Spatial Operator Factorization and Inversion of the Manipulator Mass Matrix

Guillermo Rodriguez, *Member, IEEE,* and Kenneth Kreutz-Delgado

***Abstract*—Two new recursive factorizations are developed of the mass matrix for fixed-base and mobile-base manipulators. First, the mass matrix $\mathcal{M}$ is shown to have the factorization $\mathcal{M} = H\phi M\phi^* H^*$. This is referred to here as the Newton–Euler factorization because it is closely related to the recursive Newton–Euler equations of motion. This factorization may be the simplest way to show the equivalence of recursive Newton–Euler and Lagrangian manipulator dynamics. Second, the mass matrix is shown to have a related innovations factorization $\mathcal{M} = (\mathcal{I} + H\Phi G)D(\mathcal{I} + H\Phi G)^*$. This leads to an immediate inversion $\mathcal{M}^{-1} = (\mathcal{I} - H\Psi G)^* D^{-1} (\mathcal{I} - H\Psi G)$, where H and Φ are given by known link geometric parameters, and G, Ψ and D are obtained by a discrete-step Riccati equation driven by the link masses. The factors $(\mathcal{I} + H\Phi G)$ and $(\mathcal{I} - H\Psi G)$ are lower triangular matrices that are inverses of each other, and D is a diagonal matrix. Efficient order N inverse and forward dynamics algorithms are embedded in the two factorizations. Moreover, the factorizations provide a high-level architectural understanding of the mass matrix and its inverse, which is not available readily from the detailed algorithms. The two factorizations are model-based in the sense that the manipulator model itself determines the sequence of computations. This makes the two factorizations quite distinct from more traditional Cholesky-like numerical factorizations of positive definite matrices. Because the manipulator model is used, every computational step has a corresponding physical interpretation. This adds a substantial amount of robustness, and numerical errors can be detected by physical intuition. Development of the factorizations is made simple by the use of spatial operators, such as ϕ, Φ and Ψ, which govern the propagation of forces, velocities, and accelerations from link to link along the span of the manipulator.**

NOMENCLATURE

$h(k)$	Unit vector along joint axis k.
$\theta(k)$	Angle of link k with respect to link $k+1$ about joint axis k.
$O(k)$	Fixed point on joint axis k, which can be viewed as the origin of a frame fixed in link k.
$l(k, k-1)$	Vector from $O(k)$ to $O(k-1)$.
$F(k)$	Constraint force on link k at point $O(k)$ of joint k.
$N(k)$	Constraint moment on link k at joint k.
$F^c(k)$	Net force on link k at link k mass center.
$CM(k)$	Mass center of link k.
$p(k)$	Vector from $O(k)$ to $CM(k)$.
$v(k)$	Velocity of link k at point $O(k)$ of joint k.
$\omega(k)$	Angular velocity of link k.
$v^c(k)$	Velocity of link k at link k mass center.
$m(k)$	Mass of link k.
$I^c(k)$	Inertia tensor of link k at point $CM(k)$.
$I(k)$	Inertia tensor of link k at point $O(k)$.
$T(k)$	Actuated torque at joint k.

Manuscript received May 11, 1988; revised June 13, 1991. This work was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. K. Kreutz-Delgado was partially supported by the National Science Foundation under Presidential Young Investigator Award IRI-9057631 and by the California Space Institute under Grant CS-22–90.

G. Rodriguez is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109.

K. Kreutz-Delgado is with the Electrical and Computer Engineering Department, University of California at San Diego, La Jolla, CA 92130-0407.

IEEE Log Number 9104592.

I. INTRODUCTION

A FUNDAMENTAL analogy between multibody robot dynamics and linear filtering and smoothing has been established in [1] and [2]. This analogy allows analysis of manipulator dynamics using the very well understood and highly popular recursive equations of Kalman filtering [3]. The present paper takes this analogy further by extending to mechanical systems a powerful series of results [4]–[10], which emerged after Kalman's fundamental paper [3], and which have carried filtering and smoothing theory to a very mature state of development. These results include: 1) state variable techniques [4], involving filtering and smoothing, to solve Fredholm equations analogous to the equations of robot dynamics; 2) Riccati equations, Fredholm resolvents, and Wiener–Hopf equations [5], [6] to solve estimation problems recursively; 3) the innovations approach [7], [8] to least squares estimation, which factors covariances recursively [9]. A summary of this body of work is provided in [10].

In particular, this paper establishes mass matrix factorizations similar to the covariance factorizations summarized in [10]. The innovations approach [6]–[8] plays a central role in the factorization of covariances and in the development of corresponding filtering and smoothing algorithms. Similarly, the approach for mechanical systems advanced here leads to an "innovations" factorization of the mass matrix and to corresponding recursive forward dynamics algorithms.

A. Factorizations Provide High-Level Architectural Understanding of the Mass Matrix and Its Inverse

The factorizations provide a high-level architectural understanding of the mass matrix not readily apparent from detailed algorithms. This understanding is useful in developing arm-independent hierarchical computer programs for simulation and control design. It is also useful for concisely summarizing the manipulator models required for advanced forms of manipulator motion planning and control. The two factorizations

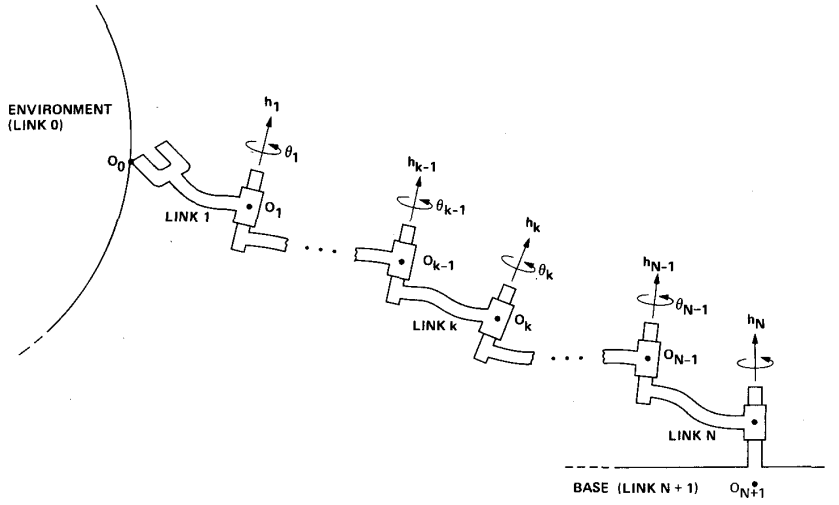


Fig. 1. Link serial manipulator.

are model based in the sense that the manipulator model itself is used to conduct the computations required. Every computational step has a simple physical interpretation. This leads to great physical understanding and insight that can be used to detect and correct numerical errors that might arise.

The spatial operator approach to manipulator modeling [11]–[14] is used to establish the mass matrix factorizations. The operators govern the propagation of forces, velocities, and accelerations through space from one rigid body to the next. A physical interpretation of the operators is easy to obtain. This allows the complex equations of robot dynamics to be understood intuitively. The spatial operators obey simple algebraic properties that allow them to be manipulated in a spatial operator algebra [12].

B. Factorizations Embed Efficient Inverse and Forward Dynamics Algorithms

Embedded in the two mass matrix factorizations are inverse and forward dynamics algorithms recognized for their numerical efficiency. The recursive Newton–Euler factorization $\mathcal{M} = H\phi M\phi^* H^*$ shows that the mass matrix $\mathcal{M}$ represents a base-to-tip recursion followed by a tip-to-base recursion. This result may be the simplest proof yet derived of the equivalence [15] of Lagrangian and Newton–Euler manipulator dynamics. It also embeds the efficient inverse dynamics algorithms of [16]–[18]. An alternative factorization $\mathcal{M} = (\mathcal{I}+H\Psi G)D(\mathcal{I}+H\Psi G)^*$, referred to here as the innovations factorization, leads to an immediate recursive inversion of the mass matrix. It also leads to equivalent $O(N)$ recursive forward dynamics algorithms [1], [19], [20] in which the number of arithmetical operations grows only linearly with the number of degrees of freedom.

C. Current Applications of the Factorization Results

The mass matrix factorization results outlined here have been used to develop hierarchical computer programs for manipulator dynamics [21]. Programs are relatively easy to write because of the high level of abstraction made possible by spatial operators. The programs can take any allowable operator expression and implement a computationally efficient recursive algorithm. It is even possible to automate the development of such algorithms with relatively simple computer programs. The factorization results are also being used to manage the complexity in several problems of current interest in robotics research: recursive implementation of operational space control [22], [23]; flexible multibody systems [24], [25]; indirect-drive geared robotic manipulators [26]; recursive linearization [27]; manipulator control design [28]; and statistical mechanics models for motion planning [29].

II. State Space Model for Multilink Dynamics

Consider a rigid N-link serial manipulator as illustrated in Figs. 1 and 2, with the symbols defined in the nomenclature list. The links and joints are numbered in an increasing order that goes from the tip of the system toward the base. Joint N is the last in the sequence, and it connects link N to a base. The base is referred to as link $N+1$. The external environment is viewed as "link 0," and the arm can contact the environment at any arbitrary point denoted by the index $k = 0$. Joint k in the sequence connects links k and $k+1$. For now, the base is assumed to be immobile, a restriction that is relaxed in Section VIII. Note that link and joint numbers increase toward the base of the system. This differs from the more common numbering approach in which the numbers increase toward the tip. This numbering scheme makes it easy to describe the mass matrix factorization techniques discussed in this paper.

The ordering scheme allows thinking of sequentially moving from joint 1 to joint N as going "inward" and moving from joint N to joint 1 as going "outward." An algorithm that processes link data by iterating in the inward direction from $k = 1$ to $k = N$ is then called tip-to-base. In contrast, an algorithm that iterates outward from $k = N$ to $k = 1$ is called base-to-tip. A complete tip-to-base iteration is called an inward sweep. A complete base-to-tip iteration is called an outward sweep. Although $\theta(k)$ is defined to be rotational,

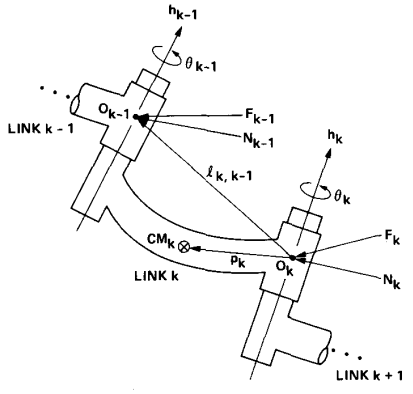


Fig. 2. Relationship of defined quantities to link k.

the extension to sliding joints is simple [11]. Note that axis $h(k)$ is associated with angle $\theta(k)$ and both are associated with link k. This scheme can be considered to be a modified Denavit–Hartenberg formulation with reversed link numbering [11].

A. Spatial Velocity, Acceleration, Force, and Inertia

Robot dynamics equations are expressed very concisely in terms of quantities referred to [1] as spatial velocity $V(k)$, spatial acceleration $\alpha(k)$, spatial force $f(k)$, and spatial inertia $M(k)$:

$$V(k) = \begin{pmatrix} \omega(k) \\ v(k) \end{pmatrix}$$

$$\alpha(k) = \begin{pmatrix} \dot{\omega}(k) \\ \dot{v}(k) \end{pmatrix}$$

$$f(k) = \begin{pmatrix} N(k) \\ F(k) \end{pmatrix} \tag{1}$$

$$M(k) = \begin{pmatrix} I(k) & m(k)\tilde{p}(k) \\ -m(k)\tilde{p}(k) & m(k)\mathcal{I} \end{pmatrix}. \tag{2}$$

The spatial forces $f(k)$ belong to a state space very similar to those encountered in filtering and smoothing [1]. The spatial accelerations are the corresponding costates. The spatial quantities defined by (1) and (2) are closely related to those of [19], but they are not identical. The difference is quite significant because it implies that only the rules of ordinary matrix algebra are used here without use of the nonstandard algebra of [19]. Furthermore, the formulation presented here also allows use of the very well established operations [1], [3] of state space estimation and control. One of the central ingredients in the state space approach is the state transition matrix as defined below.

B. State Transition Matrix

Define $\ell(i,j)$ to be the vector from point O_i to point O_j. These two points are illustrated in Fig. 1. Define also

$$\phi(i,j) = \begin{pmatrix} \mathcal{I} & \tilde{\ell}(i,j) \\ 0 & \mathcal{I} \end{pmatrix}. \tag{3}$$

The state-space transition matrix $\phi(i,j)$ relates the spatial force at point j to the spatial force at point i. Similarly, the adjoint matrix $\phi^*(i,j)$ relates the spatial velocity at point i to the spatial velocity at point j. It is known [1] that $\phi(i,j)$ obeys the state-space transition matrix properties

$$\phi(i,i) = \mathcal{I}$$

$$\phi^{-1}(i,j) = \phi(j,i)$$

$$\phi(i,k)\phi(k,j) = \phi(i,j) \tag{4}$$

$$\phi(i,k) = \phi(i,i-1), \cdots, \phi(k+1,k). \tag{5}$$

III. Recursive Newton–Euler Equations

In terms of the spatial quantities in Section II, the recursive Newton–Euler equations [16] for manipulator dynamics are [1], [11], assuming for now an immobile base,

$$V(N+1) = 0$$

$$\alpha(N+1) = 0 \tag{6}$$

for $k = N \cdots 1$ **loop**

$$V(k) = \phi^*(k+1,k)V(k+1) + H^*(k)\dot{\theta}(k) \tag{7}$$

$$\alpha(k) = \phi^*(k+1,k)\alpha(k+1) + H^*(k)\ddot{\theta}(k) + a(k) \tag{8}$$

end loop

$$V(0) = \phi^*(1,0)V(1)$$

$$\alpha(0) = \phi^*(1,0)\alpha(1) + a(0) \tag{9}$$

$$f(0) = f_{\text{ext}} \tag{10}$$

for $k = 1 \cdots N$ **loop**

$$f(k) = \phi(k,k-1)f(k-1) + M(k)\alpha(k) + b(k) \tag{11}$$

$$T(k) = H(k)f(k) \tag{12}$$

end loop

Observe in (7) that $\phi^*(k+1,k)$ is the link $k+1$ transition matrix that relates rates at joint $k+1$ to rates at joint k [1]. The joint-axis projection operator $H^*(k)$ in (7), (8), and (12) is defined as

$$H^*(k) = \begin{pmatrix} h(k) \\ 0 \end{pmatrix}. \tag{13}$$

Note that $V(0)$ and $\alpha(0)$ are the manipulator tip spatial velocity and acceleration, and $f(0)$ is the tip spatial force acting on the external environment. Note also that $a(k) = a[V(k+1), V(k)]$ and $b(k) = b[V(k)]$. Hence, at the kth iteration, the bias acceleration $a(k)$, and the bias spatial force $b(k)$ can be computed from available quantities by the explicit formulas given in [11]. If a joint k is a sliding joint rather than rotational, the recursive spatial dynamics equations can be easily modified [11] by redefining $H^*(k)$ and $a(k)$ appropriately.

Because the above equations are expressed in terms of coordinate-free vectors, the notation "*" is used to denote transpose. The adjoint x^* for a three-vector x is defined by the dot product $x^*y = x \cdot y$. Equations (6)–(12) are referred to as recursive. More properly, they could be described as iterative. However, this usage is consistent with that in the

robotics literature [16], where the term recursive has been made synonymous with iterative.

Assume henceforth that $[\theta(k), \dot{\theta}(k)]$ are known. Given this knowledge, the inverse dynamics problem is to compute the moments $T(k)$ from the known inputs $\ddot{\theta}(k)$. The forward dynamics problem is to obtain $\ddot{\theta}(k)$ from known inputs $T(k)$ and the recursive spatial dynamics equations (6)–(12).

IV. Spatial Vectors and Spatial Operators

Manipulator equations can be expressed very concisely in terms of the following spatial vectors and spatial operators [12].

A. Spatial Vectors

As an example of a spatial vector, consider

$$V = \text{col}\,[V(1), \cdots, V(N)] \tag{14}$$

which collects the sequence of spatial velocities $V(k)$ defined at the joints of the system into a $6N \times 1$ composite vector V. The vector V is a quantity associated with the entire system of bodies, in the sense that all of the bodies are represented in this vector. Each of its elements $V(k)$ corresponds to one of the bodies k. Similarly

$$\begin{aligned} \theta &= \text{col}\,[\theta(1), \ldots, \theta(N)] \\ T &= \text{col}\,[T(1), \ldots, T(N)] \\ a &= \text{col}\,[a(1), \ldots, a(N)] \\ \alpha &= \text{col}\,[\alpha(1), \ldots, \alpha(N)] \\ f &= \text{col}\,[f(1), \ldots, f(N)] \end{aligned} \tag{15}$$

$$b = \text{col}\,[b(1), \ldots, b(N)] \tag{16}$$

in which θ is the vector of joint angles, T is the vector of joint moments, a is the vector of bias accelerations, α is the vector of spatial accelerations, f is the vector of spatial forces, and b is the vector of bias forces.

The main motivation for introducing the composite notation above is to eliminate the argument k associated with the various links. The symbol V denotes a vector relevant to the entire manipulator system, and the need to refer subsequently to the individual elements $V(k)$ is reduced significantly.

B. Spatial Operators to Propagate Force, Velocity, and Acceleration

The spatial operator ϕ is defined as

$$\phi = \begin{pmatrix} \mathcal{I} & 0 & \cdots & 0 \\ \phi(2,1) & \mathcal{I} & \cdots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ \phi(N,1) & \phi(N,2) & \cdots & \mathcal{I} \end{pmatrix}. \tag{17}$$

This is perhaps the most fundamental spatial operator defined in this paper because many of the operators defined subsequently are dependent on it. It is defined in terms of the transition matrix $\phi(i,j)$ that governs the propagation of force from joint j to joint i. It is a composite operator in the sense that all of the possible pairs of joints are represented. The operator ϕ can be thought of as a transformation that in a global sense governs the propagation of force within the overall composite multilink system. Similarly, its adjoint ϕ^* governs the propagation of velocity and acceleration within the same system.

The spatial operators H and B^* are defined as

$$\begin{aligned} H &= \text{diag}[H(1), \ldots, H(N)] \\ B^* &= [\phi^*(1,0), 0, \ldots, 0]. \end{aligned} \tag{18}$$

Equation (18) defines a block-diagonal partitioned matrix H whose block-diagonal elements are $H(k)$. This matrix collects the projections $H(k)$ associated with the set $k = 1, \cdots, N$ of joint axes.

C. The Jacobian Operator

A widely known relationship between joint rates $\dot{\theta}$ and tip velocity $V(0)$ is

$$V(0) = J\dot{\theta} \tag{19}$$

in which J is the Jacobian operator. However, it is not widely known that the Jacobian operator J has the factorization

$$J = B^*\phi^*H^* \tag{20}$$

in terms of the spatial operators B^*, ϕ^*, and H^*. This factorization can be established easily using the kinematic relationships in Section III. In fact, (6)–(8), (17), and (18) together imply

$$\begin{aligned} V(k) &= \sum_{i=k}^{N} \phi^*(i,k)H^*(i)\dot{\theta}(i) \\ V &= \phi^*H^*\dot{\theta} \\ V(0) &= B^*\phi^*H^*\dot{\theta}. \end{aligned} \tag{21}$$

The operator factorization $J = B^*\phi^*H^*$ of the Jacobian J has an immediate physical interpretation in terms of the action of J on the joint rates $\dot{\theta}$: 1) H^* acts on $\dot{\theta}$ in a noniterative way resulting in the relative spatial velocities between the links along the joint axes; 2) the action of ϕ^* on $H^*\dot{\theta}$, in a base-to-tip iterative manner given by (7), propagates the relative link velocities to form the link spatial velocities $V = \text{col}[V(1), \cdots, V(N)]$; and 3) the operator B^* projects out $V(1)$ from V in a noniterative way and propagates it to the tip forming $V(0)$.

The important theme that emerges here is that operator factorizations have obvious physical interpretations and equivalent recursive algorithms.

V. Recursive Newton–Euler Mass Matrix Factorization

The mass matrix factorization

$$\mathcal{M} = H\phi M\phi^*H^* \tag{22}$$

emerges easily by expressing in terms of spatial operators the composite manipulator equations of motion. Here, $M = \text{diag}[M(1), \cdots, M(N)]$ is a block-diagonal matrix that collects along its diagonal the spatial inertias $M(k)$ of all of the links.

A. Equations of Motion Based on Spatial Operators

The equations of motion for the composite multilink system are

$$T = \mathcal{M}\ddot{\theta} + \mathcal{C} + J^* f(0) \tag{23}$$

where $\mathcal{C} = H\phi(M\phi^* a + b)$ and $J^* = H\phi B$. This result can be established easily by means of the following sequence of steps.

From (6), (8), (17), and (18), $\alpha = \phi^* H^* \ddot{\theta} + \phi^* a$. The accelerations α are seen to be the result of the joint accelerations $H^*\ddot{\theta}$ and the bias accelerations a propagated from the base to the tip of the manipulator under the influence of the operator ϕ^* of interlink Jacobians. From (6), (7), (9), (17), and (18), $\alpha(0) = B^*\phi^* H^* \ddot{\theta} + B^*\phi^* a + a(0)$. Since $\alpha(0) = J\ddot{\theta} + \dot{J}\dot{\theta}$, then $\dot{J}\dot{\theta} = B^*\phi^* a + a(0)$.

From (10)–(12)

$$\begin{aligned} f &= \phi[M\alpha + b + Bf(0)] \\ T &= Hf. \end{aligned} \tag{24}$$

The spatial force vector f is seen to result from the tip-to-base propagation of the D'Alembert forces $M\alpha$, the bias forces b, and the tip forces $f(0)$. This inward propagation is implied by the action of the lower block triangular operator ϕ on the quantity $[M\alpha + b + Bf(0)]$. T is seen to be the noniterative projection of f onto the joint axis.

Equation (24) can also be written as

$$\begin{aligned} T &= H\phi[M\alpha + b + Bf(0)] \\ \alpha &= \phi^* H^* \ddot{\theta} + \phi^* a \end{aligned} \tag{25}$$

which states that T is obtained by an outward operation on $\ddot{\theta}$ and a, followed by an inward operation on α, b, and $f(0)$. This is precisely the recursive Newton–Euler algorithm of [16].

Combination of the two equations in (25) leads to the sought after operator formulation of the manipulator dynamics in (23). That (23) is equivalent to (25) reflects the equivalence [15] between Lagrangian and recursive Newton–Euler dynamics. For this reason, (22) is referred to here as the recursive Newton–Euler factorization of $\mathcal{M}$.

B. Bias-Free Equations of Motion

Equation (23) can be written as

$$\mathcal{M}\ddot{\theta} = T' \tag{26}$$

in which $T' = T - \widehat{T}$, with the "bias torques" $\widehat{T}$ given by

$$\widehat{T} = H\phi[M\phi^* a + b + Bf(0)] = \mathcal{C} + J^* f(0). \tag{27}$$

Computing $\widehat{T}$ recursively via the algorithm implicit in (27) allows working with the simpler system (26). Obtaining $\widehat{T}$ from (27) is equivalent to using the recursive Newton–Euler algorithm implicit in (25) but with $\ddot{\theta} = 0$. This approach to simplifying from (23) to (26) is the standard one and is used in [16] and [19]. A choice exists, then, to solve the forward dynamics problem by either solving (23) directly for $\ddot{\theta}$ or by solving the simpler system (26) for which the bias torques have been removed. The second choice is considered in Section VI, where the focus is on operator factorization and inversion of the mass operator. The algorithmic alternative represented by (23) is developed and presented in Section VII.

C. Recursive Mass Matrix Evaluation

The recursive Newton–Euler factorization $\mathcal{M} = H\phi M\phi^* H^*$ implies, and is implied by, the recursive Newton–Euler algorithm for inverse dynamics. Hence, the factorization contains within it one of the most useful algorithms [16] for inverse dynamics. Another efficient algorithm embedded in the factorization is that referred to in [18] as the composite rigid body method for recursive computation of the mass matrix itself. This algorithm can be developed using the following spatial operator identity.

Identity 1: The matrix $\phi M\phi^*$ can be expressed as

$$\phi M\phi^* = r + \Phi r + r\Phi^* \tag{28}$$

where the matrix Φ is obtained from ϕ by subtracting the $6N \times 6N$ identity. Φ is equal to the operator $\tilde{\phi}$ used in [11]. The $6N \times 6N$ block-diagonal matrix $r = \mathrm{diag}[\mathrm{r}(1), \ldots, \mathrm{r}(\mathrm{N})]$ has blocks $r(k)$ given by

$$r(k) = \sum_{i=1}^{k} \phi(k,i) M(i) \phi^*(k,i). \tag{29}$$

Proof: Observe that the (k,j)th block element of $\phi M\phi^*$ is

$$\sum_{i=1}^{\min(k,j)} \phi(k,i) M(i) \phi^*(j,i).$$

Hence, the typical element kernel $r(k,j)$ of $\phi M\phi^*$ is $r(k,j) = \phi(k,j)r(j)$ for $j \le k$, in which $r(k)$ satisfies (29).

The algorithmic equivalent of (28) is the following inward recursion for the diagonal elements $m(k,k)$ and off-diagonal elements $m(i,k)$ of the composite multibody mass matrix $\mathcal{M}$:

$$r(0) = 0 \tag{30}$$

for $k = 1 \cdots N$ **loop**

$$r(k) = \phi(k, k-1) r(k-1) \phi^*(k, k-1) + M(k) \tag{31}$$

$$\begin{aligned} m(k,k) &= H(k) r(k) H^*(k) \\ x(k) &= r(k) H^*(k) \end{aligned} \tag{32}$$

for $i = k + 1 \cdots N$ **loop**

$$\begin{aligned} x(i) &= \phi(i, i-1) x(i-1) \\ m(i,k) &= H(i) x(i) \end{aligned} \tag{33}$$

end i **loop end** k **loop**

This recursive technique is equivalent [1] to the composite rigid-body method analyzed in [18] for its computational efficiency. To establish this equivalence, it is necessary [1] to parametrize the matrix $r(k)$ associated with joint k in terms of a minimal set of ten parameters for the composite rigid-body outboard of joint k: one for the mass, three for the mass center location, and six for the rotational inertia. This is yet another indication of the ease with which computationally efficient algorithms emerge from the spatial operator equations.

VI. Innovations Factorization

The mass matrix $\mathcal{M} = H\phi M\phi^* H^*$ cannot be inverted by inverting the individual factors $H\phi$ and $\phi^* H^*$ in the recursive Newton–Euler factorization. The factors are not invertible since they are not even square. An alternative factorization $\mathcal{M} = (\mathcal{I} + H\Phi G)D(\mathcal{I} + H\Phi G)^*$ is now obtained in which the individual factors are square and invertible. This alternative is referred to here as the innovations factorization because of its relationship to the innovations approach [7] of linear least squares filtering and estimation. The discovery of this alternative factorization is one of the central contributions of this paper.

The key ingredients in the innovations factorization are: 1) a discrete-step Riccati equation for the sequence of articulated inertias [1] and 2) the corresponding spatially recursive Kalman filtering equations of [1].

A. Discrete-Step Riccati Equation

The quantities $P(k)$, $D(k)$, and $G(k)$ for $k = 1, \ldots, N$, are defined by the following inward iteration:

$$P(0) = 0; \qquad G(0) = 0 \tag{34}$$

for $k = 1 \cdots N$ **loop**

$$P(k) = \psi(k, k-1)P(k-1)\psi^*(k, k-1) + M(k) \tag{35}$$
$$D(k) = H(k)P(k)H^*(k)$$
$$G(k) = P(k)H^*(k)D^{-1}(k) \tag{36}$$

end loop

where $\psi(k, k-1)$ is defined in (30) below.

This is a discrete Riccati equation driven by the link masses $M(k)$, which produce a sequence of spatial inertias $P(k)$. It can be shown that $P(k) = P^*(k) > 0$ for all k. Hence, the scalar $D(k)$ is always nonzero, and $D^{-1}(k) = 1.0/D(k)$ is guaranteed to exist. The matrix $P(k)$ is the articulated body inertia originally discussed in [19]. $D(k)$ is the projection of the articulated body inertia, $P(k)$, along joint axis k. Define also the noniterative operators

$$\begin{aligned} P &= \text{diag}[P(1), \ldots, P(N)] \\ D &= \text{diag}[D(1), \ldots, D(N)] \\ G &= \text{diag}[G(1), \ldots, G(N)]. \end{aligned} \tag{37}$$

The sequence of Kalman gains $G(k)$ and the corresponding Kalman gain spatial operator $G = PH^*D^{-1}$ are the central elements in the spatially recursive Kalman filter-like algorithms filter described below.

B. Kalman Filter Spatial Operator

The matrix $\psi(i, j)$ in (35) is defined for $i \geq j$ by

$$\psi(k, k) = \mathcal{I} \tag{38}$$
$$\psi(k, k-1) = \phi(k, k-1)[\mathcal{I} - G(k-1)H(k-1)] \tag{39}$$
$$\psi(i, j) = \psi(i, i-1)\psi(i-1, i-2) \cdots \psi(j+1, j); \quad i \geq j. \tag{40}$$

The matrix $\psi(i, j)$ is a Jacobian-like operator associated with articulated bodies that is analogous to the force propagation Jacobian operator, $\phi(i, j)$ for composite bodies (i.e., bodies whose joints are "frozen"). A complete discussion of the physical interpretation and properties of these spatial quantities can be found in [1].

The spatial Kalman filter transition operator is defined in (41), which appears at the bottom of this page. The operator Ψ is a lower block-triangular matrix. This operator will be referred to as a Kalman filter transition operator because its elements $\psi(i, j)$ govern the transitions of forces from one link to the next in a spatially recursive Kalman filter-like algorithm. The operator ψ when post-multiplied by $(\mathcal{I} - GH)$ results in the operator $\dot{\psi}$ of [11].

Identity 2: The matrices $\phi M\phi^*$ and P above are related by

$$\phi M\phi^* = P + \Phi P + P\Phi^* + \Phi PH^*D^{-1}HP\Phi^*. \tag{42}$$

Proof: Observe $[\mathcal{I} - G(k)H(k)]P(k)[\mathcal{I} - G(k)H(k)]^* = [\mathcal{I} - G(k)H(k)]P(k)$. Equation (35) then implies $P(k+1) = \phi(k+1, k)[P(k) - P(k)H^*(k)D^{-1}(k)H(k)P(k)]\phi^*(k+1, k) + M(k+1)$. Hence, $r(k) = P(k) + q(k)$ with $q(k) = \sum_{i=1}^{k-1} \phi(k, i)P(i)H^*(i)D^{-1}(i)H(i)P(i)\phi^*(k, i)$. Also, $r = P + q$, in which $q = \text{diag}[q(1), \ldots, q(N)]$ with $q(1) = 0$. However, $\Phi PH^*D^{-1}HP\Phi^* = q + \Phi q + q\Phi^*$, which can be shown by an argument exactly like the proof of Identity 1.

Identity 3: An alternative factorization of $\mathcal{M} = H\phi M\phi^* H^*$ is the innovations factorization

$$\mathcal{M} = (\mathcal{I} + H\Phi G)D(\mathcal{I} + H\Phi G)^* \tag{43}$$

where $\mathcal{I} + H\Phi G$ is lower block triangular and D is diagonal and invertible.

Proof: Multiply (42) by H and H^* and recall that $D = HPH^*$ and $G = PH^*D^{-1}$.

The objective now is to invert the lower triangular factor $(\mathcal{I} + H\Phi G)$. To do this it is first convenient to establish the following key identity.

Identity 4: The lower triangular operators Ψ and Φ are related by

$$(\mathcal{I} - \Psi GH)\Phi = \Psi. \tag{44}$$

$$\Psi = \begin{pmatrix} 0 & & & & & 0 \\ \phi(2,1) & 0 & & & & 0 \\ \psi(3,2)\phi(2,1) & \phi(3,2) & 0 & & & 0 \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ \psi(N,2)\phi(2,1) & \psi(N,3)\phi(3,2) & \psi(N,4)\phi(4,3) & \cdots & \phi(N,N-1) & 0 \end{pmatrix} \tag{41}$$

Proof: Observe the identity

$$\begin{aligned}\phi(k,m) - \psi(k,m) &= \sum_{i=m}^{k-1} [\psi(k,i+1)\phi(i+1,m) \\ &\quad - \psi(k,i)\phi(i,m)] \\ &= \sum_{i=m}^{k-1} \psi(k,i+1)\phi(i+1,i) \\ &\quad \cdot G(i)H(i)\phi(i,m)\end{aligned}$$

since $\psi(k,m) = \psi(k,m+1)\phi(m+1,m)[\mathcal{I} - G(m)H(m)]$. However, the (k,m)th block element of $\Psi GH\Phi$ is

$$\sum_{i=m}^{k-1} \psi(k,i+1)\phi(i+1,i)G(i)H(i)\phi(i,m).$$

Thus, $\Phi - \Psi = \Psi GH\Phi$.

Identity 5: The lower triangular operators $\mathcal{I} + H\Phi G$ and $\mathcal{I} - H\Psi G$ are mutually reciprocal

$$(\mathcal{I} + H\Phi G)^{-1} = \mathcal{I} - H\Psi G. \tag{45}$$

Proof: Observe that $(\mathcal{I} - H\Psi G)(\mathcal{I} + H\Phi G) = \mathcal{I} - H\Psi G + (\mathcal{I} - H\Psi G)H\Phi G = \mathcal{I}$.

Identities 3 and 5 imply the following factorization of the inverse of the mass matrix.

Identity 6: The operator $\mathcal{M}^{-1}$ can be factored as

$$\mathcal{M}^{-1} = (\mathcal{I} - H\Psi G)^* D^{-1} (\mathcal{I} - H\Psi G). \tag{46}$$

Since the matrix D in the innovations factorization is diagonal, inversion of D is obtained easily by inverting the N scalar diagonal elements $D(k)$. Therefore, inversion of the $N \times N$ mass matrix $\mathcal{M}$ is replaced by the simpler problem of inverting a diagonal matrix D. Furthermore, the factors $(\mathcal{I} - H\Psi G)$ and $(\mathcal{I} - H\Psi G)^*$ in (46) can be mechanized by the spatially recursive filtering and smoothing equations of [1]. This leads to relatively easy recursive solutions to forward dynamics problems.

VII. Forward Dynamics Algorithms Based on the Innovations Factorization

Four closely related forward dynamics algorithms are now obtained. The first algorithm is based on Identity 6 and on the bias-free robot dynamics equations (26).

A. Four-Sweep Bias-Free Algorithm

Algorithm 1:

$$T' = T - H\phi[M\phi^* a + b + Bf(0)] \tag{47}$$

$$\ddot{\theta} = (\mathcal{I} - H\Psi G)^* D^{-1} (\mathcal{I} - H\Psi G) T'. \tag{48}$$

Equation (48) is given by a tip-to-base sweep to produce the vector $\nu = D^{-1}\epsilon$ with $\epsilon = (I - H\Psi G)T'$, followed by a base-to-tip sweep to produce the joint accelerations $\ddot{\theta}$. The vector ϵ is the innovations process, whereas ν is the vector of weighted residuals. The algorithm, essentially that developed in [1] and [19], has been derived here by operator factorizations. Note that the bias-free moments T' in (47) can be computed by means of an outward sweep followed by an inward sweep corresponding to the Newton–Euler algorithm for $\ddot{\theta}$. Algorithm 1 is therefore a "four-sweep" algorithm, which in state-space (algorithmic) form becomes

1) Tip-to-base filtering of bias-free joint moments:

$$\begin{aligned} z(0) &= 0 \\ T'(0) &= 0 \\ G(0) &= 0 \end{aligned} \tag{49}$$

for $k = 1 \cdots N$ **loop**

$$\begin{aligned} z(k) &= \psi(k,k-1)z(k-1) \\ &\quad + \phi(k,k-1)G(k-1)T'(k-1) \end{aligned} \tag{50}$$

$$\epsilon(k) = T'(k) - H(k)z(k)$$

$$\nu(k) = D^{-1}(k)\epsilon(k) \tag{51}$$

end loop

2) Base-to-tip smoothing of weighted residuals:

$$\lambda(N+1) = 0 \tag{52}$$

for $k = N \cdots 1$ **loop**

$$\lambda(k) = \psi^*(k+1,k)\lambda(k+1) + H^*(k)\nu(k) \tag{53}$$

$$\ddot{\theta}(k) = \nu(k) - G^*(k)\phi^*(k+1,k)\lambda(k+1) \tag{54}$$

end loop

The quantities in the above recursions can be interpreted physically. For instance: $z(k)$ is the physical force felt by link k at joint k due to the outboard joint moments, $T(k)$ being nonzero; $\lambda(k)$ is the acceleration that link k has at joint k when the bias terms are zero (i.e., when there is no gravity loading and when velocities are zero so that there are no coriolis/centrifugal forces acting on the link). That is, when the bias terms are zero, $\lambda(k) = \alpha(k)$. Additional physical interpretation of the filtering and smoothing recursions can be found in [1] and [19]. Note that the above forward dynamics algorithm has an operations count that is $O(N)$. For notational simplicity, the need to construct V, a, b, P, D, and G has not been made explicit. It is understood that these quantities are computed during appropriate sweeps of the algorithm [11].

A slight modification of Algorithm 1 leads to an alternative algorithm that, in addition to providing joint accelerations, also produces the link spatial accelerations α and the tip spatial acceleration $\alpha(0)$.

B. Four-Sweep Algorithm to Compute Link and Tip Accelerations

Algorithm 2:

$$T' = T - H\phi[M\phi^* a + b + Bf(0)] \tag{55}$$

$$\alpha = (\mathcal{I} - GH)^* \Psi^* H^* \nu + H^* \nu + \phi^* a \tag{56}$$

$$\alpha(0) = B^* \alpha + a(0). \tag{57}$$

Proof: From (6), (8), (17), and (18), $\alpha = \phi^* H^* \ddot{\theta} + \phi^* a$. Since $\ddot{\theta} = (\mathcal{I} - H\Psi G)^* \nu$, then $\alpha = \phi^* H^* (\mathcal{I} - H\Psi G)^* \nu + \phi^* a$. However, the identities $\phi = \Phi + \mathcal{I}$ and $(\mathcal{I} - \Psi GH)\Phi = \Psi$ imply (56).

This means that a slight modification of the forward dynamics algorithm (49)–(54) allows the computation of the manipulator spatial acceleration α and tip acceleration, $\alpha(0)$, in addition to $\ddot{\theta}$. This occurs by changing (52), (53), and (54) to

$$\lambda(N+1) = 0; \quad \zeta(N+1) = 0 \tag{58}$$

for $k = N \cdots 1$ **loop**

$$\begin{aligned}
\lambda(k) &= \psi^*(k+1,k)\lambda(k+1) + H^*(k)\nu(k) \\
\ddot{\theta}(k) &= \nu(k) - G^*(k)\phi^*(k+1,k)\lambda(k+1) \\
\zeta(k) &= \phi^*(k+1,k)\zeta(k+1) + a(k) \\
\alpha(k) &= \lambda(k) + \zeta(k)
\end{aligned}$$

end loop

$$\alpha(0) = \phi^*(1,0)\alpha(1) + a(0). \tag{59}$$

Algorithms 1 and 2 are both "four-sweep algorithms," two sweeps being required to compute the biases, followed by two sweeps to complete the computation of joint rates or spatial accelerations. The next two are three-sweep algorithms.

C. Three-Sweep Algorithm Based on Biased Equations of Motion

Algorithm 3:

$$\zeta = M\phi^* a + b + Bf(0) \tag{60}$$

$$\epsilon = T - H\Psi[GT + (\mathcal{I} - GH)\zeta] - H\zeta \tag{61}$$

$$\ddot{\theta} = (\mathcal{I} - H\Psi G)^* D^{-1} \epsilon. \tag{62}$$

Proof: From (48) $\epsilon = (\mathcal{I} - H\Psi G)(T - H\phi\zeta)$. However, the identities $\phi = \Phi + \mathcal{I}$ and $(\mathcal{I} - \Psi GH)\Phi = \Psi$ imply (61).

The filtering stage in this algorithm is obtained by modifying (49)–(65) to

$$\begin{aligned}
z(0) &= f(0) \\
T(0) &= 0 \\
G(0) &= 0
\end{aligned} \tag{63}$$

for $k = 1 \cdots N$ **loop**

$$\begin{aligned}
z(k) =& \psi(k,k-1)[z(k-1) + \zeta(k-1)] \\
&+ \phi(k,k-1)G(k-1)T(k-1) \qquad (64) \\
\epsilon(k) =& T(k) - H(k)z(k) - H(k) - \zeta(k) \\
\nu(k) =& D^{-1}(k)\epsilon(k) \qquad (65)
\end{aligned}$$

end loop

Algorithm 3 requires outward-inward-outward sweeps to obtain $\ddot{\theta}$ as indicated respectively by (60), (61), and (62). In Algorithm 3, V and a can be computed during the base-to-tip sweep (60), and b, P, D, and G during the tip-to-base sweep (61). In the same way that Algorithm 2 was derived from Algorithm 1, Algorithm 3 could be modified to compute link and tip spatial accelerations.

D. Three-Sweep Algorithm Not Requiring Prior Computation of D'Alembert Forces

Note that (60) reflects a need to have a preliminary step to compute the D'Alembert forces Ma. The last algorithm given in this section removes this requirement, although the need for a preliminary base-to-tip sweep for the purpose of computing V remains.

$$V = B^* \phi^* H^* \dot{\theta} \tag{66}$$

$$\begin{aligned}
\nu =& D^{-1}(\mathcal{I} - H\Psi G)T - D^{-1}H\psi \\
&\cdot [(\mathcal{I} - GH)Pa + b + Bf(0)] + D^{-1}HPa \qquad (67)
\end{aligned}$$

$$\ddot{\theta} = (\mathcal{I} - H\Psi G)^* \nu - G^* \Psi^* (\mathcal{I} - GH)^* a - G^* a. \tag{68}$$

This algorithm is established in [11] where it is also shown that (67) and (68) can be implemented as

$$\begin{aligned}
z(0) &= f(0) \\
G(0) &= 0 \\
P(0) &= 0 \\
T(0) &= 0
\end{aligned} \tag{69}$$

for $k = 1 \cdots N$ **loop**

$$\begin{aligned}
z(k) =& \psi(k,k-1)[z(k-1) + P(k-1)a(k-1)] \\
&+ \phi(k,k-1)G(k-1)T(k-1) + b(k) \qquad (70) \\
\epsilon(k) =& T(k) - H(k)z(k); \quad \nu(k) = D^{-1}(k)e(k) \qquad (71)
\end{aligned}$$

end loop for $k = N \cdots 1$ **loop**

$$\begin{aligned}
\lambda(k) =& \psi^*(k+1,k)\lambda(k+1) + H^*(k)\nu(k) \\
&+ [\mathcal{I} - G(k)H(k)]^* a(k) \qquad (72) \\
\ddot{\theta}(k) =& \nu(k) - G^*(k)\phi^*(k+1,k)\lambda(k+1) - G^*(k)a(k) \\
& \qquad (73)
\end{aligned}$$

end loop

The three-sweep algorithm (66)–(68) can also be found in [1], where it is derived by what is referred to as the "sweep" method to solve boundary-value problems. It is possible to apply the tools of this paper to obtain a two-sweep forward dynamics algorithm, avoiding the sweep contained in (60), although at the expense of greater algorithmic complexity. Another two-sweep algorithm can also be obtained by computing (66) and (68) in the same base-to-tip sweep. This, however, requires that a slightly delayed value of the spatial velocity V be tolerated.

VIII. Extension to a Mobile Base

A fictitious joint $N+1$ is introduced which can be at any prescribed location in the base link. Typically, this joint is at the base link mass center, but it could be at any other point. Associated with the joint is the joint-axes projection operator

$$H^*(N+1) = \mathcal{I} \in R^{6\times 6}. \tag{74}$$

Note that this choice of $H(N+1)$ implies that the base is fully mobile in all six degrees of freedom. There is no loss of generality in this. By appropriate selection of $H(N+1)$, base motion in fewer degrees of freedom can be analyzed [11].

A. Augmented Spatial Acceleration Vector

The base moves with the spatial velocity $V(N+1)$ and the spatial acceleration $\alpha(N+1)$ given by

$$V(N+1) = \text{col}[\omega, v]$$
$$\dot{V}(N+1) = \text{col}[\dot{\omega}, \dot{v}] \tag{75}$$

in which ω and v are, respectively, the angular and linear velocities with respect to an inertial reference, and $\dot{\omega}$ and $\dot{v}$ are the corresponding accelerations. It is convenient to collect all of the accelerations associated with the independent (joint plus base) degrees of freedom in the system into the following augmented acceleration vector:

$$\dot{W} = \text{col}[\ddot{\theta}(1), \cdots, \ddot{\theta}(N), \dot{V}(N+1)]. \tag{76}$$

This vector contains the set of N joint-angle accelerations followed by the last element $\dot{V}(N+1)$ representing the base acceleration. The problem of forward dynamics is to compute the augmented acceleration $\dot{W}$ in (76), given the set of forces that are applied to the system. This means that the base acceleration $\dot{V}(N+1)$ in (75) is an unknown quantity that must be determined together with all of the joint accelerations $\ddot{\theta}$.

B. Augmented Applied Force Vector

It is assumed that a prescribed force $T(N+1)$ is applied at the fictitious joint $N+1$. Due to (12), this force is related to the spatial force $f(N+1)$ at the same joint by

$$T(N+1) = H(N+1)f(N+1) = f(N+1) \in R^6. \tag{77}$$

This force is put together with the remaining joint moments $T(1), \cdots, T(N)$ in order to form the augmented applied force vector

$$T = \text{col}[T(1), \cdots, T(N), T(N+1)]. \tag{78}$$

The forward dynamics problem can now be stated as that of computing the augmented acceleration vector $\dot{W}$, given the augmented applied force vector T.

C. Composite Manipulator Dynamics

The extension is now complete and $\dot{W}$ and T are related by

$$T = \mathcal{M}\dot{W} + \mathcal{C} + J^* f(0)$$
$$\dot{W} = \mathcal{M}^{-1}[T - \mathcal{C} - J^* f(0)]. \tag{79}$$

The operator forms and interpretations of $\mathcal{M}$, $\mathcal{C}$, and J^* still hold. In particular, these operators can be mechanized recursively by the operator factorization and inversion given in previous sections of this paper. The only change is in the dimensions of the operators.

IX. Hierarchical Factorization Software

The mass matrix factorization results outlined here have a built-in hierarchical architecture that leads to very simple reconfigurable computer programs [21]. The programs translate user-defined blocks of high-level operators into more detailed algorithms and programs. Fig. 3 illustrates schematically the operation of a forward dynamics program based on the innovations factorization.

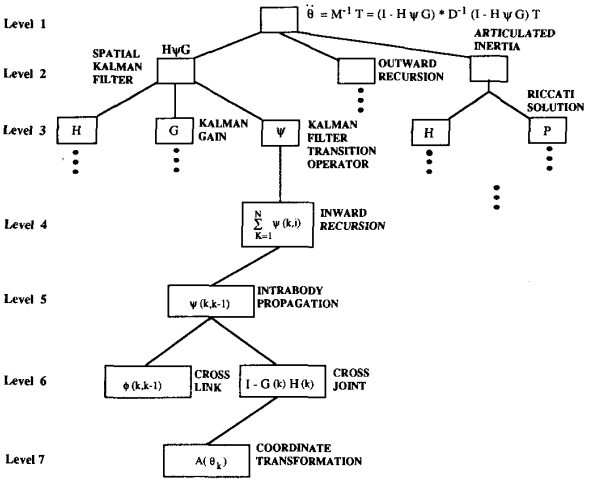


Fig. 3. Spatial operators map to efficient recursive algorithms.

At the highest level (Level 1 in Fig. 3) of abstraction, the user inputs the factorization $\ddot{\theta} = (\mathcal{I} - H\Psi G)^* D^{-1}(\mathcal{I} - H\Psi G)T$. This input requires that the program decompose the operators H, Ψ, and G that appear at this level. This decomposition occurs at Level 2 in the hierarchy. At the next level, labeled Level 3 in the figure, it is recognized that Ψ is what is referred to as the Kalman filter transition operator. This in turn implies that Ψ can be mechanized using an inward Kalman filtering recursion, shown at Level 4, from the tip of the manipulator to the base. This recursion is built up as a sequence of transitions in each of the links characterized by the operator $\psi(k, k-1)$ appearing at Level 5. There are two transitions per link at Level 6: 1) propagation of forces within a link represented by the operator $\phi(k, k-1)$ and 2) computing the effects of crossing a joint involving the operator $\mathcal{I} - G(k)H(k)$. This second transition calls the necessary coordinate transformation to go from one link to the next. This coordinate transformation occurs at Level 7, the lowest level in the hierarchy.

This programming approach achieves a very high level of abstraction. The number of symbols visible to the analyst at any level in the hierarchy is very small. This means that the corresponding computer programs are also very simple. The programs are modular and map to a modular software architecture. The programs are arm independent in the sense that going from one arm to another arm is easy to do.

Although the programs are simple, computational efficiency is not lost. Embedded in the programs are efficient algorithms equivalent to those of [1], [16], [17], [19], [20]. Other computations, such as inverse kinematics and trajectory design, have also been implemented [21]. An investigation of the number of arithmetical operations for these algorithms is presented in [30] for rigid multilink manipulators and in [25] for flexible multibody systems. Additional efficiency can be gained by taking advantage of the choice of coordinate frames and of the specific structure of the manipulator.

X. Relationship to Other Work

The primary contribution of this paper is to show that there are two alternative recursive factorizations of the manipulator mass matrix. These factorizations are referred to as the recursive Newton–Euler factorization and the innovations factorization. These two factorizations embedd in a high-level analytical framework, that of spatial operators, very efficient algorithms for inverse and forward kinematics. Furthermore, the spatial operator approach reveals very simple abstract structural properties of the manipulator mass matrix that cannot be seen easily from the detailed algorithms. The highly abstract spatial operator notation reduces the number of symbols needed to solve a given dynamics problem. This leads to a higher level abstract methodology [11] for complexity management in modeling, analysis, software development, robot programming, simulation, and control of complex robotic systems. This augments significantly the analytical tools available [31] to analyze general multibody system dynamics.

A. Significance of the Recursive Newton–Euler Factorization

The recursive Newton–Euler factorization $\mathcal{M} = H\phi M\phi^* H^*$ embedds the following efficient inverse dynamics algorithms: 1) recursive Newton–Euler and 2) the composite rigid-body method for recursive evaluation of the manipulator mass matrix.

The Newton–Euler recursion for inverse dynamics is, of course, a very well established [16] algorithm in robotics. This paper sheds additional light on this algorithm by showing that it is equivalent to the Newton–Euler mass matrix factorization in the sense that both the algorithm and the factorization can be derived easily from each other. This leads to what is believed to be the simplest derivation found to date of the equivalence of Lagrangian and recursive Newton–Euler dynamics. This equivalence is embedded in the single spatial operator equation $\mathcal{M} = H\phi M\phi^* H^*$. Establishing this result by more detailed methods [15] requires significantly more work.

Closely related to the recursive Newton–Euler algorithms of [16] is the composite-body method for inverse dynamics analyzed in [18] for its numerical efficiency. This method consists of an inward iteration that begins at the tip of the manipulator and ends at its base. For every joint, it computes the mass, mass center location, and rotational inertia of the composite body outboard of that joint. This method can be recovered easily from the single spatial operator identity $\phi M\phi^* = r + \Phi r + r\Phi^*$ which has been stated as Identity 1 of Section V. This is another example of a very efficient algorithm being embedded in the high-level spatial operator equations.

The recursive Newton–Euler factorization of this paper is closely related to a mass matrix factorization in [32]. When the notation of this paper is used, the factorization in [32] becomes $\mathcal{M} = \mathcal{P}\mathcal{P}^*$, where $\mathcal{P}$ is lower triangular. The recursive Newton–Euler factorization presented here shows that $\mathcal{P}$ of [32] has an operator factorization $\mathcal{P} = H\phi M^{1/2}$. This factorization shows explicitly the force propagation embedded in the operator $\mathcal{P}$. This complements the results of [32], where $\mathcal{P}$ is evaluated numerically and the related forward dynamics problem is solved numerically using Householder transformations.

B. Significance of the Innovations Factorization

The innovations factorization $\mathcal{M} = (\mathcal{I} + H\Phi K)D(\mathcal{I} + H\Phi K)^*$ and its corresponding inverse embedd the recursive algorithms for forward dynamics of [1] and [19]. They also embedd recursive algorithms for computation of the mass matrix inverse as discussed in [1] and [13]. These algorithms are order N in the sense that the number of arithmetic computations grows only linearly with the number of degrees of freedom. They also can be cast [1] within the highly developed filtering and smoothing algorithm architecture of state estimation theory. Algorithms are easy to implement in computer programs because the filtering and smoothing architecture can be used as a global guide or road map in program development. The numerical stability of the algorithms can also be evaluated easily because computational characteristics of filters and smoothers are very well understood [33].

One of the main contributions of the present paper is to show that the filtering and smoothing algorithms for forward dynamics are embedded in the innovations factorization. The algorithms can be derived using the single operator equation $\mathcal{M}^{-1} = (\mathcal{I} - H\Psi G)^* D^{-1}(\mathcal{I} - H\Psi G)$ expressing the innovations factorization of the mass matrix inverse. Development of the forward dynamics algorithms using more detailed methods [1], [19], [20] requires significantly more work.

C. Relationship to Numerical Factorization Methods

There is a critical difference between the factorization results of this paper and common [34] numerical factorization techniques, such as LDU decomposition or Cholesky factorization, of positive definite matrices. The key difference is that the spatial operator approach used here leads to *model-based factorizations*, in which the manipulator model itself is used to conduct the factorizations and related inversions. Both the recursive Newton–Euler factorization and the innovations factorization are model based. This means that every computational step has a corresponding physical interpretation. Intermediate quantities, such as articulated inertias, Jacobian operations, projections along joint axes, etc., that appear at certain steps of the factorization and inversion process can easily be checked for consistency with physical understanding and intuition. Another way to state this is to say that the spatial operator factorizations are completely symbolic, in the sense that there is a symbolic expression for every step in the computations. Further evidence of this is that selection of coordinate frames is not needed to state the factorizations.

Numerical factorization techniques [34] can of course be applied to factor and invert the manipulator mass matrix, since this matrix is a special case of a general positive definite symmetric matrix. This approach would begin by first assembling the mass matrix numerically by obtaining explicit numerical values for each of its elements. Any of a number of methods, the recursive Newton–Euler factorization of this paper for example, could be used to do this. Then

standard numerical techniques [34] could be used to compute the mass matrix inverse. This approach is strictly numerical. The physical model of the manipulator is not used at all in conducting the mass matrix inversion, although it may be used to some extent in the initial evaluation of mass matrix itself. Because the inversion process is not model based, not every step has a corresponding physical interpretation. This means that intermediate steps are not easy to interpret physically, and there is possible loss of physical insight and understanding.

Standard methods for numerical inversion for positive definite matrices, of course, are quite useful in solving manipulator dynamics problems. These techniques have a solid analytical foundation [34] in numerical analysis and are widely used in many applications. The factorization results of this paper do not aim to replace the numerical techniques as general-purpose tools for matrix inversion. Rather, the aim is to point out and take advantage of certain properties of the manipulator mass matrix that make it distinct from more general positive definite matrices. The major distinctive characteristic is that the manipulator mass matrix emerges from manipulator mechanics, whereas a more general positive definite matrix typically does not. The manipulator mass matrix is generated by a manipulator model. This allows use of the model itself to determine the computations required for mass matrix inversion. In this sense, the innovations factorization then represents a new way, one not found in standard numerical linear algebra references [34], to obtain a model-based and numerical Cholesky-like factorization of the mass matrix and its corresponding inverse.

XI. Concluding Remarks

This paper advances two linear operator factorizations of the manipulator mass matrix. Embedded in the factorizations are many of the techniques that are regarded as very efficient computational solutions to inverse and forward dynamics problems. The operator factorizations provide a high-level architectural understanding of the mass matrix and its inverse, which is not visible in the detailed algorithms. They also lead to a new approach to the development of computer programs to organize complexity in robot dynamics.

Acknowledgment

Thanks are due A. Jain of the Jet Propulsion Laboratory for many discussions and suggested improvements to this paper.

References

[1] G. Rodriguez, "Kalman filtering, smoothing and recursive robot arm forward and inverse dynamics," *IEEE J. Robotics Automat.*, vol. RA-3, no. 6, pp. 624–639, Dec. 1987.

[2] ——, "Recursive forward dynamics for multiple robot arms moving a common task object," *IEEE J. Robotics Automat.*, vol. 5, no. 4, pp. 510–521, Aug. 1989.

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Trans. J. Basic Eng.*, vol. D, pp. 35–45, Mar. 1960.

[4] A. E. Baggeroer, "A state-variable technique for the solution of Fredholm integral equations," *IEEE Trans. Info. Theory*, vol. IT-15, pp. 557–570, 1969.

[5] A. Schumitzky, "On the Equivalence Between Riccati Equations and Fredholm Resolvents," *J. Computer SysT. Sci.*, vol. 2, pp. 76–87, 1968.

[6] T. Kailath, "Fredholm resolvents, Wiener–Hopf equations, and Riccati differential equations," *IEEE Trans. Info. Theory*, vol. IT-15, pp. 665–672, 1969.

[7] ——, "The innovations approach to detection and estimation theory," *Proc. IEEE*, vol. 58, pp. 680–695, 1970.

[8] ——, "A view of three decades of linear filtering theory," *IEEE Trans. Info. Theory*, vol. IT-20, pp. 147–181, 1974.

[9] M. Gevers and T. Kailath, "An innovations approach to least-squares estimation—Part VI: discrete-time innovations representations and recursive estimation," *IEEE Trans. Automat. Control*, vol. AC-18, pp. 588–600, 1973.

[10] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[11] G. Rodriguez and K. Kreutz, "Recursive mass matrix factorization and inversion: An operator approach to manipulator forward dynamics," JPL Pub. 88–11, 1988.

[12] G. Rodriguez, K. Kreutz-Delgado, and A. Jain, "A spatial operator algebra for manipulator modeling and control," *Int. J. Robotics Res.*, Aug. 1991.

[13] G. Rodriguez, "Random field estimation approach to robot dynamics," *IEEE Trans. Syst. Man Cybern.*, vol. 20, no. 5, pp. 1081–1093, Sept. 1990.

[14] A. Jain, "Unified formulation of dynamics for serial rigid multibody systems," *J. Guidance, Control Dynam.*, May 1991.

[15] D. B. Silver, "On the equivalence of Lagrangian and Newton–Euler dynamics for manipulators," *Int. J. Robotics Res.*, vol. 1, 1982.

[16] J. Luh, M. Walker, and R. Paul, "On-line computational scheme for mechanical manipulators," *ASME J. Dynam. Syst., Measurement, Control*, vol. 102, no. 2, pp. 69–76, 1980.

[17] M. Walker and D. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *ASME J. Dynam. Syst., Measurement, Control*, vol. 104, pp. 205–211, Spring 1982.

[18] C. G. Lee and P. R. Chang, "Efficient parallel algorithms for robot inverse dynamics computations," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-16, 1986.

[19] R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *Int. J. Robotics Res.*, vol. 2, no. 1, pp. 13–30, 1983.

[20] M. G. Nasser, "Recursive Newton–Euler formulation of manipulator dynamics," in *Proc. NASA Conf. Space Telerobotics* (JPL Pub. 89–7), Jan. 1989, pp. 309–318.

[21] K. Kreutz and A. Jain, "Ada dual arm spatial algebra packages for closed chain motion and force control," Jet Propulsion Lab., Pasadena, CA, Eng. Memo. 347–89–266, Nov. 1989.

[22] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE J. Robotics Automat.*, vol. RA-3, pp. 43–53, Feb. 1987.

[23] K. Kreutz-Delgada, A. Jain, and G. Rodriguez, "Recursive formulation of operational space control," in *Proc. 3rd ISRAM Int. Symp. Robotics Manufact.* (Vancouver, Canada), July 1990.

[24] G. Rodriguez, "Spatial operator approach to flexible multibody manipulator inverse and forward dynamics," in *Proc. IEEE Int. Conf. Robotics Automat.* (Cincinnati, OH), May 1990.

[25] A. Jain, "Recursive dynamics for flexible multibody systems using spatial operators," in *Proc. JPL IEEE Int. Conf. Robotics Automat.*, Apr. 1991.

[26] A. Jain and G. Rodriguez, "Recursive dynamics of geared robotic manipulators," in *Proc. IEEE Conf. Decision Control* (Honolulu, HI), Dec. 1990.

[27] ——, "Recursive linearization of manipulator dynamics models," in *Proc. IEEE Conf. Syst. Man Cybern.* (Los Angeles), Nov. 1990.

[28] J. T. Wen and K. Kreutz, "Globally stable tracking control laws for the attitude maneuver problem," in *Proc. 1988 Automat. Control Conf.* (Atlanta, GA), 1988.

[29] G. Rodriguez, "Statistical mechanics models for motion and force planning," in *Proc. SPIE Conf. Intelligent Control Adaptive Syst.* (Philadelphia, PA), Nov. 1989.

[30] A. Fijany and A. K. Bejczy, "An efficient algorithm for computation of the manipulator inertia matrix," *J. Robotic Syst.*, vol. 7, pp. 57–80, Feb. 1990.

[31] J. Wittenberg, *Dynamics of Systems of Rigid Bodies*. Stuttgart: B. G. Teubner, 1977.

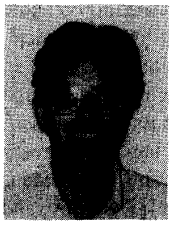
[32] J. Angeles and O. Ma, "Dynamic simulation of n-axis serial robotic manipulators using a natural orthogonal complement," *Int. J. Robotics Res.*, vol. 7, no. 5, pp. 32–47, Oct. 1988.

[33] J. Charlier and P. Van Dooren, "A systolic algorithm for Riccati and Lyapunov equations," *Math. Control, Signals, Syst.*, vol. 2, no. 2, pp. 109–136, 1989.

[34] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins Univer. Press, 1983.

Guillermo Rodriguez (S'64–M'84) received the Ph.D. degree in control theory from the University of California at Los Angeles in 1974.

He has been with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, since graduating. He has participated in the development of on-board guidance and control systems for several automnomous planetary spacecraft. He has also been the supervisor of technical groups conducting research in the control of large flexible space systems. He is currently the supervisor of a group responsible for research in machine intelligence applications to space telerobtoics. His research interests include estimation theory, control of multibody systems, and control architectures for autonomous robots.



Kenneth Kreutz-Delgado received the M.S. degree in physics and the Ph.D. degree in engineering system science from the University of California at San Diego (UCSD), La Jolla.

He is currently on the faculty of the UCSD Electrical and Computer Engineering Department. Prior to his appointment at UCSD, he was a researcher at the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, where he worked on the development of intelligent telerobotic systems. He is affiliated with both the California Space Institute and the UCSD Institute for Neural Computation.

Dr. Kreutz-Delgado is a member of the AAAS and of the IEEE Societies on Robotics and Automation, Computers, Control, and Systems, Man, and Cybernetics. In 1990, he received a National Science Foundation Presidential Young Investigator Award.