Multibody System Dynamics **6:** 377–396, 2001.
© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

A Gluing Algorithm for Network-Distributed Multibody Dynamics Simulation

FANG-CHUNG TSENG and GREGORY M. HULBERT
Mechanical Engineering Department, University of Michigan, 2250 G.G. Brown, Ann Arbor, MI 48109-2125, U.S.A.; E-mail: hulbert@umich.edu

(Received: 22 November 1999; accepted in revised form: 19 April 2001)

Abstract. The improved performance and capacity of networks has made the combined processing power of workstation clusters a potentially promising avenue for solving computationally intensive problems across such distributed environments. Moreover, networks provide an ideal platform to employ heterogeneous hardware and software to solve multibody dynamics problems. One fundamental difficulty with distributed simulation is the requirement to couple and synchronize the distributed simulations. This paper focuses on the algorithms necessary to couple together separately developed multibody dynamics modules so that they can perform integrated system simulation. To identify a useful coupling strategy, candidate numerical algorithms in the literature are reviewed briefly – namely, stiff time integration, local parameterization, waveform relaxation, stabilized constraint and perturbation. An unobtrusive algorithm that may well serve this 'gluing' role is presented. Results from numerical experiments are presented and the performance of the gluing algorithm is investigated.

Key words: multibody dynamics, integrated system simulation, distributed computing.

1. Introduction

With the ongoing improvements in computer hardware and in modeling software, it has become common practice to perform high-fidelity dynamics simulations of complex systems to evaluate their overall performance. For example, in the case of the ground vehicles, high-fidelity models and simulations are available for engines, powertrains and vehicle dynamics. Despite the availability of such high-fidelity component models and simulations, it has been difficult to combine these resources to produce an integrated, complete vehicle simulation capability. Few efforts [1, 2] have gone beyond the common practice of treating simulation and analysis packages as stand-alone tools and simplifying the interactions amongst the disparate models and codes. This lack of system integration has inhibited the ability to design complex systems beyond the subsystem level.

Instead of employing a traditional partitioning mindset, we address system integration in the context of distributed computing [3] and the new notion of 'gluing'. The distribution of resources is not seen as a way to achieve goals such as massively parallel, high-speed computing. Rather it is viewed as the operating environment imposed on the analyst because of the existing overriding situations [4]. In our scenario, we are exposed naturally to a distributed environment: components are designed by different engineering groups, employing various CAD/CAE tools that are dispersed over many locations. The goal is to execute coupled system simulation without sacrificing the integrity of subsystem modeling and solution and to maintain the efficacy of the overall results. Thus, the ideal role of such system algorithms is to 'glue' all the modules together, coordinating the modules without excessive intrusions into the subsystem modules.

To attain this ultimate plug-and-play framework, four attributes of an effective gluing algorithm are proposed as guidelines:

- (I) *Sticky*: The inter-connection relations between subdomains should be well satisfied, i.e. coupling between subdomains should be resolved and captured.
- (II) *Green*: It should not contaminate subdomain solution strategy. The integrity of the individual model and solution methods should be maintained. Minimum modification of the original solution scheme is desired.
- (III) *Inexpensive*: The overhead should be minimized.
- (IV) *Pretty*: The results should be pretty; that is, the overall solution should be numerically correct within the bounds of the desired accuracy.

With proper glue, individual modules are naturally encapsulated as black boxes with consistent interfaces defined by the respective algorithms. Then, network deployment is simplified to the tasks of distributed computing. The Common Object Request Broker Architecture (CORBA) [5] was chosen for building the distributed simulation environment, using the commercial Orbix implementation [6]. CORBA is an industry standard for distributed, heterogeneous, object-oriented applications; it is open, robust, interoperable, and is supported across multiplatforms and multivendors.

We shall consider, in particular, a system composed of n_b multiple subdomains that can be modeled as a multibody system. The system joints characterize the interconnections of such a multibody system. When the multibody system is disconnected at the joints, the system is naturally decomposed into separate parts that represent a mapping to the individual engineering units responsible for developing the specific parts. Concisely, this framework defines a constrained mechanical system that can be described by the semi-discrete DAEs

$$\begin{aligned} \mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Psi}^T\boldsymbol{\lambda} &= \mathbf{Q}, \\ \boldsymbol{\Psi}\dot{\mathbf{q}} + \boldsymbol{\psi} &= \mathbf{0}, \end{aligned} \tag{1}$$

where $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} : t \rightarrow \mathbf{R}^n$ represent the generalized position, velocity and acceleration vectors; $\mathbf{M} : \mathbf{q} \times t \rightarrow \mathbf{R}^{n\times n}$ is the symmetric positive-definite generalized inertia matrix of the system; $\mathbf{Q} : \mathbf{q} \times \dot{\mathbf{q}} \times t \rightarrow \mathbf{R}^n$ is the generalized force vector; $\boldsymbol{\lambda} : t \rightarrow \mathbf{R}^m$ is the vector of the Lagrange multipliers; $\boldsymbol{\Psi} : \mathbf{q} \times t \rightarrow \mathbf{R}^{m\times n}$ is the constraint Jacobian matrix with $\operatorname{rank}(\boldsymbol{\Psi}) = m$; $\boldsymbol{\psi} : \mathbf{q} \times t \to \mathbf{R}^m$ is the kinematic excitation. (In the following, where we need to make reference to a specific equation line in a set of equations, the equation line number will be denoted by an appended lower case alphabetic letter. For example, in (1) above, the dynamical equation is referred to as (1a) while the Pfaffian constraint form is denoted by (1b).) Here the constraints are assumed to be of Pfaffian form, (1b). If the Pfaffian constraints can be integrated to the form, $\boldsymbol{\Phi} = \mathbf{0}$, $\boldsymbol{\Phi} : \mathbf{q} \times t \to \mathbf{R}^m$, the constraints are holonomic and the system is governed by the well-known index-3 DAEs

$$\begin{aligned} &\mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^T \boldsymbol{\lambda} = \mathbf{Q}, \\ &\boldsymbol{\Phi} = \mathbf{0}, \end{aligned} \tag{2}$$

with

$$\boldsymbol{\Phi}_{\mathbf{q}} \equiv (\partial \boldsymbol{\Phi} / \partial \mathbf{q}) : \mathbf{q} \times t \to \mathbf{R}^{m \times n}$$

assumed to be of full row rank. In what follows, we shall focus on this specific type of system model. The inertia matrix and the generalized forcing vector can be visualized as being decomposed across the subdomains as

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}^{(1)} & 0 & \cdots & 0 & 0 \\ 0 & \mathbf{M}^{(2)} & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & \mathbf{M}^{(n_b-1)} & 0 \\ 0 & 0 & \cdots & 0 & \mathbf{M}^{(n_b)} \end{bmatrix}, \tag{3}$$

$$\mathbf{Q} = [\mathbf{Q}^{(1)^T}, \ \mathbf{Q}^{(2)^T}, \ \ldots, \ \mathbf{Q}^{(n_n-1)^T}, \ \mathbf{Q}^{(n_b)^T}]^T, \tag{4}$$

where the superscripts identify each subdomain. The subdomain mass matrix, $\mathbf{M}^{(i)}$, $i = 1, n_b$, is assumed to be invertible.

In the context of network-distributed simulation, the system of Equations (2–4), is considered to be distributed across multiple simulations. In particular, the inertia matrix, (3), is never assembled completely; instead it is distributed across as many as n_b processors/simulations. The goal of a gluing algorithm is to satisfy (2–4) while maintaining this distributed equation structure. Of particular importance is the satisfaction of the constraint relations and the appropriate communication of the generalized constraint forces across the subdomains such that the constraint relations and system dynamics are computed accurately and stably.

Section 2 gives a brief overview of the numerical methods for DAEs. Section 3 presents the gluing algorithm with numerical examples that highlight their performance. Conclusions are drawn and directions for future work are provided in Section 4.

2. Classification of DAE Algorithms

Numerical approaches for the solution of differential equations defined on a manifold have attracted wide attention over the past several years. It is logical to evaluate these existing algorithms in the context of their efficacy as a gluing algorithm. To assist with this evaluation, in this section we classify numerical algorithms based upon their underlying intrinsic structure, and we present some representative algorithms within each category. The intent of this classification is not to review the body of literature on numerical solution of DAEs, but rather to illuminate the construction of gluing algorithms presented in the subsequent section. In particular, we classify algorithms that have been developed to address the complexities associated with constraint equations and constraint forces.

2.1. STIFF NUMERICAL TIME-INTEGRATION APPROACH

Orlandea et al. [7, 8], first took advantage of the sparse structure of (2) and directly applied stiff time-integration algorithms. The well-known computer software ADAMS employs this technique for simulating constrained mechanical systems.

Extending the work of Cardona and Géradin [9], Farhat et al. [10] presented a spectral stability theory for the incomplete field formulation. They showed that the linear constraints associated the incomplete field formulations introduce a destabilizing effect in linear dynamical systems that can be analyzed by investigating the behavior of time-integration algorithms at infinite and/or zero frequencies. In particular, they showed the Newmark method, having no numerical dissipation, engenders a weak instability that is excited for any time step size. Numerical simulations of multibody systems with nonlinear constraints also show this weak instability that rapidly destabilizes the acceleration field and corrupts the displacement solution as well. On the other hand numerical experiments with dissipative algorithms, like the generalized-α method [11], show stable results. Their work indicates that the success of stiff time-integration algorithms on high index DAEs relies on the numerical dissipation introduced by the algorithms.

Stiff integrators, like backward difference formula (BDF), often do not attain the same order of accuracy in some variables for higher index systems. It is particularly difficult to obtain an accurate solution for the algebraic variables, $\boldsymbol{\lambda}$. Often, the acceleration, velocity and the Lagrange multipliers suffer from an order reduction that causes the step size control to break down [12]. Thus, they are often removed from the error estimates in standard BDF-based routines such as DASSL [13].

2.2. LOCAL PARAMETERIZATION APPROACH

The m constraint equations (2b) define an n–m dimension manifold $\mathbf{M}$ in $\mathbf{R}^n$. In theory we can explore $\mathbf{M}$ by finding n–m independent variables, at least locally, to describe the dynamics of the constrained system. Many seemingly totally different numerical algorithms are based upon this concept. Their underlying merits can be discussed in the context of analytical dynamics via Maggi's equations

$$\begin{aligned}&\mathbf{P}(\mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}) = \mathbf{0},\\ &\mathbf{\Phi} = \mathbf{0},\end{aligned} \tag{5}$$

where

$$\mathbf{P}\mathbf{\Phi}_{\mathbf{q}}^{T} = \mathbf{0}. \tag{6}$$

For the holonomic constraints, the Jacobian matrix $\mathbf{\Phi}_{\mathbf{q}}$ is the gradient of $\mathbf{\Psi}$, which means the range space of $\mathbf{\Phi}_{\mathbf{q}}^{T}$ spans the normal space of the constrained manifold $\mathbf{M}$. It is further induced from (6) that the row space of $\mathbf{P}$ spans the n–m dimensional tangent space. Maggi's equations (5) concisely tell us that the dynamics of the constrained system occur only on the tangent space, which makes the solution always stay on $\mathbf{M}$.

Kurdila et al. [14] wisely point out the role of Maggi's equations for constrained mechanical systems and they provide an excellent discussion on various algorithms based on this concept. They argue that all the variants simply employ different bases to span the tangent space and to project the dynamics. Algorithms belonging to this class include the zero eigenvalue method [15], coordinate partition method [16], QR decomposition method [17], singular value decomposition method [18] and Kane's equations [19]. A convergence theorem and a numerical implementation of applying linear multistep numerical integration methods to local parameterization can be found in [20, 21].

In contrast to the stiff integrators, with the local parameterization approach, generalized constraint forces, $\boldsymbol{\lambda}$, can be easily recovered using

$$\boldsymbol{\lambda} = (\mathbf{\Phi}_{\mathbf{q}}^{+})^{T}(\mathbf{Q} - \mathbf{M}\ddot{\mathbf{q}}). \tag{7}$$

The superscript '$+$' operator represents the Moore–Penrose generalized inverse operation on the operand. Equation (7) is a by-product from the derivation of Maggi's equations, as shown by Papastravridis [22].

2.3. WAVEFORM RELAXATION (WR) METHOD

Relaxation methods afford a means of parallel numerical solution of systems of differential equations. The key is to partition the whole problem into several subdomains; coupling between each partition is relaxed in order to solve smaller size sub-problems separately. Iterations are usually needed to attain convergence. The waveform relaxation method is an iterative method for analyzing nonlinear dynamical systems in the time domain. It was originally applied to the time-domain analysis of large-scale integrated circuits [23]. Leimkuhler [24] extended the method to solve DAEs for constrained mechanical systems and considered two important non-linear cases: (i) a pair of multibody systems connected by 'soft' elements, and (ii) a pair of multibody systems connected through joints.

In the first case, a convergent iteration of the natural decoupling of the equations is reported. Considering a pair of bodies, the system can be modeled by

$$\begin{aligned} &\mathbf{M}_i \ddot{\mathbf{q}}_i + \boldsymbol{\Phi}^T_{i\mathbf{q}_i} \boldsymbol{\lambda}_i = \mathbf{Q}_i, \\ &\boldsymbol{\Phi}_i = \mathbf{0}, \end{aligned} \tag{8}$$

where $\mathbf{q}_i$, $\dot{\mathbf{q}}_i$ and $\ddot{\mathbf{q}}_i : t \to \mathbf{R}^{n_i}$; $\mathbf{M}_i : \mathbf{q}_i \times t \to \mathbf{R}^{n_i \times n_i}$; $\mathbf{Q}_i : \mathbf{q}_1 \times \dot{\mathbf{q}}_1 \times \mathbf{q}_2 \times \dot{\mathbf{q}}_2 \times t \to \mathbf{R}^{n_i}$; $\boldsymbol{\lambda}_i : t \to \mathbf{R}^{m_i}$; $\boldsymbol{\Phi}_i : \mathbf{q}_i \times t \to \mathbf{R}^{m}$; $\boldsymbol{\Phi}_{i\mathbf{q}_i} \equiv (\partial \boldsymbol{\Phi}_i / \partial \mathbf{q}_i) : \mathbf{q}_i \times t \to \mathbf{R}^{m_i \times n_i}$, $i = 1, 2$. Note the two systems are coupled only through the forcing terms $\mathbf{Q}_1$ and $\mathbf{Q}_2$. The constraint equations $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ are not coupled. Physically, this corresponds to systems that are interconnected only by spring and damper elements. Applying WR leads to iteration of the following form

$$\begin{aligned} &\mathbf{M}_1^{k+1} \ddot{\mathbf{q}}_1 + \boldsymbol{\Phi}_1^{k+1\,T}{}_{\mathbf{q}_1} \boldsymbol{\lambda}_1 = \mathbf{Q}_1(\mathbf{q}_1^{k+1}, \dot{\mathbf{q}}_1^{k+1}, \mathbf{q}_2^{k}, \dot{\mathbf{q}}_2^{k}, t), \\ &\boldsymbol{\Phi}_1 = \mathbf{0}, \end{aligned} \tag{9}$$

and

$$\begin{aligned} &\mathbf{M}_2^{k+1} \ddot{\mathbf{q}}_2 + \boldsymbol{\Phi}_2^{k+1\,T}{}_{\mathbf{q}_2} \boldsymbol{\lambda}_2 = \mathbf{Q}_2(\mathbf{q}_1^{k}, \dot{\mathbf{q}}_1^{k}, \mathbf{q}_2^{k+1}, \dot{\mathbf{q}}_2^{k+1}, t), \\ &\boldsymbol{\Phi}_2 = \mathbf{0}, \end{aligned} \tag{10}$$

where k is the iteration counter. During a given iteration, each subdomain only needs the coupling information of the other subdomains from the previous iteration. Parallel implementation is straightforward. Convergence of the above iterations was demonstrated, provided consistent initial conditions are given. The remaining challenge is then how to choose a proper time step size in order to obtain rapid convergence properties for use in practical applications.

The results seem promising in the first case. However, two attempted partitions of the second scenario lead to iteration divergence. In practical applications, many multibody systems of interest belong to the latter type: the system is coupled through the nonlinear constraint equations. Thus, the strategy of splitting systems at the constraint interface and applying WR iteration will fail to converge. One simple remedy may be to derive the underlying ODEs before WR can be applied. More research is required to devise a generic relaxation algorithm for multibody systems.

2.4. STABILIZED CONSTRAINT APPROACH

One can reduce the index of DAEs by repeatedly differentiating the constraint to take advantage of their better numerical properties for general time integration schemes. Differentiation of the constraint equations will transform the original high index DAEs to a lower index system with invariants. One well-known problem associated with this transformation is that the solution may drift off from the non-differentiated invariant manifolds because of accumulated errors from the time integration scheme. Some stabilization measures must be taken if longer time simulation is required. Among the numerous stabilization approaches, some representative ideas are mentioned in the following. Many stabilization methods can be seen as variants or combinations of these key ideas.

Baumgarte's stabilization [25] is probably the most widely known and used scheme for engineering applications. It replaces the holonomic constraint (2b) by a linear combination of the constraints and their time derivatives in such a way that the differential equations for the constraints are stable. Recent efforts have been reported that apply modern control theory to derive better stabilization forms, see, e.g., [26, 27].

GGL index-2 formulation [28] is another widely used method in simulation. Both the velocity constraints and the position invariants are imbedded in the equations of motion. The resulting index-2 system can be solved more easily than the original higher index system. The price to be paid is solving the augmented set of equations simultaneously.

Park et al. [29] adopted the penalty procedure $\boldsymbol{\lambda} = \boldsymbol{\varepsilon}^{-1}\boldsymbol{\Phi}$, $\boldsymbol{\varepsilon} \to 0$ by differentiating it once with $\boldsymbol{\varepsilon}$ assumed constant to obtain

$$\dot{\boldsymbol{\lambda}} = \boldsymbol{\varepsilon}^{-1}\dot{\boldsymbol{\Phi}}. \tag{11}$$

Formula (11) together with (2a) are a set of ODEs that can be integrated using standard numerical algorithms. Park further showed that the errors committed in the constraint relation would decay according to the different corresponding response time constants of the system, in contrast to the single time constant of Baumgarte's stabilization. Park also reported that even when $\boldsymbol{\Phi}_{\mathbf{q}}\mathbf{M}^{-1}\boldsymbol{\Phi}_{\mathbf{q}}^{T}$ is almost singular, their stabilization scheme experiences no numerical difficulty. A parallel multibody dynamics analysis algorithm that combines Park's stabilization and an explicit-implicit staggered procedure [30] can be found in [31].

Bayo et al. [32] presented a modified Lagrangian formulation for the dynamics of constrained mechanisms. They proposed to append the constraint relation to the Lagrange equations, by means of a penalty. The resulting augmented formulation excludes the Lagrange multipliers but includes parameters similar to those of Baumgarte's methods. The total number of equations is independent of the number of constraints and remains the same. Better yet, the resulting system is a set of ODEs, so standard numerical integration can be used. One variant, obtained by combining mass-orthogonal projection and position, velocity invariant projection, can be found in [33].

2.5. PERTURBATION METHOD

The perturbation approach also is a constraint stabilization method. It differs from the standard constraint stabilization approach in that the stabilization is performed independently from the numerical integration. The kernel of the method is to solve ODEs with invariants. By themselves, the ODEs are sufficient to uniquely determine the solution with consistent initial conditions; i.e. IC's that satisfy the invariants. The challenge is to ensure that the invariants are satisfied at any subsequent time step by the numerical algorithms. These invariants typically have specific physical meaning, e.g., mechanical systems with energy conservation or momentum conservation. Conservation of these invariants not only justifies the solution but also can improve the solution precision.

The basic idea of maintaining the invariants is to perturb the solution after each time step by just the minimum amount necessary to satisfy the invariants. The underlying ODEs of the DAEs (2) can be derived by first solving for $\boldsymbol{\lambda}$ from the index-1 DAEs

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}^T \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ -(\dot{\boldsymbol{\Phi}}_{\mathbf{q}} + \dot{\boldsymbol{\Phi}}_t) \end{bmatrix} \tag{12}$$

and substituting the result into (12a), to obtain the explicit form of the underlying ODEs

$$\mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^T(\boldsymbol{\Phi}_{\mathbf{q}}\mathbf{M}^{-1}\boldsymbol{\Phi}_{\mathbf{q}}^T)^{-1}(\dot{\boldsymbol{\Phi}}_{\mathbf{q}}\dot{\mathbf{q}} + \dot{\boldsymbol{\Phi}}_t + \boldsymbol{\Phi}_{\mathbf{q}}\mathbf{M}^{-1}\mathbf{Q}) = \mathbf{Q}. \tag{13}$$

Two extra invariants associated with (13) arise because the differentiated acceleration constraints are used in (12). If the system begins with consistent initial conditions, the solution should satisfy the displacement and velocity invariants at any subsequent time step. Using any time integration method, the ODEs (13) are solved; the results are then perturbed so that the invariants are satisfied. This process is characterized by the projection of the solution onto the invariant manifolds. It is usually referred as the coordinate projection method in the applied mathematics literature. The projection can be regarded as the solution of the minimization problem

$$\begin{aligned} &\text{Given } \mathbf{q}, \mathbf{v} \\ &\min_{\tilde{\mathbf{q}}} \|\tilde{\mathbf{q}} - \mathbf{q}\| \quad \text{subject to } \boldsymbol{\Phi}(\tilde{\mathbf{q}}) = \mathbf{0} \\ &\min_{\tilde{\mathbf{v}}} \|\tilde{\mathbf{v}} - \mathbf{v}\| \quad \text{subject to } \dot{\boldsymbol{\Phi}}(\tilde{\mathbf{q}}, \tilde{\mathbf{v}}) = \mathbf{0} \end{aligned} \tag{14}$$

for the perturbed values $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$. By proper definition of the norm operator $\| \bullet \|$, (14) represents a wide spectrum of projection choices.

Shampine [34] first showed how the perturbations by (14) are contained and proved the convergence of the perturbed solutions with the use of one-step methods for solving ODEs. He showed that the perturbations made to the solution do not seriously disturb the step size selection process for one-step methods. In particular, if in each step the local error is controlled within tolerance by the step size selection, a perturbation to the solution at the end of the step to satisfy the invariants will not induce an error larger than the tolerance, provided the norm to control the local error and the norm to compute the minimum norm perturbation are the same.

However, he warned of a side effect of this perturbation process when employing multistep methods, noting that modern multistep methods monitor the smoothness of the solution and adapt the order of methods appropriately. Perturbations to solution values can cause the solution to look rough. Such roughness can cause the code to lower the order. The low average order and the constant fluctuation of the order can disastrously affect the efficiency of the numerical integration.

Eich [35] gave a convergence theorem for the combination of the BDF method and the coordinate projection method. He showed that only errors that lie in the manifold given by the invariants are propagated. In particular, the projected k-step BDF method ($1 \leq k \leq 6$) has the same order of convergence as the corresponding non-projected method if the time step size is sufficiently small. As opposed to Shampine's comments about projected multistep methods, Eich argued that good results with variable order and step size can be obtained, because in no part of his proof was the constancy of step size and order invoked. His only assumption is that the non-projected method is stable. He also wisely pointed out it is not necessary to explicitly use the underlying ODEs (13). The discretized version of the index-1 system (12) can be used instead because semi-explicit index-1 DAEs behave like ODEs for most integration methods, especially for the BDF methods. This observation will relieve us from the expensive decomposition of $\mathbf{\Phi_q M}^{-1}\mathbf{\Phi_q}^T$ at every time step.

3. Gluing Algorithm

Yen and coworkers [36, 37], presented a coordinate-split (CS) technique for the numerical solution of the index-2 form DAEs for flexible mechanism dynamics. These methods, which extend the α-methods for ODEs of structural dynamics to DAEs, possess numerical dissipation that can be controlled by the user. A coordinate-split modification (CM) to the Newton iteration was further adopted to improve the convergence for highly oscillatory systems.

We recognize that the CS technique is another numerical implementation of Maggi's equations. We construct a simplified Newton iteration in such a way that the solution of the dynamics and the constraint satisfaction are separated. The solution of the dynamics can be easily mapped into parallel executions for individual subdomains while the constraint satisfaction efforts can be regarded as the coordination actions. Therefore, this strategy is a fine candidate for a gluing algorithm.

3.1. DERIVATION

Suppose the orthonormal subspaces of the constraint Jacobian matrix $\mathbf{\Phi_q}$ have been identified as $\mathbf{U}$ and $\mathbf{V}$, where $\mathbf{U} \in \mathbf{R}^{(n-m)\times n}$, $\mathbf{V} \in \mathbf{R}^{m\times n}$. The rows of $\mathbf{U}$ and $\mathbf{V}$ constitute the standard basis for $\mathbf{R}^n$. We define the following operators for later use.

$$\mathbf{P}_\perp \equiv \mathbf{\Phi_q^+ \Phi_q} = \mathbf{V}^T\mathbf{V},$$

$$\mathbf{P}_{||} \equiv \mathbf{I} - \mathbf{P}_{\perp} = \mathbf{U}^T\mathbf{U}. \tag{15}$$

The system is written in stabilized index-2 GGL formulation of the constrained equations of motion

$$\begin{aligned}
&\dot{\mathbf{q}} - \mathbf{v} + \mathbf{\Phi}_{\mathbf{q}}^T \boldsymbol{\mu} = \mathbf{0},\\
&\mathbf{M}\dot{\mathbf{v}} + \mathbf{\Phi}_{\mathbf{q}}^T \boldsymbol{\lambda} = \mathbf{0},\\
&\mathbf{\Phi}_{\mathbf{q}}\mathbf{v} = \mathbf{0},\\
&\mathbf{\Phi} = \mathbf{0}.
\end{aligned} \tag{16}$$

Like Maggi's equations, we pre-multiply (16a) and (16b) by an annihilation matrix $\mathbf{P}$ such that $\mathbf{P\Phi}_{\mathbf{q}}^T = \mathbf{0}$, resulting in the following index-1 DAEs

$$\begin{aligned}
&\mathbf{P}(\dot{\mathbf{q}} - \mathbf{v}) = \mathbf{0},\\
&\mathbf{P}(\mathbf{M}\dot{\mathbf{v}} - \mathbf{Q}) = \mathbf{0},\\
&\mathbf{\Phi}_{\mathbf{q}}\mathbf{v} = \mathbf{0},\\
&\mathbf{\Phi} = \mathbf{0},
\end{aligned} \tag{17}$$

where the CS matrix $\mathbf{P}$ is chosen to be $\mathbf{U}$. Applying a kth-order BDF formula to (17) yields the nonlinear system

$$\begin{aligned}
&\mathbf{P}(\mathbf{q}_n)h(\rho_h \mathbf{q}_n - \mathbf{v}_n) = \mathbf{0},\\
&\mathbf{P}(\mathbf{q}_n)h(\mathbf{M}(\mathbf{q}_n)\rho_h \mathbf{v}_n - \mathbf{Q}(\mathbf{v}_n, \mathbf{q}_n, t_n)) = \mathbf{0},\\
&\mathbf{\Phi}_{\mathbf{q}}(\mathbf{q}_n)\mathbf{v}_n = \mathbf{0},\\
&\mathbf{\Phi}(\mathbf{q}_n) = \mathbf{0},
\end{aligned} \tag{18}$$

where ρ_h is the discretization operator,

$$\rho_h \mathbf{q}_n = \frac{1}{h}\sum_{i=0}^{k} \alpha_i \mathbf{q}_{n-1}$$

in which α_i are the coefficients of the BDF method and h the step size of the time discretization. Applying Newton-type methods to (18) requires the solution of the linear system

$$\mathbf{J}(\mathbf{q}_n, \mathbf{v}_n)\begin{bmatrix} \mathbf{\Delta q}_n \\ \mathbf{\Delta v}_n \end{bmatrix} = -\mathbf{r}(\mathbf{q}_n, \mathbf{v}_n), \tag{19}$$

where $\mathbf{\Delta q}_n$ and $\mathbf{\Delta v}_n$ are the increments of $\mathbf{q}_n$ and $\mathbf{v}_n$,

$$\mathbf{J}(\mathbf{q}_n, \mathbf{v}_n) = \begin{bmatrix} \mathbf{P}(\mathbf{q}_n)\left(\dfrac{\partial \boldsymbol{\Phi}_{\mathbf{q}}^T(\mathbf{q}_n)}{\partial \mathbf{q}_n}\mathbf{s}_1 + h\dfrac{\partial \rho_h \mathbf{q}_n}{\partial \mathbf{q}_n}\right) & -h\mathbf{P}(\mathbf{q}_n) \\ \mathbf{P}(\mathbf{q}_n)\left(\dfrac{\partial \boldsymbol{\Phi}_{\mathbf{q}}^T(\mathbf{q}_n)}{\partial \mathbf{q}_n}\mathbf{s}_2 + \dfrac{\partial \mathbf{r}_2(\mathbf{q}_n, \mathbf{v}_n)}{\partial \mathbf{q}_n}\right) & \mathbf{P}(\mathbf{q}_n)\dfrac{\partial \mathbf{r}_2(\mathbf{q}_n, \mathbf{v}_n)}{\partial \mathbf{v}_n} \\ \dfrac{\partial(\boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q}_n)\mathbf{v}_n)}{\partial \mathbf{q}_n} & \boldsymbol{\Phi}_q(\mathbf{q}_n) \\ \boldsymbol{\Phi}_q(\mathbf{q}_n) & \mathbf{0} \end{bmatrix}, \quad (20)$$

and where

$$\mathbf{r}(\mathbf{q}_n, \mathbf{v}_n) = \begin{bmatrix} \mathbf{P}(\mathbf{q}_n)\mathbf{r}_1 \\ \mathbf{P}(\mathbf{q}_n)\mathbf{r}_2 \\ \boldsymbol{\Phi}_{\mathbf{q}}(\mathbf{q}_n)\mathbf{v}_n \\ \boldsymbol{\Phi}(\mathbf{q}_n) \end{bmatrix} \quad (21)$$

in which

$$\begin{aligned} \mathbf{s}_1 &= -(\mathbf{V}\boldsymbol{\Phi}_q^T)^{-1}\mathbf{V}\mathbf{r}_1, \quad \mathbf{s}_2 = -(\mathbf{V}\boldsymbol{\Phi}_q^T)^{-1}\mathbf{V}\mathbf{r}_2, \\ \mathbf{r}_1 &= h(\rho_h \mathbf{q}_n - \mathbf{v}_n) \quad \text{and} \quad \mathbf{r}_2 = h(\mathbf{M}(\mathbf{q}_n)\rho_h \mathbf{v}_n - \mathbf{Q}(\mathbf{v}_n, \mathbf{q}_n, t_n)). \end{aligned}$$

It is easy to verify that

$$\mathbf{P}\mathbf{P}_{||} = \mathbf{P}. \quad (22)$$

Then, we can rewrite the first two formulae of (19) as

$$\begin{bmatrix} \mathbf{P}(\mathbf{q}_n) & \mathbf{0} \\ \mathbf{0} & \mathbf{P}(\mathbf{q}_n) \end{bmatrix}\left(\mathbf{J}_h(\mathbf{q}_n, \mathbf{v}_n)\begin{bmatrix} \Delta\mathbf{q}_n \\ \Delta\mathbf{v}_n \end{bmatrix} + \begin{bmatrix} \mathbf{P}_{||}(\mathbf{q}_n)\mathbf{r}_1 \\ \mathbf{P}_{||}(\mathbf{q}_n)\mathbf{r}_2 \end{bmatrix}\right) = 0, \quad (23)$$

where, after we adopt the CM modification,

$$\mathbf{J}_h(\mathbf{q}_n, \mathbf{v}_n) = \begin{bmatrix} h\dfrac{\partial \rho_h \mathbf{q}_n}{\partial \mathbf{q}_n} & -h\mathbf{I} \\ \dfrac{\partial \mathbf{r}_2(\mathbf{q}_n, \mathbf{v}_n)}{\partial \mathbf{q}_n} & \dfrac{\partial \mathbf{r}_2(\mathbf{q}_n, \mathbf{v}_n)}{\partial \mathbf{v}_n} \end{bmatrix}. \quad (24)$$

The simplified Newton iteration of the linear system, (23) and the last two formulae of (19), can be shown to be

$$\begin{aligned} \Delta\mathbf{q}_n &= \mathbf{P}_{||}(\mathbf{q}_n)\Delta\hat{\mathbf{q}}_n - \boldsymbol{\Phi}_{\mathbf{q}}^+(\mathbf{q}_n)\boldsymbol{\Phi}(\mathbf{q}_n), \\ \Delta\mathbf{v}_n &= \mathbf{P}_{||}(\mathbf{q}_n)\Delta\hat{\mathbf{v}}_n - \boldsymbol{\Phi}_{\mathbf{q}}^+(\mathbf{q}_n)\dot{\boldsymbol{\Phi}}(\mathbf{v}_n, \mathbf{q}_n + \Delta\mathbf{q}_n), \end{aligned} \quad (25)$$

in which $\Delta\hat{\mathbf{q}}_n$ and $\Delta\hat{\mathbf{q}}_n$ are solved from

$$\mathbf{J}_h\begin{bmatrix} \Delta\hat{\mathbf{q}}_n \\ \Delta\hat{\mathbf{q}}_n \end{bmatrix} = -\begin{bmatrix} \mathbf{P}_{||}(\mathbf{q}_n)\mathbf{r}_1 \\ \mathbf{P}_{||}(\mathbf{q}_n)\mathbf{r}_2 \end{bmatrix}. \quad (26)$$

An interesting feature is noted from the simplified Newton iterations (25) and (26). The modified linear problem (26) is engaged in solving the dynamics only. It preserves the sparsity of the system and is independent of the constraint equations. After these relaxed incremental values, $\mathbf{\Delta\hat{q}}_n$ and $\mathbf{\Delta\hat{v}}_n$, are calculated, they are corrected by (25), which involves diminishing any constraint violations. This is a nice general structure that warrants further exploration. We would like to extend this to the acceleration level, because the acceleration generally is one of the desired solution variables. A natural extension of (25) for the incremental acceleration, with known $\mathbf{\Delta q}_n$ and $\mathbf{\Delta v}_n$, is

$$\mathbf{\Delta a}_n = \mathbf{P}_{||}(\mathbf{q}_n)\mathbf{\Delta\hat{a}}_n - \mathbf{\Phi}^{+}_{\mathbf{q}}(\mathbf{q}_n)\ddot{\mathbf{\Phi}}(\mathbf{a}_n, \mathbf{v}_n + \mathbf{\Delta v}_n, \mathbf{q}_n + \mathbf{\Delta q}_n). \tag{27}$$

The convergence of the dynamics and the constraints are monitored separately by respective residues of the dynamics, the acceleration, velocity and position constraints

$$\begin{aligned}
&\mathbf{r}_{\text{dynamic}} \equiv \mathbf{P}(\mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}),\\
&\mathbf{r}_{\text{acc}} \equiv \ddot{\mathbf{\Phi}},\\
&\mathbf{r}_{\text{vel}} \equiv \dot{\mathbf{\Phi}},\\
&\mathbf{r}_{\text{pos}} \equiv \mathbf{\Phi}.
\end{aligned} \tag{28}$$

Assuming basic update formula are used in the discretization (see, e.g., [11], for more details on these equations and choices for the Newmark parameters γ and β),

$$\begin{aligned}
\mathbf{q}_{n+1} &= \mathbf{q}_n + h\mathbf{v}_n + \frac{h^2}{2}((1-2\beta)\mathbf{a}_n + 2\beta\mathbf{a}_{n+1}),\\
\mathbf{v}_{n+1} &= \mathbf{v}_n + h((1-\gamma)\mathbf{a}_n + \gamma\mathbf{a}_{n+1}),
\end{aligned} \tag{29}$$

the first version of the iteration loop for the gluing algorithm is proposed as follows (in pseudo code). The subscript n and function variables are droped for clarity; $\bar{\mathbf{K}}$ is the Jacobian matix associated with the discretized unconstrained problem.

```
1. IF $||\mathbf{r}_{\text{dynamics}}|| \leq$ Dynamic Tolerance AND $||\mathbf{r}_{\text{acc}}|| \leq$ Acc. Constraint Tolerance
     then problem solved

       BREAK

   ELSE IF $||\mathbf{r}_{\text{dynamics}}|| >$ Dynamic Tolerance
     incrementally solve the unconstrained problem

       $\bar{\mathbf{K}}\mathbf{\Delta\hat{a}} = -\mathbf{P}_{||}(\mathbf{Ma} - \mathbf{Q})$          (GL-DYN)

     and perform the acceleration projection (27)

       $\mathbf{\Delta a} = \mathbf{P}_{||}\mathbf{\Delta\hat{a}} - \mathbf{\Phi}^{+}_{\mathbf{q}}\ddot{\mathbf{\Phi}}$          (GL-ACC)
```

```
    ELSE
      perform the acceleration projection only
          $\mathbf{\Delta a} = \mathbf{P}_{\|}\mathbf{\Delta \hat{a}} - \mathbf{\Phi}_{\mathbf{q}}^{+}\ddot{\mathbf{\Phi}}$                (GL-ACC)
    END
2. Recover the incremental displacement and velocity values.
          $\mathbf{\Delta q} = h^2\boldsymbol{\beta}\mathbf{\Delta a}$,
          $\mathbf{\Delta v} = h\boldsymbol{\gamma}\mathbf{\Delta a}$.
3. IF $\|\mathbf{r}_{\mathrm{vel}}\| >$ Vel. Constraint Tolerance OR $\|\mathbf{r}_{\mathrm{posl}}\| >$ Pos. Constraint Tolerance
      perform the position and velocity projection (25)
          $\mathbf{\Delta q} = \mathbf{P}_{\|}\mathbf{\Delta q} - \mathbf{\Phi}_{\mathbf{q}}^{+}\mathbf{\Phi}$,
          $\mathbf{\Delta v} = \mathbf{P}_{\|}\mathbf{\Delta v} - \mathbf{\Phi}_{\mathbf{q}}^{+}\dot{\mathbf{\Phi}}$.          (GL-POS, GL-VEL)
    END
```

After attaining convergence, the generalized constraint forces can be recovered using (7).

In structure, this version is as close to an ideal gluing algorithm as possible. We first solve the *unconstrained* problem (GL-DYN) iteratively and then perform the projection to satisfy the constraints (GL-ACC), (GL-VEL) and (GL-POS). The operators involved in the projection can be constructed once the constraint Jacobian $\mathbf{\Phi}_{\mathbf{q}}$ is known. No other information is required from the system. However, when implemented for a simple parallel four-bar linkage problem, the algorithm does not converge. Before abandoning hope to derive a practical gluing algorithm, we refer to one lesser-known result from analytical dynamics for inspiration.

3.2. GAUSS'S PRINCIPLE OF LEAST CONSTRAINT

Closely related to the Gibbs–Appell equations of motion, this beautiful and powerful theorem was discovered by Gauss in 1829 [38]. It states that the acceleration of the constrained mechanical system, $\mathbf{a}$, can be determined from the following minimization problem, provided that the configuration and the velocity, $\mathbf{q}$ and $\mathbf{v}$, of the system are known,

$$\min_{\mathbf{a}}(\mathbf{a}-\mathbf{a}_u)^T\mathbf{M}(\mathbf{a}-\mathbf{a}_u) \quad \text{subject to } \ddot{\mathbf{\Phi}}(\mathbf{a},\mathbf{v},\mathbf{q},t)=0, \tag{30}$$

where $\mathbf{a}_u = \mathbf{M}^{-1}\,\mathbf{Q}(\mathbf{v},\mathbf{q},t)$ is the unconstrained acceleration. Udwadia and Kalaba [39] showed the explicit form of the Gauss's principle to be

$$\mathbf{a} = \mathbf{a}_u - \mathbf{M}^{-1/2}(\mathbf{\Phi}_q\mathbf{M}^{-1/2})^{+}\ddot{\mathbf{\Phi}}(\mathbf{a}_u,\mathbf{v},\mathbf{q},t). \tag{31}$$

In incremental form we obtain

$$\mathbf{\Delta a} = \mathbf{\Delta \hat{a}} - \mathbf{M}^{-1/2}\left(\mathbf{\Phi}_q\, \mathbf{M}^{-1/2}\right)^{+} \ddot{\mathbf{\Phi}}. \tag{32}$$

This acceleration projection formula must be used instead of the naïve extension, (27), in order to satisfy the Gauss principle of least constraint. The new version of the gluing algorithm is obtained by the replacement of (GL-ACC) with (32). Gauss's principle informs us that we cannot minimize exchange of data between the solution of the dynamics and constraint satisfaction as ideally as we had hoped. The acceleration projection needs information about the mass matrix of the system. This extra information implicitly couples these two procedures together.

3.3. NUMERICAL EXAMPLE

A planar four-bar linkage, as presented in [17], is used to demonstrate the features of the proposed algorithm. The generalized coordinates q_1, q_2, and q_3 are defined by the relative rotation angles as in Figure 1. Body 1 of the linkage is ground and revolute joints connect all the bodies. If bodies 2 and 4 have the same link length, they will rotate in a parallel motion. The joint between bodies 1 and 4 is selected as a cut-joint to make an open chain. This will give us the constraint equations of the three generalized coordinates as

$$\mathbf{\Phi} = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) - l_2 \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) + l_3 \sin(q_1 + q_2 + q_3) \end{bmatrix} = \mathbf{0}. \tag{33}$$

Simulation is carried out for the specific case: $m_2 = m_4 = 10$ kg, $m_3 = 20$ kg, $l_1 = l_3 = 1$ m, $l_2 = 2$ m, $J_2 = J_4 = 1$ kg $\times$ m^2, $J_3 = 2$ kg$\times$m^2, $q_1(0) = \pi/2$ rad, $\dot{q}(0) = 2\pi$ rad/s,which m_i and J_i are the mass and the mass moment of inertia of body i, respectively. To check the results of the gluing algorithm, the solution of the reduced equation of motion

$$\left(J_2 + J_4 + (m_2 + 4m_3 + m_4)\frac{l_1^2}{4}\right)\ddot{q}_1 = \tau \tag{34}$$

is used as a reference, where $\tau = -2.0 \times t$ nt $\cdot$ m is the driving torque on body 2.

We take a rather large time step, 0.4 sec, to illustrate some interesting characteristics of the algorithm. As seen in Figures 2–5, it is quite impressive that with such large time step sizes the responses are well captured without too much overhead effort; generally, two iterations per time step suffice. We attribute these excellent results to two main characteristics.

First, we observe that if the three generalized coordinates satisfy constraints, i.e. $q_1 + q_2 = 2\pi$, $-q_1 + q_3 = \pi$, the Jacobian matrix for the cut-joint constraint equations (33), can be simplified as

$$\mathbf{\Phi}_q = \begin{bmatrix} 0 & l_1 \sin q_1 & l_1 \sin q_1 \\ l_2 & l_2 - l_1 \cos q_1 & -l_1 \cos q_1 \end{bmatrix}. \tag{35}$$

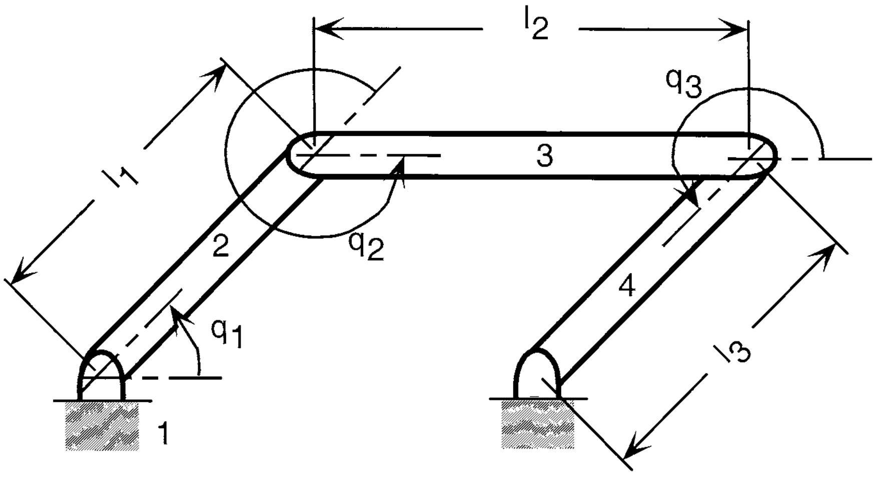

Figure 1. Parallel four-bar linkage.

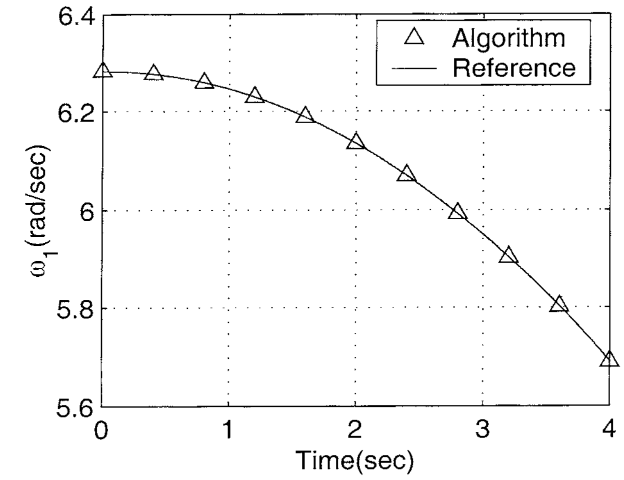

Figure 2. $\dot{q}_1$.

The QR decomposition of the Jacobian matrix is

$$\Phi_q^T = \begin{bmatrix} 0 & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{3}} \end{bmatrix} \begin{bmatrix} \sqrt{2} l_1 \sin q_1 & \sqrt{2}\left(\frac{l_2}{2} - l_1 \cos q_1\right) \\ 0 & \frac{\sqrt{6}}{2} l_2 \\ 0 & 0 \end{bmatrix}. \quad (36)$$

The above decomposition shows that the tangent space of the constraint manifold is a constant vector; i.e. we have $\mathbf{P} = [1/\sqrt{3}, -1/\sqrt{3}, 1/\sqrt{3}]$. Thus the Jacobian used in the simplified Newton iteration, (24), is exact and convergence is well achieved with this exact sensitivity analysis.

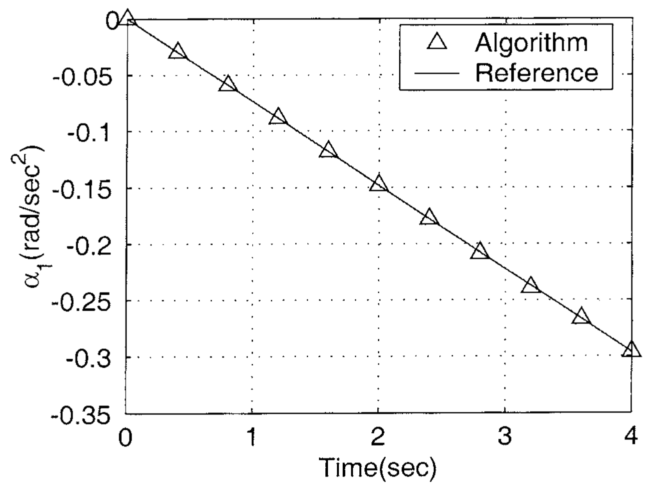

Figure 3. $\ddot{q}_1$.

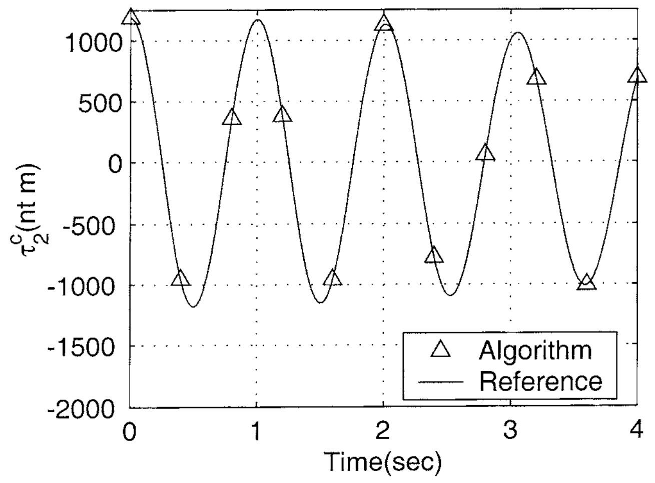

Figure 4. Constraint torque on body 2.

Second we emphasize that the constraint forces are not solved simultaneously with the dynamics of the constrained system. They are recovered after we get converged solutions of the dynamics. In this example the choice of the generalized coordinates as joint coordinates results in the system dynamics (Figures 2 and 3), having longer time scales than the constraint force (Figure 4). Thus, we can use a larger time step to solve the dynamics. The slower system dynamics are successfully insulated from the higher frequency content of the algebraic variables.

If we use Cartesian coordinates to describe the same problem, neither the tangent space is constant nor are the dynamics slower. To capture the dynamics using Cartesian coordinates, a smaller time step size is anticipated. Figures 6 and 7 illus-

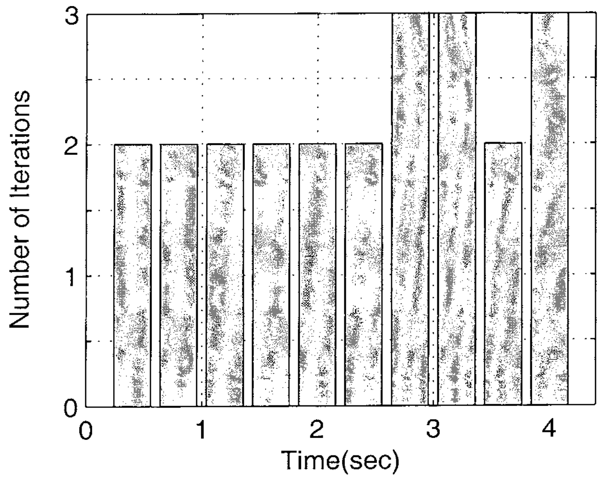

Figure 5. Iteration history.

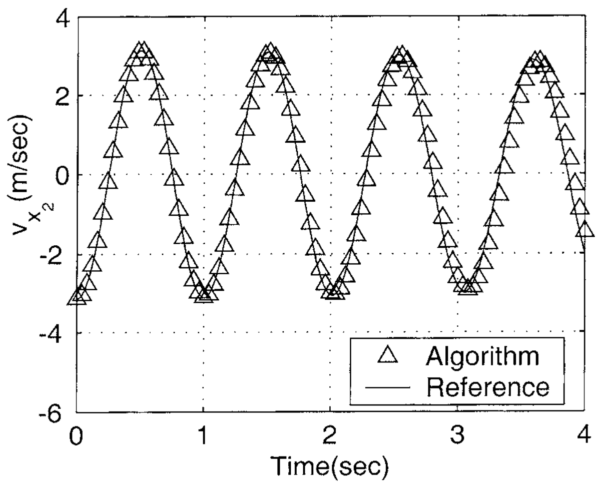

Figure 6. $\dot{x}$ of Body 2.

trate solutions of the x coordinate of the center of mass of body 2, using a time step size one-tenth the time step size of the previous results. The solution drifts a little even with the time step size of 0.04 sec.

4. Conclusions

Despite advances in computer hardware and software, there does not exist a practical, network-distributed dynamics simulation environment. To tackle this problem, instead of the traditional divide-and-conquer paradigm, an integrate-and-collaborate paradigm is exploited in this paper. This new approach naturally

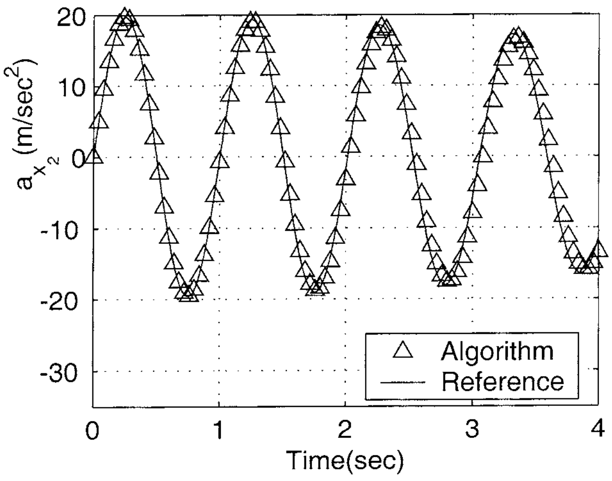

Figure 7. $\ddot{x}$ of Body 2.

encapsulates individual modules and makes them well suited for deployment over the modern network-distributed computing environments.

As the first step towards building a network-distributed dynamics simulation environment, an unobtrusive algorithm is presented after a brief discussion of the classification of the DAE algorithms. The proposed algorithm separates the solution of the dynamics from the constraint equations. The solution of the dynamics can be readily mapped into individual modules and the coordination module to satisfy the constraint Equations fits the integrate-and-collaborate role perfectly. A planar parallel four-bar problem is used to illustrate the effectiveness of the algorithm.

A framework that implements the gluing algorithm over the network is under development. The ultimate goal is to couple legacy codes for integrated system simulation. The feasibility of wrapper interfaces for such legacy packages needs to be evaluated on a case-by-case basis. Also the impact of the subdomain solution strategy on the system solution requires additional, careful studies.

Acknowledgement

The authors gratefully acknowledge support of this research by the Automotive Research Center at the University of Michigan, an U. S. Army Center of Excellence in Modeling and Simulation of Ground Vehicles, under Contract DAAE07-94-C-R094.

References

1. Mousseau, C.W., Laursen, T.A., Lidberg, M. and Taylor, R.L., 'Vehicle dynamics simulations with coupled multibody and finite element models', *Finite Elements in Analysis and Design* **31**(4), 295–315 (1999).
2. Hulbert, G.M., Michelena, N., Ma, Z.-D., Tseng, F.-C., Fellini, R., Scheffer, C., Choi, K.K., Tang, J., Ogarevic, V. and Hardee, E., 'A case study for network-distributed collaborative design and simulation: Extended life optimization for M1 Abrams tank road arm', *Mechanics of Structures and Machines* **27**(4), 1999, 423–451.
3. Enslow, Jr., P.H., 'What is a distributed data processing system?', *IEEE Computer* **11**(1), 1978, 13–21.
4. Singhal, M. and Casavant, T.L., 'Distributed computing systems', *IEEE Computer* **24**(8), 1991, 12–15.
5. OMG, Object Management Group, *The Common Object Request Broker: Architecture and Specification, Revision 2.0*, Framingham, MA, 1995.
6. IONA Technologies, *ORBIX 2, Distributed Object Technology: Programming Guide, Version 2.1*, 1996.
7. Orlandea, N., Chace, M.A. and Calahan, D.A., 'A sparsity-oriented approach to the dynamic analysis and design of mechanical systems – Part I', *Journal of Engineering for Industry* **99**(3), 1977, 773–779.
8. Orlandea, N., Calahan, D.A. and Chace, M.A., 'A sparsity-oriented approach to the dynamic analysis and design of mechanical systems – Part II', *Journal of Engineering for Industry* **99**(3), 1977, 780–784.
9. Cardona, A. and Géradin, M., 'Time integration of the equations of motion in mechanism analysis', *Computers and Structures* **33**(3), 1989, 801–820.
10. Farhat, C., Crivelli, L. and Géradin, M., 'Implicit time integration of a class of constrained hybrid formulations – Part I: Spectral stability theory', *Computer Methods in Applied Mechanics and Engineering* **125**, 1995, 71–107.
11. Chung, J. and Hulbert, G.M., 'A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method', *Journal of Applied Mechanics* **60**, 1993, 371–375.
12. Brenan, K.E., Campbell, S.L. and Petzold, L.R., *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier, New York, 1989.
13. Petzold, L.R., 'A description of DASSL: A differential/algebraic system solver', in *Transactions of the Tenth IMACS World Congress on Systems Simulation and Scientific Computation*, International Association for Mathematics & Computers in Simulation, 1982, Vol. 1, 430–432.
14. Kurdila, A., Papastavridis, J.G. and Kamat, M.P., 'Role of Maggi's equations in computational methods for constrained multibody systems', *Journal of Guidance, Control and Dynamics* **13**(1), 1990, 113–120.
15. Walton, Jr., W.C. and Steeves, E.C., 'A new matrix theorem and its application for establishing independent coordinates for complex dynamical systems with constraints', NASA Technical Report, NASA TR R-326, 1969, 1–27.
16. Wehage, R.A. and Haug, E.J., 'Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems', *Journal of Mechanical Design* **104**, 1982, 247–255.
17. Kim, S.S. and Vanderploeg, M.J., 'QR decomposition for state space representation of constrained mechanical dynamic systems', *Journal of Mechanisms, Transmissions, and Automation in Design* **108**, 1986, 183–188.
18. Singh, R.P. and Likins, P.W., 'Singular value decomposition for constrained dynamical systems', *Journal of Applied Mechanics* **52**, 1985, 943–948.
19. Kane, T.R. and Levinson, D.A., *Dynamics: Theory and Applications*, McGraw-Hill, New York, 1985.

20. Yen, J., 'Constrained equations of motion in multibody dynamics as ODEs on manifolds', *SIAM Journal on Numerical Analysis* **30**(2), 1993, 553–568.
21. Yen, J., Haug, E.J. and Tak, T.O., 'Numerical methods for constrained equations of motion in mechanical system dynamics', *Mechanics of Structures and Machines* **19**(1), 1991, 41–76.
22. Papastavridis, J.G., 'Maggi's equations of motion and the determination of constraint reactions', *Journal of Guidance, Control and Dynamics* **13**(2), 1990, 213–220.
23. Lelarasmee, E., Ruehli, A.E. and Sangiovanni-Vincentelli, A.L., 'The waveform relaxation method for time-domain analysis of large scale integrated circuits', *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **CAD-13**, 1982, 131–145.
24. Leimkuhler, B., 'Relaxation techniques in multibody dynamics', *Transactions of the CSME* **17**(4A), 1993, 459–471.
25. Baumgarte, J., 'Stabilization of constraints and integrals of motion in dynamical systems', *Computer Methods in Applied Mechanics and Engineering* **1**, 1972, 1–16.
26. Eich, E. and Hanke, M., 'Regularization methods for constrained mechanical multibody systems', *Zeitschrift für Angewandte Mathematik und Mechanik* **75**(10), 1995, 761–773.
27. Lin, S.-T. and Hong, M.-C., 'Stabilization method for numerical integration of multibody mechanical systems', *Journal of Mechanical Design* **120**, 1998, 565–572.
28. Gear, C.W., Leimkuhler, B.J. and Gupta, G.K., 'Automatic integration of Euler–Lagrange equations with constraints', *Journal of Computational and Applied Mathematics* **12**, 1985, 77–90.
29. Park, K.C. and Chiou, J.C., 'Stabilization of computational procedures for constrained dynamical systems', *Journal of Guidance, Control and Dynamics* **11**(4), 1988, 365–370.
30. Park, K.C., Chiou, J.C. and Downer, J.D., 'Explicit-implicit staggered procedure for multibody dynamics analysis', *Journal of Guidance, Control and Dynamics* **13**(3), 1990, 562–570.
31. Chiou, J.C., Park, K.C. and Farhat, C., 'A natural partitioning scheme for parallel simulation of multibody systems', *International Journal for Numerical Methods in Engineering* **36**, 1993, 945–967.
32. Bayo, E., Garcia De Jalon, J. and Serna, M.A., 'A modified Lagrangian formulation for the dynamic analysis of constrained mechanical systems', *Computer Methods in Applied Mechanics and Engineering* **71**, 1988, 183–195.
33. Bayo, E. and Ledesma, R., 'Augmented Lagrangian and mass-orthogonal projection methods for constrained multibody dynamics', *Nonlinear Dynamics* **9**, 1996, 113–130.
34. Shampine, L.F., 'Conservation laws and the numerical solution of ODEs', *Computers and Mathematics with Applications* **12B**(5/6), 1986, 1287–1296.
35. Eich, E., 'Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints', *SIAM Journal on Numerical Analysis* **30**(5), 1993, 1467–1482.
36. Yen, J., Petzold, L. and Raha, S., 'A time integration algorithm for flexible mechanism dynamics: The DAE α-method', *Computer Methods in Applied Mechanics and Engineering* **158**, 1998, 341–355.
37. J. Yen, J. and Petzold, L.R., 'An efficient Newton-type iteration for the numerical solution of highly oscillatory constrained multibody dynamic systems', *SIAM Journal on Scientific Computing* **19**(5), 1998, 1513–1534.
38. Pars, L.A., *A Treatise on Analytical Dynamics*, Heinemann, London, 1965.
39. Udwadia, F.E. and Kalaba, R.E., 'A new perspective on constrained motion', *Proceedings of the Royal Society of London Series A – Mathematical, Physical and Engineering Sciences* **439**, 1992, 407–410.