A Parallel $O(N)$ Formulation for General Multibody Dynamics*

Takashi NAGATA**

A new methodology for dynamical analyses applicable to a very large class of rigid and flexible multibody systems is presented. It is based on a variable-gain error correction method with scaling, and has the following distinctive features: (i) All kinds of holonomic and nonholonomic equality constraints can be treated in a plain and unified manner; (ii) Stability of the constraints is always attained; (iii) The formulation has an order N computational cost in terms of both the constrained and unconstrained degrees of freedom, regardless of the system topology; (iv) Unlike the traditional recursive order N algorithms, it is quite amenable to parallel computation; (v) Since no matrix operations are involved, it can be implemented to very simple general-purpose simulation programs. Versatility, dynamical validity and efficiency of the approach are checked through numerical studies of several particular systems.

Key Words: Computer Aided Analysis, Dynamics of Machinery, Flexible Structure, Nonholonomic Constraints, Penalty Method, Parallel Computation, $O(N)$ Algorithm

1. Introduction

Analytical dynamics provides quite powerful tools for modeling a broad variety of multibody systems. The concept of kinematical constraint plays an essential role in reducing the degrees of freedom of the resulting equations. However, its treatment often yields theoretical and numerical difficulties. Holonomic constraints are described by equalities involving only generalized coordinates and time, whereas the other constraints are all nonholonomic and they have no general mathematical representation. This complexity discourages development of general formulation. Furthermore, when a system is subjected to nonholonomic or intricate holonomic constraints, its explicit formulation through introduction of independent generalized coordinates is impossible or impractical, and hence its dynamics is often expressed by a set of differential-algebraic equations, which tend to be stiff suffering instabilities or unacceptably large computational cost during numerical integration.

All the above have been longstanding problems, and numerous investigations were carried out to devise algorithms with a high level of stability, efficiency and generality.

Baumgarte⁽¹⁾ proposed a stabilization technique where the kinematic constraints are replaced with differential equations yielding exponential convergence of the constraint errors. Although the concept is simple, it is generally not practical for large systems, since the procedure requires $O(N^3)$ computational effort for obtaining the constraint force, where N is the number of the constraints. Moreover, the algorithm is not valid when two or more constraints are nearly dependent and the solution matrix becomes ill-conditioned⁽²⁾. The staggered stabilization technique suggested by Park and Chio^{(2),(3)} is based on a penalty method alleviating the above shortcomings. However, the methodology brings forth another problem of choosing the right penalty number. Since small penalty can give rise to instability and its minimum stable value generally varies with time, it is usually set very large requiring additional computational effort. Approaches by Bayo et al.⁽⁴⁾ and Kurdila et al.⁽⁵⁾ fall into the same category,

* Received 30th September, 2002 (No. 02-4161)

** Integrated Volume - CAD System Program, the Institute of Physical and Chemical Research (RIKEN), 2-1 Hirosawa, Wako, Saitama, 351-0198 Japan.
E-mail: tnagata@postman.riken.go.jp

although they included artificial mass and dissipative force in addition to the proportional feedback of the constraint errors. It should also be pointed out that almost all the approaches presently available can deal with only equality constraints.

With these as background, a novel simulation methodology applicable to a large class of multibody systems is presented in this paper. It is based on a variable-gain error correction method developed by the author^{(6),(7)} improved by an adaptive control of the stabilization parameter. The formulation can deal with holonomic, nonholonomic equality and a class of inequality constraints in a plain and unified manner. Unlike the traditional penalty methods, the parameter value need not be provided a priori and the numerical stability is assured. The proposed algorithm is robust against singular constraints with which Baumgarte's approach fails. In addition to the reliability, it is quite efficient with an order N computational cost in terms of both the constrained and unconstrained degrees of freedom, regardless of the system topology. Moreover, since virtually no matrix operations are involved, the associated numerical procedure can be implemented to very simple general-purpose simulation programs. The matrixless property also implies amenability to parallel computation.

Based on the approach, a general multibody dynamics software has been developed. It supports distributed processing using a PC cluster to accelerate the calculation. To demonstrate its versatility, dynamical validity and efficiency, numerical examples of several particular systems are also given.

2. Formulation

2.1 System equations

Consider a system consisting of N_B bodies. Before imposing kinematical constraints, dynamics of each body can be described by the equation of motion^{(8)*1}.

$$[M]_i \ddot{\mathbf{u}}_i = \mathbf{Q}_i + \mathbf{F}_i \equiv \mathbf{G}_i \quad (i=1, 2, \dots, N_B) \quad (1)$$

where: $\mathbf{u}_i$ denotes an arbitrary nonholonomic velocity vector for the body i ; $[M]_i$ and $\mathbf{Q}_i$ are the corresponding mass matrix and generalized force vector, respectively; $\mathbf{F}_i$ contains force terms other than $\mathbf{Q}_i$, such as the Coriolis and centrifugal forces; and $\mathbf{G}_i$ is the total force vector. Collecting Eq.(1) for all the bodies readily leads to the system equations before interconnection as:

$$[M] \ddot{\mathbf{u}} = \mathbf{G} \quad (2.a)$$

$$[M] \equiv \text{diag}[[M]_1, [M]_2, \dots, [M]_{N_B}] \quad (2.b)$$

$$\mathbf{u} \equiv \{\mathbf{u}_1^T, \mathbf{u}_2^T, \dots, \mathbf{u}_{N_B}^T\}^T \quad (2.c)$$

$$\mathbf{G} \equiv \{\mathbf{G}_1^T, \mathbf{G}_2^T, \dots, \mathbf{G}_{N_B}^T\}^T \quad (2.d)$$

When auxiliary kinematical constraints are introduced to the system, constraint force $\mathbf{C}$ needs to be applied and hence the equation of motion (2.a) changes to

$$[M] \ddot{\mathbf{u}} = \mathbf{G} + \mathbf{C}. \quad (3)$$

In what follows, a new methodology for efficient computation of the force vector $\mathbf{C}$ stabilizing kinematical constraints is presented.

2.2 Relaxation of the constraints

The following equality constraints appear most widely in application:

$$\mathbf{A}(t, \mathbf{q}, \mathbf{u}) = \mathbf{0} \quad (4.a)$$

$$\delta(t, \mathbf{q}) = \mathbf{0} \quad (4.b)$$

where $\mathbf{q}$ denotes the generalized coordinate vector, and $\mathbf{0}$ is the zero vector. Equations (4.a) and (4.b) are generally nonholonomic and holonomic constraints, respectively. The latter can easily be converted into the former. For instance, introducing a positive*2 scalar parameter ω_δ with dimension of reciprocal time and defining

$$\mathbf{A} = \delta + \omega_\delta \delta \quad (5)$$

make the right-hand side dependent on the generalized velocity $\mathbf{u}$, assuring the convergence of δ to zero provided $\mathbf{A} \rightarrow \mathbf{0}$, thus enabling treatment as a 'non-holonomic' constraint. Latch-type inequality constraints can also be transformed to the equalities⁽⁷⁾. Hence all the constraints are assumed to be in the form of Eq.(4.a) hereafter.

The error vector $\mathbf{A}$ in Eq.(4.a) should be constantly zero for the ideal kinematical constraints. However, as mentioned in Section 1, it can give rise to numerical instability, and hence the constraints are often relaxed to allow small errors. Baumgarte⁽¹⁾ assumed exponential convergence of each component in the constraint error vector. The postulate is neither physics-based nor practical since the procedure requires $O(N^3)$ computational effort for obtaining the constraint force, where N is the number of the constraints. Moreover, the approach is not valid for systems where two or more constraints are nearly dependent and the solution matrix becomes ill-conditioned⁽²⁾. Noting the fact, a simple and effective alternative is proposed here. Consider the weighted norm of the constraint error vector $\mathbf{A}$ defined as

$$J \equiv (\mathbf{A}^T [\mathbf{S}] \mathbf{A})^{1/2}, \quad (6)$$

where $[\mathbf{S}]$ is a diagonal, constant, positive definite matrix for improving the numerical behavior. It is called scaling matrix, and an algorithm for determination of its components will be given later in Section 2.4. In this investigation, the norm J rather than the error vector $\mathbf{A}$ itself is assumed to diminish

*1 In this paper, a bold typeface is used for vectors, and bold symbols in square brackets denote matrices.

*2 For backward integration, the value of ω_δ should be negative.

exponentially with time, governed by the differential equation

$$\dot{J} + \omega J = 0 \quad (7)$$

to maintain the numerical stability. Here ω is the reciprocal of the time constant^{*3} for the error dynamics, and is referred to as constraint convergence parameter in this paper.

The parameter ω can be a constant^{(6),(7)}. However, to dispense with a priori choice of its value, the simple algorithm

$$\omega = \beta_{\text{int}} / (\Delta t_{\text{int}})_{\text{ave}} \quad (8)$$

is proposed in this paper. In the equation, $(\Delta t_{\text{int}})_{\text{ave}}$ denotes the integration step size averaged over an given period, and β_{int} is a constant dependent only on the selection of integrator. The approach is approximately equivalent to assuming the smallest time scales of the constraints and the state variables to be proportional. Updating the convergence parameter ω using Eq.(8) at appropriate intervals suffices most applications, as will be demonstrated in Section 4. Since the interval is extremely larger than the integration step, the parameter ω is regarded as a constant in the following formulation.

2.3 Constraint force

Since the constraint dynamics equation (7) is not sufficient to uniquely determine the constraint force vector $\mathbf{C}$ in Eq.(3), two additional conditions are introduced. The first one requires that the constraint force and the virtual displacement keep orthogonal, just as in the ideal kinematical constraints. The constraint force conforming with the postulate can be described as

$$\mathbf{C} = [\mathbf{A}_u]^T \boldsymbol{\lambda} \quad (9)$$

Here $[\mathbf{a}_b]$ is an abbreviation for the Jacobian matrix $\partial \mathbf{a} / \partial \mathbf{b}$, where $\mathbf{a}$ and $\mathbf{b}$ are arbitrary vectors. The second condition supposes that the Lagrangian multiplier vector in the right-hand side of the above equation is given in the form

$$\begin{aligned} \boldsymbol{\lambda} &= \boldsymbol{\lambda}_0 - \alpha [\mathbf{S}] \boldsymbol{\Delta} \\ &= \boldsymbol{\lambda}_0 - \alpha \boldsymbol{\Delta}_s \end{aligned} \quad (10)$$

where is the scaling matrix in Eq.(6), and

$$\boldsymbol{\Delta}_s = [\mathbf{S}] \boldsymbol{\Delta} \quad (11)$$

is the scaled constraint error vector. The vector $\boldsymbol{\lambda}_0$ in Eq.(10) is an estimate for the multiplier $\boldsymbol{\lambda}$, updated at each integration step as will be described later. The second term in the right-hand side of the equation corrects the estimate $\boldsymbol{\lambda}_0$ through feedback of the constraint error using the variable scalar gain α . Although the existing penalty methods⁽²⁾⁻⁽⁴⁾ adopt similar conditions, they fix the gains and hence the constraint error dynamics no more shows the simple

exponential convergence following Eq.(7), generally suffering long calculation time during the integration. The traditional penalty approaches have another disadvantage of the constraint instability for small gains. Since their minimum stable values generally vary with time and their evaluation is not straightforward, very large gains are usually selected demanding additional computational effort. All of these shortcomings are avoided by the proposed variable-gain error feedback.

Substituting Eq.(10) into Eq.(9) leads to the constraint force satisfying the two conditions as

$$\mathbf{C} = \mathbf{C}_0 - \alpha \mathbf{c} \quad (12)$$

where

$$\mathbf{C}_0 = [\mathbf{A}_u]^T \boldsymbol{\lambda}_0 \quad (13)$$

is the estimate for the constraint force, and

$$\mathbf{c} = [\mathbf{A}_u]^T [\mathbf{S}] \boldsymbol{\Delta} = [\mathbf{A}_u]^T \boldsymbol{\Delta}_s \quad (14)$$

is interpreted as its correction by unit gain ($\alpha = -1$) and hence is named direction vector of constraint force correction.

Substituting the constraint force of Eq.(12) into Eq.(3) and solving it for the generalized acceleration $\dot{\mathbf{u}}$ yield

$$\begin{aligned} \dot{\mathbf{u}} &= [\mathbf{M}]^{-1} (\mathbf{G} + \mathbf{C}_0 - \alpha \mathbf{c}) \\ &= \dot{\mathbf{u}}_0 - \alpha \dot{\mathbf{u}}_c \end{aligned} \quad (15)$$

Here the following symbols are introduced:

$$\dot{\mathbf{u}}_0 = [\mathbf{M}]^{-1} (\mathbf{G} + \mathbf{C}_0) \quad (16.a)$$

$$\dot{\mathbf{u}}_c = [\mathbf{M}]^{-1} \mathbf{c} \quad (16.b)$$

In Eq.(15), $\dot{\mathbf{u}}_0$ is the generalized acceleration before the correction of constraint force, from which $\dot{\mathbf{u}}_c$ is subtracted per unit gain. Appropriate control of the gain α can realize the desired response of the constraint error norm J described by Eq.(7). Differentiating Eq.(6) and manipulating Eq.(7) using the result as well as Eq.(15) reduce to the simple linear equation for the gain α

$$A \alpha = A_0 \quad (17)$$

yielding the desired error dynamics, where the scalars A and A_0 are defined as:

$$A = \mathbf{c}^T \dot{\mathbf{u}}_c \quad (18.a)$$

$$A_0 = \mathbf{c}^T \dot{\mathbf{u}}_0 + \boldsymbol{\Delta}_s^T \dot{\boldsymbol{\Delta}}|_{\dot{\mathbf{u}}=0} + \omega J^2 \quad (18.b)$$

Noting Eq.(16.b) and Eq.(18.a), the coefficient A is identical to $\mathbf{c}^T [\mathbf{M}]^{-1} \mathbf{c}$, and since the mass matrix $[\mathbf{M}]$ of the free system is always positive definite, Eq.(17) has a unique solution as long as the direction vector $\mathbf{c}$ is not zero. Even when the constraints become singular with the vectors in the Jacobian matrix $[\mathbf{A}_u]$ being linearly dependent, there is usually little possibility for the vector $\boldsymbol{\Delta}_s$ to lie in the corresponding null space yielding $\mathbf{c} = \mathbf{0}$. Hence this condition for unique solution of Eq.(17) is weaker than regularity of the constraint, implying robustness of the approach to singular constraints⁽⁶⁾. It is obvious from Eq.(14) that the direction vector $\mathbf{c}$ vanishes when the constraint error

^{*3} Similarly to ω_s in Eq.(5), ω takes positive values except for backward integration.

is exactly zero and then the gain α is no more obtainable from Eq.(17). However, since the correction of constraint force is unnecessary if the constraints are completely satisfied, the gain can safely be set to zero. In numerical applications, the same treatment is recommended when the scaled constraint error norm J reaches near machine accuracy of the computer, to circumvent problems due to round-off errors.

The proposed approach is rather insensitive to the choice of the estimate λ_0 for the multiplier vector, since the scaled error norm J converges to zero following Eq.(7) for an arbitrary value of λ_0 . In fact, the algorithm is valid even without λ_0 in Eq.(10) when the accuracy requirement is not severe^{(6),(7)}. However, since the error vector Δ generally has small components with less precision than that of the state variables, it should naturally be used only for correction of the constraint force C . The simplest way is to adopt the multiplier λ updated by Eq.(10) as the estimate λ_0 for the next integration step. It is desirable to hold λ_0 constant during the step for suppression of undesirable fluctuation in the constraint force.

Note, Eqs.(6), (10), (15) and (18) used in the algorithm involve only a fixed number of linear combinations and inner products, where the associated vectors have the dimension of the free system velocity u or the constraint error Δ . Therefore the proposed formulation has an $O(N)$ computational cost in terms of both the constrained and unconstrained degrees of freedom. It also makes complete exploitation of the sparsity of the Jacobian matrix $[\Delta_u]$ quite straightforward, and hence no matrix operations with dimensions of the order N are necessary in practice. This enables implementation to very simple general-purpose simulation programs. As mentioned before, the formulation has a higher level of robustness against singular constraints compared to Baumgarte's approach. Unlike the fixed-gain penalty methods⁽²⁾⁻⁽⁴⁾, the gain is obtained explicitly from Eq.(17) so that the convergence obeying Eq.(7) is attained, and hence it need not be provided a priori and stability of the constraints is automatically assured. Furthermore, the forward dynamics algorithm has no recursive operations and almost all the numerical procedures can be carried out separately for each body and joint, implying its fundamental amenability to parallel computation.

2.4 Scaling of the constraint error

The scaled norm J of the error vector Δ always satisfies Eq.(7) and converges exponentially, provided the gain α is given by Eq.(17). However, choice of the scaling matrix $[S]$ greatly affects the behavior of each component in Δ and hence numerical efficiency as well as accuracy. This can be viewed

clearly through rewriting the coefficient A of Eq.(18.a) using Eq.(16.b) and Eq.(14) as

$$A = \Delta^T [S] [D] [S] \Delta \quad (19)$$

where the matrix

$$[D] \equiv [\Delta_u] [M]^{-1} [\Delta_u]^T \quad (20)$$

does not depend on the scaling factors. Equation (19) implies that, without the scaling (i.e. when $[S]$ is the unit matrix) and the matrix $[D]$ has a large condition number, the algorithm tends to generate constraint force intensively in the direction corresponding to the largest eigenvalue until it reaches the limit, thus leading to unrealistic response in the constraint force. The scaling matrix $[S]$ is required to prevent such situations through reduction of the condition number.

Obtaining the exact eigenvalues is neither practical nor necessary. Since Eq.(20) shows that the matrix $[D]$ is symmetric and positive semidefinite (positive definite if the constraints are all independent), its eigenvalues can be roughly estimated from its diagonal elements. Hence selection of $[S]$ such that multiplication of $[D]$ by it from both sides makes all the diagonal components identical, i.e.

$$[S] = \text{diag}[\dots, \frac{1}{s_i}, \dots] / s_{\max} \quad (21.a)$$

$$s_i \equiv \begin{cases} 1/\sqrt{D_{i,i}} & (0 < D_{i,i}) \\ s_{\max} & (\text{otherwise}) \end{cases} \quad (21.b)$$

effectively decreases the condition number. Here $D_{i,i}$ denotes the i th diagonal element of the matrix $[D]$ and $s_{\max}$ is the maximum value of s_i ($0 < D_{i,i}$). Division by $s_{\max}$ in Eq.(21.a) normalizes the scaling matrix $[S]$ to have the maximum component being unity. The diagonal value $D_{i,i}$ can be obtained through multiplication of Eq.(20) from both sides by the unit vector

$$e_i = \{0, \dots, 0, \overset{i}{1}, 0, \dots, 0\}^T \quad (22)$$

which yields

$$D_{i,i} = c_i^T \ddot{u}_{ci} \quad (23.a)$$

$$c_i = [\Delta_u]^T e_i \quad (23.b)$$

$$\ddot{u}_{ci} = [M]^{-1} c_i \quad (23.c)$$

The above procedure is equivalent to the Jacobi preconditioning for iterative solution of linear equations⁽⁹⁾. Note, all the calculation involved in Eq.(23) can be accomplished using the subroutines for the direction vector c (Eq.(14)), the associated acceleration $\ddot{u}_c$ (Eq.(16.b)) and the factor A (Eq.(18.a)).

Although the scaling matrix $[S]$ should be evaluated at least once at the initial time, it need not be updated at every integration step. It can be kept constant throughout the simulation for moderate cases. When the condition number of the matrix $[D]$ of Eq.(20) varies widely with time, however, the scaling by Eq.(21) should be applied at proper intervals to renew the matrix $[S]$. Numerical experience implies that scaling once in around 1000 steps is enough for most systems, when the fifth-order Runge-

Fig. 1 PC cluster for the parallel computation

Kutta-Fehlberg method⁽¹⁰⁾ is used for integration.

3. Parallel Computation Using a PC Cluster

As already mentioned in the previous section, the proposed formulation has extreme generality and efficiency, and the latter can be improved further through parallel computation. Noting the advantages, a general multibody dynamics simulation program has been developed by the author. It is written in ANSI C++, and can deal with arbitrary types of structural members, as long as they are properly discretized. All the kinematical constraints described in Section 2.2 are supported.

The software can naturally be executed on stand-alone computers, and it also switches to the distributed algorithm on parallel processors. The variable-step fifth-order Runge-Kutta-Fehlberg method⁽¹⁰⁾ is adopted for integration, and since it is an explicit algorithm, the distributed calculation can be accomplished just dividing the state vector.

To test the present program, a PC cluster has been constructed at Kyushu Institute of Technology. A photograph of the hardware is shown in Fig. 1. The system consists of nine processors (Plat' Home Personal Station DP600 with Pentium III 600 MHz and Red Hat Linux 6.2J) connected through a 100 Base-TX network with a switching hub (Planex Communications FX-16NW). The processors exchange data using LAM (Local Area Multicomputer) conforming with the MPI (Message-Passing Interface) standard.

4. Examples

To validate the proposed approach, numerical

Fig. 2 Configuration of the crawler

analysis of particular systems is carried out.

4.1 Simulation of a crawler

First, dynamics of the crawler illustrated in Fig. 2 is studied. The system model consists of a chassis with a sprocket, rollers and an idler, enclosed by an endless chain of 42 track links interconnected by pin joints. Each track link has three grousers to grip the ground. The sprocket and the rollers are attached to the main body through revolute joints. The idler is allowed to slide relatively to the main body, and is pressed forward with a constant force of 20 000 N to generate tension of the chain. The sprocket has 25 gear teeth, and performs forced rotation at a specified angular velocity by a kinematical constraint. The teeth push the pins of the track links, thus driving the system. The cab etc. are included in the main body in terms of the inertia parameters, and hence are omitted in the figures of this paper. All the structural members are modeled as two-dimensional rigid bodies, whose degree of freedom is 153 in total. The whole system is subjected to uniform gravity and 101 constraint equations. Since the system contains a closed topology, its treatment using independent generalized coordinates is rather difficult. The number of the dynamical equations is rather small, and hence the simulation is carried out on a stand-alone notebook computer (SONY VAIO PCG-Z505CR/K with Mobile Pentium III 750 MHz) in this section.

The track links get in contact with the ground, the idler, the rollers, and the sprocket with its teeth. The associated force is modeled as

$$N(d_N, v_N) = (k + cv_N)d_N, \quad (24.a)$$

$$F(d_N, v_N, v_F) = \mu(v_F)N(d_N, v_N), \quad (24.b)$$

where d_N , v_N and v_F denote the normal displacement, its rate, and the sliding velocity, respectively. The normal force N accounts for the nonlinear damping⁽¹¹⁾. The factor μ for the friction force F is given as

$$\mu(v_F) = \frac{v_F}{|v_F| + v_0} \mu_0, \quad (25)$$

where μ_0 and v_0 are constants. Major physical and geometrical parameters are listed in Table 1.

At the initial time $t=0$, the system is assumed to

Table 1 Physical and geometrical parameters of the crawler

Mass [kg]	3373.33
Full length [m]	3.582
Length of a track link [m]	0.135
Radius of the idler [m]	0.2145
Radius of the sprocket [m]	0.22
Height of a sprocket tooth [m]	0.04
Radius of a roller [m]	0.05

Contact Force Parameters	
k [N/m]	1.0×10^6
c [Ns/m ²]	5.0×10^6
v_0 [m/s]	0.01
μ_0	0.3 (for the ground) 0.1 (for the others)

be at rest. The static equilibrium of the system is obtained through numerical simulation until $t=60$ s without driving the sprocket. Using the state as the initial condition, response of the system climbing over a bump is calculated. The bump is modeled using a sixth-degree polynomial. The width of the bump is fixed to $L=1.5H_c$, where $H_c=0.651$ m is the height of the chassis, and two cases with the bump height $H=0.5H_c$ and $0.25H_c$ are considered. Figure 3 shows the result up to $t=10$ s, where the maximum admissible error for the integration is taken as 10^{-5} . As clear from Fig. 3(a), the angular velocity of the sprocket relative to the main body is completely controlled by the imposed kinematical constraint resulting in rotation with a constant speed of 180 deg/s. Figure 3(b) represents the variation in the velocity component of the main body in the forward direction with respect to the body-fixed coordinate system. The figure implies that the crawler successfully clears the bump with the height $H=0.25H_c$ and keeps to proceed, whereas the system becomes stuck by the larger bump with $H=0.5H_c$. In the successful case, the chassis undergoes rapid pitching motion with respect to the inertial system, as can be seen from Fig. 3(c) and the frames of animation below in the figure. The torque λ_s of Fig. 3(d) is applied to the sprocket by the kinematical constraint to obtain the uniform rotation indicated in Fig. 3(a). The error norm J defined by Eq.(6) is maintained around 10^{-3} throughout the simulation in both cases (Fig. 3(e)), and hence the numerical stability of the algorithm has been validated. The constraint convergence parameter ω in Eq.(7) is adaptively controlled in inverse proportion to the averaged integration step size (Eq.(8)), as shown in Fig. 3(f). The successful results demonstrate the effectiveness of the control. The simulation

Fig. 3 Response of the crawler climbing over a bump

on the notebook computer required approximately 18 min and 32 min for the cases with $H=0.25H_c$ and $0.5H_c$, respectively, including the cost for the data output and the animation.

4.2 Parallel simulation of a flexible space structure

The simulation program used for the analysis in the previous section is able to handle any flexible systems, provided that its dynamics is written in the form of Eq.(1) through appropriate discretization. To demonstrate this capability, numerical study of a flexible frame structure depicted in Fig. 4 is tackled. The system is a rahmen consisting of 20 regular hexagonal frames with tubular members on their sides

Fig. 4 Configuration of the C60 fullerene structure with a thruster

and diagonals, and has 12 regular pentagonal holes. Its configuration is analogous to the C60 fullerene (soccer-ball shaped carbon molecule). All the tubular members are made of the 7075Al-T6 aluminum alloy ($\rho=2.80 \times 10^3 \text{ kg/m}^3$, $E=7.26 \times 10^{10} \text{ Pa}$) with a natural length of 2 m, an outer radius of 1 cm and a thickness of 1 mm. Each rod is divided into n_e finite elements, whose flexibility is discretized using absolute nodal coordinates through cubic Hermite interpolation. For the simplest case of $n_e=1$, the members have 2520 degree of freedom in total, and 1560 constraints are imposed to interconnect them forming the system model. The complexity of the dynamical equations increases with n_e . The challenging scale of the problem is quite appropriate for testing the parallel algorithm on the PC cluster described in Section 3.

Yamawaki⁽¹²⁾ suggested that the fullerene structure is suitable for application to communication, observation, solar-power satellites, etc., due to its omnidirectional nature and high rigidity per unit surface area. Hence an electric propulsion mission is assumed, where the system has solar arrays on the hexagonal frames, one of which has a thruster placed at the center (Fig. 4). The thruster starts to generate a constant thrust $f=0.1 \text{ N}$ at $t=0$ in free space without other external forces. To focus on the flexibility of the frame itself, mass and rigidity of the solar arrays are ignored.

Figure 5 shows the simulation result for $n_e=1$ up to $t=1.0 \text{ s}$, where the maximum admissible error for integration is set to 10^{-6} . Figure 5(a) represents change in the height of the structure (distance between the thruster and the node in its symmetrical position), caused by the step thrust. Results obtained using varied numbers of the processors ($N_{PC}=1$ to 9) are plotted in the graph, and they agree almost completely indicating the validity of the distributed algorithm as well as its implementation. Although the system dynamics entirely differs from that of the rigid

Fig. 5 Response of the C60 fullerene structure undergoing uniform thrust

crawler in the previous section, the proposed variable-gain error correction functions quite well. It can be confirmed from Fig. 5(b) and Fig. 5(c) showing time histories of the error norm and the controlled convergence parameter, respectively. The integration step size Δt_{int} in Fig. 5(d) averages out to a small value around $5 \times 10^{-4} \text{ s}$, which is attributed to the longitudinal vibration modes of the finite elements. As Fig. 5(a) implies the existence of system modes slower than 1 Hz, the equations of motion are extremely stiff. The explicit Runge-Kutta-Fehlberg method adopted in the present study is generally inefficient to deal with such models due to large number of steps involved.

Effectiveness of the distributed algorithm varies according to system degrees of freedom as well as the number of processors. To assess this, simulation of the flexible fullerene structure with different numbers of elements n_e for each rod is performed. Figure 6 shows the result in terms of the relative computational speed S_w defined as

$$S_w(N_{PC}) \equiv \frac{T_w(1)}{T_w(N_{PC})}, \quad (26)$$

where $T_w(N_{PC})$ denotes real time (wall-clock time) required for the simulation using N_{PC} processors. If the acceleration proportional to the number of processors is attained, $S_w=N_{PC}$ holds. The ideal state is represented by the straight line in Fig. 6. The graph indicates that the efficiency is improved as the system degree of freedom increases. For the case with $n_e=30$ where the structural members have 75 600 degree of freedom, nearly ideal acceleration of $S_w \approx 8$ at $N_{PC}=9$ is achieved. Since the overall speed of the

Fig. 6 Relative computational speed as a function of the number of the processors

simulator is greatly affected by the communication time, higher efficiency may be expected using faster network as well as the load balancing.

5. Concluding Remarks

A formulation useful for analyzing dynamics of multibody systems has been presented. It is based on a variable-gain error correction method, and its virtually matrixless property reflects remarkable generality and efficiency with other desirable features on the numerical program.

As mentioned in Section 4.2, equations of motion for detailed mathematical models tend to be fundamentally stiff. It is completely different from the artificial stiffness caused by approximation of the kinematical constraints, which can already be avoided using the proposed algorithm. To overcome the difficulty resulting from the physical stiffness, a numerical integrator with improved stability is required. Most of the implicit integration methods currently available are computationally expensive, spoiling the $O(N)$ property of the present forward dynamics algorithm. Noting the problem, future efforts are aimed at development of a stable, efficient numerical solver for ordinary differential equations with large degrees of freedom. Parallel processing of the integration is also an imperative issue.

Acknowledgements

The investigation in Section 4.1 was partially

supported by Takasago R & D Center of Mitsubishi Heavy Industries, LTD. The numerical results in Section 4.2 were obtained using the PC cluster of Kyushu Institute of Technology.

References

- (1) Baumgarte, J., Stabilization of Constraints and Integrals of Motion in Dynamical Systems, *Computer Methods in Applied Mechanics and Engineering*, Vol. 1 (1972), pp. 1-16.
- (2) Park, K.C. and Chiou, J.C., Stabilization of Computational Procedures for Constrained Dynamical Systems, *Journal of Guidance, Control, and Dynamics*, Vol. 11, No. 4, (1988), pp. 365-370.
- (3) Park, K.C., Chiou, J.C. and Downer, J.D., Explicit-Implicit Staggered Procedure for Multibody Dynamics Analysis, *Journal of Guidance, Control, and Dynamics*, Vol. 13, No. 3 (1990), pp. 562-570.
- (4) Bayo, E., de Jalon, J.G. and Serna, M.A., A Modified Lagrangian Formulation for the Dynamic Analysis of Constrained Mechanical Systems, *Computer Methods in Applied Mechanics and Engineering*, Vol. 71 (1988), pp. 183-195.
- (5) Kurdila, A.J., Junkins, J.L. and Hsu, S., Lyapunov Stable Penalty Methods for Imposing Holonomic Constraints in Multibody System Dynamics, *Non-linear Dynamics*, Vol. 4 (1993), pp. 51-82.
- (6) Nagata, T., A Unified Approach to Multibody Dynamics Formulation (Variable-Gain Penalty Algorithm), *Trans. Jpn. Soc. Mech. Eng.*, (in Japanese), Vol. 66, No. 649, C (2000), pp. 2919-2925.
- (7) Nagata, T., A Unified Approach to Multibody Dynamics Formulation (Inequality Constraints), *Trans. Jpn. Soc. Mech. Eng.*, (in Japanese), Vol. 66, No. 649, C (2000), pp. 2926-2930.
- (8) Nagata, T., Dynamics of Flexible Multibody Systems: a Formulation with Applications, Ph.D. Thesis, The University of Tokyo, Tokyo, Japan, 1995.
- (9) Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and der Vorst, H.V., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, 2nd Edition, SIAM, Philadelphia, PA, 1994.
- (10) Fehlberg, E., Classical Fifth-, Sixth-, Seventh-, and Eighth-Order Runge-Kutta Formulas with Step-size Control, NASA TR R-287, (1968).
- (11) Hunt, K.H. and Crossley, F., Coefficient of Restitution Interpreted as Damping in Vibroimpact, *Journal of Applied Mechanics*, Vol. 42, No. 2 (1975), pp. 440-445.
- (12) Yamawaki, K., Proposal for Construction of C60 Fullerene Space Structure, *Proceedings of the 43rd Space Sciences and Technology Conference* (in Japanese), (1999), p. 43.