P}}_i^k, \\ \underline{\mathcal{R}}_{2i}^k = \underline{\mathcal{R}}_{1i}^k + \overleftarrow{\underline{\mathbf{g}}}^{k+1} \underline{\mathcal{R}}_{2i}^{k+1}, \end{cases} \tag{80}$$

$$\begin{cases} \underline{\mathcal{R}} \underline{\mathbf{l}}_{1i}^k = \underline{\mathbf{l}}_{1,p_j}^k \underline{\mathcal{P}}_i^k, \\ \underline{\mathcal{R}} \underline{\mathbf{l}}_{2i}^k = \underline{\mathcal{R}} \underline{\mathbf{l}}_{1i}^k + \overleftarrow{\underline{\mathbf{g}}}^{k+1} \underline{\mathcal{R}} \underline{\mathbf{l}}_{2i}^{k+1}, \end{cases} \tag{81}$$

$$\begin{cases} \underline{\mathcal{R}} \underline{\mathcal{P}}_{1i}^k = \underline{\mathbf{l}}_1^k \underline{\mathcal{P}}_{i,p_j}^k, \\ \underline{\mathcal{R}} \underline{\mathcal{P}}_{2i}^k = \underline{\mathcal{R}} \underline{\mathcal{P}}_{1i}^k + \overleftarrow{\underline{\mathbf{g}}}^{k+1} \underline{\mathcal{R}} \underline{\mathcal{P}}_{2i}^{k+1}, \end{cases} \tag{82}$$

$$\begin{cases} \prod_{\alpha=k+1}^j \underline{\mathcal{J}}_{,p_j}^{\leftarrow} = \underline{\mathcal{J}}^{\leftarrow k+1} \underline{\mathcal{J}}^{\leftarrow k+2} \underline{\mathcal{J}}^{\leftarrow k+3} \dots \underline{\mathcal{J}}^{\leftarrow j-1} \underline{\mathcal{J}}_{,p_j}^{\leftarrow j}, \\ \underline{\mathcal{R}}\underline{\mathcal{R}}\underline{\mathcal{P}}_2^k = \sum_{j=k+1}^n \left( \prod_{\alpha=k+1}^j \underline{\mathcal{J}}_{,p_j}^{\leftarrow} \right) (\underline{\mathcal{R}}_1^j + \underline{\mathcal{J}}^{\leftarrow j+1} \underline{\mathcal{R}}_2^{j+1}). \end{cases} \quad (83)$$

Similarly, one obtains the derivatives of  $\underline{\mathcal{M}}_{ki}$  with respect to the generalized coordinates  $q_r$

$$\begin{aligned} \underline{\mathcal{M}}_{ki,q_r} &= (\underline{\mathcal{P}}_k^k)^T_{,q_r} [\underline{\mathcal{R}}_{1i}^k + \underline{\mathcal{J}}^{\leftarrow k+1} \underline{\mathcal{R}}_{2i}^{k+1}] \\ &\quad + (\underline{\mathcal{P}}_k^k)^T [\underline{\mathcal{R}}\underline{\mathcal{Q}}_{1i}^k + \underline{\mathcal{J}}^{\leftarrow k_1} \underline{\mathcal{R}}\underline{\mathcal{Q}}_{2i}^{k+1}] \\ &\quad + (\underline{\mathcal{P}}_k^k)^T \left[ \sum_{j=k+1}^n \left( \prod_{\alpha=k+1}^j \underline{\mathcal{J}}_{,q_r}^{\leftarrow} \right) (\underline{\mathcal{R}}_1^j + \underline{\mathcal{J}}^{\leftarrow j+1} \underline{\mathcal{R}}_2^{j+1}) \right] \end{aligned} \quad (84)$$

$$= (\underline{\mathcal{P}}_k^k)^T_{,q_r} \underline{\mathcal{R}}_{2i}^k + (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{R}}\underline{\mathcal{Q}}_{2i}^k + (\underline{\mathcal{P}}_k^k)^T \underline{\mathcal{R}}\underline{\mathcal{R}}\underline{\mathcal{Q}}_2^k. \quad (85)$$

The recursive procedures illustrated above for deriving  $\underline{\mathcal{M}}_{ki,p_j}$  and  $\underline{\mathcal{M}}_{ki,q_r}$  can be applied to the  $\underline{\mathcal{K}}_k$  matrix equally well. For instance, at terminal body  $n$ , Equation (26) contains one term so its derivative can be written as

$$\underline{\mathcal{K}}_{n,p_j} = (\underline{\mathcal{P}}_r^k)^T_{,p_j} \underline{\mathcal{F}}_{-1}^k + (\underline{\mathcal{P}}_r^k)^T \underline{\mathcal{F}}_{-1,p_j}^k. \quad (86)$$

Working inward to body  $n-1$  yields

$$\begin{aligned} \underline{\mathcal{K}}_{n-1,p_j} &= [(\underline{\mathcal{P}}_{n-1}^{n-1})^T_{,p_j} \underline{\mathcal{F}}_{-1}^{n-1} + (\underline{\mathcal{P}}_{n-1}^{n-1})^T \underline{\mathcal{F}}_{-1,p_j}^{n-1}] \\ &\quad + [(\underline{\mathcal{P}}_{n-1}^n)^T_{,p_j} \underline{\mathcal{F}}_{-1}^n + (\underline{\mathcal{P}}_{n-1}^n)^T \underline{\mathcal{F}}_{-1,p_j}^n] \\ &= (\underline{\mathcal{P}}_{n-1}^{n-1})^T_{,p_j} (\underline{\mathcal{F}}_{-1}^{n-1} + \underline{\mathcal{J}}^{\leftarrow n} \underline{\mathcal{F}}_{-1}^n) \\ &\quad + (\underline{\mathcal{P}}_{n-1}^{n-1})^T (\underline{\mathcal{F}}_{-1,p_j}^{n-1} + \underline{\mathcal{J}}^{\leftarrow n} \underline{\mathcal{F}}_{-1,p_j}^n) + (\underline{\mathcal{P}}_{n-1}^{n-1})^T \underline{\mathcal{J}}^{\leftarrow n}_{,p_j} \underline{\mathcal{F}}_{-1}^n. \end{aligned} \quad (87)$$

A generalized formulation for the derivative of  $\underline{\mathcal{K}}_r$  with respect to  $p_j$  is thus obtained

$$\begin{aligned} \underline{\mathcal{K}}_{r,p_j} &= (\underline{\mathcal{P}}_r^r)^T_{,p_j} [\underline{\mathcal{F}}_{-1}^r + \underline{\mathcal{J}}^{\leftarrow r+1} \underline{\mathcal{F}}_{-2}^{r+1}] \\ &\quad + (\underline{\mathcal{P}}_r^r)^T [\underline{\mathcal{F}}\underline{\mathcal{P}}_{-1}^r + \underline{\mathcal{J}}^{\leftarrow r+1} \underline{\mathcal{F}}\underline{\mathcal{P}}_{-2}^{r+1}] \\ &\quad + (\underline{\mathcal{P}}_r^r)^T \left[ \sum_{j=k+1}^n \left( \prod_{\alpha=k+1}^j \underline{\mathcal{J}}_{,p_j}^{\leftarrow} \right) (\underline{\mathcal{F}}_{-1}^j + \underline{\mathcal{J}}^{\leftarrow j+1} \underline{\mathcal{F}}_{-2}^{j+1}) \right] \end{aligned} \quad (88)$$

$$= (\underline{\mathcal{P}}_r^r)^T_{,p_j} \underline{\mathcal{F}}_{-2}^r + (\underline{\mathcal{P}}_r^r)^T \underline{\mathcal{F}}\underline{\mathcal{P}}_{-2}^r + (\underline{\mathcal{P}}_r^r)^T \underline{\mathcal{F}}\underline{\mathcal{R}}\underline{\mathcal{P}}_{-2}^r, \quad (89)$$

where, by definitions,

$$\underline{\mathcal{F}}_2^r = \underline{\mathcal{F}}_1^r + \underline{\delta}^{\leftarrow r+1} \underline{\mathcal{F}}_2^{r+1}, \quad (90)$$

$$\underline{\mathcal{F}} \underline{\mathcal{P}}_2^r = \underline{\mathcal{F}} \underline{\mathcal{P}}_1^r + \underline{\delta}^{\leftarrow r+1} \underline{\mathcal{F}} \underline{\mathcal{P}}_2^{r+1}, \quad (91)$$

$$\begin{cases} \prod_{\alpha=k+1}^j \underline{\delta}_{,p_j}^{\leftarrow} = \underline{\delta}^{\leftarrow k+1} \underline{\delta}^{\leftarrow k+2} \underline{\delta}^{\leftarrow k+3} \dots \underline{\delta}^{\leftarrow j-1} \underline{\delta}_{,p_j}^{\leftarrow j}, \\ \underline{\mathcal{F}} \underline{\mathcal{R}} \underline{\mathcal{P}}_2^r = \sum_{j=r+1}^n \left( \prod_{\alpha=r+1}^j \underline{\delta}_{,p_j}^{\leftarrow} \right) (\underline{\mathcal{F}}_1^j + \underline{\delta}^{\leftarrow j+1} \underline{\mathcal{F}}_2^{j+1}). \end{cases} \quad (92)$$

Likewise, the derivative of  $\underline{\mathcal{K}}$  with respect to  $q_r$  and  $\dot{q}_r$  can be obtained

$$\begin{aligned} \underline{\mathcal{K}}_{r,q_r} &= (\underline{\mathcal{P}}_r^r)^T_{,q_r} [\underline{\mathcal{F}}_1^r + \underline{\delta}^{\leftarrow r+1} \underline{\mathcal{F}}_2^{r+1}] + (\underline{\mathcal{P}}_r^r)^T [\underline{\mathcal{F}} \underline{\mathcal{Q}}_1^r + \underline{\delta}^{\leftarrow r+1} \underline{\mathcal{F}} \underline{\mathcal{Q}}_2^{r+1}] \\ &\quad + (\underline{\mathcal{P}}_r^r)^T \left[ \sum_{j=k+1}^n \left( \prod_{\alpha=k+1}^j \underline{\delta}_{,q_r}^{\leftarrow} \right) (\underline{\mathcal{F}}_1^j + \underline{\delta}^{\leftarrow j+1} \underline{\mathcal{F}}_2^{j+1}) \right], \end{aligned} \quad (93)$$

$$= (\underline{\mathcal{P}}_r^r)^T_{,p_j} \underline{\mathcal{F}}_2^r + (\underline{\mathcal{P}}_r^r)^T \underline{\mathcal{F}} \underline{\mathcal{Q}}_2^r + (\underline{\mathcal{P}}_r^r)^T \underline{\mathcal{F}} \underline{\mathcal{R}} \underline{\mathcal{Q}}_2^r, \quad (94)$$

$$\begin{aligned} \underline{\mathcal{K}}_{r,\dot{q}_r} &= (\underline{\mathcal{P}}_r^r)^T_{,\dot{q}_r} [\underline{\mathcal{F}}_1^r + \underline{\delta}^{\leftarrow r+1} \underline{\mathcal{F}}_2^{r+1}] + (\underline{\mathcal{P}}_r^r)^T [\underline{\mathcal{F}} \underline{\dot{\mathcal{Q}}}_1^r + \underline{\delta}^{\leftarrow r+1} \underline{\mathcal{F}} \underline{\dot{\mathcal{Q}}}_2^{r+1}] \\ &\quad + (\underline{\mathcal{P}}_r^r)^T \left[ \sum_{j=k+1}^n \left( \prod_{\alpha=k+1}^j \underline{\delta}_{,\dot{q}_r}^{\leftarrow} \right) (\underline{\mathcal{F}}_1^j + \underline{\delta}^{\leftarrow j+1} \underline{\mathcal{F}}_2^{j+1}) \right], \end{aligned} \quad (95)$$

$$= (\underline{\mathcal{P}}_r^r)^T_{,p_j} \underline{\mathcal{F}}_2^r + (\underline{\mathcal{P}}_r^r)^T \underline{\mathcal{F}} \underline{\dot{\mathcal{Q}}}_2^r + (\underline{\mathcal{P}}_r^r)^T \underline{\mathcal{F}} \underline{\mathcal{R}} \underline{\dot{\mathcal{Q}}}_2^r. \quad (96)$$

Explicit recursive relationships for computing the partial derivatives of matrices  $\underline{\mathcal{M}}$  and  $\underline{\mathcal{K}}$  have been demonstrated and generalized in Equations (79), (85), (89), (94), and (96). Since each quantity is itself expressed in a body frame (local coordinates), the local differentiation property ensures the simplification of its calculation. One obtains  $O(1)$  operation cost for calculating the individual elements of  $\underline{\mathcal{M}}_{r_i,p_j}$ ,  $\underline{\mathcal{M}}_{r_i,q_r}$ ,  $\underline{\mathcal{K}}_{r,p_j}$ ,  $\underline{\mathcal{K}}_{r,q_r}$ , and  $\underline{\mathcal{K}}_{r,\dot{q}_r}$ . Consequently, producing the partial derivatives of  $\underline{\mathcal{M}}$  requires  $O(n^2)$  operation and the partial derivatives of  $\underline{\mathcal{K}}$  can be determined in  $O(n)$ . Unfortunately, the overall cost associated with solving for  $d\dot{q}_r/dp_j$  will still require  $O(n^3)$  operations overall due to the explicit decomposition and solve of the system of equations which is associated with  $\underline{\mathcal{M}}^{-1}$  appearing in Equation (9). This *partial recursive* approach offers a definite improvement in performance over the  $O(n^4)$  performance of the conventional direct differentiation approach, which involves no recursive relationships whatsoever. However, the cost

predicted for this method is far greater than that realized via the presented *fully-recursive* first-order sensitivity method for systems involving large numbers of degrees of freedom  $n$ .

### Acknowledgment

Support for this work received under National Science Foundation through award No. 9733684 is gratefully acknowledged.

### References

1. Anderson, K.S., 'Recursive derivation of explicit equations of motion for efficient dynamic/control simulation of large multibody systems', Ph.D. Dissertation, Stanford University, CA, 1990.
2. Anderson, K.S., 'An order- $N$  formulation for motion simulation of general constrained multi-rigid-body systems', *Computers and Structures* **43**(3), 1992, 565–572.
3. Barthelemy, J.-F.M. and Hall, L.E., 'Automatic differentiation as a tool in engineering design', *Structural Optimization* **9**, 1995, 76–82.
4. Bestle, D. and Eberhard, P., 'Analyzing and optimizing multibody systems', *Mechanics of Structures and Machines* **20**(1), 1992, 67–92.
5. Bestle, D. and Seybold, J., 'Sensitivity analysis of constrained multibody systems', *Archive of Applied Mechanics* **62**, 1992, 181–190.
6. Bischof, C., Khademi, P. and Mauer, A., 'ADIFOR 2.0 – Automatic differentiation of Fortran 77 programs', *IEEE Computational Science and Engineering* **3**(3), 1996, 18–32.
7. Bischof, C.H., 'On the automatic differentiation of computer programs and an application to multibody systems', in *IUTAM Symposium on Optimization of Mechanical Systems*, D. Bestle and W. Schiehlen (eds), Kluwer Academic Publishers, Dordrecht, 1996, 41–48.
8. Eberhard, P., 'Adjoint variable method for sensitivity analysis of multibody systems interpreted as a continuous, hybrid form of automatic differentiation', in *Proceedings of the Second International Workshop of Computational Differentiation*, Santa Fe, NM, February 12–14, 1996, 215–227.
9. Chang, C.O. and Nikravesh, P.E., 'Optimal design of mechanical systems with constraint violation stabilization method', *Journal of Mechanisms, Transmissions, and Automation in Design* **107**, 1985, 493–498.
10. Campbell, S.L. and Hollenbeck, R., 'Automatic differentiation and implicit differential equations', in *Proceedings of the Second International Workshop of Computational Differentiation*, Santa Fe, NM, February 12–14, 1996, 319–328.
11. Etman, L.F.P., 'Optimization of multibody systems using approximation concepts', Ph.D. Dissertation, Technical University Eindhoven, The Netherlands, 1997.
12. Featherstone, R., 'The calculation of robot dynamics using articulated-body inertias', *International Journal of Robotics Research* **2**(1), 1983, 13–30.
13. Gabriele, G.A., 'Optimization in mechanisms', in *Modern Kinematics: Developments in the Last Forty Years*, A.G. Erdman (ed.), John Wiley & Sons, New York, 1993, Chapter 11.
14. Haug, E.J., Wehage, R.A. and Mani, N.K., 'Design sensitivity analysis of large-scaled constrained dynamic mechanical systems', *Transactions of the ASME* **106**, 1984, 156–162.
15. Haug, E.J. and Arora, J.S., *Applied Optimal Design*, John Wiley & Sons, New York, 1979.
16. Haug, E.J., 'Design sensitivity analysis of dynamic systems', *Computer-Aided Design: Structural and Mechanical Systems*, C.A. Mota-Soares (ed.), Springer-Verlag, Berlin, 1987.

17. Hsieh, C.C. and Arora, J.S., 'Design sensitivity analysis and optimization of dynamic response', *Computer Methods in Applied Mechanics and Engineering* **43**, 1984, 195–219.
18. Kane, T.R. and Levinson, D.A., *Dynamics: Theory and Applications*, McGraw-Hill, New York, 1985.
19. Mani, N.K. and Haug, E.J., 'Singular value decomposition for dynamic system design sensitivity analysis', *Engineering with Computers* **1**, 1985, 103–109.
20. Pagalday, J.M. and Aranburu, I., 'Multibody dynamics optimization by direct differentiation methods using object oriented programming', in *IUTAM Symposium on Optimization of Mechanical Systems*, D. Bestle and W. Schiehlen (eds), Kluwer Academic Publishers, Dordrecht 1996, 213–220.
21. Rosenthal, D.E., 'An order n formulation for robotic systems', *Journal of Astronautical Sciences* **38**(4), 1990, 511–529.
22. Schaechter, D.B., Levinson, D.A. and Kane, T.R., *AUTOLEV User's Manual V. 3.2*, Online Dynamics, Sunnyvale, CA, 1997.
23. Schwertassek, R. and Roberson, R.E., *Dynamics of Multibody Systems*, Springer-Verlag, Berlin, 1988.
24. Serban, R. and Freeman, J.S., 'Direct differentiation methods for the design sensitivity of multi-body dynamic systems', *Journal of Mechanical Design*, submitted.
25. Sohoni, V.N. and Haug, E.J., 'A state space technique for optimal design of mechanisms', *Transactions of the ASME* **104**, 1982, 792–798.