Nonlinear Dynamics **34:** 159–188, 2003.

A Gluing Algorithm for Distributed Simulation of Multibody Systems

JINZHONG WANG, ZHENG-DONG MA and GREGORY M. HULBERT
Department of Mechanical Engineering, The University of Michigan, 2250 G.G. Brown, 2350 Hayward, Ann Arbor, MI 48109-2125, U.S.A.; E-mail: mazd@umich.edu

(Received: 24 January 2003; accepted: 24 April 2003)

Abstract. A new gluing algorithm is presented that can be used to couple distributed subsystem models for dynamics simulation of mechanical systems. Using this gluing algorithm, subsystem models can be analyzed at their distributed locations, using their own independent solvers, and on their own platforms. The gluing algorithm developed relies only on information available at the subsystem interfaces. This not only enables efficient integration of subsystem models, but also engenders model security by limiting model access only to the exposed interface information. These features make the algorithm suitable for a real and practical distributed simulation environment.

Keywords: Distributed simulation, gluing algorithm, multibody system simulation, structural dynamics.

Nomenclature

m^i, $\mathbf{M}^i$	=	mass, mass matrix
$\mathbf{C}^i$	=	damping matrix
$\mathbf{K}^i$, $\mathbf{K}^*$	=	stiffness matrix, effective stiffness matrix
$\mathbf{k}^i_{ab}$	=	components of $\mathbf{K}^i$
$\mathbf{G}^i$	=	interface matrix
$\mathbf{g}^i_{AB}$	=	components of $\mathbf{G}^i$
t_n	=	time step
$\mathbf{Q}^i$	=	generalized force vector
$\mathbf{q}^i$	=	generalized coordinate vector
$\mathbf{X}$, $\mathbf{D}$	=	interface kinematic information vector
$\mathbf{u}$, u_m, u, v	=	displacement
$\mathbf{v}^i$	=	velocity vector
$\mathbf{a}^i$	=	acceleration vector
$\mathbf{\Phi}^i$	=	constraints in multibody dynamic system
$\mathbf{e}$, e_x, e_y	=	error measures that represent the violation of the compatibility conditions at the interfaces
$\mathbf{\Lambda}$	=	gluing (Lambda) matrix
$\mathbf{T}$, $\mathbf{F}$	=	interface force information vector
$\mathbf{f}^i_a$, $\mathbf{f}^i$, f_n	=	external or interface force
$\mathbf{\Phi}^i_q$	=	derivate of $\mathbf{\Phi}$ with respective to $\mathbf{q}$
λ^i	=	Lagrange multiplier in EOM of multibody dynamic system
$\mathbf{\Gamma}_i$	=	interface
$\mathbf{S}$	=	body shape function
$\mathbf{B}_i$, $\mathbf{C}_i$	=	assembly matrix

1. Introduction

Computer simulation has been used extensively for virtual prototyping of engineering designs as a substitute for actual physical prototyping. This simulation use has reduced the cost involved in product development, and has accelerated the design process to produce more reliable and more functional products. Rapid advances in software and hardware have accelerated the reliance on virtual prototyping. Modern simulation tools, e.g., commercial finite element codes and multibody dynamics codes, are now able to model complex mechanical systems with a large number of components, such as a complete automotive vehicle.

Independent of simulation, automobile manufacturers are operating in a way that distributes the production processes with multi-layered supply chains [1]. These supply chains, which are naturally, functionally, and geographically distributed, result in very complicated design and manufacturing systems. Even within a supplier unit, it is common practice for different groups to work on different components of the product. It is very difficult, if not impossible, to couple all detailed component models into a monolithic, stand-alone simulation model. Hence, there is a great need for methodologies that can be exploited to simulate complex mechanical systems whose models are distributed amongst disparate production units. Such methodologies need to maintain simulation fidelity, must be efficient and must maintain the 'privacy' of the individual component models amongst potentially competing supply chain units.

This paper addresses the need for a simulation environment that can incorporate distributed heterogeneous mechanical systems models and couple them together to perform dynamics simulations to assist the virtual prototyping process. Two significant challenges must be addressed. First, the distributed subsystems models might be developed independently – they might use different software packages, run on different computers, and/or reside at different geographical locations. Second, the model developers, to protect proprietary information, may not be willing to share their models directly. This suggests that only minimal information should be exchanged during the coupled simulation, and the local developer should be able to control the accessibility of the model developed. In short, the distributed simulation platform should:

1. Couple different models and software codes in a plug-and-play manner;
2. Communicate across distributed computing resources;
3. Maintain the integrity (independence) of the separated component models.

These requirements call for a new integration algorithm that can efficiently and sufficiently integrate subsystems models in a distributed environment and does not require internal details of the component models; this is the focus of the present paper.

There are two different perspectives on the decomposition and coupling of complex systems, as laid out by Tseng [2], namely, 'divide-and-conquer' and 'integrate-and-collaborate', or, in other words, decomposition and gluing. The former focuses on how to actively partition a large problem in order to take advantage of parallel computing. In contrast, the gluing perspective starts from the fact that many systems are already partitioned and distributed. In other words, gluing does not involve active decomposition.

Many researchers have studied the decomposition of large mechanical systems with a primary focus on the decomposition strategy. Arising from the field of parallel computing applied to finite element analysis, many have explored extending the concept of parallel algo-

rithms to distributed simulation. A primitive version of a distributed finite element simulation is presented in [3], in which stiffness matrices and load vectors are generated concurrently on clients and sent to a central server to be assembled. Other researchers adopted a different approach, using parallel solution of the system of equations. The parallelization of both direct solution methods [4] and iterative methods [5] have been studied. A substructuring method called FETI was presented in [6], introducing extra traction variables and exhibiting more flexibility for model reduction and coupling, compared to competing schemes. (See [6] for a good literature review on implicit parallel computing.) As discussed in [2], these efforts all focus on active partitioning of an existing large-scale system rather than coupling an already distributed system, employing, e.g., substructuring [7] or domain decomposition methods [8].

In the multibody dynamics arena, researchers also have studied how to partition and parallelize systems [2, 9–20]. One strategy adopted by researchers is similar to substructuring in FEM, i.e., a small global problem is formed by incorporating condensed subsystem models and then is solved to provide necessary information to subsystem models. The subsystem models subsequently can be solved based on this information. In [9], a subsystem synthesis method was proposed for dynamic analysis of vehicle multibody systems, in which each subsystem is independently analyzed with a virtual reference body and the overall vehicle system analysis is formed by synthesizing the effective inertia matrix and force vector from the virtual reference body of each subsystem. A divide-and-conquer algorithm was presented in [10, 11] for rigid body dynamics, which reduces the system to an 'articulated-body' by recursively applying a formula that constructs the articulated-body equations of motion of the system from those of its constituent parts. Both the inputs and outputs of the formula are equations of motion. Another approach was given in [12, 13], in which the equations of the subsystem models are evaluated in parallel, and the results are loaded into a single system wide equation to explicitly calculate the constraint forces. The strategy adopted in [14] follows a similar path.

Treating the subsystem models as control blocks and taking advantage of many sophisticated control-based simulation software packages is another common modeling approach. In [15], a modular formulation for multibody systems is proposed, based on the block representation of a multibody system with corresponding input and output quantities. This 'block diagram' representation of the system can then be embedded into appropriate simulation packages, e.g., SIMULINK. In [16], 'Co-simulation', is presented, which employs a new discrete time sliding mode controller (DTSM) to satisfy the algebraic constraints among the subsystem models and to solve the causal conflicts associated with the algebraic constraints.

The methods reviewed above either involve the active decomposition of the full system and require more information than just that associated with the subsystem interfaces or mandate specific requirements or structure on the formulation of the subsystems. In the context of coupling already distributed subsystems, the gluing perspective is preferred. A study is presented in [2], in which the terminology 'gluing algorithm' is first suggested to describe a class of algorithms that can be used to glue distributed component models for use in dynamics simulations. Several gluing algorithms are studied in [2, 17, 18], including MEPI (Maggi's Equations with Perturbed Iteration) and MOP (Manifold Orthogonal Projection Method). The word 'glue' will be used hereafter instead of 'couple' in places where the gluing perspective is implied.

We have been developing a concept platform for simulating general distributed mechanical systems. Here, the mechanical system may have a large number of components represented by either finite element models and/or multibody dynamics models. The goal is to fill the

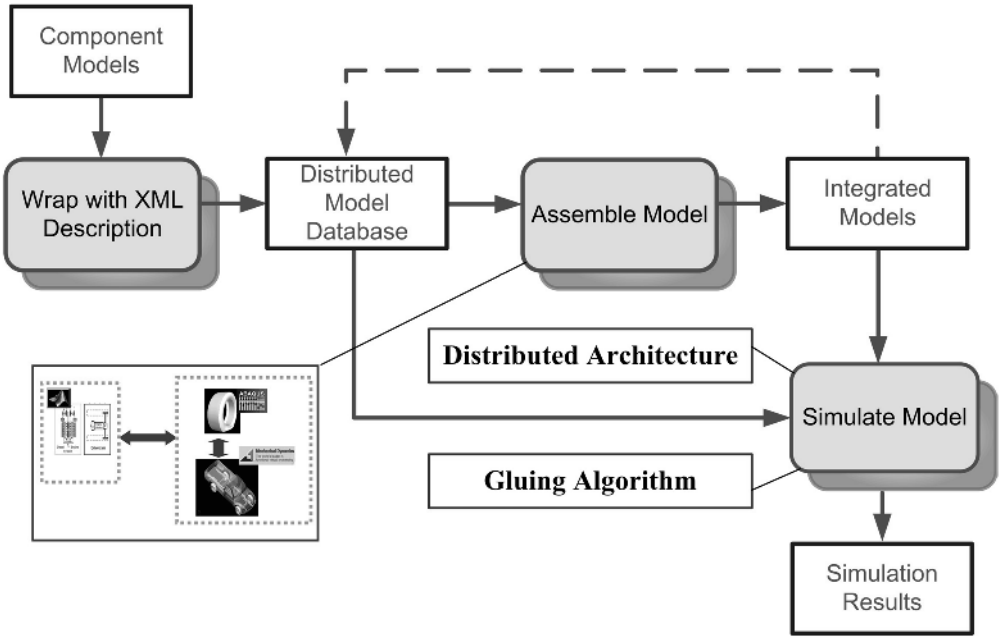


Figure 1. Outline of overall solution.

gap between state-of-the-art simulation techniques and practical product development systems described previously. Figure 1 shows an outline of the proposed overall solution, comprising three major elements. First, component models (including integrated models) are described in XML and stored in a model database. Second, integrated models are instantiated by assembling proper component models. Third, the mechanical system simulation is performed by executing the separate integrated model simulations and by gluing these separate simulations together using appropriate gluing algorithms. To achieve this requires:

1. a standardized description of models implemented in XML;
2. a distribution architecture that can be realized using available technologies;
3. a gluing algorithm that can couple component models without requiring modification to the model details.

In the following sections, we present a new gluing algorithm, denoted as the T–T method, which can be used to glue static, dynamic, and multibody dynamic component models of a mechanical system. Each component model is treated as a black box, and only minimal data at the interfaces are required. That is, only interface information is transmitted across the network during the mechanical system simulation.

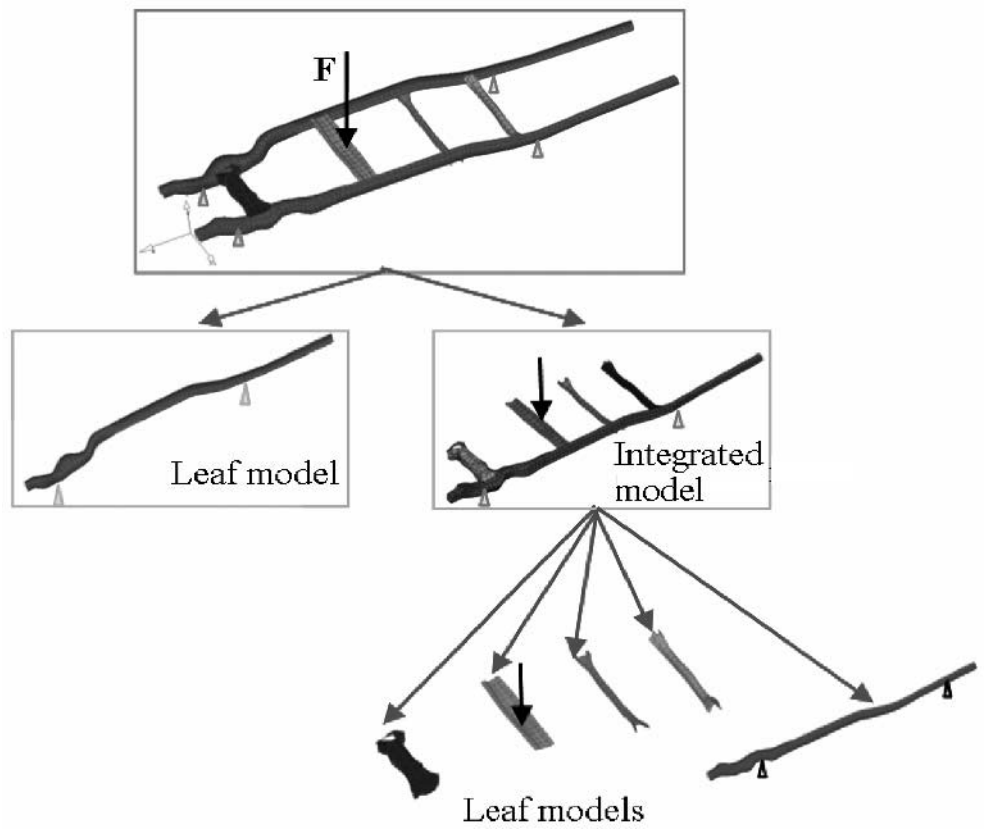


Figure 2. Distributed simulation of a truck chassis frame.

2. Basic Concept

2.1. DEFINITIONS

Standard simulation practice involves the use of a dataset, which includes, e.g., structural geometry, material data, loading conditions; this dataset describes the simulation model being used to represent a specific mechanical component or group of components along with the simulation scenario in which to exercise the model. To obtain simulation results, the dataset must be input to a specific simulation code. Using the nomenclature of data trees, this combination of dataset and simulation code is defined as a *leaf model*. Thus, in the usual design and simulation environment, the analyst works with leaf models. Instead of using the term parent model, we use integrated model to refer to models that are assembled from leaf models and/or lower-level integrated models, i.e., children models. As such, integrated models include the information required to couple/assemble its children models. Within the proposed framework, an integrated model also contains a strategy for gluing together its children models.

An example of an integrated model is the truck chassis frame, shown in Figure 2; the model is formed by gluing two subsystem (children) models: a leaf model of the right rail and an integrated model comprising a left rail model and models for the four cross- members. These models are coupled using the gluing algorithm described herein. In the coupling process,

the models exchange interface information and the interface information is updated based on the gluing algorithm.

2.2. GLUING STRATEGIES

The proposed gluing algorithm relies only on the information at the interfaces of the models that are to be coupled. Here, interface refers to the connections or common surfaces of two models. An interface can be represented by a set of interface nodes in a finite element model, or by a set of connecting joints in a multibody dynamics model. The typical information available at the interface can be classified as kinematic information and force information. The kinematic information contains displacements, velocities, and accelerations of the interface. Force information refers to action-reaction forces at the interface.

Mechanics principles require that at any interface the force quantities, namely action-reaction forces, satisfy the equilibrium equations, and the kinematic quantities satisfy the compatibility conditions, where it is assumed that the equilibrium[1] and compatibility conditions in the internal domain of each subsystem are satisfied *a priori*. The proposed gluing algorithm employs an iterative process, starting with an initial guess of some of the interface quantities. These interface quantities are then updated using a prescribed iteration process to satisfy the equilibrium and/or compatibility conditions at the interface.

In general, if a proper set of interface force variables is defined such that the equilibrium conditions are satisfied, then only the compatibility conditions need to be considered during the iteration process. In this case, the interface force variables can be considered as functions of the interface kinematic quantities, and these interface force variables can be updated using the kinematic information and compatibility conditions. Similarly, if a proper set of the interface kinematic variables is defined such that that the compatibility conditions are satisfied, then only the equilibrium conditions need to be considered during the iteration process. In this latter case, the interface kinematic variables are functions of the force quantities at the interface, and they can be updated by satisfying the equilibrium conditions. Different gluing algorithms ensue, depending on which group of interface quantities is considered as the defined input.

Figure 3 illustrates three typical coupling strategies, which are candidates for a gluing algorithm. Here, $\mathbf{X}$ represents the vector of kinematic quantities, and $\mathbf{T}$ the vector of force quantities at the interfaces of the two models to be glued together.

Figure 3a illustrates the T–T coupling strategy, for a two- subsystem case. In this strategy, kinematic quantities at the interfaces of both subsystem models, i.e., $\mathbf{X}_n^I$ and $\mathbf{X}_n^{II}$, are used as inputs to the coordinator. The interface force vectors of the two models, $\mathbf{T}_n^I$ and $\mathbf{T}_n^{II}$, are coordinator outputs, which will be applied to the subsystem models for next time step calculations. We defer discussion of the detailed coupling structure.

Figure 3b illustrates the X–X coupling strategy. In this method, interface force vectors $\mathbf{T}_n^I$ and $\mathbf{T}_n^{II}$ of the two models are used as inputs to the coordinator. The kinematic quantity vectors $\mathbf{X}_n^I$ and $\mathbf{X}_n^{II}$ are the coordinator outputs. Note that the MEPI algorithm developed in [2, 18] has the form of an X–X method. Figure 3c illustrates a mixed coupling strategy, the X–T method, in which the interface force vector $\mathbf{T}_n^I$ of Model 1 and kinematic quantity vector $\mathbf{X}_n^{II}$ of Model 2 are used as inputs to the coordinator, while the kinematic quantity vector $\mathbf{X}_n^I$ of Model 1 and the interface force vector $\mathbf{T}_n^{II}$ of Model 2 become outputs from the coordinator. This strategy was adopted in [19] to simulate the behavior of an army tank using a distributed

[1] Here the equilibrium will also consider the inertial forces and dynamic loads in the case of a dynamic problem.

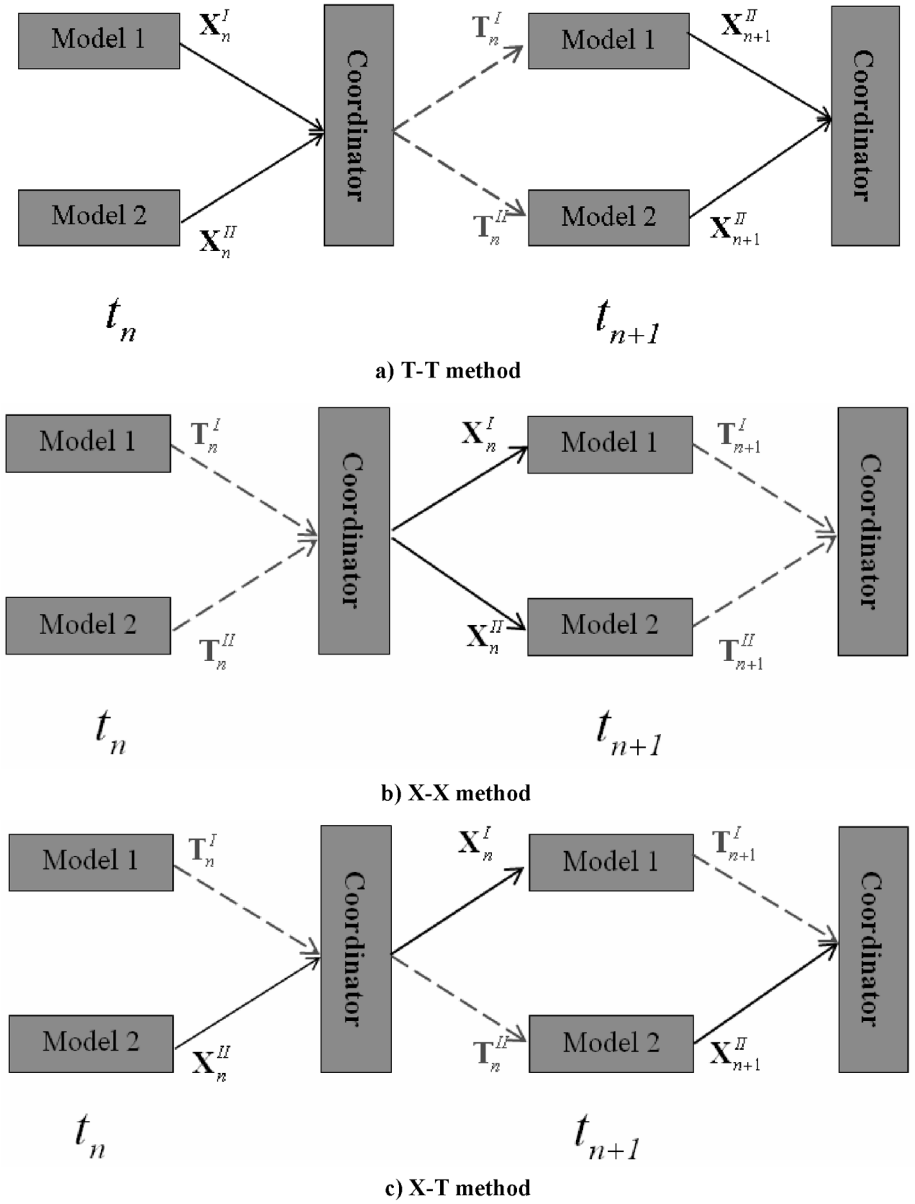


Figure 3. Coupling strategies (**T**: force quantity vector, **X**: kinematic quantity vector, t_n: time at the n-th step, t_{n+1}: time at the $(n+1)$-th step).

computing facility. This strategy is adopted by MATLAB/SIMULINK, and describes the so-called 'across and through' variables method adopted in 20-Sim and is employed in the control block strategy of [15].

Clearly the defined structure of the coordinator plays an important role in the problem and must be different for different coupling strategies. In the following, we will focus only on the T–T method. The major advantage of using a T–T strategy in a general problem of mechanical system simulation is that the forces can be easily applied to the subsystems when solving

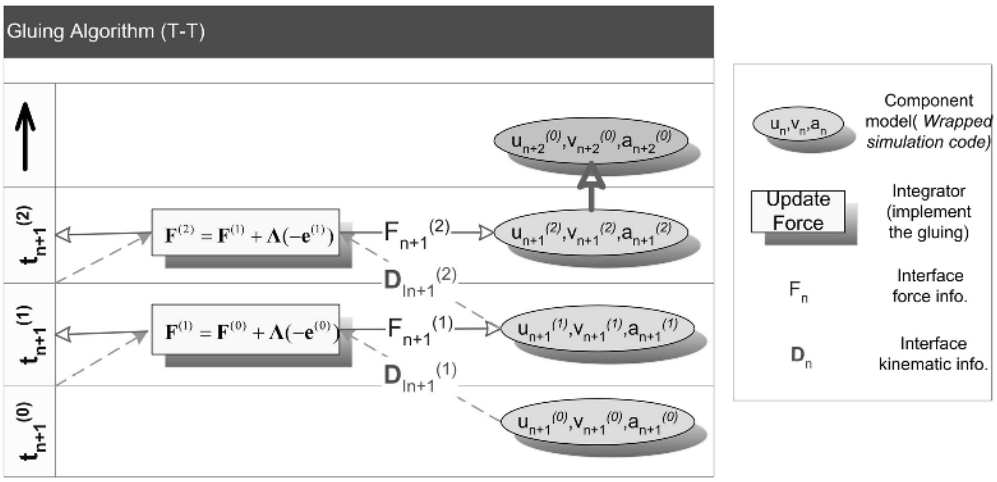


Figure 4. A stepping method with gluing algorithm (T–T method).

the subsystems equations, compared with the need to prescribe the kinematical conditions at the interfaces. This T–T strategy is more suitable to the standard setup of simulation codes that can be employed for the subsystems analyses, and thus it improves the efficiency of the integration process and enhances the independency of the subsystems models.

2.3. Iteration Method

Various iteration methods within the time integration loop can be considered depending on whether or not there is a leading subsystem and depending on how the time steps are arranged for information exchange between the different models. Figure 4 illustrates a typical iteration method we use currently with the T–T method. Here, **u**, **v**, and **a** represent the state variables of component model and **F**, **D** represent the interface forces and the kinematic information at the interface, respectively.

As shown in Figure 4, first the states of each component model are obtained from the previous time step. Then the interface kinematic information is extracted and sent to the coordinator. Next, the coordinator updates the interface forces and sends these forces back to the component models. Independently, the corrected states of each component model are solved for with respect to the updated interface forces. This procedure is repeated until convergence is attained. Finally, the integration proceeds to the next time step. Note that other iteration methods can also be considered, e.g., [19].

2.4. General Concept of the Gluing Algorithm (T–T method)

Assume that **F** is a properly defined interface force vector; that is, **F** contains the necessary and sufficient set of variables that can represent the force space at the interfaces considered and **F** is self-balanced, i.e., the equilibrium conditions at the interfaces are automatically satisfied if **F** is employed. Let **e** be an error measure vector that represents the violation of the compatibility conditions at the interfaces, where $\mathbf{e} = \mathbf{0}$ indicates that the compatibility conditions are fully satisfied. In the general case, **e** can be considered as a function of **F**, namely

$$\mathbf{e} = \mathbf{e}(\mathbf{F}). \tag{1}$$

Since $\mathbf{F}$ is defined in such a way that the equilibrium conditions can be automatically satisfied, the only objective of the gluing algorithm is to bring $\mathbf{e}$ to zero, namely to find a proper $\mathbf{F}$ that satisfies

$$\mathbf{e} \to \mathbf{0}. \tag{2}$$

Equation (2) defines a set of (linear or nonlinear) equations, which can be solved by a properly chosen algorithm of (linear or nonlinear) equation solvers.

Assuming an initial guess $\mathbf{F} = \mathbf{F}^{(i)}$ $(i = 0)$, we have $\mathbf{e}^{(i)} = \mathbf{e}(\mathbf{F}^i)$, then in the general case, a gluing algorithm (T–T method) can be proposed as

$$\mathbf{F}^{(i+1)} = \mathbf{F}^{(i)} + \mathbf{\Lambda}(-\mathbf{e}^{(i)}), \tag{3}$$

where $\mathbf{\Lambda}$ is called the *gluing matrix* or *lambda matrix*, which will be a constant matrix if Equation (2) is linear, or a function of $\mathbf{F}$ if Equation (2) is nonlinear. A gluing matrix can be obtained, for example, by using a standard Newton–Raphson method, and then we have

$$\mathbf{\Lambda} = \left(\frac{\partial \mathbf{e}}{\partial \mathbf{F}}\right)^{-1}_{\mathbf{F}=\mathbf{F}^{(i)}}. \tag{4}$$

Equation (3) simply implies that the interface forces can be updated (to satisfy the compatibility conditions) using only the kinematic information at the interface with an update rule such as that shown in Equation (3) provided the gluing matrix is obtained. Therefore, the key issue becomes how to obtain the lambda matrix in a systematic and efficient way. It should be mentioned that the adoption of a Newton–Raphson method here is to illustrate the general concept of the T–T method. In practice, other, possibly more efficient methods, such as the BFGS quasi-Newton method, may be employed [21].

2.5. Calculation of Gluing Matrix

For the sake of exposition, let us first consider an elastostatics problem of a distributed system. Using a finite element method, the equation of motion of each distributed subsystem can be written as

$$\mathbf{K}^i \mathbf{u}^i = \mathbf{f}^i, \quad (i = 1, 2, \ldots, n) \tag{5}$$

or

$$\begin{bmatrix} \mathbf{k}^i_{oo} & \mathbf{k}^i_{oc} \\ \mathbf{k}^i_{co} & \mathbf{k}^i_{cc} \end{bmatrix} \begin{Bmatrix} \mathbf{u}^i_o \\ \mathbf{u}^i_c \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}^i_o \\ \mathbf{f}^i_c \end{Bmatrix}, \quad (i = 1, 2, \ldots, n), \tag{6}$$

where $\mathbf{u}^i_o$ denotes displacements of the internal nodes of the ith subsystem, $\mathbf{f}^i_o$ denotes forces applied at the internal nodes, $\mathbf{u}^i_c$ denotes displacements of the interface nodes, $\mathbf{f}^i_c$ denotes action-reaction forces at the interface, and n is the total number of the subsystems to be integrated. The problem here is to couple the subsystems equations in Equation (5) so that the equilibrium and compatibility conditions at the subsystems interfaces can be satisfied while individual equations in Equation (5) are solved independently.

In the general case, the subsystem interface force vector $\mathbf{f}^i_c$ in Equation (6) can be represented by a subset of the variables in $\mathbf{F}$, and therefore $\mathbf{f}^i_c$ can be written as

$$\mathbf{f}^i_c = \mathbf{f}^i_c(\mathbf{F}), \quad (i = 1, 2, \ldots, n). \tag{7}$$

Now we define a matrix $\mathbf{C}_i$, for the ith subsystem as

$$\mathbf{C}_i = \frac{\partial \mathbf{f}_c^i}{\partial \mathbf{F}}, \quad (i = 1, 2, \ldots, n). \tag{8}$$

Typically, $\mathbf{C}_i$ will be a simple, constant matrix.

Assume $\mathbf{U}$ is an assembly of the interface displacements of all subsystems, namely,

$$\mathbf{U} = \{\mathbf{u}_c^i\}^T = \begin{Bmatrix} \mathbf{u}_c^1 \\ \vdots \\ \mathbf{u}_c^n \end{Bmatrix}. \tag{9}$$

In the general case (for the static problem considered here), the error measure $\mathbf{e}$ can be written as

$$\mathbf{e} = \mathbf{e}(\mathbf{U}), \tag{10}$$

and now we can define a matrix, $\mathbf{B}_i$, for the ith subsystem as

$$\mathbf{B}_i = \frac{\partial \mathbf{e}}{\partial \mathbf{u}_c^i}, \quad (i = 1, 2, \ldots, n). \tag{11}$$

Using Equations (8) and (11), Equation (4) can be rewritten as

$$\mathbf{\Lambda} = \left(\sum_{i=1}^{n} \bar{\mathbf{G}}^i \right), \tag{12}$$

where,

$$\bar{\mathbf{G}}^i = \mathbf{B}_i \mathbf{G}^i \mathbf{C}_i, \tag{13}$$

and

$$\mathbf{G}^i = \frac{\partial \mathbf{u}_c^i}{\partial \mathbf{f}_c^i}. \tag{14}$$

$\mathbf{G}^i$ is called the interface flexibility matrix of subsystem i, which can be calculated by solving Equation (6) independently for each subsystem, where $i = 1, 2, \ldots, n$. It is important to note that $\mathbf{\Lambda}$ is essentially the inverse matrix of an assembly of the subsystem interface flexibility matrices defined in Equation (14). Therefore, $\mathbf{C}_i$ and $\mathbf{B}_i$ are called *assembly matrices* of subsystem i, and we will discuss them further below.

To show more specific structure of $\bar{\mathbf{G}}^i$ in Equation (12) as well as of $\mathbf{C}_i, \mathbf{B}_i$ and $\mathbf{G}^i$ in Equation (13), we discuss some typical cases. First, if components in $\mathbf{F}$ can be directly selected from $\mathbf{f}_c^i$, $(i = 1, 2, \ldots, n)$, then we have

$$\mathbf{F} = \left\{ (\mathbf{f}_A^1)^T, (\mathbf{f}_A^2)^T, \ldots, (\mathbf{f}_A^n)^T \right\}^T, \tag{15}$$

where $\mathbf{f}_A^i$ is the subset of independent force variables selected from $\mathbf{f}_c^i$, and thus

$$\mathbf{f}_c^i = \begin{Bmatrix} \mathbf{f}_A^i \\ \mathbf{f}_R^i \end{Bmatrix}, \tag{16}$$

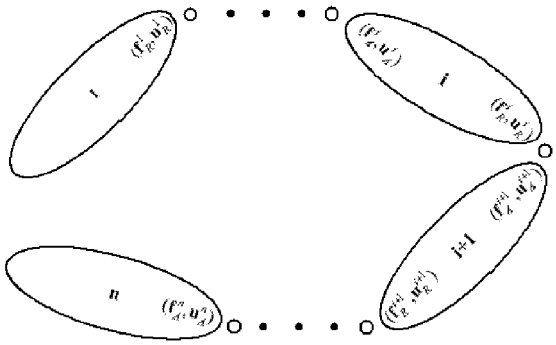


Figure 5. Gluing simple-connected components.

where $\mathbf{f}_R^i$ contains the remaining force components in $\mathbf{f}_c^i$. For convenience, we call $\mathbf{f}_A^i$ 'active' or 'action' forces and $\mathbf{f}_R^i$ 'passive' or 'reaction' forces. Passive forces are in general determined by the active forces of the adjacent subsystems. For example, for a system with the simple connections shown in Figure 5, we have

$$\mathbf{f}_R^i = -\mathbf{f}_A^{i+1}, \quad (i = 1, 2, \ldots, n-1). \tag{17}$$

In this case, $\mathbf{C}_i$ in Equation (8) becomes a $m \times n$ matrix with only terms of 1, -1 or 0. To be more specific, for the subsystem i $(1 < i < n)$, we have

$$\mathbf{C}_i = \begin{bmatrix} \mathbf{0} & \cdots & \overbrace{\mathbf{I}}^{i} & \overbrace{\mathbf{0}}^{i+1} & \cdots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & -\mathbf{I} & \cdots & \mathbf{0} \end{bmatrix}. \tag{18}$$

For the same problem, assume the error measure is

$$\mathbf{e} = \left\{ \begin{array}{c} \vdots \\ \mathbf{u}_A^i - \mathbf{u}_R^{i-1} \\ \mathbf{u}_A^{i+1} - \mathbf{u}_R^i \\ \vdots \end{array} \right\} \} i\text{-th subsystem} . \tag{19}$$

Then from Equation (11) we have

$$\mathbf{B}_i = \mathbf{C}_i^T = \left[\begin{array}{cc} \mathbf{0} & \mathbf{0} \\ \vdots & \vdots \\ \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} \\ \vdots & \vdots \\ \mathbf{0} & \mathbf{0} \end{array} \right] \} i\text{-th subsystem}, \quad (1 < i < n). \tag{20}$$

Assuming matrix $\mathbf{G}^i$ of each component in Figure 5 can be expressed as

$$\mathbf{G}^i = \begin{bmatrix} \mathbf{g}_{AA}^i & \mathbf{g}_{AR}^i \\ \mathbf{g}_{RA}^i & \mathbf{g}_{RR}^i \end{bmatrix}, \quad (i < i < n), \tag{21}$$

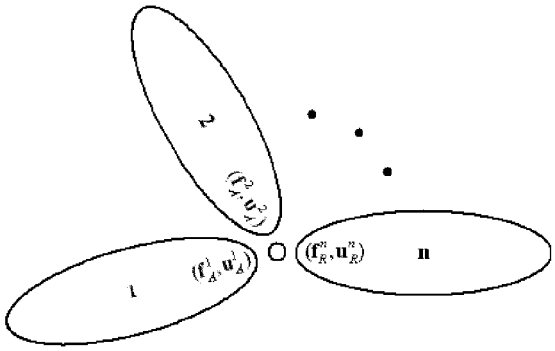


Figure 6. Gluing multiple-connected components.

where subscripts indicate the relation of each sub-matrix in $\mathbf{G}^i$ with respect to the interface sets A and R, using Equation (12), we have

$$\bar{\mathbf{G}}^i = \begin{bmatrix} \ddots & & & \\ & \mathbf{g}^i_{AA} & -\mathbf{g}^i_{AR} & \\ & -\mathbf{g}^i_{RA} & \mathbf{g}^i_{RR} & \\ & & & \ddots \end{bmatrix}. \tag{22}$$

Another typical case is when the system has multiple components being connected at the same interface. For example, for the system shown in Figure 6, an error measure can be taken as

$$\mathbf{e} = \begin{Bmatrix} \mathbf{u}^1_A - \mathbf{u}^n_R \\ \mathbf{u}^2_A - \mathbf{u}^n_R \\ \vdots \\ \mathbf{u}^{n-1}_A - \mathbf{u}^n_R \end{Bmatrix}, \tag{23}$$

and we select $\mathbf{F}$ as

$$\mathbf{F} = \{(\mathbf{f}^1_A)^T, (\mathbf{f}^2_A)^T, \ldots, (\mathbf{f}^{n-1}_A)^T\}. \tag{24}$$

Then from Equation (8) we have

$$\mathbf{C}_i = \begin{cases} \begin{bmatrix} \cdots \underbrace{\mathbf{I}}_{i-\text{th}} \cdots \end{bmatrix} & (\text{for } i = 1, 2, \ldots, n-1), \\ [-\mathbf{I} - \mathbf{I} \cdots - \mathbf{I}] & (\text{for } i = n). \end{cases} \tag{25}$$

From Equation (11), we have

$$\mathbf{B}_i = \mathbf{C}_i^T, \quad (i = 1, 2, \ldots, n). \tag{26}$$

Solving Equation (5) or (6), we obtain

$$\mathbf{G}^i = [\mathbf{g}^i_{AA}]. \tag{27}$$

Finally we obtain $\bar{\mathbf{G}}^i$ as

$$\bar{\mathbf{G}}^i = \begin{cases} \begin{bmatrix} \ddots & & \\ & \mathbf{g}^i_{AA} & \\ & & \ddots \end{bmatrix} & (\text{for } i = 1, 2, \ldots n-1), \\ \begin{bmatrix} \mathbf{g}^n_{AA} & \cdots & \mathbf{g}^n_{AA} \\ \vdots & \ddots & \vdots \\ \mathbf{g}^n_{AA} & \cdots & \mathbf{g}^n_{AA} \end{bmatrix} & (\text{for } i = n). \end{cases} \tag{28}$$

Note that subsystem assembly matrices $\mathbf{C}_i$ and $\mathbf{B}_i$ defined in Equations (8) and (11) are general and they can have more complex form rather than shown in Equations (18), (20), (25) and (26) depending on how the independent force vector $\mathbf{F}$ is selected. This provides a means to deal with different kinds of connections for the subsystems, for example, connections associated with inconsistent finite element meshes at the interface of the adjacent subsystems, and various kinds of mechanical joints.

Note that $\mathbf{G}^i$ in Equation (14) can be approximated as

$$\mathbf{G}^i \cong \left[\frac{\Delta u_m}{\Delta f_n}\right]_{\Delta f_n \to 0}, \tag{29}$$

where u_m is the m-th component of $\mathbf{u}^i_c$ and f_n is the n-th component of $\mathbf{f}^i_c$, Δu_m is the change of u_m with respect to an incremental interface force Δf_n. For a linear system, Equation (29) is exact for any Δf_n. In other words, for a linear system, $\mathbf{G}^i$ is independent of the external force, and $\Delta u_m / \Delta f_n$ is independent of the amplitude of the Δf_n used; therefore we can assume

$$\Delta f_n = 1 \tag{30}$$

and

$$\Delta u_m = u^1_m - u^0_m, \tag{31}$$

where u^0_m is calculated by solving Equation (6) without applying any interface force, and u^1_m is calculated by applying a unit interface force at the n-th interface degree of freedom. Note that for each Δf_n we obtain a vector $\mathbf{\Delta u}_n = \{\Delta u_m\}^T$.

In summary, the procedure to calculate $\mathbf{G}^i$ is as follows:

ALGORITHM 1 (Calculating interface matrix).

1. *Calculate initial* $\mathbf{u}^0$ *without applying any interface force,*
2. *Apply a unit force to a degree of freedom, n, in the interface,*
3. *Obtain* $\Delta \mathbf{u}_n$ *by solving the subsystem's equation Equation (6) (using its own independent solver),*
4. *Repeat steps 2 and 3 for all interface degrees of freedom to obtain* $\mathbf{G}^i = [\Delta \mathbf{u}_1, \Delta \mathbf{u}_2, \ldots, \Delta \mathbf{u}_N]$*, where N is the total number of interface degrees of freedom.*

Usually, the number of interface degrees of freedom is much smaller than the number of total degrees of freedom of the subsystem model. Therefore, $\mathbf{\Lambda}$ can be easily calculated when $\mathbf{G}^i$

($i = 1, 2, \ldots, n$) are obtained. The proposed approach treats each subsystem as a black box without accessing its internal information. Subsystem interface matrices, $\mathbf{G}^i$, can be calculated by calling the independent solvers associated with the subsystem models, and the subsystems can be glued together using only the interface information. Note that for a linear system, there is no need for iteration when Equation (3) is used. Thus, the gluing process converges in one iteration.

The gluing algorithm described above can be extended for solving dynamics or multibody dynamics problems. For a dynamics or multibody dynamics problem, the error measure can include accelerations, velocities and displacements at the interfaces or a combination of them. A similar process can be conducted to calculate the gluing matrix and thus couple the subsystem models.

3. Structural Dynamics Problem

For linear system dynamics, the initial-value problem of each subsystem can be written as

$$\begin{cases} \mathbf{M}^i\mathbf{a}^i + \mathbf{C}^i\mathbf{v}^i + \mathbf{K}^i\mathbf{u}^i = \mathbf{f}^i, \\ \mathbf{u}^i(0) = \mathbf{u}_0^i, \\ \mathbf{v}^i(0) = \mathbf{v}_0^i, \end{cases} \tag{32}$$

in which $\mathbf{M}^i$, $\mathbf{C}^i$ and $\mathbf{K}^i$ are the respective mass, damping, and stiffness matrices, $\mathbf{f}^i$ denotes the external forces, and $\mathbf{u}^i$, $\mathbf{v}^i$ and $\mathbf{a}^i$ are the nodal displacement, velocity and acceleration vectors, respectively. A number of different methods can be employed to solve Equation (32). For example, employing the Newmark algorithm,

$$\begin{cases} \mathbf{M}^i\mathbf{a}_{n+1}^i + \mathbf{C}^i\mathbf{v}_{n+1}^i + \mathbf{K}^i\mathbf{u}_{n+1}^i = \mathbf{f}_{n+1}^i, \\ \mathbf{u}_{n+1}^i = \mathbf{u}_n^i + \Delta t_n\mathbf{v}_n^i + \frac{\Delta t_n^2}{2}[(1-2\beta)\mathbf{a}_n^i + 2\beta\mathbf{a}_{n+1}^i], \\ \mathbf{v}_{n+1}^i = \mathbf{v}_n^i + \Delta t_n[(1-\gamma)\mathbf{a}_n^i + \gamma\mathbf{a}_{n+1}^i], \end{cases} \tag{33}$$

or

$$\mathbf{K}_{n+1}^*\mathbf{a}_{n+1} = \mathbf{f}_{n+1}^*, \tag{34}$$

where

$$\begin{aligned} \mathbf{K}_{n+1}^* &= \mathbf{M}^i + \gamma\Delta\, t_n\mathbf{C}^i + \beta\Delta\, t_n^2\mathbf{K}^i, \\ \mathbf{f}_{n+1}^* &= \mathbf{f}_{n+1}^i - \mathbf{C}^i\tilde{\mathbf{v}}_{n+1}^i - \mathbf{K}^i\tilde{\mathbf{u}}_{n+1}^i, \end{aligned} \tag{35}$$

and

$$\begin{aligned} \tilde{\mathbf{u}}_{n+1}^i &= \mathbf{u}_n^i + \Delta\, t_n\mathbf{v}_n^i + \frac{\Delta t_n^2}{2}(1-2\beta)\mathbf{a}_n^i, \\ \tilde{\mathbf{v}}_{n+1}^i &= \mathbf{v}_n^i + \Delta\, t_n(1-\gamma)\mathbf{a}_n^i. \end{aligned} \tag{36}$$

Note that Equation (34) is similar to Equation (5). Here, in Equation (34), $\mathbf{a}_{n+1}$ is the acceleration vector at the time step $t = t_{n+1}$, $\mathbf{K}_n^*$ and $\mathbf{f}_n^*$ are effective mass matrix and effective force vectors, respectively, Δt_n is the integration time interval at the n-th step, and β and γ are the standard parameters in Newmark algorithm. If the compatibility condition (Equation (2)) is

imposed on the interface accelerations, which is a subset of $\mathbf{a}^i_{n+1}$, and if the force variables are selected from the corresponding interface components of $\mathbf{f}^i_{n+1}$, then the approaches discussed for the statics case apply here as well, provided the velocity and displacement vectors are updated appropriately. In particular, the subsystem interface matrices can be calculated as in Equation (14), and the lambda matrix can be calculated as in Equation (12). The difference between the dynamics problem and the statics problem is that Equation (34) has to be solved at each time step for the dynamics problem. Note that for a linear system, if the time interval Δt_n is constant, i.e., $\Delta t_n = \Delta t\,(n = 1, 2, \ldots)$, then the lambda matrix needs to be calculated only once . Clearly, this provides a cost saving during the solution process.

The gluing algorithm for the dynamics problem is summarized as:

ALGORITHM 2 (Algorithm 2: Gluing algorithm for a dynamic problem).

1. *Calculate the interface matrices using Algorithm 1 (for the equivalent algebraic problem in Equation (34)),*
2. *Assemble the lambda matrix at the starting time,*
3. *In each time step, solve Equation (34) and update the interface forces using Equation (3),*
4. *Repeat step 3 for all time steps.*

Note that for a linear system, there is no need for iteration at each time step when the update equation Equation (3) is used. In fact, at each time step, Equation (3) needs to be employed only once. In Example 1, we will illustrate how *Algorithm 2* can be applied to a real engineering problem, namely, a distributed simulation problem of the truck chassis frame shown in Figure 2.

4. Multibody Dynamics Problem

The equation of motion of a multibody dynamics model for a subsystem can be written, in general, as an index-3 problem

$$\begin{cases} \mathbf{M}^i\mathbf{a}^i + \mathbf{\Phi}^i_q\,\boldsymbol{\lambda}^i = \mathbf{Q}^i, \\ \mathbf{\Phi}^i = \mathbf{0}, \end{cases} \tag{37}$$

where $\mathbf{M}^i$ denotes the inertia matrix, $\mathbf{a}^i$ denotes the acceleration vector, $\mathbf{Q}^i$ denotes the generalized force vector, $\boldsymbol{\lambda}^i$ is the vector of Lagrange multipliers, $\mathbf{\Phi}^i$ represents the internal constraints, which are assumed to be holonomic. Here we will take the double pendulum as an example to demonstrate the applying of the T–T method to a multibody dynamics problem. In Figure 7, the revolute joint (Joint A) connecting body 1 and body 2 is cut, forming two subsystems. Then, the T–T gluing algorithm is applied.

The equations of motion for the two subsystems are

$$\mathbf{a}^i = \ddot{\mathbf{q}}^i = \begin{Bmatrix} \ddot{x}^i \\ \ddot{y}^i \\ \ddot{\theta}^i \end{Bmatrix}, \quad \mathbf{M}^i = \begin{bmatrix} m^i & 0 & 0 \\ 0 & m^i & 0 \\ 0 & 0 & I^i \end{bmatrix} \quad (i = 1, 2), \tag{38}$$

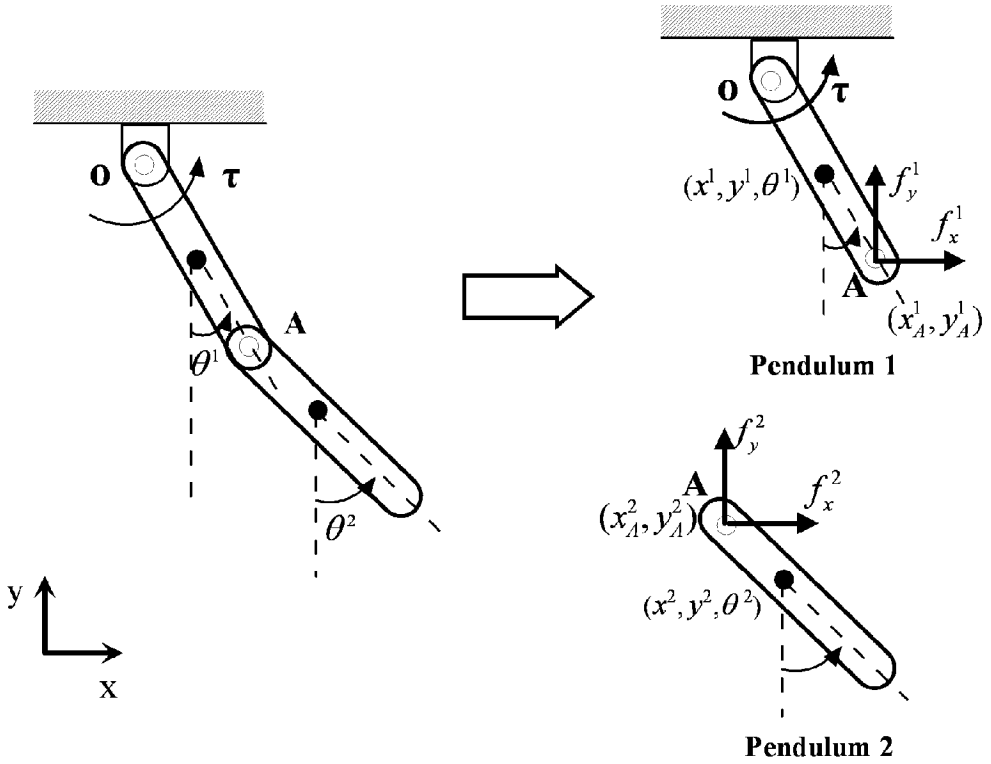


Figure 7. A double pendulum divided as two subsystems.

where $\mathbf{q}^i = \{x^i, y^i, \theta^i\}^T$ $(i = 1, 2)$, $I^i = 1/12 m^i (l^i)^2$, and m^i and l^i are the mass and length of the i-th pendulum $(i = 1, 2)$. For the first pendulum we have

$$\mathbf{Q}^1 = \begin{Bmatrix} f_x^1 \\ f_y^1 \\ \tau + \frac{l^1}{2}(f_x^1 \cos\theta^1 + f_y^1 \sin\theta^1) \end{Bmatrix}, \tag{39}$$

$$\mathbf{\Phi}^1 = \begin{Bmatrix} x^1 - \frac{l^1}{2}\cos\theta^1 \\ y^1 + \frac{l^1}{2}\sin\theta^1 \end{Bmatrix}, \quad \mathbf{\Phi}_q^1 = \begin{bmatrix} 1 & 0 & \frac{l^1}{2}\sin\theta^1 \\ 0 & 1 & \frac{l^1}{2}\cos\theta^1 \end{bmatrix}, \tag{40}$$

and for the second pendulum we have

$$\mathbf{Q}^2 = \begin{Bmatrix} f_x^2 \\ f_y^2 \\ \frac{l^2}{2}(-f_x^2 \cos\theta^2 - f_y^2 \sin\theta^2) \end{Bmatrix}, \tag{41}$$

$$\mathbf{\Phi}^2 = 0, \quad \text{and} \quad \mathbf{\Phi}_q^2 = 0. \tag{42}$$

The compatibility condition for the system is the continuity of displacement at joint A, which can be written as

$$\mathbf{e} = \begin{Bmatrix} e_x \\ e_y \end{Bmatrix} = \begin{Bmatrix} x_A^1 - x_A^2 \\ y_A^1 - y_A^2 \end{Bmatrix} = \mathbf{0}, \tag{43}$$

where x_A^i, y_A^i are the displacements of the two pendulums ($i = 1, 2$) at the connecting joint A. Here we could also employ acceleration or velocity in the compatibility condition Equation (43).

The equilibrium condition at the joint becomes

$$\begin{cases} f_x^1 + f_x^2 = 0, \\ f_y^1 + f_y^2 = 0. \end{cases} \tag{44}$$

As before, we define an interface force vector,

$$\mathbf{F} = \begin{Bmatrix} f_x \\ f_y \end{Bmatrix} = \begin{Bmatrix} f_x^1 \\ f_y^1 \end{Bmatrix}. \tag{45}$$

Then

$$\begin{cases} f_x^2 = -f_x \\ f_y^2 = -f_y \end{cases}. \tag{46}$$

Applying the T–T method, the $\mathbf{\Lambda}$ matrix can be calculated as

$$\mathbf{\Lambda} = \left[\bar{\mathbf{G}}^1 + \bar{\mathbf{G}}^2\right]^{-1}, \tag{47}$$

where

$$\bar{\mathbf{G}}^i = \begin{bmatrix} \frac{\partial x_A^i}{\partial f_x^i} & \frac{\partial x_A^i}{\partial f_y^i} \\ \frac{\partial y_A^i}{\partial f_x^i} & \frac{\partial y_A^i}{\partial f_y^i} \end{bmatrix} \quad (i = 1, 2). \tag{48}$$

Now Equation (3) can be used to update the interface forces so that the two pendulums can be glued together. In this multibody dynamics system, the relationships between the force quantities and kinematic quantities are no longer linear, which can be observed from the subsystems' equations, Equations (37–42).

For a nonlinear subsystem we approximate

$$\frac{\partial x_m}{\partial f_n} \approx \frac{\Delta x_m}{\Delta f_n}, \tag{49}$$

where $\partial x_m / \partial f_n$ represents a component in the right side matrix of Equation (48). Obviously, the amplitude of the Δf_n will affect the outcome of the lambda matrix, and in general, the lambda matrix will be a function of the interface forces calculated at the previous step. In this paper, Δf_n is determined by a percentage of the corresponding interface force component, namely

$$\Delta f_n = \varepsilon f_n, \tag{50}$$

where, ε is a small ratio, which can be typically selected as $\varepsilon = 0.01$.

The gluing algorithm (T–T method) for a multibody dynamics system is given by:

ALGORITHM 3 (Gluing algorithm for a multibody dynamics system).

1. $n = 0$,
2. *At each time step n, update the gluing matrix* $\mathbf{\Lambda}_n$ *as needed,*

Table 1. Component models of the chassis frame system using a finite element representation.

Component	Number of nodes	Number of elements	Number of interface nodes
Left Rail	622	562	31
Right Rail	623	564	31
Connector 1	192	187	28
Connector 2	105	84	8
Connector 3	74	51	10
Connector 4	108	83	16
Coupled System	1662	1531	62

3. *Let* $i = 0$,
4. *Solve each subsystems' equations with the interface forces predicted from the gluing algorithm, and predict the states of the subsystems for next time step, namely obtain* $\mathbf{q}_{n+1}^{(i)}$, $\dot{\mathbf{q}}_{n+1}^{(i)}$, *and* $\ddot{\mathbf{q}}_{n+1}^{(i)}$,
5. *Check whether* $||\mathbf{e}|| < \delta$ *is satisfied, where* δ *is a given error tolerance. If satisfied, then* $n = n + 1$ *and go to step (2),*
6. *Update the interface force vector* $\mathbf{F}$ *using Equation (3), namely,* $\mathbf{F}^{(i+1)} = \mathbf{F}^{(i)} + \mathbf{\Lambda}_n(-\mathbf{e}^{(i)})$,
7. *Let* $i = i + 1$ *and go to step (4).*

Normally, the $\mathbf{\Lambda}$ matrix need not be updated at each time step if the configuration of the subsystems does not change much within a time step. Therefore, we can use the $\mathbf{\Lambda}$ matrix calculated from a previous time step, and only update it when, for example, the iteration number during a present time step exceeds a certain value.

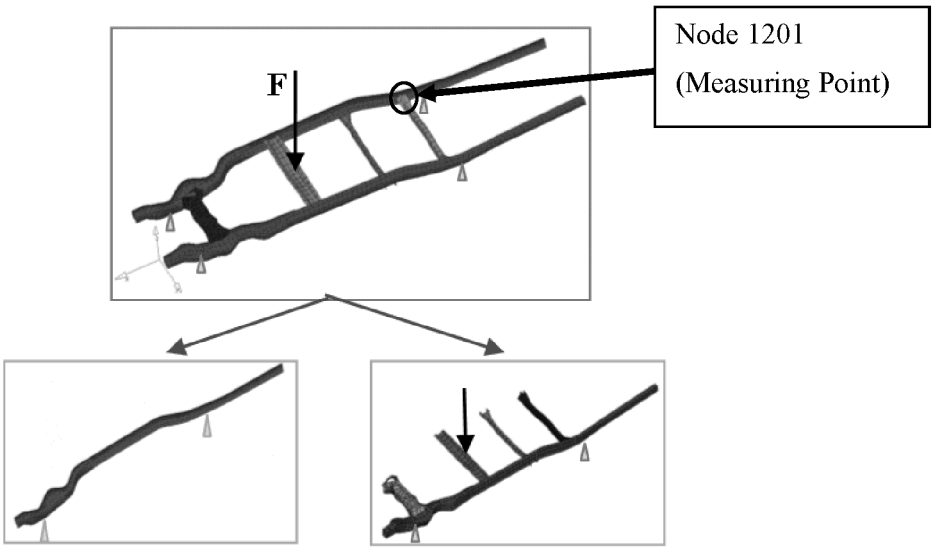
One concern is that this method may be problematic for a highly nonlinear system if the gluing matrix is obtained from the standard Newton–Raphson method, namely if Equation (4) is used. As mentioned previously, the T–T method is, however, not limited to the use of the standard Newton–Raphson method. The T–T method is even not limited to the one-step method discussed in this paper. The scope of this paper is to layout the basic concept of the T–T method and to demonstrate its capabilities for solving general classes of dynamic and multibody dynamic problems. Research is still ongoing and more refined results will be published in a subsequent paper.

5. Examples

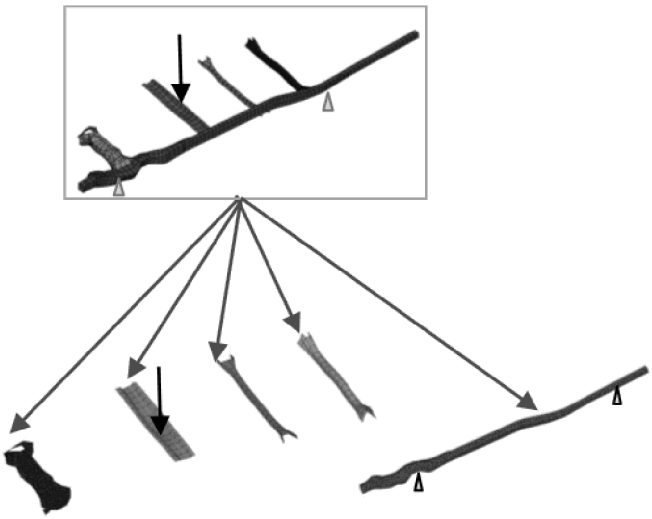
In this section, three examples are presented to demonstrate the gluing algorithm (T–T method) developed. A finite element truck frame model is considered first, then a double pendulum model and a four-bar link multibody dynamics model are discussed.

5.1. EXAMPLE 1: GLUING A TRUCK CHASSIS FRAME OF FINITE ELEMENT MODELS

The first example demonstrates the gluing algorithm developed in this paper applied to a distributed simulation problem of the truck chassis frame shown in Figure 2. This simulation



a) First layer



b) Second layer

Figure 8. A two-layer distributed simulation of a truck chassis frame system.

problem has two gluing layers as shown in Figure 8, which simulates the generality of our coupling process for a distributed structural system. In the first layer (Figure 8a), the frame structure is divided into two subsystems, which include a right rail model and a left-side model that contains the rest of the components of the frame system. In the second layer (Figure 8b), the left-side subsystem comprises 5 components, namely a left rail and 4 connectors. The simulation system couples the second layer components first to form a higher-level subsystem

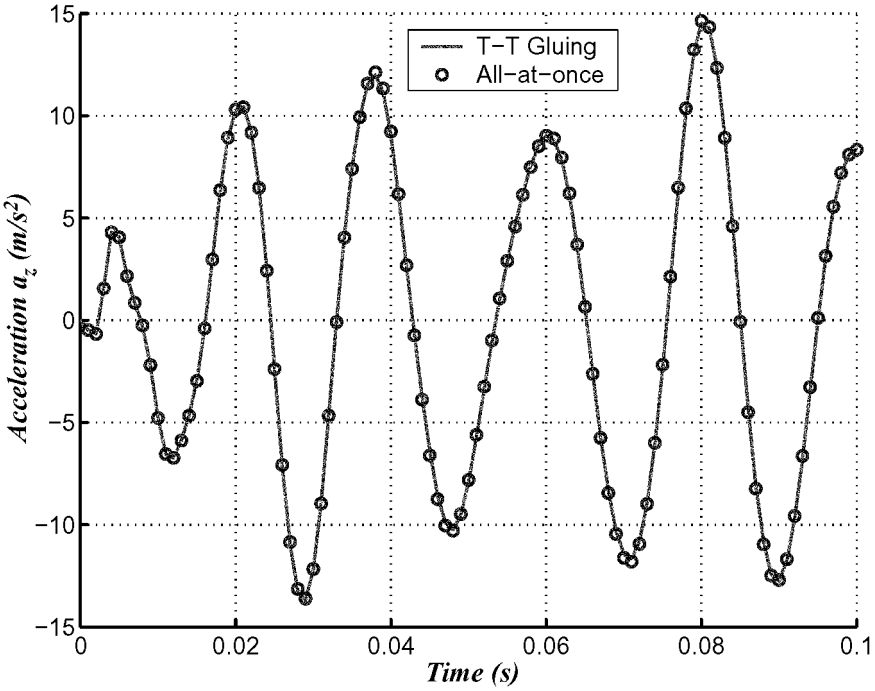


Figure 9. Nodal acceleration in z direction at node 1201.

model, and then couples the first layer subsystems' models to form the frame system. This nesting process can continue until all the components in the vehicle system are included. As shown in Figure 8, each component consists of a finite element model, which is wrapped by its own solver (standalone finite element code). Table 1 shows the modeling details of each component; including the numbers of nodes, elements, and interface nodes used in each component model, with a total number of 1662 nodes and 1531 shell elements in the integrated model. During the gluing process, the models at the two levels communicate by exchanging their interface information at their own level, and the gluing coordinators update the interface variables using the gluing algorithm 1. Note that each layer has its own coordinator and its own gluing process. Since the problem is linear, there is no need for iteration when updating the interface forces using Equation (3).

Figure 9 shows the results obtained using the gluing system developed compared with results obtained using an 'all-at-once' finite element analysis. Here, a dynamic load $f = 2000\sin(100\pi t)$N is applied at the middle point of the connector 2, along the global z (vertical) direction, and the chassis frame is supported at the four points shown in Figure 8. Figure 9 shows the acceleration at a selected node (node 1201) along the vertical direction. It is clear that the gluing process induces no additional error (beyond the round-off errors).

Figure 10 illustrates one of the interface force components obtained during the simulation using the developed gluing algorithm, which can be easily obtained from the available information.

5.2. EXAMPLE 2: GLUING THE DOUBLE PENDULUM

The double pendulum problem, depicted in Figure 7 was solved using the proposed gluing algorithm and using the commercial multibody dynamics code ADAMS. The parameters for

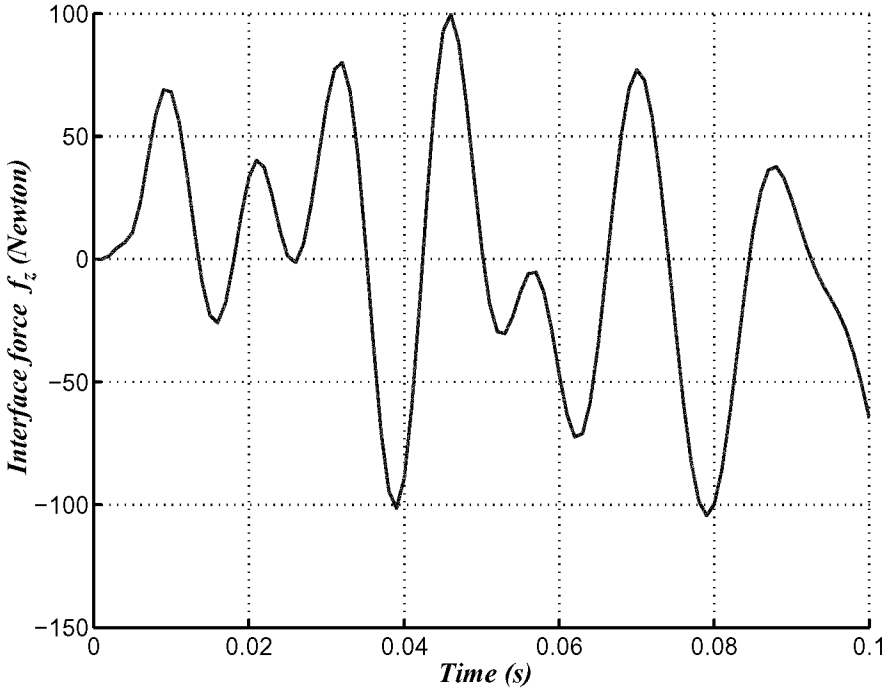


Figure 10. Interface force in z direction at node 1201.

the double pendulum are (see Equations (38–42) and Figure 7)

$$\begin{aligned} m^1 &= 10\,\text{kg}, \quad m^2 = 20\,\text{kg}, \\ l^1 &= 1\,\text{m}, \quad l^2 = 2\,\text{m}, \\ \tau &= -200\sin(2\pi\ t)\ \text{N}\cdot\text{m}. \end{aligned}$$

The two pendulums are driven from rest with zero initial speed. The equations of motion of the pendulums are first reduced to ODEs in terms of θ^1 and (x^2, y^2, θ^2), respectively. Then both subsystems are solved separately, using the *ode*45 solver in Matlab. The tolerance for the compatibility condition is $||\mathbf{e}|| \leq 1.0E-10$ and Δt is selected as $1.0E-2$ s.

Figures 11 and 12 illustrate the interface forces predicted using the T–T method and using ADAMS (which employs an all-at-once model.) It is clear that the results obtained using the gluing algorithm are in good agreement with the ADAMS results.

Figures 13 and 14 depict the displacement and velocity obtained at the center of mass of the second pendulum. It can be seen that the displacement and velocity calculated using the gluing algorithm also are in good agreement with the ADAMS results.

Figure 15 shows the iteration count in each time step for updating the interface forces using Equation (3) when the $\mathbf{\Lambda}$ matrix is updated at each time step. It is seen that 3–6 iterations were needed per time step for this typical multibody dynamics problem for an error tolerance $||\mathbf{e}|| \leq 1.0E-10$ and $\Delta t = 1.0E-2$ s. For a comparison, Figure 16 further shows the updating frequency of the $\mathbf{\Lambda}$ matrix obtained from another calculation when the maximal iteration number is limited to 6. In this case, a total 61 updates are needed during the 300 steps of the simulation. Note that the $\mathbf{\Lambda}$ matrix only needs to be updated at one-fifth of the time steps.

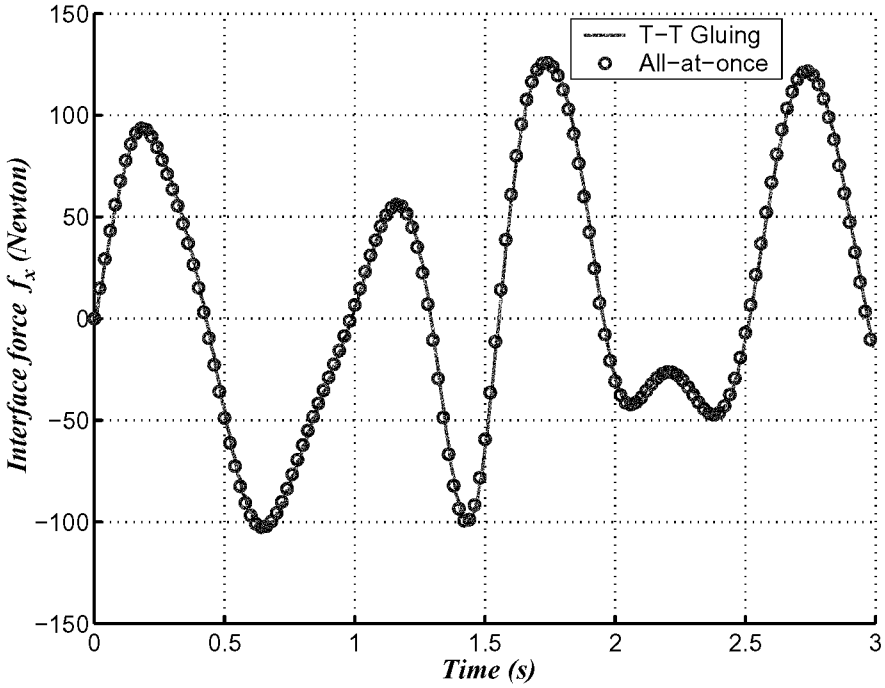


Figure 11. Interface force f_X.

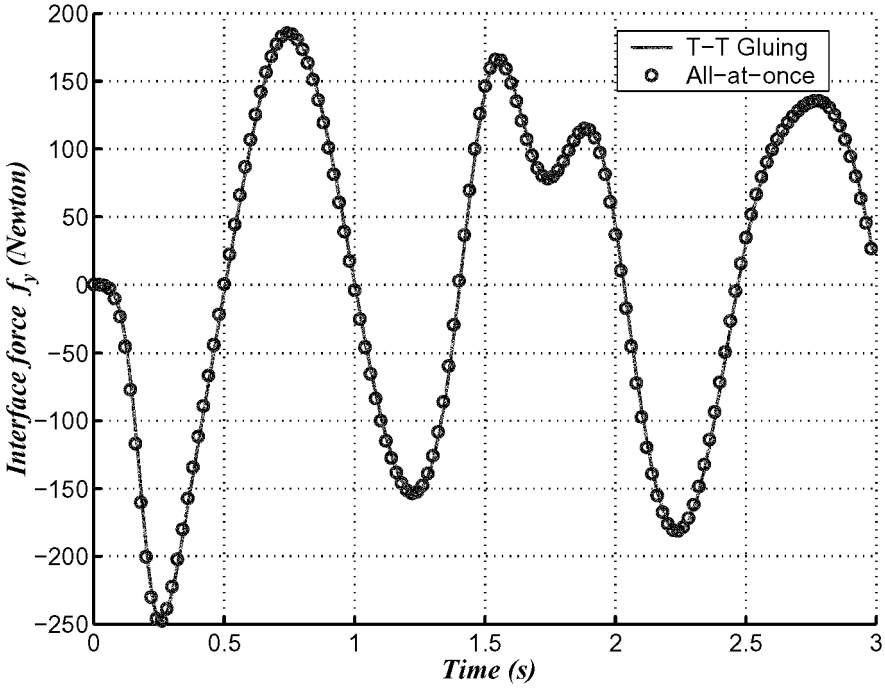


Figure 12. Interface force f_y.

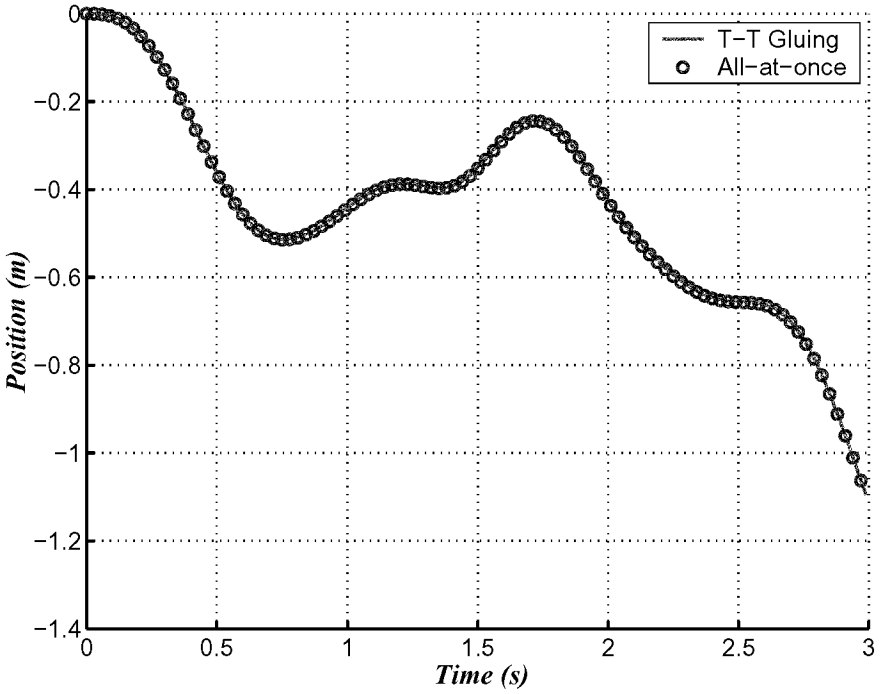


Figure 13. Center of mass position of pendulum 2 in x-direction.

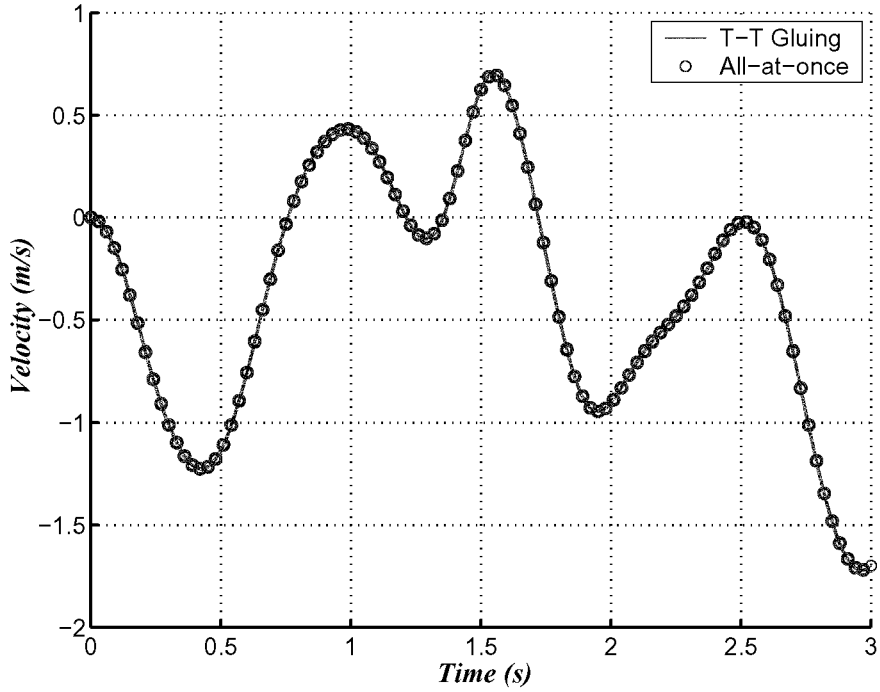


Figure 14. Center of mass velocity of pendulum 2 in x-direction.

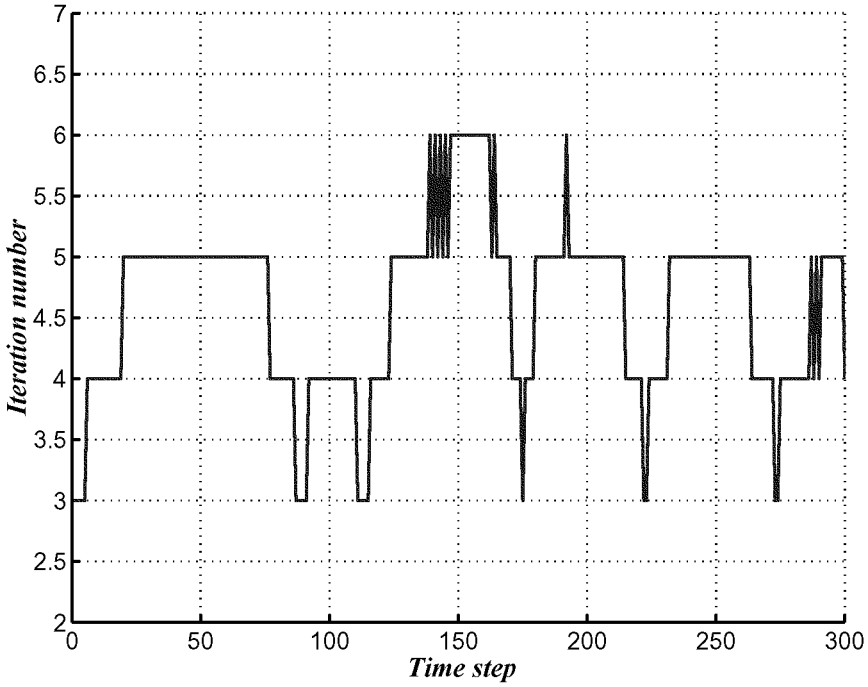


Figure 15. Iteration time history in the gluing process.

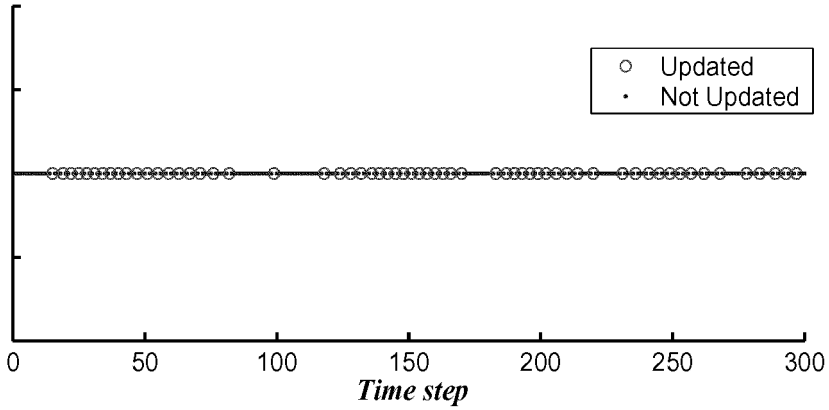


Figure 16. Updating frequency of $\mathbf{\Lambda}$ matrix.

Figure 17 illustrates the effect of ε in Equation (50) on the iteration count through a number of numerical experiments for the same problem. It can be seen that for a wide range of values, i.e., from $1.04E-4$ to $1.0E2$, the average number of iterations changes little. This suggests that the value of ε in Equation (50) can be selected in a relatively wide range.

5.3. EXAMPLE 3: GLUING A FOUR-BAR LINK MECHANISM WITH A FLEXIBLE COMPONENT

In this example, a four-bar link mechanism is considered. To demonstrate the applicability of the methodology developed for a general mechanical system, one bar in the four-bar link system is considered flexible, and the rest bars are considered rigid, as shown in Figure 18.

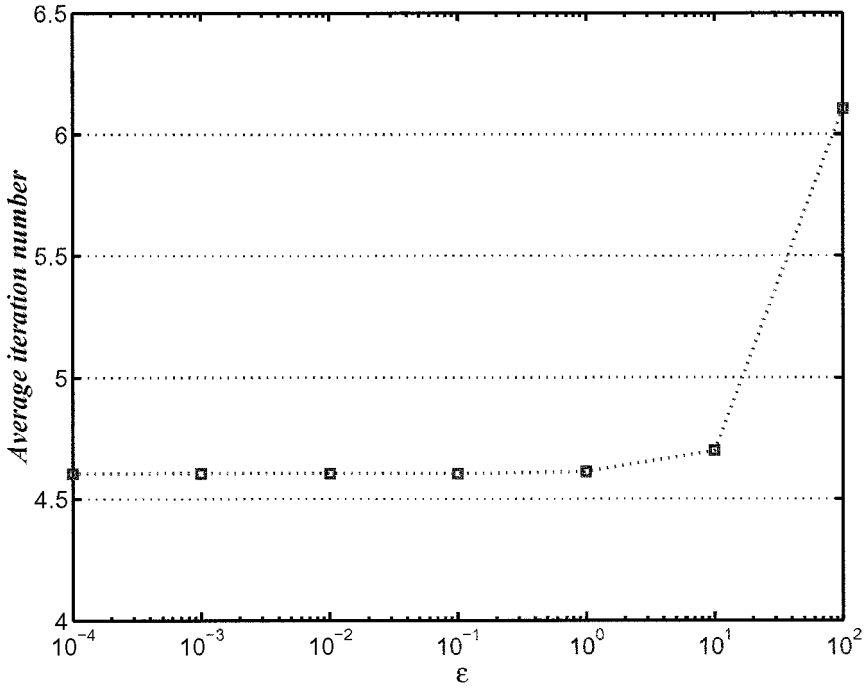


Figure 17. Effect of ε in Equation (50).

Figure 18 shows the four-bar link is separated into two subsystems, which will be glued together using the gluing algorithm developed. As shown in the figure, the separation is at joint C, which connects Bodies 2 and 3. The first subsystem includes Body 1 and 2, both of which are modeled as rigid bodies. The second subsystem comprises only Body 3, which is modeled as a flexible Euler–Bernoulli beam. Using the floating frame of reference coordinate system shown in Figure 18, the deformation shape of the beam is represented by the assumed modes as

$$\begin{Bmatrix} u \\ v \end{Bmatrix} = \mathbf{Sq} = \begin{bmatrix} \sin\left(\frac{\pi x}{2l}\right) & 0 & 0 \\ 0 & \sin\left(\frac{\pi x}{l}\right) & \sin\left(\frac{2\pi x}{l}\right) \end{bmatrix} \begin{Bmatrix} q^1 \\ q^2 \\ q^3 \end{Bmatrix}. \tag{51}$$

Parameters used for each body are

$$\begin{aligned} m^1 &= 10\ \text{kg}, \quad m^2 = 20\ \text{kg}, \quad m^3 = 5\ \text{kg}, \\ l^1 &= 1\ \text{m}, \quad l^2 = 2\ \text{m}, \quad l^3 = 1\ \text{m}, \end{aligned}$$

where m^i $(i = 1, 2, 3)$ are the masses of bar 1, 2 and 3 and l^i $(i = 1, 2, 3)$ are the link lengths of the bars. A driving torque is applied at joint A with a profile as

$$\tau = \begin{cases} -1{,}000^* t\ \text{N} \cdot \text{m}, & t < 0.1\ \text{s}, \\ -100\ \text{N} \cdot \text{m}, & t \geq 0.1\ \text{s}. \end{cases}$$

For the flexible beam, the cross-section is assumed to be round with a radius $r = 0.015$ m and uniform along the axial direction. Young's modulus and mass density of the beam are $E = 210$ GPa and $\rho = 7.0E3$ kg/m^3.

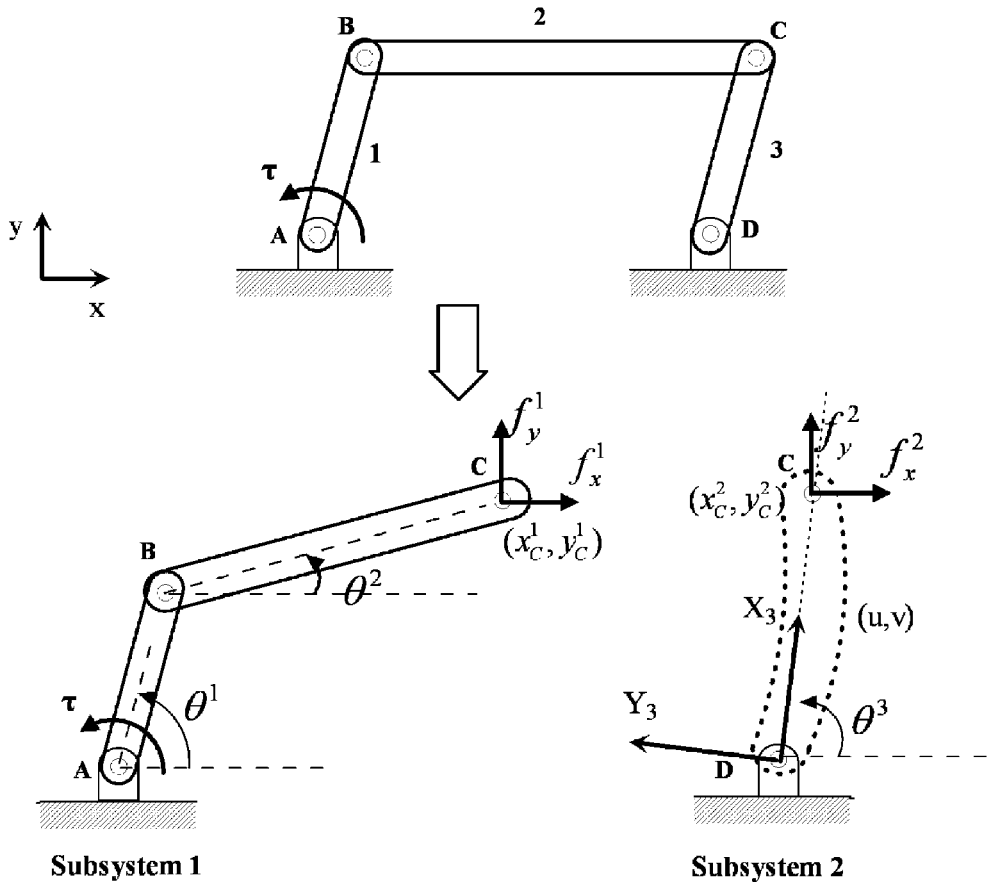


Figure 18. Gluing simulation of a four-bar link mechanism.

The mechanism is driven from an initial position $\theta^1 = \theta^3 = 0.3$ Rad with initial speeds $\dot{\theta}^1 = \dot{\theta}^3 = -2$ Rad/s and all other initial conditions are zero. The equations of motion of both subsystems are first reduced to ODEs in terms of (θ^1, θ^2) and $(\theta^3, q^1, q^2, q^3)$, respectively. Then both subsystems are solved using the *ode*45 solver in Matlab. The error tolerance for the gluing is $||\mathbf{e}|| \leq 1.0E - 10$ and the time step size Δt is selected as $1.0E - 3$ s. The compatibility condition at joint C is used to update the interface forces with the use of Equation (3). ADAMS/Flex was employed as the all-at-once system benchmark, in which the flexible bar is modeled using 10 beam elements, which are finally reduced to three modal coordinates as used in the gluing simulation. A damping coefficient of 0.1 is applied to all three modes in both simulations. Figures 19–22 compare the results obtained from the T–T method and the ADAMS simulation. Figure 19 and 20 compare the displacement at the cut joint (joint C); Figures 21 and 22 compare the velocity at the same joint. Here all the measurements are in the global coordinate system. It is seen that good agreement is obtained between the two simulations except for the small oscillation in the velocity as seen in Figure 22, which arises when the mechanism passes through the 'singular points'. These oscillations are observed in the results from both the T–T method and the ADAMS simulation with a slight difference. Note that the singular point is at the time when all bars lie on the same line when ignoring

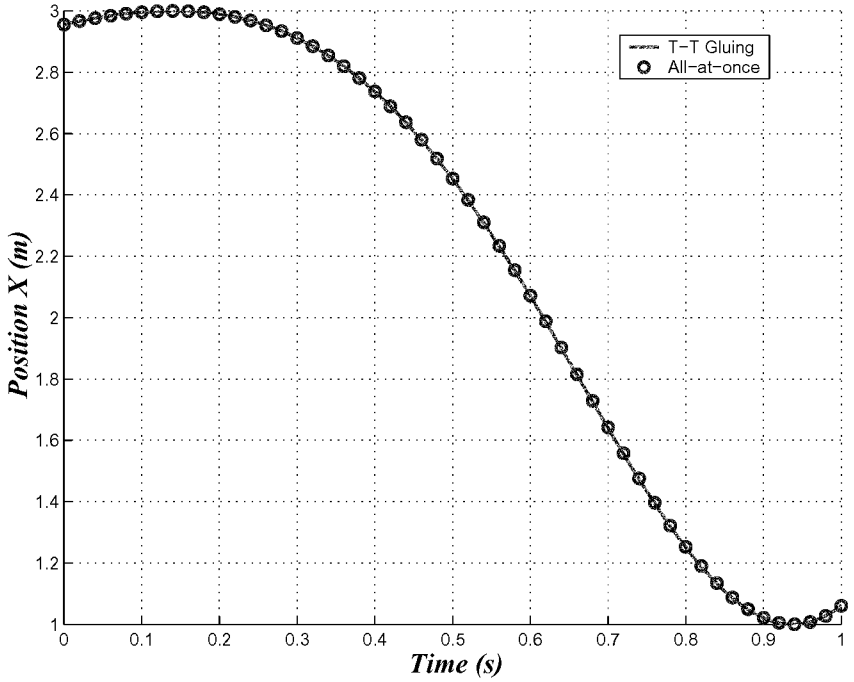


Figure 19. Displacement along global X direction of the interface joint C.

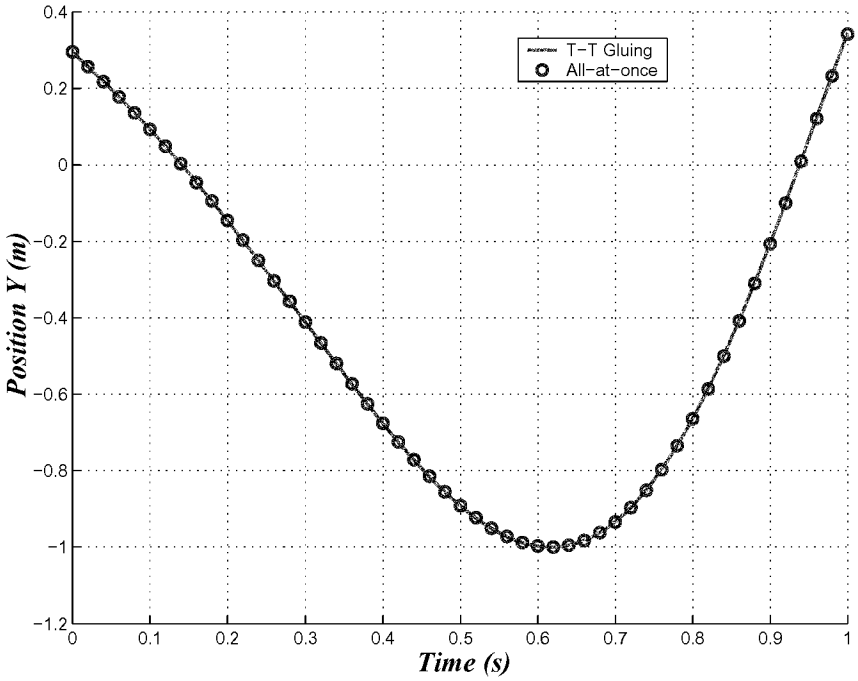


Figure 20. Displacement along global Y direction of the interface joint C.

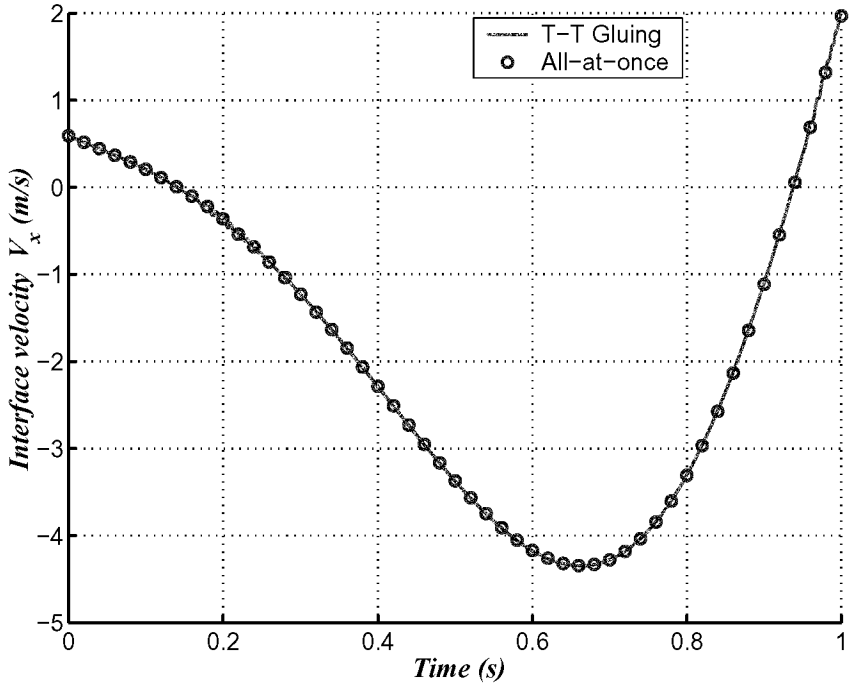


Figure 21. Velocity along global X direction of the interface joint C.

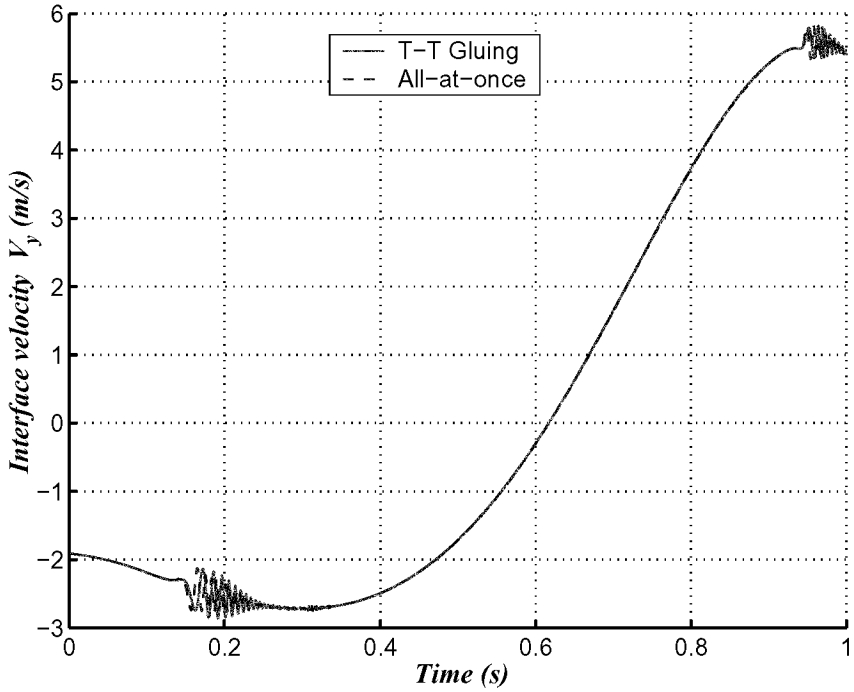


Figure 22. Velocity along global Y direction of the interface joint C.

the deformation. Further investigation is still needed for understanding the mechanism of this oscillation.

6. Conclusion

A general gluing algorithm, the T–T method, is presented in this paper, which can be used to glue, in an effective and accurate way, distributed subsystem models for both structural dynamics and multibody dynamics problems. It can also be used to solve static problems for distributed systems. The formulation of the gluing algorithm is general so it can be extended to deal with a variety of different gluing problems, including linear and non-linear problems. The proposed gluing algorithm relies only on the interface information exposed by the subsystem models, without requiring internal details of the model. Therefore, each subsystem model can be treated as a black box, regardless of the model and its inherent solution scheme. From the outset of the algorithm development, we considered that subsystem models do not possess knowledge of the gluing algorithm, and that the models may be built using different commercial packages, which usually do not communicate well with each other. These features make the algorithm suitable for use in a practical environment of distributed simulation within a real distributed production system.

We have demonstrated that the T–T method can produce exact solutions for linear systems, including static and dynamics problems, without any iteration of the updating equation, and with only one-time calculation of the gluing matrix at the beginning of the simulation. With iteration, the gluing algorithm can be used to solve nonlinear multibody dynamics problems, including rigid and flexible multibody dynamics systems. A number of multibody dynamics problems, including flexible members, have been successfully solved, and two examples are shown in this paper. Future development will focus on the applicability of the gluing algorithm to a broader class of distributed system simulations.

Acknowledgments

The authors would like to acknowledge the support provided by the U.S. Army Tank-Automotive and Armaments Command (TACOM) through the Automotive Research Center at the University of Michigan under contract DAAE07-98-3-0022.

References

1. Viswanadham, N., 'The past, present, and future of supply-chain automation', *IEEE Robotics and Automation Magazine* **9**(2), 2002, 48–56.
2. Tseng, F. C., 'Multibody dynamics simulation in network- distributed environments', Ph.D Dissertation, University of Michigan, 2000.
3. Huddi, A. V. and Pidaparti, R. M. V., 'Distributed finite element structural analysis using the client-server model', *Communications in Numerical Methods in Engineering* **11**, 1995, 227–233.
4. Kumar, S. and Adeli, H., 'Distributed finite-element analysis on network of workstations – Algorithms', *Journal of Structural Engineering* **121**(10), 1995, 1448–1455.
5. Farhat, C. H. and Wilson, E., 'A parallel active column equation solver', *Computers & Structures* **28**, 1988, 289–304.
6. Farhat, C. H. and Roux, F. X., 'Implicit parallel processing in structural mechanics', *Computational Mechanics Advances* **2**, 1994, 1–124.

7. Craig, R. R., 'Coupling of substructures for dynamic analyses: An overview', in *Collection of Technical Papers – AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Reston, VA, Vol. 5, 2000, pp. 3–14, AIAA-2000-1573.
8. Tallec, P. L., 'Domain decomposition methods in computational mechanics', *Computational Mechanics Advances* **1**, 1994, 121–220.
9. Kim, S. S., 'A subsystem synthesis method for an efficient vehicle multibody dynamics', *Multibody System Dynamics* **7**, 2002, 189–207.
10. Featherstone, R., '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), 1999, 867–875.
11. Featherstone, R., '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), 1999, 876–892.
12. Anderson, K. S. and Duan, S., 'Highly parallelizable low-order dynamics simulation algorithm for multi-rigid-body systems', *Journal of Guidance, Control, and Dynamics* **23**(2), 2000, 355–364.
13. Duan, S. and Anderson, K.S., 'Parallel implementation of a low order algorithm for dynamics of multibody systems on a distributed memory computing system', *Engineering with Computers* **16**(2), 2000, 96–108.
14. Sharf, I. and D'Eleuterio, G. M. T., 'Parallel simulation dynamics for elastic multibody chains', *IEEE Transactions on Robotics and Automation* **8**, 1992, 597–606.
15. Kübler, R. and Schiehlen, W., 'Modular simulation in multibody system dynamics', *Multibody System Dynamics* **4**, 2000, 107–127.
16. Gu, B. and Asada, H. H., 'Co-simulation of algebraically coupled dynamic subsystems', in *Proceedings of the American Control Conference*, Arlington, VA, Vol. 3, 2001, pp. 2273–2278 (IEEE cat n 01CH37148).
17. Tseng, F. C. and Hulbert, G. M., 'A gluing algorithm for network-distributed dynamics simulation', *Multibody System Dynamics* **6**(4), 2001, 377–396.
18. Tseng, F. C., Ma, Z. D., and Hulbert, G. M., 'Efficient numerical solution of constrained multibody dynamics systems', *Computer Methods in Applied Mechanics and Engineering* **192**, 2003, 439–472.
19. Hulbert, G. M., Michelena, N., and Ma, Z. D., et al., '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.
20. Fisette, P. and Peterkenne, J. M., 'Contribution to parallel and vector computation in multibody dynamics', *Parallel Computing* **24**(5–6), 1998, 717–728.
21. Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, 1992, pp. 425–428.