N90-22989

SPATIAL OPERATOR ALGEBRA FRAMEWORK FOR MULTIBODY SYSTEM DYNAMICS

G. Rodriguez, A. Jain, and K. Kreutz
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, MS 198-219, Pasadena, CA 91109

Abstract: This paper describes the Spatial Operator Algebra framework for the dynamics of general multibody systems. The use of a spatial operator–based methodology permits the formulation of the dynamical equations of motion of multibody systems in a concise and systematic way. The dynamical equations of progressively more complex rigid multibody systems are developed in an evolutionary manner beginning with a serial chain system, followed by a tree topology system and finally, systems with arbitrary closed loops. Operator factorizations and identities are used to develop novel recursive algorithms for the forward dynamics of systems with closed loops. Extensions required to deal with flexible elements are also discussed.

1 Introduction

The field of multibody dynamics is currently being challenged in two major ways. The increase in the size and complexity of spacecraft systems requires the development of tools that not only help manage the complexity of such systems, but also facilitate the development of novel dynamics formulation techniques and solution algorithms. Areas such as robotics involve multibody systems consisting of multiple robot manipulators interacting with each other and with complex environments. These are multibody systems with not only constantly time–varying topological structure, but also ones in which the constituent bodies change with time. Coping with this aspect requires versatile and flexible dynamics simulation tools.

In this paper, the Spatial Operator Algebra Framework [1] is used to develop a systematic procedure for concisely formulating the equations of motion and derive spatially recursive forward dynamics algorithms for multibody systems. The equations of motion of progressively more complex rigid multibody systems such as serial chains, tree topology systems and finally closed chain systems are developed. Operator factorizations and identities are then used to obtain efficient spatially recursive algorithms for the forward dynamics of such systems. Extensions to handle flexible link elements are also discussed.

2 Equations of Motion

We begin by briefly describing the coordinate–free *spatial notation* used throughout this paper. Given the linear and angular velocities v and ω, the linear force F, and moment N at a point on a body, the *spatial velocity* V, *spatial acceleration* α and the *spatial force* f in $\mathcal{R}^6$ are defined as follows:

$$V \triangleq \begin{pmatrix} \omega \\ v \end{pmatrix}, \quad \alpha \triangleq \dot{V}, \quad f \triangleq \begin{pmatrix} N \\ F \end{pmatrix}$$

The *rigid body transformation operator* $\phi(.) \in \mathcal{R}^{6\times6}$ is defined as:

$$\phi(l) \triangleq \begin{pmatrix} I & \tilde{l} \\ 0 & I \end{pmatrix}$$

CODE	DOCUMENTATION	WELCOME FEATURES	ADDITIONAL FEATURES DESIRED	USER EXPERIENCE	TYPICAL APPLICATIONS	EASY TO USE	RUN TIME ACCEPTABILITY	FLEXIBLE COMPONENT DATA
NBOD2	• Fairly well documented • Source available and commented	• Preprocessor code is helpful	• Indexing scheme on free and constrained joints can be improved • Simple application examples desired	limited	• Attitude control of spacecraft	mixed	yes, but should be improved	Does not model flexibility
SD/EXACT	• Good manual • Source code not available	• Configuration parameters can be changed at run time • Fast	• More efficient memory use • Handle flexibility • Graphics • Variable degree of freedom	extensive	• Spacecraft • Robot • Robot experiment • Code checking	yes	yes	Does not model flexibility
SPASIS	• Well Documented • Source available and commented	• Good documentation • User friendly inputs • Batch processing utilities • Extensive earth orbital dynamics	• More efficient aero-dynamics and propellant dynamics options • Animation • Reboost ability	moderate	• Large space platforms • Small space platforms • Propellant dynamics • Docking	yes	yes	Does not model flexibility
Station Control Simulation	• Code documented • Source available and commented	• Modular design • Self contained	• Model reduction code • Faster code	limited	• Space station GN&C design • RCS reboost maneuver	yes	Excessive when flexible model used	Outputs 2 and 4 from NASTRAN
TREETOPS	• Reasonably documented • Source is available but not thoroughly commented	• Interactive preprocessor • Speedup options • Library of actuators and sensors	• Certain flex inputs not fully defined • Graphics preprocessor • Cumbersome output • Better interface with user subroutines	moderate	• Shuttle and payloads • Spacestation • Wing flap • Check other codes	usually	Usually, high for flexbody payloads	• Difficult, additional software needed • Modes are restrictive

where l is a vector joining two points, and $\tilde{l}$ is the cross-product matrix associated with l which acts on a vector to produce the cross-product of l with the vector. $\phi(l)$ and $\phi^*(l)$ transform spatial forces and spatial velocities respectively between two points on a rigid body seperated by the vector l. For a rigid body, its spatial inertias M_C and M_O at its center of mass C and at another point O respectively, are defined as

$$M(C) \triangleq \begin{pmatrix} \mathcal{J}(C) & 0 \\ 0 & mI \end{pmatrix}, \quad \text{and} \quad M(O) \triangleq \phi(p)M(C)\phi^*(p) = \begin{pmatrix} \mathcal{J}(O) & m\tilde{p} \\ -m\tilde{p} & mI \end{pmatrix}$$

where p is the vector from O to C, m is the mass of the body, and $\mathcal{J}(C)$ and $\mathcal{J}(O)$ are the inertia tensors for the body about C and O respectively. The reader is referred to [2] for additional discussion on the use of spatial notation.

2.1 Dynamics of a Serial Rigid Multibody System

Serial rigid multibody systems form basic subsystems from which the dynamics of more general rigid multibody systems can be generated. In this section we derive the equations of motion of a serial multibody system consisting of n rigid links connected together by multiple dof joints. The links are numbered 1 through n from tip to base. We use the terms ***outboard*** (***inboard***) link to refer to a link on the path towards the tip (base).

The set of configuration variables for the serial chain are the collection of the joint configuration parameters. It is assumed that the k^{th} joint possesses $r_p(k)$ positional dofs parameterized by the vector of configuration variables $\theta(k)$ (of dimension at least $r_p(k)$), and that its $r_v(k)$ motion dofs are parameterized by the $r_v(k)$ dimensional joint velocity vector $\beta(k)$. The kinematical equations which relate $\dot{\theta}(k)$ to $\beta(k)$ depend on the specific nature of the k^{th} joint. It is assumed for notational convenience that all the joint constraints are homogeneous (i.e., catastatic). $H(k)$ is defined such that $H^*(k)$ is the $6 \times r_v(k)$ joint map matrix for the k^{th} joint whose columns span the space of permissible relative spatial velocities $\Delta_V(k)$ across the joint. The complexity of the dynamics algorithms for the serial chain is determined by the number of overall motion dofs $\mathcal{N} \triangleq \sum_{k=1}^{n} r_v(k)$ for the chain. The state of the multibody system is defined by the collection of $[\theta(.), \beta(.)]$ for all the joints, and is assumed known.

Since each link is rigid, it suffices to develop the equations of motion at a single reference point on each link, which is taken to be the inboard joint location $\mathcal{O}_k$ for the k^{th} link. With $V(k)$ denoting the spatial velocity, $\alpha(k)$ the spatial acceleration, $f(k)$ the spatial force and $T(k)$ the joint force at $\mathcal{O}_k$ for the k^{th} link, the following Newton–Euler recursive equations describe the equations of motion for the serial rigid multibody chain:

```
    $V(n+1) = 0, \quad \alpha(n+1) = 0$
for k = n ··· 1
    $V(k) = \phi^*(k+1,k)V(k+1) + H^*(k)\beta(k)$
    $\alpha(k) = \phi^*(k+1,k)\alpha(k+1) + H^*(k)\dot{\beta}(k) + a(k)$
end loop
                                                                  (2.1)
    $f(0) = 0$
for k = 1 ··· n
    $f(k) = \phi(k+1,k)f(k-1) + M(k)\alpha(k) + b(k)$
    $T(k) = H(k)f(k)$
end loop
```

$a(k)$ and $b(k)$ are the velocity dependent Coriolis acceleration and gyroscopic forces repectively for the k^{th} link at $\mathcal{O}_k$. $\phi(k,k-1)$ denotes the transformation operator from $\mathcal{O}_{k-1}$ to $\mathcal{O}_k$. For additional details regarding the derivation of these equations of motion, see [1]. We have made the simplifying assumption that the tip force $f(0)$ is zero. Attaching a full 6 motion dof joint between the physical base and the inertial frame allows us to easily deal with the mobile base situation. For the inverse dynamics problem, the joint accelerations $\dot{\beta}$ are known, and Eq. (2.1) represents an $O(\mathcal{N})$ computational process involving a base–to–tip recursion to compute the velocities and accelerations, followed by a tip–to–base recursion to compute the joint forces.

In order to express the equations of motion given by Eq. (2.1) in a more compact form, we define the ***stacked notation***. In this notation, the $V(k)$'s, $\alpha(k)$'s etc are viewed as components of vectors V, α etc. Then Eq. (2.1) can be written in the following compact form:

$$\begin{aligned} V &= \mathcal{E}_\phi^* V + H^* \beta \\ \alpha &= \mathcal{E}_\phi^* \alpha + H^* \dot{\beta} + a \\ f &= \mathcal{E}_\phi f + M\alpha + b \end{aligned} \tag{2.2}$$

where,

$$\mathcal{E}_\phi \triangleq \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \phi(2,1) & 0 & \dots & 0 & 0 \\ 0 & \phi(3,2) & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \phi(n,n-1) & 0 \end{pmatrix}, \quad M \triangleq \begin{pmatrix} M(1) & 0 & \dots & 0 \\ 0 & M(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M(n) \end{pmatrix},$$
$$H \triangleq \begin{pmatrix} H(1) & 0 & \dots & 0 \\ 0 & H(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & H(n) \end{pmatrix} \tag{2.3}$$

However, since $\mathcal{E}_\phi$ is nilpotent ($\mathcal{E}_\phi^n = 0$),

$$\phi \triangleq (I - \mathcal{E}_\phi)^{-1} = I + \mathcal{E}_\phi + \mathcal{E}_\phi^2 + \cdots + \mathcal{E}_\phi^{n-1} = \begin{pmatrix} I & 0 & \dots & 0 \\ \phi(2,1) & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \phi(n,1) & \phi(n,2) & \dots & I \end{pmatrix} \tag{2.4}$$

where,

$$\phi(i,j) \triangleq \phi(i,i-1) \cdots \phi(j+1,j)$$

Thus Eq. (2.2) can be reexpressed in the form,

$$\begin{aligned} V &= \phi^* H^* \beta \\ \alpha &= \phi^*(H^*\dot{\beta} + a) \\ f &= \phi(M\alpha + b) = \phi M \phi^* H^* \dot{\beta} + \phi(M\phi^* a + b) \\ T &= Hf = H\phi M\phi^* H^* \dot{\beta} + H\phi(M\phi^* a + b) \\ &= \mathcal{M}\dot{\beta} + \mathcal{C}, \quad \text{where } \mathcal{M} \triangleq H\phi M\phi^* H^*, \quad \text{and } \mathcal{C} \triangleq H\phi(M\phi^* a + b) \end{aligned} \tag{2.5}$$

$\mathcal{M} \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$ is the ***mass matrix*** for the serial chain and $\mathcal{C} \in \mathcal{R}^{\mathcal{N}}$ consists of the velocity dependent Coriolis, centrifugal and gyroscopic joint forces. In the terminology of Kane's method [3], β are the *generalized speeds* and the elements of $\phi^* H^*$ are the ***partial (spatial) velocities***.

$\mathcal{E}_\phi$, ϕ, H, and M are the first of the ***spatial operators*** that will be encountered. Recursive dynamical algorithms can be derived naturally by exploiting the special *state transition* properties [1] of the elements of spatial operators such as $\mathcal{E}_\phi$, ϕ etc.. For instance, given a vector y, the evaluation of the matrix–vector product ϕy *does not* require an $O(n^2)$ matrix-vector product computation, and not even the explicit computation of the elements of ϕ, but rather, it can be evaluated using an $O(n)$ recursive algorithm involving only the elements of $\mathcal{E}_\phi$ and y. This is precisely the correspondence between the concise operator based high-level description of the equations of motion in Eq. (2.5) and the recursive algorithmic description in Eq. (2.1).

Spatially recursive $O(\mathcal{N})$ forward dynamics algorithms for serial chains have been developed in [4] based on the recognition of the isomorphism between the structure of the dynamics equations and the equations encountered in Kalman Filtering theory. These insights have formed the basis for the development of the Spatial Operator Algebra Framework for multibody dynamics.

2.2 Tree Topology Systems

In this section, the dynamics of rigid multibody systems with tree topological structure are discussed. A tree topology system may be viewed as a set of component serial chains (referred to as *branches*) coupled together via joints at their *terminal* links. The total number of branches is denoted ℓ. The index for the branches thus ranges from $1 \cdots \ell$, and consistent with the link numbering scheme in the previous section, the inboard branches are assigned indices larger than those for the outboard ones. The *connectivity function* $\imath(k)$ is defined as the index of the *direct predecessor* branch, i.e., the inboard branch to which the k^{th} branch is connected. The j^{th} branch is simply denoted a *predecessor* branch for the k^{th} branch if it belongs on the unique path from the k^{th} branch to the base, i.e., if $\imath^p(k) = j$ for some integer $p > 0$. The joint coupling two branches is assigned to the outboard branch. Figure 1 illustrates the link/branch numbering convention for tree topology systems.

The notation for serial chains from Section 2.1 is carried over to describe the branches in the tree structure, and an additional subscript is used to identify the specific branch in the system. Thus n_j and $\mathcal{N}_j$ denote the number of links and the number of motion dofs respectively, while V_j, M_j, $\mathcal{E}_{\phi_j}$, ϕ_j etc. denote the appropriate spatial velocity etc. quantities for the j^{th} branch. A link/joint is identified by the index of the branch it is on, plus its location within the branch. For instance, $V(k_j)$ (or more accurately $V_j(k)$) denotes the spatial velocity of the the k^{th} link of the j^{th} branch at its inboard joint location $\mathcal{O}(k_j)$. The overall stacked spatial velocity, acceleration etc. vectors for the tree are now denoted V, α, f etc. with $V \triangleq [V_1^* \cdots V_\ell^*]^*$ etc.. The total number of links n, and the total number of motion dofs $\mathcal{N}$ for the system are given by

$$n \triangleq \sum_{j=1}^{\ell} n_j \quad \text{and} \quad \mathcal{N} \triangleq \sum_{j=1}^{\ell} \mathcal{N}_j \tag{2.6}$$

Note that when the j^{th} branch is the direct predecessor of the k^{th} branch, i.e., $j = \imath(k)$, the joint connecting them is the n_k^{th} joint on the k^{th} branch and is located on link 1_j on the j^{th} branch. The transformation operator from the n_k^{th} joint to the 1_j^{th} joint is denoted $\phi(1_j, n_k)$. The spatial operator $\mathcal{E}_\phi$ is now defined in terms of its block matrix elements below. For $j, k \in 1 \cdots \ell$,

$$\mathcal{E}_\phi(j,k) = \begin{cases} \mathcal{E}_{\phi_j} & \text{for} \quad j = k \\ \begin{pmatrix} 0 & \cdots & 0 & \phi(1_j, n_k) \\ 0 & \cdots & 0 & 0 \\ \vdots & \cdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{pmatrix} & \text{for} \quad j = \imath(k), \text{ i.e. if } j \text{ is the direct predecessor branch of } k \\ 0 & \text{for} \quad j \neq \imath(k), \text{ i.e. if } j \text{ is } \textit{not} \text{ the direct predecessor of } k \end{cases} \tag{2.7}$$

0 denotes a zero matrix of dimension appropriate for the context. As a consequence of the numbering scheme used here, for $j < k$, the j^{th} branch cannot be a predecessor to the k^{th} branch and thus the $(j,k)^{th}$ block

element, $\mathcal{E}_\phi(j,k) = 0$. Thus $\mathcal{E}_\phi$ is a strictly lower triangular matrix. The analogs of Eq. (2.2) are as follows:

$$\begin{aligned} V &= \mathcal{E}_\phi^* V + H^*\beta \\ \alpha &= \mathcal{E}_\phi^*\alpha + H^*\dot{\beta} + a \\ f &= \mathcal{E}_\phi f + M\alpha + b \end{aligned} \tag{2.8}$$

Once again (analogous to Eq. (2.4)), $\mathcal{E}_\phi$ is nilpotent ($\mathcal{E}_\phi^n = 0$), and so

$$\phi \overset{\Delta}{=} (I - \mathcal{E}_\phi)^{-1} = I + \mathcal{E}_\phi + \mathcal{E}_\phi^2 + \cdots + \mathcal{E}_\phi^{n-1} \tag{2.9}$$

The block structure of ϕ is described below:

$$\phi(j,k) = \begin{cases} \phi_j & \text{for } j = k \\ \{\phi(m_j, l_k)\}_{m,l} & \text{if } \exists\, p > 0 : j = \imath^p(k), \text{ i.e., if } j \text{ is a predecessor branch of } k \\ 0 & \text{if } j \neq \imath^p(k)\ \forall\, p > 0, \text{ i.e., if } j \text{ is } \textit{not} \text{ a predecessor branch of } k \end{cases} \tag{2.10}$$

Here $\{\phi(m_j, l_k)\}_{m,l}$ denotes a block matrix whose $(m,l)^{th}$ entry is given by $\phi(m_j, l_k)$ with $m \in 1\cdots n_j$ and $l \in 1 \cdots n_k$. $\phi(m_j, l_k)$ is the transformation operator from joint l_k (on the k^{th} branch) to joint m_j (on the j^{th} branch) and is a generalization of the transformation operator $\phi(i,j)$ in Eq. (2.4) for serial chains. It is formed by sequentially composing all the individual transformation operators that lie on the *unique* path joining the two joints. The numbering scheme used here ensures that ϕ will be a lower triangular matrix. The operator ϕ has state transition properties analogous to the ϕ for serial chains, and as a consequence, it can be used for high-level and concise description of the dynamics of tree topology systems (as in Eq. (2.11) below), but with the full understanding that from the computational perspective, these equations directly map into recursive implementation procedures. From Eq. (2.8) and Eq. (2.4) it follows that,

$$\begin{aligned} V &= \phi^* H^* \beta \\ \alpha &= \phi^*(H^*\dot{\beta} + a) \\ f &= \phi(M\alpha + b) = \phi M \phi^* H^* \dot{\beta} + \phi(M\phi^* a + b) \\ T &= Hf = H\phi M\phi^* H^*\dot{\beta} + H\phi(M\phi^* a + b) \\ &= \mathcal{M}\dot{\beta} + \mathcal{C}, \quad \text{where } \mathcal{M} \overset{\Delta}{=} H\phi M \phi^* H^*, \quad \text{and} \quad \mathcal{C} \overset{\Delta}{=} H\phi(M\phi^* a + b) \end{aligned} \tag{2.11}$$

$\mathcal{M} \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$ denotes the mass matrix for the tree system. $\hat{T} \overset{\Delta}{=} T - \mathcal{C}$ can be easily computed from the knowledge of the system state, and so the equations of motion for the system can be rewritten in the form

$$\mathcal{M}\dot{\beta} = \hat{T} \tag{2.12}$$

The forward dynamics problem requires then the solution of the joint accelerations $\dot{\beta}$ for a given set of joint forces $\hat{T}$. The mass matrix for the system is typically not available and potentially needs to be computed to solve the forward dynamics problem. However, in Section 3, a recursive $O(\mathcal{N})$ forward dynamics algorithm for tree topology systems, which does not require the explicit computation of the mass matrix $\mathcal{M}$, is derived.

Before proceeding on to closed topology systems, we first derive the structure of the Jacobian operator. Given n_C points, denoted C_k's, on the tree (see Figure 1), the Jacobian operator $J \in \mathcal{R}^{6n_C \times \mathcal{N}}$ defines the mapping between β and $\tilde{V}$, i.e., $\tilde{V} = J\beta$, where $\tilde{V} \in \mathcal{R}^{6n_C}$ denotes the vector of spatial velocities at these points. If C_k is on link m_j, then the spatial velocity at C_k is given by

$$\tilde{V}(k) = \phi^*(\mathcal{O}(m_j), C_k) V(m_j)$$

with $\phi(\mathcal{O}(m_j), C_k)$ denoting the rigid body transformation operator from C_k to the point $\mathcal{O}(m_j)$. With the block elements of $B \in \mathcal{R}^{6n \times 6n_C}$ defined as

$$B(m_j, k) = \begin{cases} \phi(\mathcal{O}(m_j), C_k) & \text{if } C_k \in m_j^{th} \text{ link} \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } k = 1 \cdots n_C \tag{2.13}$$

it follows that

$$\tilde{V} = B^*V = B^*\phi^*H^*\beta, \quad \text{i.e.,} \quad J = B^*\phi^*H^* \tag{2.14}$$

This gives us an expression for the desired Jacobian operator which will be used below when dealing with loop closure constraints for closed topology systems.

2.3 Closed Topology Systems

This paper develops a systematic procedure for the formulation of the equations of motion and derivation of forward dynamics solution algorithms for general topology multibody systems with time–varying topologies as well as changing constituent bodies. Based on the specific application, such systems may be conceptually partitioned as follows:

(a) The *primary* system consisting of the least time–variant part, i.e., the multibody subsystem with fixed topology and constituent bodies.

(b) The *secondary* system consisting of the multibody subsystem which may change from time to time.

(c) The set of *closure constraints* and/or boundary conditions between/within the primary and secondary systems which change with changes in the system topology.

Note the the subsystems described above are in the order of increasing time–variation. As an example, let us examine the robotics scenario of multiple manipulators interacting with each other and the environment to perform complex tasks. In this context, the manipulators belong to the primary system since their innate structure varies very little with time. The task objects vary from task to task and form the secondary system. The constraints between these two subsystems change during the execution of a task, such as grasping, mating, tool operation etc., and belong to the last category.

This partitioning allows us to derive a very general and yet systematic procedure for the development of dynamics algorithms which are responsive and adaptable to time–varying systems. The procedure involves a sequence of decoupled steps for each of the primary and secondary system dynamics, and one step in which they come together when the constraint forces are computed. Being structurally time–invariant, it is possible to put in place optimized algorithms for the dynamics of the primary system. The time–variant secondary system is typically of small complexity and thus the use of standard, though suboptimal, algorithms does not substantively degrade performance.

This decomposition of the closed topology system is a departure from the more traditional approach (see [5, 6]) of forming a spanning tree for the full system and computing the constraint forces at the points of closure. In these latter approaches, even small changes in the original system typically lead to whole new spanning trees for the system. This disallows any algorithmic optimization, and the algorithms are also not very amenable to coping with time–varying systems.

The primary and secondary systems in most applications have tree topological structure. However in general there may be internal closed loops within either system. In any case, by cutting an appropriate number of joints, each subsystem may be regarded as a tree topology system with additional kinematical constraints at the internal loop closure points. The equations of motion for tree topology systems derived in Eq. (2.12) will be used to describe the dynamics of the tree components of both the primary and secondary systems, with the subscripts "P" and "S" differentiating the two subsystems. Thus the dynamics of the tree part of the two systems are described by

$$\hat{T}_P = H_P\phi_P M_P\phi_P^* H_P^* \dot{\beta}_P = \mathcal{M}_P\dot{\beta}_P, \quad \text{and} \quad \hat{T}_S = H_S\phi_S M_S\phi_S^* H_S^* \dot{\beta}_S = \mathcal{M}_S\dot{\beta}_S \tag{2.15}$$

$\mathcal{M}_P$ and $\mathcal{M}_S$ denote the mass matrices, β_P and β_S the motion dof parameter vectors, $\hat{T}_P$ and $\hat{T}_S$ the bias–free internal joint forces for the primary and secondary subsystems respectively.

Combining the internal loop points of closure with the points of closure coupling the two systems, we obtain the overall points of closure for each of the subsystems. Let $\tilde{V}_P$ and $\tilde{V}_S$ denote the spatial velocities at these overall points of closure for the two systems, and following the discussion leading to Eq. (2.14), let $J_P = B_P^* \phi_P^* H_P^*$ and $J_S = B_S^* \phi_S^* H_S^*$ denote the Jacobian operators for the two systems corresponding to these points. Thus $\tilde{V}_P = J_P \beta_P$ and $\tilde{V}_S = J_S \beta_S$. The kinematical constraints due to the existence of internal closed loops within the primary and secondary systems leads to constraint equations of the form:

$$Q_P \tilde{V}_P = \tilde{U}_P \quad \text{and} \quad Q_S \tilde{V}_S = \tilde{U}_S$$

The coupling together of the primary and secondary systems via joints leeds to constraint equations of the form:

$$\tilde{Q}_P \tilde{V}_P + \tilde{Q}_S \tilde{V}_S = \tilde{U}_C$$

Defining

$$A_P \triangleq \begin{pmatrix} \tilde{Q}_P \\ Q_P \\ 0 \end{pmatrix}, \quad A_S \triangleq \begin{pmatrix} \tilde{Q}_S \\ 0 \\ Q_S \end{pmatrix}, \quad \text{and} \quad A \triangleq [A_P \;\; A_S]$$

the closure constraints can be collectively expressed in the form:

$$A \begin{pmatrix} \tilde{V}_P \\ \tilde{V}_S \end{pmatrix} = [A_P \;\; A_S] \begin{pmatrix} J_P & 0 \\ 0 & J_S \end{pmatrix} \begin{pmatrix} \beta_P \\ \beta_S \end{pmatrix} = [A_P J_P \;\; A_S J_S] \begin{pmatrix} \beta_P \\ \beta_S \end{pmatrix} = \begin{pmatrix} \tilde{U}_C \\ \tilde{U}_P \\ \tilde{U}_S \end{pmatrix} \triangleq \tilde{U} \qquad \textbf{(2.16)}$$

It is assumed that $[A_P J_P \;\; A_S J_S]$ is of full row rank $\mathcal{N}_E$. The overall number of motion dofs of the closed chain system is given by $\mathcal{N}_C \triangleq \mathcal{N} + \mathcal{N}_S - \mathcal{N}_E$, and if necessary, Eq. (2.16) can be used to find the $\mathcal{N}_C$ dimensional minimal set of motion generalized coordinates for system. Based on the principle of virtual work, Eq. (2.16) implies that the closure constraint joint forces are of the form

$$\begin{pmatrix} J_P^* A_P^* \\ J_S^* A_S^* \end{pmatrix} \tilde{f}$$

for some $\tilde{f} \in \mathcal{R}^{\mathcal{N}_E}$. This leads to the following overall equations of motion:

$$\begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S & J_S^* A_S^* \\ A_P J_P & A_S J_S & 0 \end{pmatrix} \begin{pmatrix} \dot{\beta}_P \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} \hat{T}_P \\ \hat{T}_S \\ U \end{pmatrix}, \quad \text{where} \quad U \triangleq \dot{\tilde{U}} - [(A_P \dot{J}_P) \;\; (A_S \dot{J}_S)] \begin{pmatrix} V_P \\ V_S \end{pmatrix}$$

$$\Rightarrow \quad \begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S & J_S^* A_S^* \\ 0 & 0 & -[A_P \Lambda_P A_P^* + A_S \Lambda_S A_S^*] \end{pmatrix} \begin{pmatrix} \dot{\beta}_P \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} \hat{T}_P \\ \hat{T}_S \\ U - [A_P J_P \mathcal{M}_P^{-1} \hat{T}_P + A_S J_S \mathcal{M}_S^{-1} \hat{T}_S] \end{pmatrix} \qquad \textbf{(2.17)}$$

where

$$\Lambda_P \triangleq J_P \mathcal{M}_P^{-1} J_P^*, \quad \text{and} \quad \Lambda_S \triangleq J_S \mathcal{M}_S^{-1} J_S^*$$

Note that Λ_P and Λ_S are the effective "admittances" of the primary and secondary systems reflected to the points of closure. We now describe some special cases of the above setup:

- The joint constraints coupling the primary and secondary systems are typically on the relative spatial velocity across the joints at the points of closure. When this is true for all joints, an appropriate reordering of the elements of $\tilde{V}$ will result in $\tilde{Q}_P = -\tilde{Q}_S$. Furthermore, if no relative motion is permitted across the joint, i.e., there is rigid rather than loose coupling, then in fact $\tilde{Q}_P = I$ and $\tilde{Q}_S = I$. When this is the case for only some of the joints, only the corresponding rows have these special features.
- If the secondary system has no internal actuators or source of generalized forces, then $T_S = 0$.
- If the secondary system is a free rigid body with no internal degrees of freedom, then the motion generalized coordinates vector β_S is of dimension 6 and consists of the 3 translational and 3 rotational dof parameters.

3 Forward Dynamics of Closed Chain Systems

In this section we discuss a recursive method for solving the forward dynamics of closed chain rigid multibody systems. This method does not require the explicit computation of the primary and secondary tree system mass matrices $\mathcal{M}_P$ or $\mathcal{M}_S$, but does require the computation of the constraint force parameters.

From the equations of motion of the closed chain system with dynamical closure constraints given by Eq. (2.17), the solution of the forward dynamics problem can be solved by the following sequence of steps:

(A)	Solve $\mathcal{M}_P\dot\beta^f = \hat{T}_P$ for $\dot\beta_P^f$	Solve $\mathcal{M}_S\dot\beta_S^f = \hat{T}_S$ for $\dot\beta_S^f$
(B)	Compute $\tilde\alpha_P^f = J_P\dot\beta_P^f$	Compute $\tilde\alpha_S^f = J_S\dot\beta_S^f$
(C)	Compute $\Lambda_P = J_P\mathcal{M}_P^{-1}J_P^*$	Compute $\Lambda_S = J_S\mathcal{M}_S^{-1}J_S^*$
(D)	Solve $[A_P\Lambda_PA_P^* + A_S\Lambda_SA_S^*]\tilde f = (A_P\tilde\alpha_P^f + A_S\tilde\alpha_S^f) - U$ for $\tilde f$	
(E)	Solve $\mathcal{M}_P\dot\beta_P^\delta = -J_P^*A_P^*\tilde f$ for $\dot\beta_P^\delta$	Solve $\mathcal{M}_S\dot\beta_S^\delta = -J_S^*A_S^*\tilde f$ for $\dot\beta_S^\delta$
(F)	$\dot\beta_P = \dot\beta_P^f + \dot\beta_P^\delta$	$\dot\beta_S = \dot\beta_S^f + \dot\beta_S^\delta$

As a result of the partitioning, a changes in the closure constraints only effect A and thus only STEP D, while changes in the secondary system effect only the steps in the right half column. Recursive algorithms for carrying out each of these steps are derived below. The proofs of the various lemmas are omitted due to space limitations. However they follow precisely along the lines of the proofs for serial chains discussed in [7]. The explicit use of the subscripts indentifying the primary/secondary system is dropped (except for STEP (D)) since the discussion is equally applicable to either subsystem.

STEP (A) Solve $\mathcal{M}\dot\beta^f = \hat{T}$. (Forward Dynamics of a Tree Topology System)

Note that Step (A) is equivalent to solving the forward dynamics of a tree topology system, and we develop an $O(\mathcal{N})$ recursive algorithm for this solution. This algorithm is based on a new factorization of the mass matrix $\mathcal{M}$ in terms of square factors. which may be contrasted with the earlier non-square factorization in Eq. (2.11). This square factorization is then used to obtain an explicit expression for $\mathcal{M}^{-1}$.

The *articulated body inertia matrix* P is defined as the solution to the following equation:

$$M = P - \mathcal{E}_\phi[P - PH^*(HPH^*)^{-1}HP]\mathcal{E}_\phi^* \qquad \textbf{(3.1)}$$

P is *block diagonal* and the elements on the diagonal (denoted $P(k_j)$) can be obtained using a recursive algorithm described in Eq. (A.1) in Appendix A. Physically, $P(k_j)$ is the *articulated body inertia* as seen at the k_j^{th} joint, i.e., it is the effective inertia of all the links outboard from the k_j^{th} joint assuming that the joint forces at all the outboard joints are zero.

For the subsequent development, it is convenient to define

$$\begin{aligned} D &\triangleq HPH^*, \quad G \triangleq PH^*D^{-1}, \quad K \triangleq \mathcal{E}_\phi G \\ \tau &\triangleq GH, \quad \overline{\tau} \triangleq I - \tau, \quad \mathcal{E}_\psi \triangleq \mathcal{E}_\phi\overline{\tau} \end{aligned} \qquad \textbf{(3.2)}$$

Note that D, G, τ and $\overline{\tau}$ are all block diagonal. The structure of $\mathcal{E}_\psi$ is identical to that of $\mathcal{E}_\phi$ with its elements being given by

$$\psi(k_j, k_j - 1) \triangleq \phi(k_j, k_j - 1)\overline{\tau}(k_j - 1)$$

$\mathcal{E}_\psi$ is also nilpotent ($\mathcal{E}_\psi^n = 0$), and analogous to ϕ, ψ is defined as

$$\psi \triangleq (I - \mathcal{E}_\psi)^{-1} = I + \mathcal{E}_\psi + \mathcal{E}_\psi^2 + \cdots + \mathcal{E}_\psi^{n-1} \tag{3.3}$$

The structure of ψ is very similar to that of ϕ and it also possesses the state transition properties which are used to develop recursive algorithms. ϕ may be viewed as the transformation operator for composite bodies (i.e., as if all the joints are locked), while ψ is the transformation operator for articulated bodies (i.e., as if all the joint forces were zero). The following lemma yields a square factorization of $\mathcal{M}$.

Lemma 1: The mass matrix $\mathcal{M}$ has the following factorization:

$$\mathcal{M} = [I + H\phi K]D[I + H\phi K]^*, \tag{3.4} \blacksquare$$

The following lemma gives the explicit form for the inverse of $[I + H\phi K]$.

Lemma 2:

$$[I + H\phi K]^{-1} = [I - H\psi K] \tag{3.5} \blacksquare$$

Combining Lemma 1 and Lemma 2 leads to the following form for the inverse of the mass matrix.

Lemma 3:

$$\mathcal{M}^{-1} = [I - H\psi K]^* D^{-1} [I - H\psi K] \tag{3.6} \blacksquare$$

Thus,

$$\dot{\beta}^f = \mathcal{M}^{-1}\hat{T} = [I - H\psi K]^* D^{-1} [I - H\psi K]\hat{T} \tag{3.7}$$

The $O(\mathcal{N})$ recursive computation of the expression on the right is given in Eq. (A.2) in Appendix A.

STEP (B) Compute $\tilde{\alpha}^f = J\dot{\beta}^f$

From Eq. (2.14), $\tilde{\alpha}^f = B^*\hat{\alpha}^f$, where

$$\hat{\alpha}^f \triangleq \phi^* H^* \dot{\beta}^f \tag{3.8}$$

However we have that,

Lemma 4:

$$(I - H\psi K)H\phi = H\psi \tag{3.9} \blacksquare$$

Thus using Eq. (3.6) and the above lemma in Eq. (3.8),

$$\hat{\alpha}^f = \phi^* H^* [I - H\psi K]^* D^{-1} [I - H\psi K]\hat{T} = \psi^* H^* D^{-1} [I - H\psi K]\hat{T}$$

Comparing this with Eq. (3.7) we see that $\hat{\alpha}^f$ can be evaluated as an intermediate quantity in the $O(\mathcal{N})$ recursive algorithm for computing $\dot{\beta}^f$ described in STEP (A).

STEP (C) Compute $\Lambda = J\mathcal{M}^{-1}J^*$

Using Eq. (2.14) and Eq. (3.6),

$$\begin{aligned} \Lambda &= \{[I - H\psi K]H\phi B\}^* D^{-1} \{[I - H\psi K]H\phi B\} \\ &= B^*\psi^* H^* D^{-1} H\psi B = B^*\Omega B, \quad \text{where} \quad \Omega \triangleq \psi^* H^* D^{-1} H\psi \end{aligned} \tag{3.10}$$

where Eq. (3.9) has been used to simplify the above expression. A recursive $O(\mathcal{N})$ procedure for the computation of Ω is given in Eq. (A.5) in Appendix A. Note that without the simplification resulting from the use of Eq. (3.9), the computation of Λ would be an $O(\mathcal{N}^3)$ process.

STEP (D) Solve $[A_P \Lambda A_P^* + A_S \Lambda_S A_S^*]\tilde{f} = (A_P \tilde{\alpha}_P^f + A_S \tilde{\alpha}_S^f) - U$ for $\tilde{f}$

Now,

$$\tilde{f} = [A_P \Lambda A_P^* + A_S \Lambda_S A_S^*]^{-1}[(A_P \tilde{\alpha}_P^f + A_S \tilde{\alpha}_S^f) - U] \tag{3.11}$$

In this form this step is of $O(\mathcal{N}_E^3)$ complexity. However, when $(A_P \Lambda_P A_P^*)$ is invertible, we can obtain an alternative expression for $\tilde{f}$ by reexpressing Eq. (2.17) as follows:

$$\begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S & J_S^* A_S^* \\ 0 & A_S J_S & -A_P \Lambda_P A_P^* \end{pmatrix} \begin{pmatrix} \dot{\beta} \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} = \begin{pmatrix} \hat{T}_P \\ \hat{T}_S \\ U - A_P \tilde{\alpha}_P^f \end{pmatrix}$$

and consequently,

$$\begin{pmatrix} \mathcal{M}_P & 0 & J_P^* A_P^* \\ 0 & \mathcal{M}_S + J_S^* A_S^* (A_P \Lambda A_P^*)^{-1} A_S J_S & 0 \\ 0 & A_S J_S & -A_P \Lambda_P A_P^* \end{pmatrix} \begin{pmatrix} \dot{\beta} \\ \dot{\beta}_S \\ \tilde{f} \end{pmatrix} =$$

$$\begin{pmatrix} \hat{T}_P \\ \hat{T}_S + J_S^* A_S^* (A_P \Lambda_P A_P^*)^{-1} [U - A_P \tilde{\alpha}_P^f] \\ U - A_P \tilde{\alpha}_P^f \end{pmatrix}$$

From the above equation it follows that

$$\begin{aligned} \dot{\beta}_S &= [\mathcal{M}_S + J_S^* A_S^* (A_P \Lambda_P A_P^*)^{-1} A_S J_S]^{-1} [T_S - J_S^* A_S^* (A_P \Lambda A_P^*)^{-1} (A_P \tilde{\alpha}_P^f - U)] \\ \tilde{f} &= (A_P \Lambda A_P^*)^{-1} [(A_P \tilde{\alpha}_P^f + A_S J_S \dot{\beta}_S) - U] \\ &= (A_P \Lambda_P A_P^*)^{-1} [(A_P \tilde{\alpha}_P^f + A_S \tilde{\alpha}_S) - U], \quad \text{where} \quad \tilde{\alpha}_S \triangleq J_S \dot{\beta}_S \end{aligned}$$

Note the similarity between the forms of Eq. (3.11) and the above equation for $\tilde{f}$. The computational cost of the above operation is a combination of the cost of inverting $A_P \Lambda_P A_P^*$, and the $O(\mathcal{N}_S^3)$ step of solving a square linear system of equations of size $\mathcal{N}_S$. The cost of inverting $A_P \Lambda_P A_P^*$ depends on its structure: its sparsity reflects the degree of coupling between the closed loops in the system. The cost is typically much less than the worst case of $O(\mathcal{N}_E^3)$. In many application domains such as robotics, $A_P \Lambda_P A_P^*$ is in fact block diagonal and is thus invertible in $O(\mathcal{N}_E)$ steps [1]. In addition, for most applications $\mathcal{N}_S \ll \mathcal{N}_E$, and this new formulation can lead to considerable computational savings.

The inverse of $[A_P \Lambda_P A_P^* + A_S \Lambda_S A_S^*]$ will not exist if $[A_P J_P \quad A_S J_S]$ is not of full rank, i.e., the configuration is such that the number of motion dofs for the system have changed. It is therefore necessary to reformulate the constraint equation Eq. (2.16) so as to preserve the full rank property. Such changes of rank can occur at kinematically singular configurations.

STEP (E) Compute $\beta^\delta = -\mathcal{M}^{-1} J^* A^* \tilde{f}$

We have from Eq. (3.6) and Eq. (2.14) that

$$\beta^\delta = -[I - H\psi K]^* D^{-1} [I - H\psi K] H\phi B A^* \tilde{f}$$

Using Lemma 4 this simplifies to

$$\beta^\delta = -[I - H\psi K]^* D^{-1} H\psi B A^* \tilde{f} \tag{3.12}$$

The recursive $O(\mathcal{N})$ implementation of the above step is given in Eq. (A.6) in Appendix A.

The overall complexity of this spatially recursive forward dynamics algorithm ranges between $O(\mathcal{N}+\mathcal{N}_S)+O(\mathcal{N}_E^3)$ for the worst case and $O(\mathcal{N}+\mathcal{N}_S)+O(\mathcal{N}_E)+O(\mathcal{N}_S^3)$ in the best case.

By treating the primary and secondary system as one system, which amounts to defining the quantities $\psi \triangleq diag(\psi_P, \psi_S)$, $H \triangleq diag(H_P, H_S)$ etc., and using the above results, the overall closed topology forward dynamics algorithm can be restated in the following form:

$$\dot{\beta} = [I - H\psi K]^* D^{-\frac{1}{2}} \left[I - b(A\Lambda A^*)^{-1} b^*\right] D^{-\frac{1}{2}} [I - H\psi K]\hat{T}, \quad \text{where} \quad b \triangleq H\psi B A^* \tag{3.13}$$

Note that when there are no closed loops in the overall system, $A = 0$, and the middle term reduces to I, and we recover the form for for the forward dynamics of tree topology systems in Eq. (3.7).

4 Flexible Multibody Dynamics

In this section we briefly describe the extensions to handle the case of flexible links. We use the serial chain discussed in Section 2.1 as an illustrative example, but now assume that the links in the chain are flexible. It is assumed (without losing any generality) that finite element models are available for all the links, and in particular, the k^{th} link is characterized by: n_k node points with the location of of the j^{th} node denoted $\mathcal{Q}_k(j)$'s, the vector of displacement variables $u_k^f \in \mathcal{R}^{6n_k}$, a free–free mass-matrix $m_k \in \mathcal{R}^{6n_k \times 6n_k}$, a stiffness matrix $K_k \in \mathcal{R}^{6n_k \times 6n_k}$. The ordering of the nodes is such that $\mathcal{Q}_k(1)$ is on the same element as $\mathcal{O}_{k-1}$ and $\mathcal{Q}_k(n_k)$ is on the same element as $\mathcal{O}_k$. Treating the k^{th} link as being pinned at $\mathcal{O}_k$, this implies that $u_k^f(n_k) = 0$, and thus the true flexible dofs are given by the vector $u_k = h_k u_k^f$, where $h_k^- [I, 0]$. Note that $u_k \in \mathcal{R}^{6(n_k-1)}$ and $h_k \in \mathcal{R}^{6n_k \times 6(n_k-1)}$.

With $V(k)$ denoting the spatial velocity of the k^{th} link at $\mathcal{O}_k$,

$$\begin{aligned} V(k) &= \phi^*(k+1,k)V(k+1) + H^*(k)\beta(k) + \phi^*\left(\mathcal{Q}_{k+1}(1), \mathcal{O}_k\right)u_{k+1}(1) \\ &= \phi^*(k+1,k)V(k+1) + H^*(k)\beta(k) + C_{k+1}^* h_k^* u_{k+1} \end{aligned} \tag{4.1}$$

$$\text{where} \quad C_{k+1}^* \triangleq [\phi^*\left(\mathcal{Q}_{k+1}(1), \mathcal{O}_k\right), \ 0, \ \cdots \ 0] \in \mathcal{R}^{6 \times 6n_k} \tag{4.2}$$

Thus,

$$V = \phi^*[H^*\beta + C^* h^* u]$$

with C defined as the block matrix with C_2 to C_n along its first block subdiagonal, and h is the block diagonal matrix with j^{th} block diagonal element being h_j. Let $V_k^f \in \mathcal{R}^{6n_k}$ denote the vector of spatial velocities on the k^{th} link at the n_k node points. Then

$$\begin{aligned} V_k^f &= B_k^* V(k) + h_k^* u_k, \quad \text{where} \quad B_k \triangleq [\phi(\mathcal{O}_k, \mathcal{Q}_k(1)), \cdots, \phi(\mathcal{O}_k, \mathcal{Q}_k(n_k))] \in \mathcal{R}^{6 \times 6n_k} \\ \Rightarrow \quad V^f &= B^* V + h^* u = B^*\phi^*[H^*\beta + C^* h^* u] + h^* u = B^*\phi^* H^*\beta + [I + B^*\phi^* C^*]h^* u \\ &= [I + B^*\phi^* C^*][h^* \ \ B^* H^*]X, \quad \text{where} \quad X \triangleq \begin{pmatrix} u \\ \beta \end{pmatrix} \end{aligned} \tag{4.3}$$

B denotes the block diagonal matrix with the B_k's along its diagonal. We have used the facts that,

$$B_k C_k = \phi(k, k-1) \quad \Rightarrow \quad BC = \mathcal{E}_\phi \quad \Rightarrow \quad BC\phi = \phi - I \quad \Rightarrow \quad B[I + C\phi B] = \phi B$$

Note that X is the vector of motion dofs for the serial links and includes both the rigid and flexible dof parameters.

Using Eq. (4.3), the kinetic energy for the whole chain is given by

$$T = \frac{1}{2}(V^f)^* m V^f = \frac{1}{2} X^* \mathcal{M}^f X, \quad \text{where} \quad \mathcal{M}^f \triangleq \begin{pmatrix} HB \\ h \end{pmatrix} [I + C\phi B] m [I + C\phi B]^* \begin{pmatrix} HB \\ h \end{pmatrix}^*$$

$\mathcal{M}^f$ is the mass matrix for the flexible serial chain. Given this factored form for the mass matrix, similar techniques to those used for the rigid multibody case in the earlier sections result in alternate factorizations and inversion of the mass matrix, and recursive forward dynamics algorithms. The reader is referred to [8] for additional details. Just as for the rigid multibody case, the algorithms for flexible serial chains directly extend to the flexible general topology multibody systems.

5 Conclusions

This paper describes the Spatial Operator Algebra Framework for the dynamics of general multibody systems. Based on their rate of time–variation, the multibody system is partitioned into a primary subsystem, a secondary subsystem and the set of closure constraints. This allows the development of forward dynamics algorithms which are not only recursive and efficient, but also capable of easily coping with time varying multibody systems. The solution procedure consists of a sequence of steps on parallel paths involving the dynamics of the spanning trees for the primary and secondary systems. The two paths come together for one step in order to compute the constraint forces. Using the spatial algebra techniques to develop novel factorizations of the mass matrix and operator identities, efficient recursive algorithms for carrying out each of these steps is developed. The overall algorithm does not require the computation of the mass matrix, and its complexity is linear in the number of dofs for the tree systems. In addition, the impact on the complexity of the algorithm, of the degree of coupling among the closed loops in the system topology is made clear, and it is shown that the in the best circumstance, the algorithmic complexity is also linear in the number of closure constraint equations. During the development, an $O(\mathcal{N})$ forward dynamics algorithm for tree topology systems is also developed. For the sake of clarity, the focus of much of the paper was on multibody systems with rigid links. However the extensions necessary to deal with flexible elements are discussed.

6 Acknowledgement

The research described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[1] G. Rodriguez and K. Kreutz, "Recursive mass matrix factorization and inversion: an operator approach to open and closed–chain multibody dynamics," JPL Publication 88–11, Jet Propulsion Laboratory, Pasadena, CA, 1988.

[2] A. Jain, "Unified formulation of dynamics for serial rigid multibody systems," Eng. Memo. 347–89–264, (Internal Document), Jet Propulsion Laboratory, Pasadena, CA, 1989.

[3] T. Kane and D. Levinson, *Dynamics: Theory and Applications.* McGraw–Hill, 1985.

[4] G. Rodriguez, "Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics," *IEEE J. Robotics Automat.*, vol. 3, Dec. 1987. (JPL Publication 86-48, 1986).

[5] H. Brandl, R. Johanni, and M. Otter, "An algorithm for the simulation of multibody systems with kinematic loops," in *World Congress on the Theory of Machines and Mechanisms (7th), Seville, Spain*, 1987.

[6] D. Bae and E. Haug, "A recursive formulation for constrained mechanical system dynamics: Part II. Closed loop systems," *Mech. Struct. & Mach.*, vol. 15, no. 4, pp. 481–506, 1987–88.

[7] G. Rodriguez, K. Kreutz, and A. Jain, "A spatial operator algebra for manipulator modeling and control," in *IEEE Conf. Rob. and Aut., Scottsdale, Az*, May 1989.

[8] G. Rodriguez, "Spatial operator approach to multibody manipulator inverse and forward dynamics," in *IEEE Conf. Rob. and Aut., Cincinnati, OH*, May 1990.

A Appendix

Based on the special structure of ϕ, ψ etc., it is possible to evaluate many of the dynamical expressions in a recursive manner and we describe some recursive algorithms in this appendix. First we define some notational shorthand to simplify the description of the algorithms that follow:

$$\begin{aligned}
x(n_j+1) &\Longrightarrow x(1_{\imath(j)}) \\
y(n_j+1, n_j) &\Longrightarrow y(1_{\imath(j)}, n_j) \\
y(1_j, 0_j)x(0_j) &\Longrightarrow \sum_{m \in \imath^{-1}(j)} y(1_j, n_m)x(n_m) \\
y(1_j, 0_j)x(0_j)y^*(1_j, 0_j) &\Longrightarrow \sum_{m \in \imath^{-1}(j)} y(1_j, n_m)x(n_m)y^*(1_j, n_m)
\end{aligned}$$

where $y(.,.)$ and $x(.)$ stand for some appropriate arrays. Thus wherever a term with indices as in the left column appears, its meaning is actually given by the corresponding term in the column on the right. !pr

- A recursive method for the computation of the block diagonal elements of P as defined by Eq. (3.1) and the entries of D, G, K, $\mathcal{E}_\psi$ and $\overline{\tau}$ defined in Eq. (3.2) are given by:

```
for j = $1 \cdots \ell$
    If $\imath^{-1}(j) = \emptyset$, then $P(0_j) = 0$
    for k = $1_j \cdots n_j$
        $P(k) = \psi(k, k-1)P(k-1)\psi^*(k, k-1) + M(k)$
        $D(k) = H(k)P(k)H^*(k)$
        $G(k) = P(k)H^*(k)D^{-1}(k)$
        $\overline{\tau}(k) = I - G(k)H(k)$
        $\psi(k+1, k) = \phi(k+1, k)\overline{\tau}(k)$
        $K(k+1, k) = \phi(k+1, k)G(k)$
    end loop
end loop                                                    (A.1)
```

- The recursive computation of $\beta^f = [I - H\psi K]^* D^{-1}[I - H\psi K]\hat{T}$ in Eq. (3.7) in STEP (A) can be carried out via the $O(\mathcal{N})$ tree topology forward dynamics algorithm described below. It also results in the computation of $\hat{\alpha}^f = \psi^* D^{-1}[I - H\psi K]\hat{T}$ required in STEP (B) as an intermediate quantity.

```
for j = $1 \cdots \ell$
    If $\imath^{-1}(j) = \emptyset$, then $z(0_j) = 0, \hat{T}(0_j) = 0$
    for k = $1_j \cdots n_j$
        $z(k) = \psi(k, k-1)z(k-1) + K(k, k-1)\hat{T}(k-1)$
        $\epsilon(k) = T(k) - H(k)z(k)$
        $\nu(k) = D^{-1}(k)\epsilon(k)$
    end loop
end loop
```

```
$\hat{\alpha}^f(n_\ell + 1) = 0$
for j = $\ell \cdots 1$
    for k = $n_j \cdots 1_j$
        $\hat{\alpha}^f(k) = \psi^*(k+1,k)\hat{\alpha}^f(k+1) + H^*(k)\nu(k)$
        $\dot{\beta}^f(k) = \nu(k) - K^*(k+1)\hat{\alpha}^f(k+1)$
    end loop
end loop
```
(A.2)

- STEP (C) requires the computation of $\Lambda = B^*\Omega B$. In order to obtain a $O(\mathcal{N})$ recursive scheme for the computation of Ω we first define the matrix Υ as the one satisfying the equation:

$$H^*D^{-1}H = \Upsilon - \mathcal{E}_\psi^*\Upsilon\mathcal{E}_\psi \tag{A.3}$$

Υ as defined above is a block diagonal matrix and its elements can be computed recursively. We now obtain the following decomposition of Ω.

Lemma 5:

$$\Omega = \Upsilon + \tilde{\psi}^*\Upsilon + \Upsilon\tilde{\psi} \tag{A.4}$$ ∎

Noting that $\tilde{\psi}$ is strictly lower triangular, we can then recognize that Υ as nothing but the diagonal elements of Ω. We now present a recursive scheme to compute the block diagonal elements of Υ and of Ω.

```
$\Upsilon(n_\ell + 1) = 0$
for j = $\ell \cdots 1$
    for k = $n_j \cdots 1_j$
        $\Upsilon(k) = \psi^*(k+1,k)\Upsilon(k+1)\psi(k+1,k) + H^*(k)D^{-1}(k)H(k)$
            $\Omega(k,k) = \Upsilon(k)$
            for m = $k-1 \cdots 1_j$
                $\Omega(k,m) = \Omega^*(m,k) = \Upsilon(k,m+1)\psi(m+1,m)$
            end loop
    end loop
end loop
```

The above recursion yields the elements Ω_j on the block diagonal of Ω. Since Ω is symmetric, the off–diagonal elements satisfy $\Omega_{j,l} = \Omega^*_{l,j}$, and can be computed from the diagonal elements as follows. $\Omega_{l,j}$ for $l \in 1 \cdots (j-1)$ can be obtained via the following recursive scheme:

```
if $\imath^p(l) = j$ for some $p > 0$
    for k = $n_j \cdots 1_j$
        for m = $n_l \cdots 1_l$
            $\Omega(k,m) = \Omega^*(m,k) = \Omega(k,1_j)\psi(1_j,m)$
        end loop
    end loop
else
    $\Omega_{j,l} = \Omega^*_{l,j} \equiv 0$
end if
```
(A.5)

- The $O(\mathcal{N})$ recursive implementation of $\beta^\delta = -[I - H\psi K]^* D^{-1} H\psi B A^* \tilde{f}$ in Eq. (3.12) in Step (e) is given below:

```
Define $\hat{x} \triangleq -BA^*\tilde{f}$
for j = $1 \cdots \ell$
    If $\imath^{-1}(j) = \emptyset$, then $z(0_j) = 0, \hat{x}(0_j) = 0$
    for k = $1_j \cdots n_j$
        $z(k) = \psi(k, k-1) z(k-1) + K(k, k-1)\hat{x}(k-1)$
        $\epsilon(k) = -H(k) z(k)$
        $\nu(k) = D^{-1}(k)\epsilon(k)$
    end loop
end loop
$\alpha(n_\ell + 1) = 0$
for j = $\ell \cdots 1$
    for k = $n_j \cdots 1_j$
        $\alpha(k) = \psi^*(k+1, k)\alpha(k+1) + H^*(k)\nu(k)$
        $\dot{\beta}^\delta(k) = \nu(k) - K^*(k+1)\alpha(k+1)$
    end loop
end loop                                                    (A.6)
```

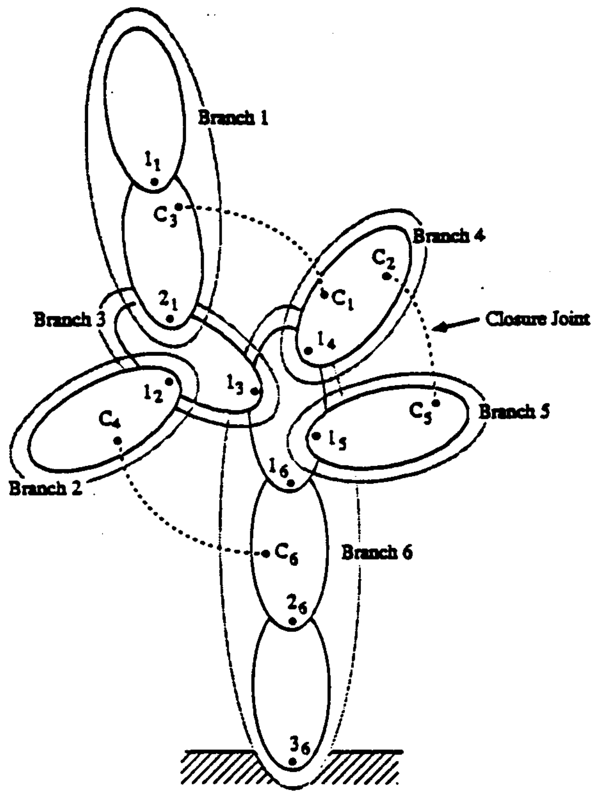

Figure 1: Illustration of link/branch numbering convention for multibody system