A Fast Recursive Algorithm for Molecular Dynamics Simulation

A. Jain
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

N. Vaidehi
A. A. Noyes Laboratory of Chemical Physics
California Institute of Technology
Pasadena, CA 91125

G. Rodriguez
Jet Propulsion Laboratory/California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109

Journal of Computational Physics, vol. 106, pp. 258-268, June 1993.

Subject classification: 65P99, 82A71.

Keywords: Molecular dynamics, algorithms, simulation.

Running head: A Fast Algorithm for MD Simulation

Please address all correspondence to:

Abhinandan Jain
Jet Propulsion Laboratory
M/S 198-219
4800 Oak Grove Drive
Pasadena, CA 91109

A Fast Recursive Algorithm for Molecular Dynamics Simulation

A. Jain, N. Vaidehi and G. Rodriguez

Abstract

In this paper, we develop a recursive algorithm for solving the dynamical equations of motion for molecular systems. We make use of internal variable models which have been shown to reduce the computation times of molecular dynamics simulations by an order of magnitude when compared with Cartesian models. The $O(\mathcal{N})$ algorithm described in this paper for solving the equations of motion provides additional significant improvements in computational speed. We make extensive use of the spatial operator methods which have been developed recently for the analysis and simulation of the dynamics of multibody systems. The spatial operators are used to derive the equations of motion and obtain an operator expression for the system mass matrix. An alternative square factorization of the mass matrix leads to a closed form expression for its inverse. From this follows the recursive algorithm for computing the generalized accelerations. The computational cost of this algorithm grows only linearly with the number of degrees of freedom. This is in contrast to conventional constrained dynamics algorithms whose cost is a cubic function of the number of degrees of freedom. For the case of a polypeptide molecule with 400 residues, the $O(\mathcal{N})$ algorithm provides computational speedup by a factor of 450 over the conventional $O(\mathcal{N}^3)$ algorithm. We also describe a simplified method for computing and handling the potential function gradients within the dynamics computations.

1 Introduction

Molecular dynamics (MD) simulations are being used increasingly to model structure–function relationships of chemical reactivity of macromolecules and to study the structural evolution of proteins, nucleic acids and other polymers [1–3]. However, large computational times are a significant limiting factor on the time–scales over which MD simulations are currently feasible. Reference [4] contains an overview of some of the literature on the development of dynamics models and computational techniques for molecular systems.

The Cartesian model for molecular systems is widely used due to its simplicity and ease of application [3,5,6]. In this model, all the atoms in the system are treated as free particles and their accelerations are computed independently using Newton's law. Thus, the equations of motion for this model are highly decoupled.

Hard constraints on bonds are often used to eliminate from molecular models lightly excited degrees of freedom (e.g., inter–atomic oscillations, rotations about double bonds) and those that have insignificant effect on long time-scale processes such as conformational changes in macromolecules [4,7–12]. This is particularly important for high-frequency degrees of freedom since they force the use of small integration step–sizes which severely limit the time–scales for MD simulations. While constrained models may be unsuitable for studying transition states during chemical reactions, models with varying levels of constraints can be very useful for large time simulations.

With the addition of constraints, the equations of motion for the Cartesian model are no longer ordinary differential equations (ODEs), but are instead more complex differential–algebraic equations (DAEs). The SHAKE algorithm is widely used to handle these inter–atomic constraints [6]. In this algorithm, the "unconstrained" ODE component of the equations of motion is used to compute an initial estimate of the change in the conformational state of the system. This estimate is modified iteratively until it satisfies the constraints to within acceptable error limits.

An alternative approach, for molecular models with constraints, is to use internal coordinates to directly incorporate the constraints into the dynamical model [4,9,10]. In such *internal variable models*, the number of generalized coordinates for the molecular system is smaller than in the corresponding Cartesian models. The internal variable model we describe and use in this paper consists of rigid subunits of atoms – referred to as *clusters* – whose relative motion is described by the internal coordinates. We limit our attention here to tree–topology (branched) molecular models, that is, models in which the clusters do not form any closed loops. Individual clusters can however contain closed loops formed by atoms. For tree topology molecular models, the generalized coordinates are of minimum dimension and the equations of motion are ODEs rather than the DAEs obtained using Cartesian models. The integration of the equations of motion is thus considerably simpler.

The equations of motion for an internal variable model for a tree–topology molecular system with $\mathcal{N}$ degrees of freedom can be expressed as follows:

$$\mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta, \dot{\theta}) = T(\theta) \tag{1.1}$$

Here θ denotes the $\mathcal{N}$ dimensional vector of generalized coordinates, T is the $\mathcal{N}$ dimensional vector of generalized forces, $\mathcal{M}$ is the $\mathcal{N} \times \mathcal{N}$ symmetric and positive definite mass matrix, and $\mathcal{C}$ is the $\mathcal{N}$ dimensional vector of Coriolis and other (velocity dependent) nonlinear forces. While the dimension of the mass matrix $\mathcal{M}$ is less than that for the Cartesian model, it is no longer diagonal, and it also depends nonlinearly upon θ. Lagrangian analysis has been used to derive expressions for the components of $\mathcal{M}$ and $\mathcal{C}$ [4, 9, 10]. The dynamics algorithm in [10] for computing the acceleration vector $\ddot{\theta}$ consists of computing $\mathcal{M}$, $\mathcal{C}$ and T explicitly and then solving the linear matrix equation in (1.1). Due to the coupled structure of $\mathcal{M}$, the computational cost of this step is typically a cubic function of $\mathcal{N}$. The cost of this $O(\mathcal{N}^3)$ algorithm increases rapidly for large molecules. Even though an order of magnitude reduction in computational time is achieved compared to the conventional Cartesian model, the sharp increase in computational cost as molecular size increases is a major limiting factor for the internal variables approach.

In this paper we describe a fast recursive algorithm for solving the equations of motion for internal variable molecular models. The computational cost of this algorithm is a linear function of the number of degrees of freedom. Thus, computation times are significantly reduced. The algorithm does not require either the explicit computation of the mass matrix $\mathcal{M}$ or the explicit solution of the linear matrix equation in (1.1). It also provides a simplified method for computing the forces arising from the potential functions by eliminating the need to compute the gradient of the Cartesian potential functions with respect to the internal variables.

The algorithm is based on the recently developed *spatial operator algebra* which has been used for the analysis and high–speed simulation of the dynamics of complex multibody systems such as robots, spacecraft, and vehicles [13, 14]. This methodology uses spatial operators to concisely derive the equations of motion, to reduce the complexity of dynamics models, and to develop fast recursive computational algorithms. We include here for completeness the relevant aspects of the formulation and derivation of the algorithm and refer the reader to the above references for additional details. The spatial operator algebra algorithms are very similar to those used in the well–studied areas of Kalman filtering and smoothing [15]. Thus, a substantial body of knowledge exists about issues such as numerical stability and software architectures. Extensions of the algorithm to closed–topology molecular models, i.e. models containing clusters in closed loops are straightforward [14].

2 Dynamical Models for Molecular Systems

All methods for simulating the dynamics of molecular systems require two major computational steps: (i) the computation of forces for the current conformational state of the molecule; (ii) the solution of the equations of motion and their subsequent integration for the calculation of the trajectories.

The forces of interaction between atoms in a molecule are typically described by means of potential functions. The potentials are classified as being due to bonding or non–bonding interactions. The bond potential is classically taken to be a quadratic, harmonic or a Morse–type potential. The non–bonded interactions include van der Waal attraction, electrostatic interaction, dipole–dipole interaction, and dispersion forces [3,6]. Several levels of quantum mechanical calculations of the potential energy surface are available [16–18]. The expression for the potential function PE can be written as the combination of a function of Cartesian coordinates, $\mathcal{P}[x]$, and another function of internal coordinates, $\mathcal{P}[\theta]$, as follows:

$$PE = \mathcal{P}[\theta] + \mathcal{P}[x]$$

The gradients of $\mathcal{P}[x]$ and $\mathcal{P}[\theta]$ are used to calculate the inter–atomic forces required for solving the equations of motion.

The free atom Cartesian model for molecules constitutes the simplest of possible dynamics models with each atom regarded as an unconstrained point mass particle. Let n denote the number of atoms in the molecular model. Once the $3n$–dimensional vector of inter–atomic forces f is computed, the acceleration of each atom is obtained using Newton's law. The equations of motion for the atoms in the Cartesian model can be written in matrix form as:

$$\mathcal{M}_c \ddot{x} = f \tag{2.1}$$

where the mass matrix $\mathcal{M}_c$ is a $3n \times 3n$ diagonal matrix whose diagonal elements are the masses of the atoms, while $x \in \mathbb{R}^{3n}$ is the vector of Cartesian coordinates of the atoms.

In order to increase integration step size, hard constraints are often imposed on the lightly excited degrees of freedom in the molecular model. Assuming that there are m such constraints, they can be expressed as instantaneous linear constraints on the atomic velocities as follows:

$$A\dot{x} = B \tag{2.2}$$

Here $A \in \mathbb{R}^{m \times 3n}$ denotes the constraint matrix and $B \in \mathbb{R}^{m}$ is the constraint vector and in general they both depend upon the configuration of the molecule. Solving for the accelerations, $\ddot{x}$, requires the solution of (2.1) subject to the constraint defined by (2.2). Due to the constraints, the equations of motion are now no longer ordinary differential equations (ODEs) but are instead the more complicated differential–algebraic equations (DAEs). The computation of the accelerations $\ddot{x}$ can be carried out using iterative methods such as the SHAKE algorithm [6] or by using Lagrange multipliers.

An alternative approach for handling constraints is to use (2.2) to numerically eliminate some of the components of $\dot{x}$ from (2.1) to obtain a generalized velocity vector $\dot{x}_r$ of minimal dimension $\mathcal{N} \overset{\triangle}{=} 3n - m$. $\mathcal{N}$ represents the number of degrees of freedom remaining in the molecular model after the imposition of the constraints. This results in a dimensionally reduced form of the equations of motion:

$$\mathcal{M}_r \ddot{x}_r = f_r \tag{2.3}$$

There is no longer a separate equation for the constraints and the equations of motion are once again in the form of ODEs. While the size of the mass matrix $\mathcal{M}_r$ is $\mathcal{N} \times \mathcal{N}$ and of minimal dimension, it no longer has a simple diagonal structure.

Lower dimensional (and an ODE form) for the equations of motion for molecular models with constraints can also be obtained in a more natural manner by using *internal variable models*. The use of internal variables preserves key structural properties of the mass matrix and provides computational advantages over the model defined by (2.3).

3 Internal Variable Models

In this section we describe an internal variable molecular model consisting of *clusters of atoms* coupled together with the permissible relative motion between them being partially constrained. A *cluster* is defined to be a rigid body formed by a collection of atoms or a section of a molecule with frozen covalent structure such as the benzene rings in polystyrene and the rings of certain amino acids in proteins and polypeptides. A single atom can be regarded as a cluster with zero extent and zero rotational inertia.

The topological structure of internal variable models for a molecular system can vary based upon the nature of the study. For instance, the tyrosine residue in the the BPTI side–chains can be considered as rigid for studying the overall motion of BPTI. In this case the molecular model will have a tree-topology structure (i.e. a structure with branching but no closed loops formed by the clusters). However, when the side–chain motions are studied (which are known to be important in the function of the protein [2]) they should be modeled as being flexible and closed–chain topology models are more appropriate. For simplicity, we restrict our attention here to tree–topology molecular models. The internal variable modeling scheme described here does not require the use of virtual atoms and virtual bonds [10].

One particular cluster in the tree is designated the *base* cluster. It is slightly advantageous computationally to choose as the base cluster one that minimizes the maximum branch length (as measured in number of clusters), i.e., the base cluster atom should be somewhat mid–centered.

We adopt a *parent/child* designation for adjoining clusters – with the one on the path to the base being the *parent* cluster and the other one being the *child* cluster. While each cluster has a unique parent, it can have zero or more children. The parent/child designation is completely determined for all the clusters in a molecule once a base cluster has been designated. In Figure 1, if cluster A is chosen as the base cluster, it is the parent of clusters B and C, and dually, B and C are its children. However, if cluster B is chosen as the base cluster, then cluster A is the child of cluster B, while cluster C is the child of cluster A.

We use the concept of a *hinge* to characterize the permissible relative motion between adjoining clusters. Each hinge can possess between zero and six degrees of freedom of relative motion. A zero degree of freedom hinge corresponds to clusters that are coupled together rigidly. On the other hand, a six degree of freedom hinge implies that there is no coupling between the two clusters as is the case for a pair of independent molecules. A bond for which only changes in the torsional bond angle are permitted is modeled as a one degree of freedom rotational hinge. On the other hand, clusters coupled by a bond whose length alone can vary is modeled as a one degree of freedom translational (or sliding) hinge.

As shown in Figure 1, for each hinge, we assign a coordinate reference frame to each of the clusters coupled by the hinge and designate them $\mathcal{O}^+$ and $\mathcal{O}^-$. The motion of the hinge is characterized by the motion of these hinge frames with respect to each other. For a hinge with m degrees of freedom, θ denotes the vector of m generalized coordinates which describe the state of the hinge. In the case of a one degree of freedom rotational hinge, θ is just the value of the hinge angle. For a one degree of freedom translational hinge, θ is the value of the bond displacement. The time derivative vector $\dot{\theta}$ provides an obvious and common choice for the generalized hinge velocity vector β. However, for multiple degree of freedom hinges, alternative choices for β – referred to as *differentials of quasi-coordinates* [19] – help simplify the equations of motion and are often preferable. A case in point is a 3 degree of freedom (ball and socket) rotational hinge for which the θ vector may consist of an Euler angle or quaternion representation for the hinge orientation. For this case, the use of the relative angular velocity in place of $\dot{\theta}$ as the generalized velocity vector simplifies the equations of motion. The full unconstrained mobility of the base cluster of a molecule is modeled by a 6 degree of freedom hinge between the base cluster and the inertial frame. The generalized velocity vector for this hinge is simply the 6–dimensional spatial velocity vector [1] of the base cluster. In general, there is always an (invertible) kinematic function, $\Pi(\theta)$, that maps the hinge velocity β to $\dot{\theta}$ via $\dot{\theta} = \Pi(\theta)\beta$. For the sake of notational simplicity and with no loss in generality, we assume here that β is indeed $\dot{\theta}$ for all the hinges.

---

[1] The definition of spatial velocities and other spatial quantities is given in Appendix A.

The six dimensional relative spatial velocity across a hinge can be written in the form $H^*\dot{\theta}$, where H^* is the $6 \times m$ dimensional *hinge matrix* representing the relative motion characteristics of the m degrees of freedom hinge. Some examples of the structure of the H^* matrix are given below:

$$\begin{array}{ccc} \begin{pmatrix}0\\0\\1\\0\\0\\0\end{pmatrix} & \begin{pmatrix}0\\0\\0\\0\\0\\1\end{pmatrix} & \begin{pmatrix}0&0\\0&0\\1&0\\0&0\\0&0\\0&1\end{pmatrix} \\ \text{Torsional mode alone} & \text{Stretching mode alone} & \text{Torsional \& Stretching modes} \\ \text{(1 dof)} & \text{(1 dof)} & \text{(2 dof)} \end{array}$$

In the above examples, the components of the H matrix remain constant in the $\mathcal{O}^+$ and $\mathcal{O}^-$ frames for arbitrary hinge motions. This is true even for 6 degree of freedom hinges for whom H^* is the 6×6 identity matrix. In other instances of multiple degree of freedom hinges, the axis of motion of a component degree of freedom can depend upon the motion of other component degrees of freedom. When this is the case, we decompose the hinge degrees of freedom into component hinges and separate them by *pseudo-clusters* (clusters with zero mass and zero extent) such that the component hinges can be characterized by constant H matrices. Thus for the example of a hinge with free bond and torsion angle degrees of freedom, we introduce a pseudo-cluster so that there is only a 1 degree of freedom bond angle hinge between the inboard cluster and the pseudo–cluster, and a 1 degree of freedom torsional hinge between the pseudo–cluster and the outboard cluster. We will assume below that this type of a decomposition has been carried out throughout the molecular model.

4 Equations of Motion

In this section, we describe the equations of motion for internal variable models for molecular systems. We make extensive use of the spatial operator algebra approach that has been used in the past for multibody dynamics applications [13,14]. We adopt a Newton–Euler approach since it readily reveals the inherent recursive relationships among the various dynamical quantities. We use coordinate–free spatial notation in our development and a brief overview of some of the definitions and notation is described in Appendix A.

At first we restrict our attention to branchless molecular models, i.e. models whose structure is in the form of a serial chain. Later, we describe the straightforward extensions required to handle tree topology molecular models. In a serial chain, each atom cluster has a unique parent and a unique child, and this helps simplify the notation.

We assume that the serial chain contains n clusters and that the total number of degrees of freedom in the system is $\mathcal{N}$. One of the extremal clusters is designated the base cluster, and the cluster at the other end of the chain is referred to as the "tip" cluster. We assign numbers from 1 through n to the clusters in increasing order from the tip to the base. The k^{th} hinge in the serial chain couples together the $(k+1)^{th}$ and k^{th} clusters, and associated with it are the pair of hinge frames $\mathcal{O}_k^+$ and $\mathcal{O}_k^-$ shown in Figure 2. The generalized coordinates and the hinge matrix for the k^{th} hinge are given by $\theta(k)$ and $H^*(k)$ respectively. The parent and child of the k^{th} cluster are the $(k+1)^{th}$ and the $(k-1)^{th}$ clusters respectively. The point denoted CM on each cluster designates its center of mass. The mobility of the base cluster (the n^{th} cluster) is accounted for by attaching a 6 degree of freedom hinge between it and the inertial frame. The index $(n+1)$ is used for the inertial frame.

The spatial velocity, $V(\mathcal{O}_k^-)$, of frame $\mathcal{O}_k^-$ can be computed recursively as described by the following equation (see (A.3)):

$$V(\mathcal{O}_k^-) = \phi^*(\mathcal{O}_{k+1}^-, \mathcal{O}_k^-)V(\mathcal{O}_{k+1}^-) + H^*(k)\dot{\theta}(k) \tag{4.1}$$

Here $\phi(\mathcal{O}_{k+1}^-, \mathcal{O}_k^-)$ is a spatial transformation matrix described inAppendix A. We use the notational shorthand $x(k)$ to denote a quantity of the form $x(\mathcal{O}_k^-)$ in the remainder of this paper. Thus, (4.1) can be rewritten as:

$$V(k) = \phi^*(k+1,k)V(k+1) + H^*(k)\dot{\theta}(k) \tag{4.2}$$

The recursive expression for the spatial acceleration $\alpha(k)$ of frame $\mathcal{O}_k^-$ is obtained by taking the time derivative of both sides of (4.2)

$$\alpha(k) = \dot{V}(k) = \phi^*(k+1,k)\alpha(k+1) + H^*(k)\ddot{\theta}(k) + a(k) \tag{4.3}$$

where the expression for the Coriolis acceleration term $a(k)$ is given by:

$$a(k) = \begin{pmatrix} 0 \\ \tilde{\omega}(k+1)[v(k) - v(k+1)] \end{pmatrix} + \begin{pmatrix} \tilde{\omega}(k) & 0 \\ 0 & \tilde{\omega}(k) \end{pmatrix} H^*(k)\dot{\theta}(k) \tag{4.4}$$

The notation $\tilde{x}$ denotes the cross product tensor associated with a 3–dimensional vector x. Since $H(k)$ is constant in the frames $\mathcal{O}_k^-$ and $\mathcal{O}_k^+$, $\omega(k+1)$ can be used in place of $\omega(k)$ in the second term on the right hand side of (4.4).

The gradient of the potential function, PE $(= \mathcal{P}[x] + \mathcal{P}[\theta]))$, with respect to the generalized coordinates $\theta(k)$ gives the vector of effective generalized forces for the k^{th} hinge. These are usually computed for all the hinges prior to solving the equations of motion. Due to its functional form, the computation of

$$T(k) \overset{\triangle}{=} \nabla_{\theta(k)} \mathcal{P}[\theta] \tag{4.5}$$

is straightforward. On the other hand, the computation of the gradient $\nabla_{\theta(k)}\mathcal{P}[x]$ is more complex and reference [10] describes a procedure for its computation. Our algorithm avoids the computation of $\nabla_{\theta(k)}\mathcal{P}[x]$. In its place, we make use of the simpler gradient of $\mathcal{P}[x]$ with respect to the Cartesian position of the atoms. With $x_i(k)$ denoting the position of the i^{th} atom of the k^{th} cluster, the gradient, $\hat{f}_i(k) = \nabla_{x_i(k)}\mathcal{P}[x]$, denotes the 3-dimensional Cartesian force acting on the atom. The Cartesian forces for all the atoms in a cluster are combined together to yield a single effective 6-dimensional Cartesian spatial force on the cluster. If the k^{th} cluster has $r(k)$ atoms, the effective spatial force $\hat{f}_c(k)$ on the cluster is given by

$$\hat{f}_c(k) \overset{\triangle}{=} \begin{pmatrix} \sum_{i=1}^{r(k)} \tilde{l} \left[\mathcal{O}_k^-, x_i(k)\right] \hat{f}_i(k) \\ \sum_{i=1}^{r(k)} \hat{f}_i(k) \end{pmatrix} \tag{4.6}$$

where $l\left[\mathcal{O}^-(k), x_i(k)\right]$ is the vector from the k^{th} hinge frame $\mathcal{O}_k^-$ to the location of the i^{th} atom of the k^{th} cluster. We assume that the gradients $T(k)$ and $\hat{f}_c(k)$ have been computed and are available for all the clusters and hinges.

With $f(k)$ denoting the spatial force of interaction at $\mathcal{O}_k^-$ between the $(k+1)^{th}$ and the k^{th} clusters, and $M(k)$ denoting the spatial inertia of the k^{th} cluster about the frame $\mathcal{O}_k^-$, the equations of motion of the k^{th} cluster about $\mathcal{O}_k^-$ are (see (A.5)):

$$\begin{aligned} f(k) &= \phi(k, k-1) f(k-1) + M(k)\alpha(k) + b(k) + \hat{f}_c(k) \\ T(k) &= H(k) f(k) \end{aligned} \tag{4.7}$$

$b(k)$ is the spatial gyroscopic force for the k^{th} cluster (defined in (A.6)). The incorporation of the gradients $T(k)$ and $\hat{f}_c(k)$ in separate ways into the equations of motion in (4.7) allows us to avoid computing the gradient $\nabla_{\theta(k)}\mathcal{P}[x]$.

Putting together (4.2)–(4.7), leads to the following Newton-Euler recursive equations of motion for the whole system:

$$\left\{\begin{array}{rcl} & & V(n+1)=0, \quad \alpha(n+1)=0 \\ \textbf{for } \boldsymbol{k} & = & \boldsymbol{n}\cdots\mathbf{1} \\ V(k) & = & \phi^*(k+1,k)V(k+1)+H^*(k)\dot{\theta}(k) \\ \alpha(k) & = & \phi^*(k+1,k)\alpha(k+1)+H^*(k)\ddot{\theta}(k)+a(k) \\ \textbf{end loop} & & \end{array}\right.$$

$$\left\{\begin{array}{rcl} & & f(0)=0 \\ \textbf{for } \boldsymbol{k} & = & \mathbf{1}\cdots\boldsymbol{n} \\ f(k) & = & \phi(k,k-1)f(k-1)+M(k)\alpha(k)+b(k)+\hat{f}_c(k) \\ T(k) & = & H(k)f(k) \\ \textbf{end loop} & & \end{array}\right. \tag{4.8}$$

We now introduce *spatial operators* and use them to express the equations of motion in a more concise form. The operators $M \in \mathbb{R}^{6n\times 6n}$ and $H \in \mathbb{R}^{\mathcal{N}\times 6n}$ are block diagonal matrices defined as $M = \text{diag}\left\{M(1) \ \cdots \ M(n)\right\}$ and $H = \text{diag}\left\{H(1) \ \cdots \ H(n)\right\}$, while the operator $\mathcal{E}_\phi$ is defined as the following lower–triangular matrix:

$$\mathcal{E}_\phi \quad \triangleq \quad \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \phi(2,1) & 0 & \dots & 0 & 0 \\ 0 & \phi(3,2) & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \phi(n,n-1) & 0 \end{pmatrix} \in \mathbb{R}^{6n\times 6n} \tag{4.9}$$

The 0 entries denote 6 × 6 dimensional zero matrices. We use the notation V to denote the $6n$–dimensional stacked column vector $[V^*(1) \ \cdots \ V^*(n)]^*$. Similarly we define the vectors θ, α, a and so on by stacking up the contributions from each cluster. Using these newly defined quantities, it is easy to see that the recursive equation for $V(k)$ in (4.8) can be collectively rewritten as:

$$V = \mathcal{E}_\phi^* V + H^*\dot{\theta} \tag{4.10}$$

The inverse of $[I - \mathcal{E}_\phi]$ can be readily shown to be the lower–triangular spatial operator ϕ given by

$$\phi \triangleq [I - \mathcal{E}_\phi]^{-1} = \begin{pmatrix} I & 0 & \dots & 0 \\ \phi(2,1) & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \phi(n,1) & \phi(n,2) & \dots & I \end{pmatrix} \in \mathbb{R}^{6n\times 6n} \tag{4.11}$$

where,

$$\phi(i,j) \overset{\triangle}{=} \phi(i,i-1)\ \cdots\ \phi(j+1,j) \quad \text{for} \quad i > j$$

In (4.11), the 0 and I entries denote 6×6 dimensional zero and identity matrices respectively. Using (4.11) in (4.10) leads to the following equation for V:

$$V = \phi^* H^* \dot{\theta} \tag{4.12}$$

The special structure of the spatial operators allows high-level operator expressions involving them to be directly mapped into recursive algorithms, and the explicit computation of their elements is not required. The correspondence between operator expressions and recursive computations is illustrated by the equivalence of the operator expression in (4.12) and the recursive expression for $V(k)$ in (4.8).

Continuing along these lines, all the component level equations in (4.8) can be rewritten using operators as follows:

$$\begin{aligned} V &= \phi^* H^* \dot{\theta} \\ \alpha &= \phi^*(H^* \ddot{\theta} + a) \\ f &= \phi(M\alpha + b + \hat{f}_c) = \phi M \phi^* H^* \ddot{\theta} + \phi(M\phi^* a + b + \hat{f}_c) \\ T &= Hf = H\phi M \phi^* H^* \ddot{\theta} + H\phi(M\phi^* a + b + \hat{f}_c) \end{aligned} \tag{4.13}$$

In particular, the equations of motion have the form

$$T = \mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta, \dot{\theta}) \tag{4.14}$$

where

$$\mathcal{M}(\theta) \overset{\triangle}{=} H\phi M \phi^* H^* \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}} \tag{4.15a}$$

$$\mathcal{C}(\theta, \dot{\theta}) \overset{\triangle}{=} H\phi(M\phi^* a + b + \hat{f}_c) \in \mathbb{R}^{\mathcal{N}} \tag{4.15b}$$

Here, $\mathcal{M}$ is the *mass matrix* of the serial chain and $\mathcal{C}$ is the vector of Coriolis, centrifugal, gyroscopic and Cartesian forces. Note that $\mathcal{M}$ and $\mathcal{C}$ are nonlinear functions of θ and $\dot{\theta}$. The factorization in (4.15a) of the mass matrix $\mathcal{M}$ is referred to as the *Newton–Euler Operator Factorization* [13] because it is equivalent to the Newton–Euler recursive algorithm in (4.8).

The solution of the equations of motion in (4.14) for the accelerations vector $\ddot{\theta}$ is required by the integrator to propagate the state of the system during MD simulations. However, (4.14) represents only a conceptual statement of the dynamics problem since $\mathcal{M}$ and $\mathcal{C}$ are not explicitly available. The conventional approach for computing the accelerations $\ddot{\theta}$ consists of first computing both $\mathcal{M}$ and $\mathcal{C}$, and solving the linear matrix equation for the vector $\ddot{\theta}$. In general, $\mathcal{M}$ is fully populated and as a result, the computational cost of solving the equations of motion using this method grows cubically with the number of degrees of freedom in the system, i.e., this is an $O(\mathcal{N}^3)$ method. Even though an order of magnitude reduction in computational times is achieved compared to the conventional Cartesian model based methods [10], the cubic dependency leads to large computational costs for large molecules.

In the next section we describe a recursive algorithm for computing the vector of generalized accelerations $\ddot{\theta}$ without having to explicitly compute the mass matrix. The complexity of this method is only $O(\mathcal{N})$, i.e., its computational cost grows only linearly with the number of degrees of freedom in the model.

5 Recursive Solution of Equations of Motion

An operator factorization of the system mass matrix $\mathcal{M}$, denoted the *Innovations Operator Factorization*, is derived in this section. This factorization is an alternative to the Newton–Euler factorization in (4.15a) and, in contrast with the latter, the factors in the Innovations factorization are square and invertible. Operator expressions for the inverse of these factors are derived and lead to an expression for the inverse of the mass matrix. Using further operator identities, we obtain an operator expression for the generalized accelerations $\ddot{\theta}$. The recursive implementation of this expression leads to the $O(\mathcal{N})$ recursive dynamics algorithm. Following this, we describe the extensions required to handle tree topology molecular models and the computational cost for the algorithm.

5.1 Operator Expression for the Mass Matrix Inverse

Given below is a recursive algorithm which defines some required quantities such as $P(k)$, $D(k)$ etc. for each of the clusters:

$$\left\{\begin{array}{rcl} & P^{+}(0) = 0 \in \mathbb{R}^{6\times 6} & \\ \textbf{for } \boldsymbol{k} & \boldsymbol{=} & \boldsymbol{1 \cdots n} \\ P(k) & = & \phi(k,k-1)P^{+}(k-1)\phi^{*}(k,k-1) + M(k) \\ D(k) & = & H(k)P(k)H^{*}(k) \\ G(k) & = & P(k)H^{*}(k)D^{-1}(k) \\ K(k+1,k) & = & \phi(k+1,k)G(k) \\ \overline{\tau}(k) & = & I - G(k)H(k) \\ P^{+}(k) & = & \overline{\tau}(k)P(k) \\ \psi(k+1,k) & = & \phi(k+1,k)\overline{\tau}(k) \\ \textbf{end loop} & & \end{array}\right. \tag{5.1}$$

This recursive equation for $P(k)$ is a discrete Riccati equation and has been extensively studied in control and estimation theory [20]. The operator $P \in \mathbb{R}^{6n\times 6n}$ is defined as a block diagonal matrix with the k^{th} diagonal element being $P(k)$. The quantities defined in (5.1) form the component elements of the following spatial operators:

$$\begin{array}{rcl} D & \stackrel{\triangle}{=} & HPH^{*} = \text{diag}\Big\{D(k)\Big\} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}} \\ G & \stackrel{\triangle}{=} & PH^{*}D^{-1} = \text{diag}\Big\{G(k)\Big\} \in \mathbb{R}^{6n\times\mathcal{N}} \\ K & \stackrel{\triangle}{=} & \mathcal{E}_{\phi}G \in \mathbb{R}^{6n\times\mathcal{N}} \\ \overline{\tau} & \stackrel{\triangle}{=} & I - GH = \text{diag}\Big\{\overline{\tau}(k)\Big\} \in \mathbb{R}^{6n\times 6n} \end{array} \tag{5.2}$$

The only nonzero block elements of K are the $K(k+1,k)$ elements along the first sub-diagonal. We define the operators $\mathcal{E}_\psi$ and ψ as follows:

$$\mathcal{E}_\psi \overset{\triangle}{=} \mathcal{E}_\phi \overline{\tau} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ \psi(2,1) & 0 & \dots & 0 & 0 \\ 0 & \psi(3,2) & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \psi(n,n-1) & 0 \end{pmatrix} \in \mathbb{R}^{6n \times 6n} \tag{5.3}$$

$$\psi \overset{\triangle}{=} [I - \mathcal{E}_\psi]^{-1} = \begin{pmatrix} I & 0 & \dots & 0 \\ \psi(2,1) & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \psi(N,1) & \psi(N,2) & \dots & I \end{pmatrix} \in \mathbb{R}^{6n \times 6n} \tag{5.4}$$

where

$$\psi(i,j) \overset{\triangle}{=} \psi(i,i-1) \ \cdots \ \psi(j+1,j) \quad \text{for} \quad i > j \tag{5.5}$$

The structure of the operators $\mathcal{E}_\psi$ and ψ is identical to that of the operators $\mathcal{E}_\phi$ and ϕ except that the component elements are now $\psi(i,j)$'s rather than $\phi(i,j)$'s. Also, the elements of ψ have the same semigroup properties ((5.5)) as the elements of the operator ϕ, and as a consequence, high-level operator expressions involving them can be directly mapped into recursive algorithms. The explicit computation of all the elements of the operator ψ is not required.

The Innovations Operator Factorization of the mass matrix is an alternative to the factorization in (4.15a) and is described by the following lemma.

Lemma 5.1

$$\mathcal{M} = [I + H\phi K] D [I + H\phi K]^* \tag{5.6}$$

Proof: See Appendix B. ∎

Note that the factor $[I + H\phi K] \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ is square, block lower triangular and nonsingular, while D is a block diagonal matrix. This factorization provides a closed form operator expression for the block LDL^* decomposition of $\mathcal{M}$. The following lemma gives the closed form operator expression for the inverse of the factor $[I + H\phi K]$.

Lemma 5.2

$$[I + H\phi K]^{-1} = [I - H\psi K] \tag{5.7}$$

Proof: See Appendix B. ∎

It follows from Lemmas 5.1 and 5.2 that the operator expression for the inverse of the mass matrix is given by:

Lemma 5.3

$$\mathcal{M}^{-1} = [I - H\psi K]^* D^{-1} [I - H\psi K] \tag{5.8}$$

∎

Once again, note that the factor $[I - H\psi K]$ is square, block lower triangular and nonsingular and so Lemma 5.3 provides a closed form expression for the block LDL^* decomposition of $\mathcal{M}^{-1}$.

5.2 Recursive Computational Algorithm

The following lemma describes the operator expression for the generalized accelerations $\ddot{\theta}$ in terms of the hinge forces T and Cartesian spatial forces $\hat{f}_c$.

Lemma 5.4

$$\ddot{\theta} = [I - H\psi K]^* D^{-1} \Big[T - H\psi\{KT + Pa + b + \hat{f}_c\}\Big] - K^*\psi^* a \tag{5.9}$$

Proof: See Appendix B. ∎

(5.9) can be decomposed into the following sequence of expressions:

$$\begin{aligned} z &= \psi[KT + Pa + b + \hat{f}_c] \\ \epsilon &= T - Hz \\ \nu &= D^{-1}\epsilon \\ \alpha &= \psi[H^*\nu + a] \\ \ddot{\theta} &= \nu - K^*\alpha \end{aligned} \tag{5.10}$$

The recursive implementation of (5.10) leads to the following $O(\mathcal{N})$ computational algorithm for the accelerations, $\ddot{\theta}$:

$$\left\{\begin{array}{rcl} & z^{+}(0) = 0 & \\ \textbf{for } \boldsymbol{k} & = & \boldsymbol{1 \cdots n} \\ z(k) & = & \phi(k,k-1)z^{+}(k-1) + P(k)a(k) + b(k) + \hat{f}_c(k) \\ \epsilon(k) & = & T(k) - H(k)z(k) \\ \nu(k) & = & D^{-1}(k)\epsilon(k) \\ z^{+}(k) & = & z(k) + G(k)\epsilon(k) \\ \textbf{end loop} & & \end{array}\right. \tag{5.11a}$$

$$\left\{\begin{array}{rcl} & \alpha(n+1) = 0 & \\ \textbf{for } \boldsymbol{k} & = & \boldsymbol{n \cdots 1} \\ \alpha^{+}(k) & = & \phi^{*}(k+1,k)\alpha(k+1) \\ \ddot{\theta}(k) & = & \nu(k) - G^{*}(k)\alpha^{+}(k) \\ \alpha(k) & = & \alpha^{+}(k) + H^{*}(k)\ddot{\theta}(k) + a(k) \\ \textbf{end loop} & & \end{array}\right. \tag{5.11b}$$

This algorithm does not require either the explicit computation of the mass matrix $\mathcal{M}$, nor the numerical solution of the matrix equation (4.14). The steps in the above algorithm can be summarized as follows:

1. The first step is a recursion from the base to the tip to compute the orientation, location and spatial velocities, $V(k)$, and the Coriolis and gyroscopic terms $a(k)$ and $b(k)$ for each of the clusters using (4.2), (4.4) and (A.6).
2. Next follows a recursion from the tip towards the base as defined by (5.1) to compute the $P(k)$'s etc.
3. The recursion in (5.11a) from the tip to the base is used next to compute the residual forces $z(k)$ etc. This recursion can be combined with the tip to base recursion in the previous step to obtain a single tip to base recursion sequence.
4. Finally, the base to tip recursion described by (5.11b) computes the $\ddot{\theta}(k)$ accelerations for all the clusters.

The computational cost of this algorithm depends only linearly on the number of clusters. This is discussed in more detail in Section 5.4. The structure of this algorithm closely resembles those found in Kalman filtering and smoothing theory [15, 20].

This algorithm has been implemented and tested quite extensively for robot and spacecraft dynamics applications. It is currently being used for MD simulations and the results from the ongoing studies will be reported in a forthcoming publication.

5.3 Extensions to Branched Molecular Structures

The dynamics algorithm described above is for molecular models with unbranched, serial chain structure. The extension of the algorithm to molecular models with general tree–topology structure is simple [14]. In tree–topology models, each cluster can have more than one child. Thus, the structure of the algorithm now consists of a tips-to-base recursion followed by a base-to-tips recursion. The changes required in the recursion steps are:

- During a tips–to–base inward recursion, at each cluster, the results from each of the children clusters are summed up before proceeding with the recursion.
- During a base–to–tips outward recursion, at each cluster, the recursions are continued separately along the outgoing children branches.

Extensions of the algorithm to molecular models with closed topology can be carried out using the analysis and algorithms described in reference [14]. A major step of the algorithm for closed–chain models requires solving the equations motion of a tree–topology subsystem using precisely the algorithm described here.

5.4 Computational Costs

As mentioned earlier, the dynamics algorithm in Section 5.2 is of $O(\mathcal{N})$ complexity, i.e., its complexity grows only linearly with the number of degrees of freedom. For a *fixed number of degrees of freedom*, $\mathcal{N}$, the computational cost is maximal for a molecular system with serial chain structure, no point mass clusters and only single degree of freedom hinges. For such systems, the computational cost in floating point operations is roughly $500\mathcal{N}$. The presence of point–masses, multiple degree of freedom hinges, or branches reduces the computational cost.

Recall that the $O(\mathcal{N}^3)$ method requires first the computation of the mass matrix $\mathcal{M}$ and the vector $\mathcal{C}$ followed by the solution of a linear matrix equation of size $\mathcal{N}$. The computational cost in floating point operations for such $O(\mathcal{N}^3)$ algorithms is given roughly by the polynomial $\mathcal{N}^3/3 + 19\mathcal{N}^2 + 350\mathcal{N}$ for a serial chain system with single degree of freedom hinges and no point–masses.

Figure 3 compares the computational cost of the $O(\mathcal{N})$ algorithm with the $O(\mathcal{N}^3)$ algorithm. The computational efficiency of the $O(\mathcal{N})$ algorithm increases rapidly with the size of the molecule. Consider the example of a polypeptide molecule with each residue regarded as a rigid atomic cluster. We assume that there are two bending degrees of freedom between neighboring clusters. For this molecular system, the number of degrees of freedom is approximately twice the number of residues. For a polypeptide molecule with 400 residues, the cost for the $O(\mathcal{N}^3)$ algorithm is approximately 450 times larger than that for the $O(\mathcal{N})$ algorithm.

6 Conclusions

The use of constrained molecular models allow the use of significantly larger integration time steps during molecular dynamics simulations when compared with Cartesian models. We have studied the use of internal variable molecular models to handle inter–atomic constraints, and have developed a fast recursive $O(\mathcal{N})$ algorithm for solving the equations of motion for these models. The spatial operator algebra methods used here were originally developed for the analysis and high–speed dynamics simulation of complex multibody systems such as robots, spacecraft and vehicles. The $O(\mathcal{N})$ algorithm is obtained by developing operator expressions for the factorization and inversion of the system mass matrix. This leads to closed form expressions for the generalized accelerations. The algorithm is recursive in nature and consists of sweeps from the tips–to–the–base and from the base–to–the–tips of the molecular model.

In contrast with the $O(\mathcal{N}^3)$ algorithms typically used for internal variable models, the $O(\mathcal{N})$ algorithm developed here requires neither the computation of the mass matrix nor the solution of the linear matrix equation in (1.1). Instead, it directly solves for the accelerations of the system. Since it solves the equations of motion exactly, it provides improved numerical stability over iterative techniques such as the SHAKE algorithm.

Only gradients of the potential functions with respect to the natural coordinates are required by the algorithm. These gradients are considerably simpler to compute than the more generally used gradients with respect to internal coordinates. This simplifies the computation of the inter–atomic forces.

Based upon prior work [14] this algorithm extends readily and naturally to molecular models with closed topologies. This approach also allows changing of constraints during simulations and can be useful for modeling events such as the making and breaking of bonds.

7 Acknowledgments

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

Appendix A: Spatial Notation

The use of spatial quantities considerably simplifies the expressions and analysis of the equations of motion of multibody systems (see [21] for more details). The *spatial velocity*, $V(\mathcal{O})$, of a coordinate frame is a 6-dimensional vector defined by combining its 3-dimensional angular and linear velocity vectors $\omega(\mathcal{O})$ and $v(\mathcal{O})$. Similarly, the *spatial force*, $f(\mathcal{O})$, is a 6-dimensional vector which combines the moment and force components $N(\mathcal{O})$ and $F(\mathcal{O})$. The specific definitions are as follows:

$$V(\mathcal{O}) \triangleq \begin{pmatrix} \omega(\mathcal{O}) \\ v(\mathcal{O}) \end{pmatrix}, \quad \text{and} \quad f(\mathcal{O}) \triangleq \begin{pmatrix} N(\mathcal{O}) \\ F(\mathcal{O}) \end{pmatrix} \tag{A.1}$$

The *spatial acceleration*, $\alpha(\mathcal{O})$, of frame $\mathcal{O}$ is the time derivative, $\dot{V}(\mathcal{O})$, of the spatial velocity vector. We denote by $l(\mathcal{O}_x, \mathcal{O}_y)$ the 3-dimensional vector from the origin of frame $\mathcal{O}_x$ to the origin of frame $\mathcal{O}_y$. Associated with a pair of such frames is also a 6×6 *spatial transformation matrix*, $\phi(\mathcal{O}_x, \mathcal{O}_y)$ defined as follows:

$$\phi(\mathcal{O}_x, \mathcal{O}_y) \triangleq \begin{pmatrix} I_3 & \tilde{l}(\mathcal{O}_x, \mathcal{O}_y) \\ 0_3 & I_3 \end{pmatrix} \tag{A.2}$$

In the above, I_3 and 0_3 denote 3×3 dimensional identity and zero matrices respectively, and the notation $\tilde{x}$ denotes the cross–product tensor associated with a 3–dimensional vector x. The transformation matrix $\phi(\mathcal{O}_x, \mathcal{O}_y)$ transforms spatial quantities between the two frames $\mathcal{O}_x$ and $\mathcal{O}_y$ as shown below:

$$f(\mathcal{O}_x) = \phi(\mathcal{O}_x, \mathcal{O}_y) f(\mathcal{O}_y), \quad \text{and} \quad V(\mathcal{O}_y) = \phi^*(\mathcal{O}_x, \mathcal{O}_y) V(\mathcal{O}_x) \tag{A.3}$$

The symbol $*$ denotes the matrix transpose operation. The *spatial inertia*, $M(\mathcal{O})$ of a rigid body about the frame $\mathcal{O}$ is defined as

$$M(\mathcal{O}) \triangleq \begin{pmatrix} \mathcal{J}(\mathcal{O}) & m\tilde{p} \\ -m\tilde{p} & mI_3 \end{pmatrix} \tag{A.4}$$

Here m denotes the mass of the body, p the vector from frame $\mathcal{O}$ to the body's center of mass, and $\mathcal{J}(\mathcal{O})$ is the 3×3 inertia matrix for the body about the reference frame $\mathcal{O}$.

With $f(\mathcal{O})$ denoting the effective spatial force on a rigid body about frame $\mathcal{O}$, the equations of motion about the frame can be written as follows:

$$f(\mathcal{O}) = M(\mathcal{O})\alpha(\mathcal{O}) + b(\mathcal{O}) \tag{A.5}$$

where the *gyroscopic spatial force* $b(k)$ has the form

$$b(\mathcal{O}) = \begin{pmatrix} \tilde{\omega}(\mathcal{O})\mathcal{J}(\mathcal{O})\omega(\mathcal{O}) \\ m\tilde{\omega}(\mathcal{O})\tilde{\omega}(\mathcal{O})l(\mathcal{O}, \mathcal{O}_{CM}) \end{pmatrix} \tag{A.6}$$

Appendix B: Proofs of the Lemmas

The proofs of the lemmas in this publication are closely parallel those described in references [13,21] for rigid multibody systems.

Proof of Lemma 5.1: It is easy to verify that $\overline{\tau}P\overline{\tau}^* = \overline{\tau}P$. As a consequence, the recursion for $P(.)$ in (5.1) can be rewritten in the form

$$M \;=\; P - \mathcal{E}_\psi P\mathcal{E}_\psi^* = P - \mathcal{E}_\psi P\mathcal{E}_\phi^* = P - \mathcal{E}_\phi P\mathcal{E}_\phi^* + KDK^* \tag{B.1}$$

Pre– and post–multiplying the above by ϕ and ϕ^* respectively then leads to

$$\phi M\phi^* = P + \tilde{\phi}P + P\tilde{\phi}^* + \phi KDK^*\phi^*$$

Hence,

$$\begin{aligned} \mathcal{M} \;&=\; H\phi M\phi^* H^* = H[P + \tilde{\phi}P + P\tilde{\phi}^* + \phi KDK^*\phi^*]H^* \\ &=\; D + H\phi KD + DK^*\phi^*H^* + H\phi KDK^*\phi^*H^* = [I + H\phi K]D[I + H\phi K]^* \end{aligned}$$

∎

Proof of Lemma 5.2: Using a standard matrix identity we have that

$$[I + H\phi K]^{-1} = I - H\phi[I + KH\phi]^{-1}K \tag{B.2}$$

Note that

$$\psi^{-1} = I - \mathcal{E}_\psi = (I - \mathcal{E}_\phi) + \mathcal{E}_\phi GH = \phi^{-1} + KH \tag{B.3}$$

from which it follows that

$$\psi^{-1}\phi = I + KH\phi$$

Using this with (B.2) it follows that

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

■

Proof of Lemma 5.4: From (4.13) the expression for the generalized accelerations $\ddot{\theta}$ is given by

$$\ddot{\theta} \quad = \quad \mathcal{M}^{-1}(T - \mathcal{C}) = [I - H\psi K]^* D^{-1}[I - H\psi K]\Big[T - H\phi[M\phi^* a + b + \hat{f}_c]\Big] \tag{B.4}$$

From (B.3) we have that

$$[I - H\psi K]H\phi = H\psi[\psi^{-1} - KH]\phi = H\psi \tag{B.5}$$

Thus (B.4) can be written as

$$\ddot{\theta} = [I - H\psi K]^* D^{-1}\Big[T - H\psi[KT + M\phi^* a + b + \hat{f}_c]\Big] \tag{B.6}$$

From (B.1) it follows that

$$M = P - \mathcal{E}_\psi P \mathcal{E}_\phi^* \implies \psi M \phi^* = \psi P + P\tilde{\phi}^* \tag{B.7}$$

and so (B.6) simplifies to

$$\ddot{\theta} = [I - H\psi K]^* D^{-1}\Big[T - H\psi[KT + Pa + b + \hat{f}_c] - HP\tilde{\phi}^* a\Big] \tag{B.8}$$

From (B.3) we have that

$$[I - H\psi K]^* D^{-1} HP\tilde{\phi}^* = [I - H\psi K]^* K^* \phi^* = K^* \psi^* [\psi^{-*} - KH]^* \phi^* = K^* \psi^* \tag{B.9}$$

Using this in (B.8) leads to the result. ■

References

[1] M. Karplus and G. Petsko *Science*, vol. 347, pp. 631–639, 18 October 1990.

[2] C. Brooks, M. Karplus, and B. Pettitt, *A Theoretical Perspective of Dynamics, Structure and Thermodynamics.* Wiley, New York, 1988.

[3] A. Warshel, *Computer Simulation of Chemical Reactions in Enzymes and Solutions.* John Wiley & Sons, 1991.

[4] K. Gibson and H. Scheraga *J. Comp. Chem.*, vol. 11, no. 4, pp. 468–486, 1990.

[5] B. Brooks, R. Bruccoleri, B. Olafson, D. States, S. Swaminathan, and M. Karplus *J. Comp. Chem.*, vol. 4, no. 2, pp. 187–217, 1983.

[6] J. Mcammon and S. Harvey, *Dynamics of Proteins and Nucleic Acids.* Cambridge University Press, Cambridge, 1987.

[7] M. Pear and J. Weiner *J. Chem. Phys.*, vol. 71, pp. 212–224, July 1979.

[8] R. Edberg, D. Evans, and G. Morriss *J. Phys. Chem.*, vol. 84, p. 6933, 1986.

[9] A. Mazur and R. Abagyan *J. of Biomolecular Structure& Dynamics*, vol. 6, no. 4, pp. 815–832, 1989.

[10] A. Mazur, V. Dorofeev, and R. Abagyan *Journal of Computational Physics*, vol. 92, pp. 261–272, Feb. 1991.

[11] J. Durup *J. Phys. Chem.*, vol. 95, no. 4, pp. 1817–1829, 1991.

[12] J. Perram and H. Petersen *Molecular Physics*, vol. 65, no. 4, pp. 861–874, 1988.

[13] G. Rodriguez, K. Kreutz-Delgado, and A. Jain *The International Journal of Robotics Research*, vol. 10, pp. 371–381, Aug. 1991.

[14] G. Rodriguez, A. Jain, and K. Kreutz-Delgado *Journal of the Astronautical Sciences*, vol. 40, pp. 27–50, Jan.–March 1992.

[15] B. D. O. Anderson and J. B. Moore, *Optimal Filtering.* Prentice-Hall Inc., 1979.

[16] J. Chandrasekar, S. Smith, and W. Jorgensen *J. Am. Chem. Soc.*, vol. 107, p. 155, 1985.

[17] J. Hwang, G. King, S. Creighton, and A. Warshel *J. Am. Chem. Soc.*, vol. 110, p. 5297, 1988.

[18] N. Vaidehi and A. Warshel *J. Chem. Phys.*, 1991. In press.

[19] L. Meirovitch, *Methods of Analytical Dynamics*. McGraw-Hill, New York, 1970.

[20] G. Rodriguez *IEEE Journal of Robotics and Automation*, vol. 3, pp. 624–639, Dec. 1987.

[21] A. Jain *Journal of Guidance, Control and Dynamics*, vol. 14, pp. 531–542, May–June 1991.

Figure Captions

1. Illustration of clusters in molecular models and the hinges that couple them
2. A molecular model consisting of atomic clusters coupled together in a serial chain
3. Comparison of the computational cost (in floating point operations) for the $O(\mathcal{N})$ and $O(\mathcal{N}^3)$ algorithms for solving the equations of motion

Footnotes

1. The definition of spatial velocities and other spatial quantities is given in Appendix A.

Figure 1: Illustration of clusters in molecular models and the hinges that couple them

A. Jain, N. Vaidehi, G. Rodriguez
Journal of Computational Physics

Figure 2: A molecular model consisting of atomic clusters coupled together in a serial chain

A. Jain, N. Vaidehi, G. Rodriguez
Journal of Computational Physics

Figure 3: Comparison of the computational cost (in floating point operations) for the $O(\mathcal{N})$ and $O(\mathcal{N}^3)$ algorithms for solving the equations of motion

A. Jain, N. Vaidehi, G. Rodriguez
Journal of Computational Physics