**Proceedings of IDETC/CIE 2007**
**Proceedings of the ASME 2007 International Design Engineering Technical Conferences &**
**Computers and Information in Engineering Conference**
**September 4-7, 2007, Las Vegas, Nevada, USA**

**DETC2007-35128**

# AN EFFICIENT MULTIBODY DIVIDE AND CONQUER ALGORITHM (DETC2007-35128)

**James H. Critchley***
Modeling and Simulation Technology
General Dynamics Land Systems
Sterling Heights, Michigan 48310

**Kurt Anderson**
**Adarsh Binani**
Department of Mechanical, Aerospace and Nuclear Engineering
Rensselaer Polytechnic Institute
Troy, New York, 12180

## ABSTRACT

*A new and efficient form of Featherstone's multibody Divide and Conquer Algorithm (DCA) is presented and evaluated. The DCA was the first algorithm to achieve theoretically optimal logarithmic time complexity with a theoretical minimum of parallel computer resources for general problems of multibody dynamics, however the DCA is extremely inefficient in the presence of small to modest parallel computers. This alternative efficient DCA approach (DCAe) demonstrates that large DCA subsystems can be constructed using fast sequential techniques to realize a substantial increase in speed. The usefulness of the DCAe is directly demonstrated in an application to a four processor workstation and compared with results from the original DCA and a fast sequential recursive method. Previously the DCA was a tool intended for a future generation of parallel computers, this enhanced version delivers practical and competitive performance with the parallel computers of today.*

## NOMENCLATURE

$\mathbf{A}^k$ Spatial (6 dimensional) acceleration of body $k$.
$\widehat{\mathbf{A}}^k$ Recursive spatial acceleration of body $k$.
$\check{\mathbf{A}}^k$ Backward recursive spatial acceleration of body $k$.
$\widehat{\mathbf{A}}_i^k$ Recursive spatial acceleration of handle $i$ on body or subsystem $k$.
$\mathbf{A}_T{}^k$ Spatial state explicit acceleration of body $k$.
$\widehat{\mathbf{E}}^k$ Recursive coefficient of propagated spatial handle force for body $k$.
$\check{\mathbf{E}}^k$ Backward recursive coefficient of propagated spatial handle force for body $k$.
$\mathbf{F}^k$ Spatial applied and state explicit inertia force (6 dimensional) associated with body $k$.
$\mathbf{F}_{RHS}^k$ Spatial right hand side force (6 dimensional) associated with Newton-Euler equations of body $k$.
$\widehat{\mathbf{F}}^k$ Recursive spatial force of body $k$.
$\check{\mathbf{F}}^k$ Backward recursive spatial force of body $k$.
$\mathbf{I}^k$ Spatial (6 dimensional) inertia of body $k$.
$\widehat{\mathbf{I}}^k$ Recursive spatial inertia (Articulated body inertia) of body $k$.
$\check{\mathbf{I}}^k$ Backward recursive spatial inertia of body $k$.
$\mathbf{M}^k$ Diagonal block $k$ of a triangularized generalized mass matrix.
$\check{\mathbf{M}}^k$ Diagonal block $k$ of a triangularized generalized mass matrix.
$\mathbf{P}_\mathbf{k}^k$ Spatial (6 dimensional) partial velocity of body $k$ with respect to the coordinates defining the local joint motion.
$\check{\mathbf{P}}_\mathbf{k}^k$ Backward spatial partial velocity of body $k$ with respect to the coordinates defining the local joint motion.
$\mathbf{S}^k$ Spatial position vector cross product required to compute accelerations and forces on adjacent bodies $p[k]$ and $k$.
$\check{\mathbf{S}}^k$ Backward spatial position vector cross product required to compute accelerations and forces on adjacent bodies $\check{p}[k]$ and $k$.
$\mathbf{T}^k$ Spatial triangulatization matrix associated with body $k$.

*Address all correspondence to this author

$\check{\mathbf{T}}^k$ Backward spatial triangulatization matrix associated with body $k$.

$\mathbf{U}$ An identity matrix.

$\mathbf{W}$ An intermediate DCA term formed during subsystem combination.

$\mathbf{b}_i^k$ State explicit term in the equation for handle $i$ of DCA subsystem $k$.

$ch[k]$ Topological child body index of body $k$.

$\check{ch}[k]$ Child body index of body $k$ in a backwards topological representation (same as $p[k]$ for chain systems).

$\mathbf{d}$ An intermediate DCA term formed during subsystem combination.

$\mathbf{f}_i^k$ Spatial joint constraint force at handle $i$ of DCA subsystem $k$.

$h_i[j]$ Subassembly $i$ in subsystem $j$.

$p[k]$ Topological parent body index of body $k$.

$\check{p}[k]$ Parent body index of body $k$ in a backwards topological representation (same as $ch[k]$ for chain systems).

$\mathbf{q_k}$ Coordinates of joint $k$.

$\mathbf{u_k}$ Generalized speeds (quasi-coordinate velocities) of joint $k$.

$\Phi^k$ A handle constraint force coefficient matrix in the DCA equation for subsystem $k$.

$\Psi^k$ A handle constraint force coefficient matrix in the recursive equation for DCA subsystem $k$.

## INTRODUCTION

The speed of solution has motivated many in the computational multibody field to pursue efficient alternatives to classical formulations. Those interested in operator or hardware in the loop simulations or model based predictive control often require faster than real-time solutions. Designers may seek to perform comprehensive design optimization on high fidelity dynamic models which potentially require an unbounded number of iterations. While others simply wish to work with prohibitively large systems. Regardless of motivation the development of time efficient multibody solutions extends the domain of realizable analysis for problems involving multibody systems.

In 1999 Featherstone introduced the Divide and Conquer Algorithm (DCA) for parallel computation of multi-rigid-body dynamics [1, 2]. This algorithm boasts a logarithmic order of time complexity for the solution of the unknown generalized coordinate accelerations in the presence of a parallel computer system which scales linearly with multibody problem size, $n$ (the number of joint degrees of freedom). In other words, the algorithm is $O(\log_2 n)$ in the presence of $O(n)$ processors, which is a theoretical minimum order time and resource solution.

Prior to the DCA, optimal order time and resource methods were either topologically limited to chain systems (Fijany *et al.* [3]) or iterative and therefore not truly logarithmic (Anderson and Duan [4]). The DCA represents a milestone in efficient computation of multibody dynamics, however the DCA does not approach the efficiency of the fastest parallel algorithms until the number of available processors becomes very large [2].

Critchley and Anderson [5, 6] addressed the inefficiencies of asymptotically superior algorithms and presented a minimum order time and resource parallel algorithm called Recursive Coordinate Reduction Parallelism (RCRP). The RCRP produces efficient solutions for any number of processors because at all levels it is the efficient recursive $O(n)$ algorithm (with the addition of a linear order constraint technique). As a result the RCRP achieves $O(\log_2 n)$ scalable speed increases and highly competitive performance on parallel computer architectures with few processors.

This paper presents a new form of the Divide and Conquer Algorithm which demonstrates that scalable results similar to the RCRP are obtainable with a DCA approach. This efficient DCA (DCAe) further exploits the properties of multibody systems to implement the efficient $O(n)$ algorithm within course grain multibody subsystems (e.g. system bodies allocated to the same processor). This method ensures that speed increases are achievable with as few as two parallel processors and effectively improves performance by an order of magnitude.

## PRELIMINARIES

Before introducing the equations of the Divide and Conquer Algorithm it is necessary to establish some fundamental notations and relationships. This paper assumes that the reader is either familiar with "spatial" notations or otherwise capable of backing out the necessary details of the compact equations which are presented. The unfamiliar reader may simply regard spatial quantities as linear algebraically augmented linear and angular quantities or refer to [7] or similar texts.

In spatial matrix form, the Newton-Euler equations of motion for a single rigid body $k$ can be represented as

$$\mathbf{I}^k\mathbf{A}^k = \mathbf{F}^k_{RHS} \tag{1}$$

In Eq. (1), $\mathbf{I}^k$ is the spatial inertia of body $k$ which is a $6\times 6$ block diagonal matrix containing the central inertia matrix of body $k$ and a diagonal matrix of the mass of body $k$; $\mathbf{A}^k$ is the spatial acceleration (column matrix) of body $k$ containing the stacked matrix representations of angular and linear acceleration; $\mathbf{F}^k_{RHS}$ is the spatial force (column matrix) containing the stacked matrix representations of moments and forces which must be present on the right hand side (constraint, applied and gyroscopic inertial forces).

For simplicity, only multibody chain systems (Fig. 1) will be studied in this paper. Extension of all presented methods to tree and looped systems follows exactly as the original formulation of Featherstone [2]. In a multibody chain system, each body $k$ has exactly one child (or outboard) body $ch[k]$, with the exception of the terminal (last or outboard most) body which has no children.

Figure 1. Example of a chain system

Each body also has a single parent (or inboard) body $p[k]$ and the first body in the system has a parent which is the inertial reference frame $N$.

Relative joint coordinates $\mathbf{q}$ are used to describe the position and orientation of each body relative to its parent. The kinematic relationships between bodies can be simplified by exploiting some form of quasi coordinate velocities $\mathbf{u}$ (or *generalized speeds* [8]) which may be any linear combination of the $\dot{\mathbf{q}}$.

$$\mathbf{u} = \mathbf{X}_{(\mathbf{t},\mathbf{q})}\dot{\mathbf{q}} + \mathbf{Y}_{(\mathbf{t},\mathbf{q})} \tag{2}$$

For the purposes of recursive methods, it is common (if not required) to restrict the generalized speeds to be functions of only the local joint coordinates and coordinate velocities.

$$\mathbf{u_k} = \mathbf{X}_{(\mathbf{t},\mathbf{q_k})}\dot{\mathbf{q}}_\mathbf{k} + \mathbf{Y}_{(\mathbf{t},\mathbf{q_k})} \tag{3}$$

Equation (4) divides the spatial accelerations of Eq. (1) into parts which are functions only of the system state (and time) $\mathbf{A}_T$, and parts which are linear in the unknown $\dot{\mathbf{u}}$, $\widehat{\mathbf{A}}$. Using the prior definition of relative coordinates and generalized speeds, the kinematic relationship of Eq. (5) can be formed.

$$\mathbf{A}^k = \widehat{\mathbf{A}}^k + \mathbf{A}_T{}^k \tag{4}$$

$$\widehat{\mathbf{A}}^k = \left(\mathbf{S}^k\right)^T \widehat{\mathbf{A}}^{p[k]} + \mathbf{P}^k_\mathbf{k}\dot{\mathbf{u}}_\mathbf{k} \tag{5}$$

This decomposition and notation is essentially that of Anderson [9]. The $\widehat{\mathbf{A}}$ are called spatial recursive accelerations. $\mathbf{S}$ is the spatial shift matrix which encapsulates the necessary position vector cross product (*shift*) operation (in transpose) required to transfer accelerations from the local body center of mass to the point which is fixed in the local body and instantaneously coincident with the child body center of mass. And $\mathbf{P}^k_\mathbf{k}$ is exactly the matrix of state dependent spatial partial velocities of body $k$ with respect to the joint local generalized speeds $\mathbf{u_k}$.

It is also useful to note that pre-multiplying a spatial force applied on body $k$ by the spatial shift matrix $\mathbf{S}^k$ has the effect of moving (or shifting) spatial forces to the point which is instantaneously coincident with the center of mass of body $p[k]$.

## THE DIVIDE AND CONQUER ALGORITHM

This section presents a brief derivation of an equivalent form of Featherstone's Divide and Conquer Algorithm for chain systems and illustrates that such a method achieves $O(\log_2 n)$ time complexity of solutions in the presence of a parallel computer with resources which scale with problem size ($O(n)$ processors).

In what follows, a notational consistency has been maintained with the original work of Featherstone. While many of the symbols used by him are given different definitions, they represent analogous quantities and the resulting form of the solution is changed only slightly. This abuse of notation is designed to clearly illustrate that the underlying assembly operations are the prior work of Featherstone, while permitting the inclusion of efficient subsystems.

The DCA begins by computing all necessary kinematic information for the entire system via two sweeps of a binary assembly tree ($O(\log_2 n)$), as illustrated in Fig. 2. A binary tree solution for kinematic problems can be found in [10]. In the DCA, the kinematic quantities of interest are $\mathbf{P}^k_\mathbf{k}$, $\mathbf{S}^k$, and state dependent inertia and applied forces $\mathbf{F}^k$. The $\mathbf{P}^k_\mathbf{k}$ and $\mathbf{S}^k$ are functions of the local joint geometry and are evaluated concurrently at the leaf nodes (single bodies) of the binary tree. In general the computation of all applied forces and the state dependent linear and angular acceleration terms appearing in the $\mathbf{F}^k$ requires the evaluation of absolute positions, orientations, angular and linear velocities on every body.

The assembly tree recursively combines subsystems at the connecting joint to form a single assembled subsystem. In an assembled subsystem $s$, $h_1[s]$ refers to the parent subsystem of the assembly joint (containing subsystem handle 1) and $h_2[s]$ refers to the child subsystem (containing handle 2). Using this notation, a complete set of body frame orientations relative to the inertial frame ${}^N\mathbf{C}^k$ are constructed from the local joint transformations

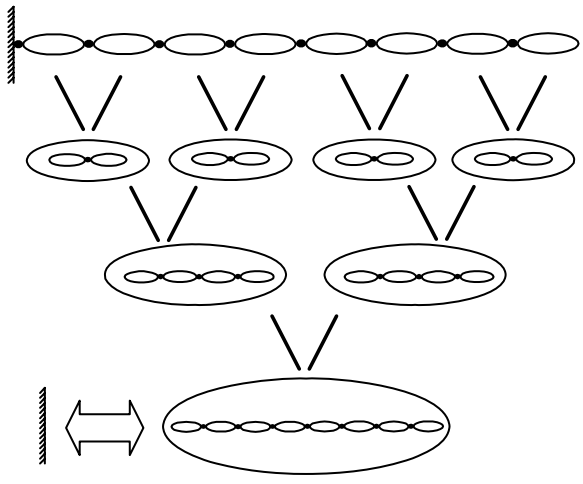

Figure 2. example of a binary assembly tree

${\mathbf{C}_T}^s$ in an inward sweep of the assembly tree using Eq. (6) and an outward sweep with Eq. (7-8) where for the root subsystem $r$, ${\mathbf{C}_T}^r$ is the desired result ${}^N\mathbf{C}^r$.

$$
\begin{aligned}
{\mathbf{C}_T}^s &= {\mathbf{C}_T}^{h_1[s]}{\mathbf{C}_T}^{h_2[s]} && (6)\\
{}^N\mathbf{C}^{h_1[s]} &= {}^N\mathbf{C}^s({\mathbf{C}_T}^{h_1[s]})^T && (7)\\
{}^N\mathbf{C}^{h_2[s]} &= {}^N\mathbf{C}^s && (8)
\end{aligned}
$$

In a similar fashion a complete set of kinematic vector quantities can be computed for each body using the generic form of Eq. (9-11). The $\mathbf{z}_\times$ terms are the additional cross products across the subsystem assembly required for each vector quantity. The $\mathbf{z}_\times$ are always zero for position and angular velocity and nonzero for angular acceleration and linear velocity and acceleration. It is useful to note that all kinematic values (including orientation transformations) can be computed during the same two tree traversals.

$$
\begin{aligned}
{\mathbf{z}_T}^s &= {\mathbf{z}_T}^{h_1[s]} + {\mathbf{z}_T}^{h_2[s]} + {\mathbf{z}_{\times 1}}^s && (9)\\
\mathbf{z}^{h_1[s]} &= \mathbf{z}^s - {\mathbf{z}_T}^{h_2[s]} + {\mathbf{z}_{\times 2}}^s && (10)\\
\mathbf{z}^{h_2[s]} &= \mathbf{z}^s && (11)
\end{aligned}
$$

Once the kinematic based information is available the *two handle* [1] spatial Newton-Euler equation of a single rigid body $p[k]$ is written as Eq. (12). In this equation subscripts 1 and 2 refer to the first and second handles of body $p[k]$. Handle 1 is the center of mass of $p[k]$ and handle 2 is the point on $p[k]$ which is instantaneously coincident with the center of mass of body $k$. $\widehat{\mathbf{A}}_1^{p[k]}$ is the spatial recursive acceleration of handle 1 (identically $\widehat{\mathbf{A}}^{p[k]}$) and $\mathbf{f}_1^{p[k]}$ and $\mathbf{f}_2^{p[k]}$ are the spatial joint constraint forces of the inboard and outboard joints, respectively (which are applied at the centers of mass of body $p[k]$ and $k$, necessitating the $\mathbf{S}^k$ applied to $\mathbf{f}_2^{p[k]}$).

$$
\mathbf{I}^{p[k]}\widehat{\mathbf{A}}_1^{p[k]} = \mathbf{f}_1^{p[k]} + \mathbf{S}^k\mathbf{f}_2^{p[k]} + \mathbf{F}^{p[k]}. \qquad (12)
$$

Equation (12) can be solved for the unknown spatial recursive acceleration as Eq. (13) and the acceleration of the second handle is then related by Eq. (14).

$$
\begin{aligned}
\widehat{\mathbf{A}}_1^{p[k]} &= \left(\mathbf{I}^{p[k]}\right)^{-1}\left\{\mathbf{f}_1^{p[k]} + \mathbf{S}^k\mathbf{f}_2^{p[k]} + \mathbf{F}^{p[k]}\right\} && (13)\\
\widehat{\mathbf{A}}_2^{p[k]} &= \left(\mathbf{S}^k\right)^T\widehat{\mathbf{A}}_1^{p[k]}\\
&= \left(\mathbf{S}^k\right)^T\left(\mathbf{I}^{p[k]}\right)^{-1}\left\{\mathbf{f}_1^{p[k]} + \mathbf{S}^k\mathbf{f}_2^{p[k]} + \mathbf{F}^{p[k]}\right\} && (14)
\end{aligned}
$$

These equations are valid for any two handle rigid body and are re-written as Eq. (15-16), where the $\Phi$ and $\mathbf{b}$ are determined through collection of coefficients.

$$
\begin{aligned}
\widehat{\mathbf{A}}_1^k &= \Phi_{11}^k\mathbf{f}_1^k + \Phi_{12}^k\mathbf{f}_2^k + \mathbf{b}_1^k && (15)\\
\widehat{\mathbf{A}}_2^k &= \Phi_{21}^k\mathbf{f}_1^k + \Phi_{22}^k\mathbf{f}_2^k + \mathbf{b}_2^k && (16)
\end{aligned}
$$

At this point it is the objective of a divide and conquer method to take the equations associated with two rigid bodies and combine them to form a new subsystem of identical structure. To this end consider pairs of neighboring bodies such that the common joint spatial constraint forces are equal and opposite.

$$
\mathbf{f}_2^{p[k]} = -\mathbf{f}_1^k \qquad (17)
$$

The kinematic definition of Eq. (5) is written in two handle form as Eq. (18) which simplifies to Eq. (19) and is expressed as Eq. (20).

$$
\begin{aligned}
\widehat{\mathbf{A}}_1^k &= \left(\mathbf{S}^k\right)^T\widehat{\mathbf{A}}_1^{p[k]} + \mathbf{P}_\mathbf{k}^k\dot{\mathbf{u}}_\mathbf{k} && (18)\\
\widehat{\mathbf{A}}_1^k &= \widehat{\mathbf{A}}_2^{p[k]} + \mathbf{P}_\mathbf{k}^k\dot{\mathbf{u}}_\mathbf{k} && (19)\\
\mathbf{P}_\mathbf{k}^k\dot{\mathbf{u}}_\mathbf{k} &= \widehat{\mathbf{A}}_1^k - \widehat{\mathbf{A}}_2^{p[k]} && (20)
\end{aligned}
$$

Equation (20) is expanded and simplified using Eq. (15-17).

$$
\begin{aligned}
\mathbf{P}_\mathbf{k}^k\dot{\mathbf{u}}_\mathbf{k} &= \Phi_{11}^k\mathbf{f}_1^k + \Phi_{12}^k\mathbf{f}_2^k + \mathbf{b}_1^k - \Phi_{21}^{p[k]}\mathbf{f}_1^{p[k]} - \Phi_{22}^{p[k]}\mathbf{f}_2^{p[k]} - \mathbf{b}_2^{p[k]} && (21)\\
\mathbf{P}_\mathbf{k}^k\dot{\mathbf{u}}_\mathbf{k} &= \left[\Phi_{11}^k + \Phi_{22}^{p[k]}\right]\mathbf{f}_1^k + \Phi_{12}^k\mathbf{f}_2^k + \mathbf{b}_1^k - \Phi_{21}^{p[k]}\mathbf{f}_1^{p[k]} - \mathbf{b}_2^{p[k]} && (22)
\end{aligned}
$$

From Eq. (22) a solution for $\dot{\mathbf{u}}_\mathbf{k}$ is available in terms of only $\mathbf{f}_2^k$ and $\mathbf{f}_1^{p[k]}$ when the orthogonality of the constraint force $\mathbf{f}_1^k$ and the admissible joint motion $\mathbf{P}_\mathbf{k}^k$ is exploited.

$$
\begin{aligned}
\dot{\mathbf{u}}_\mathbf{k} = &\left[\left(\mathbf{P}_\mathbf{k}^k\right)^T\Phi^{-1}\mathbf{P}_\mathbf{k}^k\right]^{-1}\left(\mathbf{P}_\mathbf{k}^k\right)^T\Phi^{-1}\{\Phi_{12}^k\mathbf{f}_2^k -\\
&\Phi_{21}^{p[k]}\mathbf{f}_1^{p[k]} + \mathbf{b}_1^k - \mathbf{b}_2^{p[k]}\} && (23)\\
\Phi = &\Phi_{11}^k + \Phi_{22}^{p[k]} && (24)
\end{aligned}
$$

Equation (22) also exhibits a solution for $\mathbf{f}_1^k$ as Eq. (25) which further simplifies to Eq. (26) upon substitution of (23),

where $\mathbf{U}$ is an identity matrix.

$$\mathbf{f}_1^k = \Phi^{-1}\left\{\mathbf{P}_\mathbf{k}^k\dot{\mathbf{u}}_\mathbf{k} - \Phi_{12}^k\mathbf{f}_2^k + \Phi_{21}^{p[k]}\mathbf{f}_1^{p[k]} - \mathbf{b}_1^k + \mathbf{b}_2^{p[k]}\right\} \quad (25)$$

$$\mathbf{f}_1^k = \mathbf{W}\Phi_{12}^k\mathbf{f}_2^k - \mathbf{W}\Phi_{21}^{p[k]}\mathbf{f}_1^{p[k]} + \mathbf{d} \quad (26)$$

$$\mathbf{W} = \Phi^{-1}\left\{\mathbf{P}_\mathbf{k}^k\left[\left(\mathbf{P}_\mathbf{k}^k\right)^T\Phi^{-1}\mathbf{P}_\mathbf{k}^k\right]^{-1}\left(\mathbf{P}_\mathbf{k}^k\right)^T\Phi^{-1} - \mathbf{U}\right\} \quad (27)$$

$$\mathbf{d} = \mathbf{W}\left(\mathbf{b}_1^k - \mathbf{b}_2^{p[k]}\right) \quad (28)$$

A new subsystem $j$ of identical structure can be constructed using Eq. (15–16) for a body $k$ and its parent $p[k]$ together with Eq. (17) and Eq. (26).

$$\widehat{\mathbf{A}}_1^j = \Phi_{11}^j\mathbf{f}_1^j + \Phi_{12}^j\mathbf{f}_2^j + \mathbf{b}_1^j \quad (29)$$

$$\widehat{\mathbf{A}}_2^j = \Phi_{21}^j\mathbf{f}_1^j + \Phi_{22}^j\mathbf{f}_2^j + \mathbf{b}_2^j \quad (30)$$

$$\Phi_{11}^j = \Phi_{11}^{p[k]} + \Phi_{12}^{p[k]}\mathbf{W}\Phi_{21}^{p[k]} \quad (31)$$

$$\Phi_{12}^j = -\Phi_{12}^{p[k]}\mathbf{W}\Phi_{12}^k \quad (32)$$

$$\mathbf{b}_1^j = \mathbf{b}_1^{p[k]} - \Phi_{12}^{p[k]}\mathbf{d} \quad (33)$$

$$\Phi_{22}^j = \Phi_{22}^k + \Phi_{21}^k\mathbf{W}\Phi_{12}^k \quad (34)$$

$$\Phi_{21}^j = -\Phi_{21}^k\mathbf{W}\Phi_{21}^{p[k]} \quad (35)$$

$$\mathbf{b}_2^j = \mathbf{b}_2^k + \Phi_{21}^k\mathbf{d} \quad (36)$$

Similar binary tree assembly of all subsystems produces a parallel assembly of global equations Eq. (29–30) in logarithmic ($O(\log_2 n)$) time complexity. In terms of the subsystem notation, assembly equations (31–36) are rewritten as (29–**??**).

$$\Phi_{11}^k = \Phi_{11}^{h_1[k]} + \Phi_{12}^{h_1[k]}\mathbf{W}\Phi_{21}^{h_1[k]} \quad (37)$$

$$\Phi_{12}^k = -\Phi_{12}^{h_1[k]}\mathbf{W}\Phi_{12}^{h_2[k]} \quad (38)$$

$$\mathbf{b}_1^k = \mathbf{b}_1^{h_1[k]} - \Phi_{12}^{h_1[k]}\mathbf{d} \quad (39)$$

$$\Phi_{22}^k = \Phi_{22}^{h_2[k]} + \Phi_{21}^{h_2[k]}\mathbf{W}\Phi_{12}^{h_2[k]} \quad (40)$$

$$\Phi_{21}^k = -\Phi_{21}^{h_2[k]}\mathbf{W}\Phi_{21}^{h_1[k]} \quad (41)$$

$$\mathbf{b}_2^k = \mathbf{b}_2^{h_2[k]} + \Phi_{21}^{h_2[k]}\mathbf{d} \quad (42)$$

Trivial boundary data is obtained for the entire system $S$ through the connection to the inertial frame and the lack of a connection at the second handle ($\widehat{\mathbf{A}}^N = \mathbf{0}$ and $\mathbf{f}_2^S = \mathbf{0}$). These identities solve for the local joint $\dot{\mathbf{u}}_\mathbf{1}$ and $\widehat{\mathbf{A}}_1^S$ which in turn gives $\mathbf{f}_1^S$. The constraint forces are then propagated back up the assembly tree to give solutions for the unknown coordinate accelerations in a time complexity of $O(\log_2 n)$.

To summarize, the DCA first computes all required kinematic information and state dependent inertia forces in a backward (leaf to root) and forward pass of a binary assembly tree (Fig. 2). Local equations Eq. (15–16) are then constructed on the leaf nodes and a backward sweep begins. Equations (37–42) are applied at each level of the tree during this backward sweep to generate identical forms of Eq. (29–30) for each subsystem. When all subsystems have been combined the definition of the connection to the inertial frame allows the solution of the root joint spatial constraint force (using Eq. (29)), then the root joint generalized acceleration (using Eq. (30)). The results are then propagated forward through the assembly tree and the complete solution is achieved in $O(\log_2 n)$ time complexity provided that the computations associated with branches of the tree are delegated to $O(n)$ processors.

To avoid confusion during implementation, the reader is again reminded that symbols and entire equations appearing here which are common to [1] do in fact have different numerical values. Mixing definitions from the two will most certainly yield incorrect results.

Although derived here for chain systems, Featherstone presents a complete algorithm for trees and loops [2]. A limited connectivity of general subsystems is also desired because the straightforward application of the *multi-handle* equations exhibit a local cubic growth in complexity, $O(h^3)$, with handle number $h$. The concept of *link splitting* is introduced to transform mechanisms with high inter-body connectivity into equivalent mechanisms of lower connectivity through the subdivision of individual bodies and the addition of equivalent rigid joints.

The loop solution is a direct extension of the tree formulation. Within each loop a single joint is stripped of its generalized coordinates and represented only by its constraint forces. This equivalent constraint force representation constitutes acceleration level enforcement of the joint constraint which requires the addition of constraint stabilization to remove displacement level violations that accrue during temporal integration.

## THEORETICAL DCA PERFORMANCE

This section describes the performance and related issues associated with Featherstone's DCA (and therefore the DCA form presented to this point). Unless otherwise stated, all observations and results quoted in this section can be found in [1, 2].

Featherstone presents a detailed operations count and comparison of the DCA with other algorithms. An effective operations count of $(928\,mult + 812\,add)\log_2(n)$ is given for the DCA in the presence of exactly $n$ processors and $(1150\,mult + 937\,add)n$ with one processor (an $O(n)$ algorithm). These operations counts are optimized for chain systems through the use of Denavit-Hartenberg transformations and a similarly optimized operations count for the efficient recursive $O(n)$ Articulated Body Algorithm (ABA) is also given as $(300\,mult + 279\,add)n$.

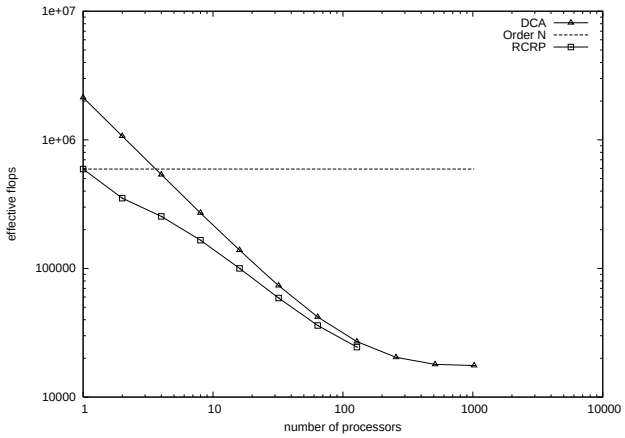


Figure 3. DCA performance for a 1024 body chain

Thus the theoretical minimum number of processors required to demonstrate any sort of speed increase over the serial $O(n)$ method is four.

In the presence of a variable number of parallel resources $p$, the operations counts of Featherstone can be used to obtain the following DCA operations count.

$$\left[928\log_2\left(\frac{p}{2}\right)+1150\left(\frac{n}{p}\right)-1041\right] \quad \text{multiplications}$$
$$+\left[812\log_2\left(\frac{p}{2}\right)+937\left(\frac{n}{p}\right)-845\right] \quad \text{additions} \qquad (43)$$

A plot of theoretical performance versus number of processors for a 1024 body chain is illustrated in Fig. 3 which also contains the results from the serial recursive $O(n)$ and RCRP for comparison.

Featherstone also notes that approximately one third of the computations involved in the DCA may be done independently, constituting a significant "overlap". This overlap is used to produce another operations count which is two thirds of the total but requires exactly $n$ processors to realize. It should be noted that all parallel algorithms and many sequential algorithms possess some form of overlap but for the purposes of straight forward comparison these properties will be ignored. One should also consider that exploiting overlap typically requires additional processor synchronization and communication which for present day computers will likely outweigh any benefits.

Another issue which adversely effects the DCA's efficiency is that its accuracy is significantly worse than the ABA. To remedy highly inaccurate results for large ($n > 1000$) system chains, Featherstone introduces a pivoting scheme which successfully avoids ill-conditioned calculations present in the formulation. The pivoted version of the DCA remains less accurate than the $O(n)$ algorithm, which should be expected of any method which introduces additional calculations in the form of inversions and propagations to induce parallelism. The additional computational cost of the pivoting scheme is said to make the equations "more complicated" and "less efficient" than the original, but is not otherwise quantified by an operations count. This pivoting scheme is also shown to recast the equations in a form which is the ABA in the case of a single rigid body.

In parallel algorithms one should expect the accuracy relative to sequential methods to be a function of available parallelism rather than system size. In this respect one of the chief problems with the DCA is that the concurrency is always generated for exactly $n$ virtual processors which are mapped to the actual number of available processors. This is in contrast to other subsysteming methods which generate concurrency only as required (preserving both accuracy and efficiency for small parallel computers).

Ideally, the DCA should only use a number of subsystems which is equal to the number of available parallel processors and use an efficient algorithm within them. Such a change would guarantee speed improvements over the ABA and other efficient recursive $O(n)$ algorithms in the presence of as few as two parallel processors, rendering it highly competitive with methods which have shown useful application.

## AN $O(n)$ ALGORITHM

There are many efficient recursive $O(n)$ algorithms to choose from, but in general they all have more in common than they have differences. The DCA has been recast to work with one such algorithm, that of Anderson [9], however it should be possible to do the same for most of the others as well. For completeness Anderson's algorithm for chain systems is briefly presented here in a notation consistent with that of this paper. For more detail refer to [9].

Given relative joint states $\mathbf{q_k}$ and relative quasi-coordinate velocities $\mathbf{u_k}$, a linear order forward topological solution for all position and velocity kinematics is automatic. For the purposes of this algorithm, this means that the terms $\mathbf{S}^k$, $\mathbf{P}^k_\mathbf{k}$, and $\mathbf{F}^k$ are computed for all bodies $k$.

A backwards system topological sweep follows computing $\widehat{\mathbf{I}}$ and $\widehat{\mathbf{F}}$. These terms are also known as the recursive spatial inertia and recursive spatial force (or articulated body inertia and articulated body force [11]), respectively.

$$\widehat{\mathbf{I}}^k = \mathbf{I}^k + \mathbf{T}^{ch[k]}\widehat{\mathbf{I}}^{ch[k]}\left(\mathbf{S}^{ch[k]}\right)^T \qquad (44)$$

$$\widehat{\mathbf{F}}^k = \mathbf{F}^k + \mathbf{T}^{ch[k]}\widehat{\mathbf{F}}^{ch[k]} \qquad (45)$$

$$\mathbf{T}^k = \mathbf{S}^k\left(\mathbf{U} - \widehat{\mathbf{I}}^k\mathbf{P}^k_\mathbf{k}\left(\mathbf{M}^k\right)^{-1}\left(\mathbf{P}^k_\mathbf{k}\right)^T\right) \qquad (46)$$

$$\mathbf{M}^k = \left(\mathbf{P}^k_\mathbf{k}\right)^T\widehat{\mathbf{I}}^k\,\mathbf{P}^k_\mathbf{k} \qquad (47)$$

Boundary relationships $\widehat{\mathbf{I}}^t = \mathbf{I}^t$ and $\widehat{\mathbf{F}}^t = \mathbf{F}^t$ are retrieved by observing that the chain system terminal body $t$ has no children.

At the completion of the backwards sweep Eq. (48) is the equation of motion associated with joint $k$.

$$\mathbf{0} = (\mathbf{P}_\mathbf{k}^k)^T \{\widehat{\mathbf{I}}^k \widehat{\mathbf{A}}^k - \widehat{\mathbf{F}}^k\} \tag{48}$$

Equation (49) is used with Eq. (5) in a final forward topological sweep solving for the unknown $\dot{\mathbf{u}}_\mathbf{k}$. The required boundary relationship is that the spatial recursive acceleration of the inertial frame (parent of the first body in the chain) is zero ( $\widehat{\mathbf{A}}^N = \mathbf{0}$).

$$\dot{\mathbf{u}}_\mathbf{k} = -\left(\mathbf{M}^k\right)^{-1} \left(\mathbf{P}_\mathbf{k}^k\right)^T \left[\widehat{\mathbf{I}}^k \left(\mathbf{S}^k\right)^T \widehat{\mathbf{A}}^{p[k]} - \widehat{\mathbf{F}}^k\right] \tag{49}$$

## TERMINAL SUBSYSTEMS

An $O(n)$ subsystem can be used within any terminal subsystem appearing in the DCA. This is accomplished by replacing the equation of a single rigid-body with the articulated body equation for the base body $b$ of the terminal subsystem. The articulated body equation for the base body which replaces Eq. (12) is given in Eq. (50).

$$\widehat{\mathbf{I}}^b \widehat{\mathbf{A}}^b = \mathbf{f}_1 + \Psi_2^b \mathbf{f}_2 + \widehat{\mathbf{F}}^b \tag{50}$$

This equation was obtained directly from Eq. (48) by definition of the velocity projection.

Within terminal subsystems the second handle does not contain a connection and thus the force $\mathbf{f}_2^b$ is identically zero and the unknown coefficient matrix $\Psi_2^b$ is of no consequence. In fact the analyst is free to choose the second handle as coincident with the first. In any case the form of Eq. (51–52) is required and can be arrived at by computing $\widehat{\mathbf{I}}^b$ and $\widehat{\mathbf{F}}^b$ with the efficient recursive $O(n)$ operations of Eq. (44–47).

$$\widehat{\mathbf{A}}_1^b = \Phi_{11}^b \mathbf{f}_1^b + \Phi_{12}^b \mathbf{f}_2^b + \mathbf{b}_1^b \tag{51}$$

$$\widehat{\mathbf{A}}_2^b = \Phi_{21}^b \mathbf{f}_1^b + \Phi_{22}^b \mathbf{f}_2^b + \mathbf{b}_2^b \tag{52}$$

Equations (51–52) are exactly those appearing in the DCA formulation and proper selection of subsystems and the order in which they are combined (load balancing) insures a speed increase relative to the $O(n)$ algorithm in the presence of as few as two parallel processors.

Locally, this equation is both formed and solved at the same cost as the $O(n)$ algorithm and provides more accurate results without the pivoted form of the DCA. Exploiting this identity and load balancing the processors reveals a significant increase in throughput. Figure 4 illustrates the theoretical performance

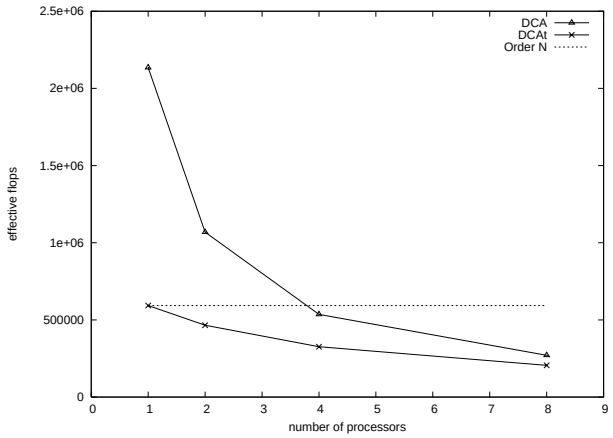


Figure 4. Improvement using $O(n)$ terminal subsystems on a 1024 body chain

benefits of this terminal $O(n)$ DCA method (referred to as DCAt) applied to a sequential chain system of 1024 bodies with *small* parallel computers.

This optimization has further significance to branching systems which may be subdivided such that the identity holds for terminal subsystems of each branch. From this one may now conclude that the worst case modified DCA system is actually a chain system and not a tree. Although there still remains additional complexity introduced at the branching bodies which should be treated as per the discussion of Featherstone [2].

## NON-TERMINAL SUBSYSTEMS

Non-terminal subsystems present a twofold problem with respect to obtaining the two handle subsystem equations required for the fundamental form of the DCA. First, in the articulated body equation for the subsystem base, Eq. (50), the $\mathbf{f}_2^b$ are neither zero or in general applied to the base body. This means that $\Psi_2^b$ is a nontrivial dynamic quantity which can not be obtained directly from the geometry. Second, the equation associated with the acceleration of handle 2 cannot be formed by the same $O(n)$ traversal as that of handle 1 because the articulated body equation for the body at handle 2 cannot contain ancestral body forces such as $\mathbf{f}_1^b$. This information is instead embedded in the forward solution phase of the $O(n)$ algorithm.

### Handle 1 Equations

To resolve the issues associated with $\Psi_2^b$ appearing in the equation for handle 1, a first step is to write out the articulated body equation for the subsystem terminal body $t$ (containing handle 2). This is given by Eq. (53) and further simplifies to Eq. (54)

by definition of a terminal body.

$$\widehat{\mathbf{I}}^t \widehat{\mathbf{A}}^t = \mathbf{S}^{ch[t]}\mathbf{f}_2 + \widehat{\mathbf{F}}^t \tag{53}$$
$$\mathbf{I}^t \widehat{\mathbf{A}}^t = \mathbf{S}^{ch[t]}\mathbf{f}_2 + \mathbf{F}^t \tag{54}$$

It should be noted that the recursive computations of $\widehat{\mathbf{I}}$ and $\widehat{\mathbf{F}}$ follow exactly as in (44–47) and ignore bodies beyond the local subsystem terminal body, however the shift operation to one body beyond the terminal body (handle 1 on the adjacent subsystem or $ch[t]$) is required.

The systematic application of recursive $O(n)$ procedures can be used to generate recursive equations of motion for the other bodies in the local subsystem. The recursive assumption is that these equations exist in the form of Eq. (55) with the trivial boundary data of Eq. (56) obtained on the subsystem terminal body.

$$\widehat{\mathbf{I}}^k \widehat{\mathbf{A}}^k = \widehat{\mathbf{E}}^k \mathbf{S}^{ch[t]}\mathbf{f}_2 + \widehat{\mathbf{F}}^k \tag{55}$$
$$\widehat{\mathbf{I}}^t = \mathbf{I}^t \qquad \widehat{\mathbf{E}}^t = \mathbf{U} \qquad \widehat{\mathbf{F}}^t = \mathbf{F}^t \tag{56}$$

Solving for the assumed form of the recursive quantities verifies that Eq. (44–47) are unchanged and further requires Eq. (57–58).

$$\dot{\mathbf{u}}_\mathbf{k} = -\left(\mathbf{M}^k\right)^{-1} (\mathbf{P}^k_\mathbf{k})^T \left[\widehat{\mathbf{I}}^k (\mathbf{S}^k)^T \widehat{\mathbf{A}}^{p[k]} - \widehat{\mathbf{E}}^k \mathbf{S}^{ch[t]}\mathbf{f}_2 - \widehat{\mathbf{F}}^k\right] \tag{57}$$
$$\widehat{\mathbf{E}}^{p[k]} = \mathbf{T}^k \widehat{\mathbf{E}}^k. \tag{58}$$

The coefficient matrices for the subsystem base body equation of motion Eq. (59) may now be obtained in linear time.

$$\widehat{\mathbf{I}}^b \widehat{\mathbf{A}}^b = \mathbf{f}_1 + \widehat{\mathbf{E}}^b \mathbf{S}^{ch[t]}\mathbf{f}_2 + \widehat{\mathbf{F}}^b \tag{59}$$

### Handle 2 Equations

Forming the necessary subsystem equation associated with the acceleration of handle 2 may be accomplished using a backwards subsystem sweep and almost exactly the equations of handle 1. To facilitate the justification of these equations, the system is given a backwards ordering and new quantities associated with the backwards solution are denoted with ($\check{}$).

The $O(n)$ algorithms exploit the forward propagation of equation (5) and the corresponding definition of the partial velocities. The first observation fundamental to the $O(n)$ derivation reduces the summation over all bodies in D'Alembert's variational form (or in this case Kane's Method) such that only one term appears at terminal bodies $t$. In the backwards system the summation at the terminal body ($\check{t} = b$) does not readily reduce to one term because the variational displacements are coupled by the subsystem generalized coordinates. To resolve this issue one observes that the DCA equations require only the matrices relating the accelerations and constraint forces and is therefore free from any assumptions about a uniform coordinate description between the handle 1 and handle 2 equations of motion. In this case a free joint (six degree of freedom) may be added at the handle 2 subsystem base body ($\check{b} = t$) and the kinematics recomputed using this definition.

Because the analyst is also free to choose a specified base motion upon which to attach the free joint, the calculation of a second set of kinematic data can be completely avoided. This is done by prescribing the base motion such that the state dependent acceleration terms absent from $\check{\mathbf{A}}^{\check{b}}$ and $\widehat{\mathbf{A}}^t$ are identical (and therefore $\check{\mathbf{A}}^{\check{b}} = \widehat{\mathbf{A}}^{\check{b}} = \widehat{\mathbf{A}}^t$) thus the spatial forces $\mathbf{F}$ also remain unchanged.

The backwards kinematic equation of Eq. (60) allows the equation of motion for body $\check{t}$ to be written as Eq. (61).

$$\widehat{\mathbf{A}}^k = (\check{\mathbf{S}}^k)^T \widehat{\mathbf{A}}^{\check{p}[k]} + \check{\mathbf{P}}^k_\mathbf{k} \dot{\mathbf{u}}_\mathbf{k} \tag{60}$$
$$\check{\mathbf{I}}^{\check{t}} \widehat{\mathbf{A}}^{\check{t}} = \mathbf{f}_1 + \check{\mathbf{F}}^{\check{t}}. \tag{61}$$

Equation (61) is then simplified by the definition of a terminal body.

$$\check{\mathbf{I}}^{\check{t}} \widehat{\mathbf{A}}^{\check{t}} = \mathbf{f}_1 + \mathbf{F}^{\check{t}} \tag{62}$$

The assumed recursive form for the equations of motion of all bodies in the local backwards subsystem is written as Eq. (63). And trivial boundary data obtained on body $\check{t}$ allows a solution for the unknown recursive quantities, which are analogous to those required for handle 1.

$$\check{\mathbf{I}}^k \widehat{\mathbf{A}}^k = \check{\mathbf{E}}^k \mathbf{f}_1 + \check{\mathbf{F}}^k \tag{63}$$
$$\check{\mathbf{I}}^k = \mathbf{I}^k + \check{\mathbf{T}}^{\check{ch}[k]} \check{\mathbf{I}}^{\check{ch}[k]} (\check{\mathbf{S}}^{\check{ch}[k]})^T \tag{64}$$
$$\check{\mathbf{F}}^k = \mathbf{F}^k + \check{\mathbf{T}}^{\check{ch}[k]} \check{\mathbf{F}}^{\check{ch}[k]} \tag{65}$$
$$\check{\mathbf{T}}^k = \check{\mathbf{S}}^k \left(\mathbf{U} - \check{\mathbf{I}}^k \check{\mathbf{P}}^k_\mathbf{k} (\check{\mathbf{M}}^k)^{-1} (\check{\mathbf{P}}^k_\mathbf{k})^T\right) \tag{66}$$
$$\check{\mathbf{M}}^k = (\check{\mathbf{P}}^k_\mathbf{k})^T \check{\mathbf{I}}^k \check{\mathbf{P}}^k_\mathbf{k} \tag{67}$$
$$\dot{\mathbf{u}}_\mathbf{k} = -(\check{\mathbf{M}}^k)^{-1} (\check{\mathbf{P}}^k_\mathbf{k})^T \left[\check{\mathbf{I}}^k (\check{\mathbf{S}}^k)^T \widehat{\mathbf{A}}^{\check{p}[k]} - \check{\mathbf{E}}^k \mathbf{f}_1 - \check{\mathbf{F}}^k\right] \tag{68}$$
$$\check{\mathbf{E}}^{\check{p}[k]} = \check{\mathbf{T}}^k \check{\mathbf{E}}^k, \tag{69}$$

Thus the coefficients for the subsystem terminal body equation of motion are obtained in linear time (Where the forward shift and child body definitions must be used exactly as shown).

$$\check{\mathbf{I}}^{\check{b}} \widehat{\mathbf{A}}^{\check{b}} = \check{\mathbf{E}}^{\check{b}} \mathbf{f}_1 + \mathbf{S}^{ch[\check{b}]}\mathbf{f}_2 + \check{\mathbf{F}}^{\check{b}} \tag{70}$$
$$= \check{\mathbf{I}}^t \widehat{\mathbf{A}}^t = \check{\mathbf{E}}^t \mathbf{f}_1 + \mathbf{S}^{ch[t]}\mathbf{f}_2 + \check{\mathbf{F}}^t \tag{71}$$

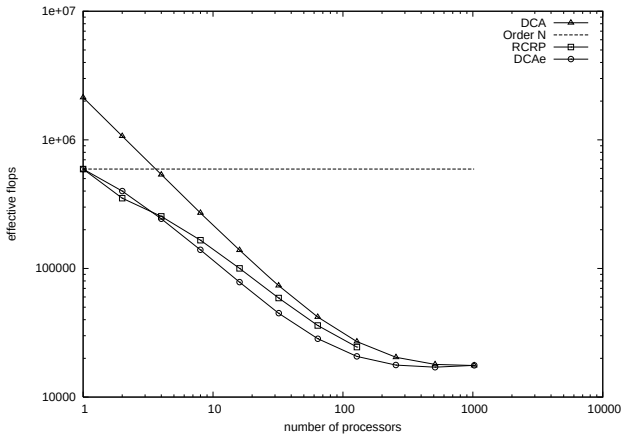


Figure 5. Efficient DCA applied to a 1024 body chain

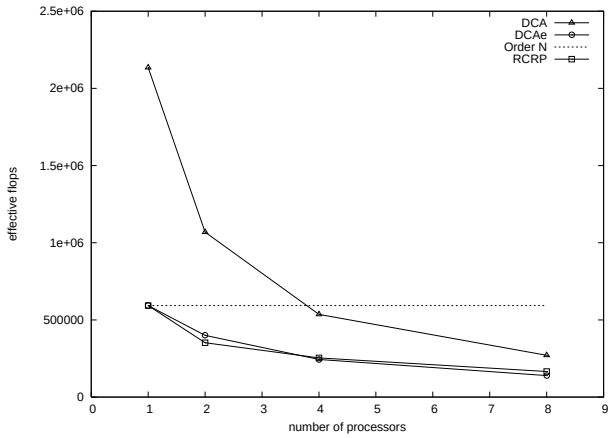


Figure 6. Efficient DCA applied to a 1024 body chain

## THEORETICAL PERFORMANCE

Utilizing the efficient terminal and non-terminal subsystem solutions within the DCA defines the efficient Divide and Conquer Algorithm (DCAe). An effective operations count for the DCAe given an arbitrary computer system with $p$ processors is obtained.

$$\begin{aligned} &\left[928\log_2\left(\frac{p}{2}\right)+637\left(\frac{n}{p}\right)-495\right] \quad \text{multiplications} \\ +&\left[812\log_2\left(\frac{p}{2}\right)+548\left(\frac{n}{p}\right)-446\right] \quad \text{additions} \end{aligned} \tag{72}$$

In the operations count, the coefficients of $\log_2(\frac{p}{2})$ is exactly the same as the DCA assembly for which the linear algebraic assembly operations of Featherstone have not changed. The coefficients of $(\frac{n}{p})$ are the processor local $O(n)$ operations which includes a single forward kinematics pass, double the usual articulated body inertia and force computations, and the two additional **E** terms. The constant term accounts for the triviality of all operations at the boundaries.

Further improvements to this count are realized through load balancing, which simply means that twice as many bodies are allocated to terminal subsystems within which the $O(n)$ algorithm is applied without modification. In the presence of a single processor, the entire system is terminal and the $O(n)$ solution is retrieved.

Figure 5 illustrates the theoretical performance of the balanced algorithm applied to a 1024 body chain system. It can be seen that the DCAe widely outperforms the traditional DCA and is a more efficient alternative to the RCRP. For today's parallel computers, the region of practical importance involves small parallel computers and is shown in Fig. 6.

The only other optimal order/resource parallel multibody method with general system applicability is the RCRP, however direct comparison of the two methods in general is difficult. The DCA (and DCAe) may encounter complications associated with branching systems while the RCRP excels under such conditions. The DCAe also enforces kinematic loop constraints at the acceleration level and therefore requires stabilization, while the RCRP maintains a velocity level description of constraints and can run for extended periods of time without any stabilization. These two drawbacks aside, in general the DCAe has a lower total operations count and is a much more practical algorithm to implement. The RCRP can enter configurations which are singular to the formulation and requires a complicated accounting system to avoid them.

Despite the fundamental formulation differences, the DCAe and RCRP have much in common. Both methods degenerate to the same $O(n)$ algorithm in the presence of serial computers and obtain optimal results for small parallel computers by allocating more bodies to terminal subsystems (load balancing). In each, concurrency of solution is introduced only as additional processors become available, this means that accuracy relative to the $O(n)$ algorithm is a function of parallelism rather than system size. And lastly, the computations associated with both algorithms are modified forms of the $O(n)$ solution and can leverage two decades of related research including extensions to flexible bodies, analytic sensitivity information, and operational control and filtering.

## IMPLEMENTATION AND RESULTS

The three multibody solution algorithms, DCA, $O(n)$, and DCAe were implemented in the object oriented programming language C++. Mathematically the DCAe is the combination of the DCA and $O(n)$ solutions and through interface inheritance, the generic forms of both algorithms can be used to construct the software implementation of the DCAe. The benefits of this approach are not only reduction in the development time of the

three solutions, but also the ability to compare algorithm performance against common code instead of three independent implementations.

The software implementation is capable of solving arbitrary three dimensional revolute chain systems with all three algorithms and was applied to systems of 128, 256, and 1024 bodies. In the case of the DCAe, a variable number of bodies can be allocated to the terminal subsystem permitting experimental validation of load balancing. All simulations were run on a 3.2 GHz quad-processor Xeon shared memory Linux computer using *gcc* and POSIX threads and semaphores for parallelization. The results presented have been obtained from the run time of 200 consecutive solves for the unknown coordinate accelerations which consist of three complete passes through the assembly tree (root to leaves and back).

The operations counts presented as the theoretical performance of the algorithms are symbolically optimized to eliminate all known zero, identity, and duplicate calculations. In contrast the validation simulation software treats all quantities as generic matrices and takes no advantage of matrix symmetry, sparsity, or identities. Given the difference in implementation the software results should not be expected to exactly match the theoretical predictions. Under the assumption of a uniform distribution of symbolic optimizations throughout the algorithms it is expected that results will be close and any discrepancies should be investigated.

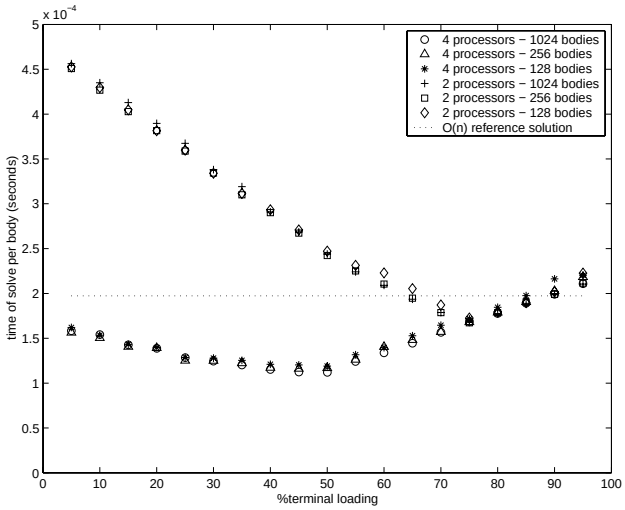


Figure 7. Load balancing results

Load balancing execution times for the DCAe are given in Figure 7. The results show that the implementation is not sensitive to multi-processor communication costs for the large problems which were considered and that the optimal terminal system loading is three times that of the non-terminal branches (75 and 50 percent for two and four processors, respectively).

The Load balancing ratio is significantly different than the result reported by the operations count (a ratio of two to one was anticipated). The additional overhead has been traced to two sources. First, the decision to use parent body frames within the recursive algorithm subsystems results in many extraneous transformations which were easily removed from the sequential solution terms but appear other terms such as $\widehat{\mathbf{E}}$ making several evaluations three times more expensive than similar operations in the standard algorithm. And second, the backward subsystem description required in the general case was not created as a preprocessing step and instead matrices were generated through inversion of the forward description to simplify the implementation. These and other minor inconsistencies are the most likely reason for the lack of relative performance. Despite the inflated solution time for general subsystems the implemented solution does outperform the theoretical sequential DCA which would provide a ratio of approximately four to one.

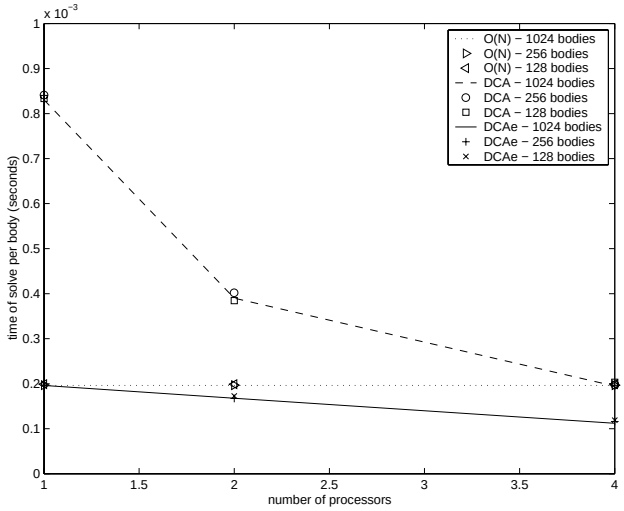


Figure 8. Algorithm timing results

Figure 8 clearly illustrates the performance advantages of the DCAe, outperforming the fast recursive method with just two processors, and demonstrates that no benefit is gained from four processors and the traditional DCA. The DCA as implemented is slightly greater than four times as computationally intensive as the sequential recursive algorithm. This result suggests that symbolic optimization in the DCA reduces a somewhat larger percentage of computations relative to the recursive solution.

## CONCLUSIONS

Reconstruction of the Divide and Conquer Algorithm to include sequential $O(n)$ computations within subsystems has produced a new and efficient multibody solution scheme with optimal $O(\log_2 n)$ time complexity. Relative to the RCRP (the only other optimal order/resource multibody solution with general system applicability) this efficient DCA offers a simplified solution and lower operations count for general bodies. The lower operations count and simple load balancing in parallel environments leads to DCAe performance which outperforms efficient higher order methods in regions of practical interest (small parallel computers). The operations count based performance predictions have been successfully validated through comparison of three solution algorithms applied to a four processor shared memory workstation. Previously the DCA was a tool intended for a future generation of parallel computers, this enhanced version delivers practical and competitive performance with the parallel computers of today and scales with computing resources for successful application to future computer systems.

## REFERENCES


[1] Featherstone, R., 1999. "A divide-and-conquer articulated body algorithm for parallel $O(\log(n))$ calculation of rigid body dynamics. Part 1: Basic algorithm". *International Journal of Robotics Research,* **18**(9), Sep., pp. 867–875.

[2] Featherstone, R., 1999. "A divide-and-conquer articulated body algorithm for parallel $O(\log(n))$ calculation of rigid body dynamics. Part 2: Trees, loops, and accuracy". *International Journal of Robotics Research,* **18**(9), September, pp. 876–892.

[3] Fijany, A., Sharf, I., and D'Eleuterio, G. M. T., 1995. "Parallel O(log n) algorithms for computation of manipulator forward dynamics". *IEEE Transactions on Robotics and Automation,* **11**(3), pp. 389–400.

[4] Anderson, K. S., and Duan, S., 1999. "A hybrid parallelizable low order algorithm for dynamics of multi-rigid-body systems: Part I, Chain systems". *Mathematical and Computer Modeling,* **30**, pp. 193–215.

[5] Critchley, J. H., 2003. "A parallel logarithmic time complexity algorithm for the simulation of general multibody system dynamics". Phd thesis, Rensselaer Polytechnic Institute, Troy, NY.

[6] Critchley, J. H., and Anderson, K. S., 2004. "Parallel logarithmic order algorithm for general multibody system dynamics". *Multibody Systems Dynamics,* **12**(1), pp. 75–93.

[7] Featherstone, R., 1987. *Robot Dynamics Algorithms*. Kluwer Academic Publishing, New York.

[8] Kane, T. R., and Levinson, D. A., 1985. *Dynamics: Theory and Application*. Mcgraw-Hill, NY.

[9] Anderson, K. S., 1993. "An order-n formulation for the motion simulation of general multi-rigid-body tree systems". *Computers and Structures,* **46**(3), pp. 547–559.

[10] Lathrop, L. H., 1985. "Parallelism in manipulator dynamics". *International Journal of Robotics Research,* **4**(2), pp. 80–102.

[11] Featherstone, R., 1983. "The calculation of robotic dynamics using articulated body inertias". *International Journal of Robotics Research,* **2**(1), Spring, pp. 13–30.