

# Modular and distributed forward dynamic simulation of constrained mechanical systems – A comparative study

Waseem A. Khan <sup>a</sup>, Chin Pei Tang <sup>b</sup>, Venkat N. Krovi <sup>b,\*</sup>

<sup>a</sup> Centre for Intelligent Machines, Department of Mechanical Engineering, McGill University, 3480 University Street, #421, Montreal, Que., Canada H3A 2A7

<sup>b</sup> Department of Mechanical and Aerospace Engineering, State University of New York at Buffalo, 318 Jarvis Hall, Buffalo, NY 14260, USA

Received 19 August 2005; received in revised form 4 May 2006; accepted 16 May 2006  
Available online 10 July 2006

---

## Abstract

In this paper, we examine the modular development of two alternate methods for distributed computation of the forward dynamics simulations of constrained mechanical systems. We exploit the natural spatial parallelism of closed-chain linkages, initially, for the modular development of overall dynamics, and subsequently, for the distributed numerical simulation of the dynamics. Traditionally, the numerical simulation problem of constrained mechanical systems has been treated as two separate stages: the problem of algorithm development and the subsequent numerical problem of advancing the discretized differential equations in time. However the potential numerical instabilities arising from the *formulation stiffness* of the algorithm development stage has the potential to hinder the subsequent numerical integration stage. These aspects are also explored during the evaluation of the two alternative approaches for the distributed forward dynamics simulation of a four-bar linkage. A series of tests were performed on a real-time synchronized distributed computational system (RT-LAB) to evaluate the performance of both methods of simulation and preliminary results to quantify the *overall* computational efficiency and accuracy are presented.

© 2006 Elsevier Ltd. All rights reserved.

---

## 1. Introduction

In the last quarter century, dynamics simulation tools have seen manifold increases in terms of their usage in the design, analysis, parametric refinement and ultimately model-based control of a variety of multibody systems such as vehicles, heavy machinery, spacecraft and robots. In the absence of efficient, general purpose, closed-form analytical methods, numerical simulation methods have taken a premier position for simulation of such multibody systems. The interested reader may refer to a number of books on the subject [1–5] for further details on the wide variety of formulations and computational methods that exist in the literature for numerical implementation of multibody simulations.

---

\* Corresponding author.

E-mail addresses: [wakhan@cim.mcgill.ca](mailto:wakhan@cim.mcgill.ca) (W.A. Khan), [chintang@eng.buffalo.edu](mailto:chintang@eng.buffalo.edu) (C.P. Tang), [vkrovi@eng.buffalo.edu](mailto:vkrovi@eng.buffalo.edu) (V.N. Krovi).

While efficient formulations exist for serial chain and tree structured multibody systems, the adaptation of these methods for the simulation of closed-chain linkages and parallel manipulators is relatively more difficult. Such systems possess one or more closed kinematic loops, requiring the introduction of algebraic (typically nonlinear) constraints into the formulation. In our work, we restrict our attention to the *forward dynamic simulation* of this class of constrained mechanical systems.

At the outset, we note that the configuration of such constrained multibody systems can be described by a variety of sets of coordinates. The suitable selection of a set of *configuration coordinates* is of particular importance due to its impact both on the ease of formulation and the subsequent computational efficiency. While the use of expanded sets of *dependent configuration coordinates* linked together by *constraining relations* is considered more appropriate for general purpose analysis than the use of a minimal set of independent configuration coordinates, considerable variety and choice in selection of such sets exists. These choices may be broadly classified into: (a) relative, (b) reference point, and (c) natural (or fully Cartesian), each bringing its corresponding share of advantages and shortcomings which are discussed in detail in [2]. In our case, we focus our attention on the use of sets of *relative coordinates*, parameterizing the relative degree-of-freedom (DOF) at kinematic articulations. By facilitating direct control of joint-based actuators and enabling a minimal description of the configuration of open-chain systems, such sets of relative coordinates have found extensive use in the mechanisms and robotics community despite other shortcomings in the form of creation of transcendental constraint relations or of relatively increased complexity of formulations of equations of motion (EOM).

In this paper, we examine both the development and performance evaluation of two alternate methods for *modular and distributed forward dynamic simulations of constrained mechanical systems using such relative coordinates*. The emphasis on modular development is to promote the reuse of existing components. Specifically, we consider that: (a) the EOM of the individual subsystems/modules are well known and available and (b) an overall system may be composed of several serial or tree structured individual subsystems plus sets of holonomic constraints. In particular, we examine exploiting the spatial parallelism [6] that is inherent in closed kinematic chains to pursue a modular composition of the overall system dynamics. Such simulations are typically also computationally expensive and hence effective methods for distribution of computational load on multiple processors (associated with the composing subsystems) are attractive. The traditionally adopted solution approach is to: (a) augment the unconstrained system dynamics with the differentiated constraints and Lagrange multipliers; (b) convert the augmented system into a system of Ordinary Differential Equations (ODEs) by a variety of methods (to be discussed in the next section); (c) which are then numerical integrated by appropriately applying stabilization or regularization techniques [1,2]. While good performance of such approaches has been reported for unified system solution, in our work we wish to examine their applicability and viability in the distributed computation domain.

Specifically, we examine two approaches – a compliance-based method [7], and a projection-based method [8] – from the viewpoint of *distributing the dynamics computation*. Compliance-based methods use artificial mechanical compliance elements (virtual springs and dampers) to *approximate* the constraint forces. Such methods are attractive for distribution of the forward dynamics simulation because they permit explicit approximation of loop-closure constraint forces and can then effectively decouple the numerical integration of the component dynamical subsystems. In contrast, projection-based methods require projection of the overall dynamics equations onto the feasible motion directions (the reduced dimensional space of independent generalized velocities), which can add to the computation complexity. However, by permitting the inclusion of various stabilization and regularization methods, this latter approach shows considerable promise for ensuring consistency of the constraints over long periods of time in the presence of numerical disturbances.

Thus, in this paper, we wish to compare the compliance-based and projection-based approaches, focusing specifically on: (a) the modular development of dynamics formulation of an entire closed-loop manipulator by a composition of the dynamics of the component subsystems; (b) re-distributing the computation of the forward dynamics simulations back to the individual subsystems; and (c) quantifying the relative merits in terms of the relative computational efficiency and accuracy of the two approaches. These aspects are studied in the context of the distributed forward dynamic simulation of a *planar four-bar linkage*, whose overall dynamic equations are assumed to result from constraining the independent dynamics of a 2 DOF left open serial-chain and a 1 DOF right open serial-chain by means of holonomic constraints (see Fig. 1).

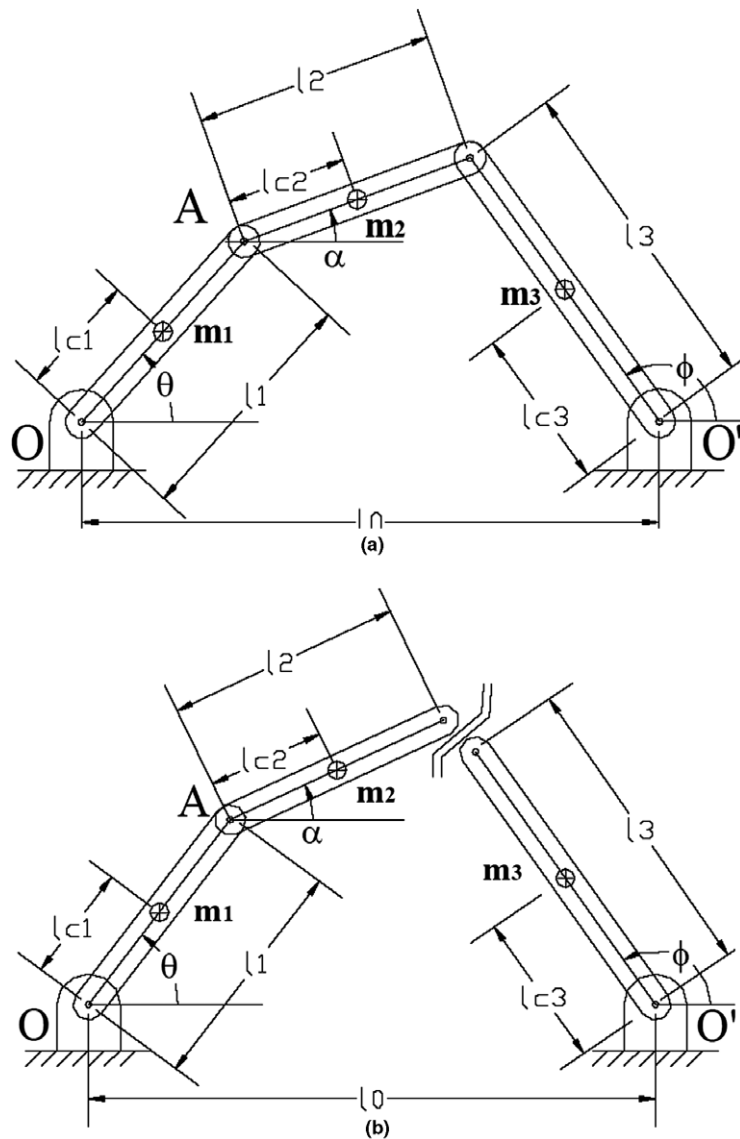


Fig. 1. The representation of the planar four-bar linkage under consideration: (a) the composite model and (b) the divided sub-chains (subsystems) model – Chain A is the left serial-chain with 2 DOF and Chain B is the right serial-chain with 1 DOF.

The rest of the paper is organized as follows: Section 2 presents a brief overview of pertinent literature, followed by a comparison of various converted ODE approaches for constrained system simulations in Section 3. The two principal formulations under consideration are presented in Section 4, while, in Section 5, we specialize these methods in distributed form in the context of a four-bar linkage. The critical implementation issues and designer selection are studied in Section 6. Section 7 presents comparative studies of the simulation results, followed by a discussion in Section 8. Finally, Section 9 concludes the paper.

## 2. Background

One promising method to overcome the complexity of robotic systems consists of breaking down the system into independent subsystems, which can be mapped onto distributed/parallel processing elements, at the algorithmic or natural body levels. Henrich and Höniger [9] present a brief review and a preliminary taxonomy

of the different levels of parallelism that have been explored in the context of robotic applications and note that parallelization at all levels may not be possible. Results obtained by parallelizing algorithms vary depending on the degree of dependency and coupling among the equations. While image processing problems [10] can be broken down quite well by dividing the image into smaller independent blocks, the problems of dynamic simulation of constrained mechanical systems is strongly coupled problem and the task is not trivially parallelizable [11,12].

Fijany and Bejczy [11] survey many of the methods developed for parallelization of dynamics algorithms, both at the algorithmic level and at the natural body level, for serial chain manipulators. Most work has focused on fine grain parallel algorithms for implementation on special purpose computational architectures [13–15]. The primary motivation behind such methods is the desire to speed up the computation to satisfy the required real-time constraints, but not the requirement for modularity. In contrast, McMillan [6] proposed and evaluated the use of a combination of spatial parallelism (based on the structural parallelism of a multi-arm or multi-legged system, and temporal parallelism) to compute the forward dynamics of individual chains, and examining the synchronization requirements and load balancing – our proposed distribution approach is more in this vein.

Several efficient algorithmic approaches have been developed for forward dynamics computation of *serial-chain* and *tree-structured* multibody systems. The two principal approaches are: Composite Rigid Body Methods (CRBM) [16] which have a computational complexity of  $O(N^3)$  but are highly effective for typical robot arms; and Articulated Body Methods (ABM) [17] with fast recursive efficient algorithms of  $O(N)$  complexity for simulation of longer serial-chain and tree structured dynamic systems. Ascher et al. [18] unified the two seemingly disparate methods by showing that the different dynamics algorithms (ABM, CRBM) can be derived as different elimination methods for solving an augmented system of Differential Algebraic Equations (DAEs). They also highlight the potential numerical instabilities (“formulation stiffness”) that can arise from separating the treatment of the forward dynamics problem of computing the system accelerations from the numerical integration problem of advancing the discretized differential equations in time and advocate a global and unified view.

The dynamics of *mechanical systems with closed loops* can be formulated as a system of ODEs whose solutions are required to satisfy additional holonomic (algebraic) equations resulting from cutting the loops [19]. The dynamics of mechanical systems with holonomic constraints can be formulated as Lagrangian equations of the first kind [20], as

$$\dot{\mathbf{q}} = \mathbf{v}, \quad (1)$$

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u}) - \mathbf{A}^T(\mathbf{q})\lambda, \quad (2)$$

$$\mathbf{C}(\mathbf{q}) = \mathbf{0}, \quad (3)$$

where  $\mathbf{q}$  is the  $n$ -dimensional vector of generalized coordinates,  $\mathbf{v}$  is the  $n$ -dimensional vector of generalized velocities,  $\mathbf{M}(\mathbf{q})$  is the  $n \times n$  dimensional inertia matrix,  $\mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u})$  is the  $n$ -dimensional vector of external forces,  $\mathbf{u}$  is the vector of actuator forces/torques,  $\mathbf{C}(\mathbf{q})$  is a  $m$ -dimensional vector of holonomic constraints,  $\mathbf{A}(\mathbf{q}) = \frac{\partial \mathbf{C}}{\partial \dot{\mathbf{q}}}$  is the  $m \times n$  constraint Jacobian matrix,  $\lambda$  is the  $m$ -dimensional vector of Lagrange multipliers.

The resulting formulations in nonminimal (redundant) sets of coordinates yield an often simpler, albeit larger, system of index-3 DAEs. The development of such models for the entire system using augmented Lagrangian-based models, initially in a redundant set of coordinates with a subsequent determination of the multipliers, is also attractive since the given models can be used for both forward and inverse dynamics.

The solution of a system of index-3 DAEs by direct finite difference discretization is not possible using explicit discretization methods. The two principal approaches adopted for the forward dynamics simulation of such systems are: (a) *Direct elimination of the surplus variables* using the position-level algebraic constraints to explicitly reduce the index-3 DAEs to an ODE in a minimal set of generalized coordinates (conversion into Lagrangian equations of the second kind). The resulting (smaller size) ODE can then be integrated using numerical ODE methods without worrying about the stability issues. However, such a reduction cannot be done in general, and even when it can, the obtained differential equations are typically complicated [21]. (b) *Converted ODE approach* wherein all the algebraic position and velocity level constraints are differentiated and represented at the acceleration level to obtain an augmented index-1 DAE (in terms of both the unknown

accelerations and the unknown multipliers). Differentiating the position constraints in Eq. (3) with respect to time yields the velocity-level constraints:

$$\dot{\mathbf{C}} = \mathbf{A}(\mathbf{q})\mathbf{v} = \mathbf{0} \quad (4)$$

and a further differentiation with respect to time yields the acceleration-level constraints as

$$\ddot{\mathbf{C}} = \mathbf{A}(\mathbf{q})\dot{\mathbf{v}} + \dot{\mathbf{A}}(\mathbf{q})\mathbf{v} = \mathbf{0}. \quad (5)$$

Thus, Eq. (2) can then be written together with Eq. (5) as an index-1 DAE as

$$\begin{bmatrix} \mathbf{M} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}_{(n+m) \times (n+m)} \begin{bmatrix} \dot{\mathbf{v}} \\ \lambda \end{bmatrix}_{(n+m) \times 1} = \begin{bmatrix} \mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u}) \\ -\dot{\mathbf{A}}(\mathbf{q})\mathbf{v} \end{bmatrix}_{(n+m) \times 1}. \quad (6)$$

Such index-1 DAEs may be solved by: simply eliminating the Lagrange multipliers in favor of the unknown accelerations leaving a system of ODEs [1]; explicitly computing the Lagrange multipliers by a projection into the constrained force space [22]; approximating the Lagrange multipliers using artificial compliance elements, such as virtual springs and dampers [7]; or projecting the EOM onto the tangent space of the constraint manifold in a variety of ways to obtain constraint-reaction free EOM [2]. These aspects are explicitly discussed in the next section.

Some of the drawbacks of the converted ODE approach include the need to provide additional consistent initial conditions and the fact that the differentiated constraint manifold is mildly unstable resulting in drift of the state from the position level constraint manifold. While the growth rate can be reduced by lowering the error tolerance and/or by using smaller step-sizes or greater numerical precision, this comes at the cost of longer and more expensive computations.

Hence, most constrained multibody methods also combine (one or more) of the following methods for improved numerical solution using explicit discretization methods [1]: (a) Coordinate projection of the state of the system onto the constraint manifold at frequent intervals to ensure maintenance of the algebraic constraint; (b) Computing a local velocity-level parameterization and integrating the ODE on the constraint manifold (in the independent coordinates); or (c) Creation of an artificial first- or second-order dynamical system which has the algebraic constraint as its attractive equilibrium configuration (Baumgarte's stabilization technique) [23]. While Baumgarte's technique is very popular in the engineering application community, principally due to the resulting augmented ODE formulation, the practical selection of the parameters of the stabilization system depends both on the discretization methods and step-size, and is widely regarded as an open research problem [24].

### 3. Converted ODE approaches

Index-1 DAE systems resulting from the converted ODE approach are typically solved using one of the following three approaches: (a) direct Lagrange multiplier elimination; (b) compliance-based; or (c) projection-based.

In the *direct Lagrange multiplier elimination* approach, a simultaneous solution of the augmented linear system of equations in Eq. (6) is possible at each time step. While we note that an explicit inversion of the augmented system may be avoided by adopting a Gaussian elimination method, the overall approach may still be denoted as

$$\begin{bmatrix} \dot{\mathbf{v}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f} \\ -\dot{\mathbf{A}}(\mathbf{q})\mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1(\mathbf{q}, \mathbf{v}) \\ \mathbf{f}_2(\mathbf{q}, \mathbf{v}) \end{bmatrix}. \quad (7)$$

Thus the overall system may now be written as a system of first-order ODEs as

$$\dot{\mathbf{x}}_{2n \times 1} = \begin{bmatrix} \dot{\mathbf{q}}_{n \times 1} \\ \dot{\mathbf{q}}_{n \times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{f}_1(\mathbf{q}, \mathbf{v}) \end{bmatrix} \quad (8)$$

which may then be integrated using standard codes. Note that, in principle, the index-reduced system Eq. (8) needs more initial conditions than the original system in Eqs. (1)–(3) to specify a unique solution. The main advantage is conceptual simplicity and simultaneous determination of the accelerations and the Lagrange multipliers by solving a *linear* system of equations. However, the constraint conditions may be progressively violated, especially in the presence of large step-sizes, leading to unacceptably large errors even for short duration simulations.

In the *compliance-based approaches*, the loop-closure constraints are *relaxed* and replaced using virtual springs and dampers. Using such virtual springs can be considered as a form of penalty formulation (see Section 5.1.4 of García de Jalón and Bayo [2]) which incorporates the constraint equations as a dynamical system penalized by a large factor. The Lagrange multipliers are *approximated* using a force-law (based the extent of the constraint violation and an assumed spring stiffness) and eliminated from the list of  $n + m$  unknowns leaving behind a system of  $2n$  first-order ODEs. While the sole initial drawback may appear to be restricted to the numerical ill-conditioning due to selection of large penalty factors, it is important to note that penalty approaches only *approximate* the true constraint forces and can create unanticipated problems (as discussed later).

Finally, the class of *projection-based approaches* seeks to take the constraint-reaction containing dynamical equations into the *orthogonal* and *tangent* subspaces of the vector space of the system's generalized velocities. Let  $\mathbf{S}(\mathbf{q})$  be an  $n \times (n - m)$ -dimensional full rank matrix whose column space is in the nullspace of  $\mathbf{A}(\mathbf{q})$ , i.e.  $\mathbf{A}(\mathbf{q})\mathbf{S}(\mathbf{q}) = \mathbf{0}$ . The orthogonal subspace is spanned by the so-called constraint vectors [forming the rows of the matrix  $\mathbf{A}(\mathbf{q})$ ] while the tangent subspace *complements* this orthogonal subspace in the overall generalized velocity vector space. All *feasible* dependent velocities  $\dot{\mathbf{q}}$  of a constrained multibody system necessarily belong to this tangent space, appropriately called the *space of feasible motions*. This space is spanned by the columns of  $\mathbf{S}(\mathbf{q})$  and is parameterized by an  $(n - m)$ -dimensional vector of independent velocities  $\mathbf{v}(t)$  yielding the expression for the feasible dependent velocities as  $\dot{\mathbf{q}} = \mathbf{S}(\mathbf{q})\mathbf{v}(t)$ .

It is very important to note two factors at this stage. *First*, the initial selection of the set of configuration coordinates plays an important role here. In particular, while it is always possible to create a Riemannian configuration space (and, consequently, the vector spaces for the generalized velocities) using the sets of relative coordinates, special care needs to be exercised when treating configuration spaces created with other sets of generalized coordinates [such as Cartesian coordinates of the bodies in  $SE(2)$  or  $SE(3)$ ]. This is because the generalized velocity space for Cartesian coordinates need not necessarily form a vector space, and, hence, the notion of orthogonal complement subspaces (which exists only in a Riemannian setting) needs to be examined carefully. For example, in Blajer et al. [25], the local task-space coordinates (consisting only translations) form a Riemannian space permitting the orthogonal decomposition to carry out. This is the other motivating factor for restricting ourselves to joint-based relative-coordinate descriptions of the configuration of the system. *Second*, a family of choices exists for the selection of projection between dependent and independent velocities (including the case where the set of independent velocities form a proper subset of the original set of dependent velocities) and each of such choice can give rise to a different  $\mathbf{S}(\mathbf{q})$ . See García de Jalón and Bayo [2] for a description of the many possibilities as well as the determination of the projection matrices determining the transformations between dependent and independent velocities.

However, once a projection is selected, the dynamic EOM can now be projected onto the instantaneous feasible motion directions to obtain the so-called constraint-reaction-free EOM. Pre-multiplying both sides of Eq. (2) by  $\mathbf{S}^T$  and noting that  $\mathbf{S}^T\mathbf{A}^T = \mathbf{0}$ , we get

$$\mathbf{S}^T\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{S}^T\mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u}). \quad (9)$$

Note that this is a system of  $n - m$  second-order differential equations in the  $n$  dependent accelerations. All further steps for the solution of this system of equations may be performed either in terms of *dependent* or *independent* coordinates, as discussed in Chapter 5 of García de Jalón and Bayo [2].

For example, the  $m$  acceleration-level constraints shown in Eq. (5) may be appended to this system resulting in a system of  $2n$  first-order ODEs in the state vector consisting of the  $n$  dependent velocities and  $n$  dependent accelerations. Note that while the notion of dependent and independent velocities is not explicitly considered, this is implicit in the selection of  $\mathbf{S}^T$ . Alternatively, by explicitly considering the same local projection used to determine the feasible motion directions, the constraint-reaction-free EOM may be expressed in terms of the

independent coordinates [26]. The final solution may be obtained: either by numerically integrating a system of  $2(n - m)$  first-order ODEs in the  $(n - m)$  independent velocities and  $(n - m)$  independent accelerations and solving the position problem at each step; or by numerically integrating a system of  $(2n - m)$  first-order ODEs in the  $n$  dependent velocities and  $(n - m)$  independent accelerations. The benefits accumulate from two sources. *First*, since the feasible motion directions are guaranteed to be tangent to the holonomic constraint manifold, with an adequately small step-size, the resulting integrated solution is guaranteed to maintain the constraints. *Second*, the integration of the reduced-order system of ODEs [with either  $(n - m)$  or  $(2n - m)$  states] also reduces the error due to numerical integration.

#### 4. The two approaches under consideration

##### 4.1. Method A: compliance-based method

In Wang et al. [7], the Lagrange multiplier  $\lambda$  in Eq. (2) is explicitly calculated as a restoring force provided by a virtual spring. This restoring force, which is proportional to the extent of constraint violation, can be expressed as  $\lambda_i = k_i c_i(\mathbf{q})$ , where  $k_i$  is the spring constant and  $c_i(\mathbf{q})$  is the constraint violation in the direction of the respective  $\lambda_i$ . By substituting the value of  $\lambda$  in Eq. (2), the final ODE system can be written as

$$\dot{\mathbf{x}}_{2n \times 1} = \begin{bmatrix} \dot{\mathbf{q}}_{n \times 1} \\ \ddot{\mathbf{q}}_{n \times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{v} \\ \mathbf{M}^{-1} \{ \mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u}) - \mathbf{A}^T(\mathbf{q}) [\mathbf{K}\mathbf{C}(\mathbf{q})] \} \end{bmatrix}, \quad (10)$$

where for  $m$  number of constraints  $\mathbf{K} = \text{diag}[k_i]$  is the  $m \times m$  constant “stiffness” matrix,  $\mathbf{C}(\mathbf{q})$  is the vector of  $m$ -dimensional constraint violations.

##### 4.2. Method B: projection-based method

The derivation process discussed by Yun and Sarkar [8] very closely resembles the work of Serna et al. [26], but with the addition of Baumgarte’s stabilization [23]. The formulation in terms of rheonomous constraints permits easy incorporation of Baumgarte’s stabilization, and additionally can be easily specialized for the scleronomous case. The holonomic constraints  $\mathbf{C}(\mathbf{q}) = \mathbf{0}$  are approximated by a first-order system of the form

$$\dot{\mathbf{C}}(\mathbf{q}) + \sigma \mathbf{C}(\mathbf{q}) = \mathbf{0}, \quad \sigma > 0, \quad (11)$$

where  $\sigma$  is the rate of convergence. The equilibrium condition for this first-order system is the constraint manifold  $\mathbf{C}(\mathbf{q}) = \mathbf{0}$ , and for any initial condition  $\mathbf{q}_0 = \mathbf{q}(0)$  that may not satisfy the holonomic constraint equation  $\mathbf{C}(\mathbf{q}_0) = \mathbf{0}$ , the above first-order equation guarantees exponential convergence of  $\mathbf{C}(\mathbf{q}(t))$  to  $\mathbf{0}$  as the time  $t$  progresses. The rate of convergence is determined by  $\sigma$ , which can be chosen based on specific application. By taking  $\mathbf{A}(\mathbf{q})$  as the Jacobian matrix of  $\mathbf{C}(\mathbf{q})$ , Eq. (11) can be written as

$$\mathbf{A}(\mathbf{q})\dot{\mathbf{q}} = -\sigma \mathbf{C}(\mathbf{q}) = \mathbf{a}(\mathbf{q}). \quad (12)$$

Then, the general solution of Eq. (12) is given by

$$\dot{\mathbf{q}} = \mathbf{v} = \mathbf{S}(\mathbf{q})\mathbf{v} + \eta(\mathbf{q}), \quad (13)$$

where  $\mathbf{S}(\mathbf{q})$  is an  $n \times (n - m)$ -dimensional full rank matrix whose column space is in the nullspace of  $\mathbf{A}(\mathbf{q})$ , i.e.  $\mathbf{A}(\mathbf{q})\mathbf{S}(\mathbf{q}) = \mathbf{0}$ ,  $\mathbf{v}$  is an  $(n - m)$ -dimensional vector of independent velocities,  $\eta(\mathbf{q})$  is the  $n$ -dimensional particular solution of Eq. (11).<sup>1</sup>

Differentiating once we get

$$\dot{\mathbf{v}} = \mathbf{S}(\mathbf{q})\dot{\mathbf{v}} + \dot{\mathbf{S}}(\mathbf{q})\mathbf{v} + \dot{\eta}(\mathbf{q}) = \mathbf{S}(\mathbf{q})\dot{\mathbf{v}} + \gamma(\mathbf{q}, \mathbf{v}). \quad (14)$$

Pre-multiplying both sides of Eq. (2) by  $\mathbf{S}^T$  and noting that  $\mathbf{S}^T \mathbf{A}^T = \mathbf{0}$ , we get

$$\mathbf{S}^T \mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{S}^T \mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u}). \quad (15)$$

<sup>1</sup> Note that this is where the contribution of  $\sigma$  in this formulation.

By substituting  $\dot{\mathbf{v}}$  from Eq. (14) into Eq. (15), and solving for  $\dot{\mathbf{v}}$  we get

$$\dot{\mathbf{v}} = -[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} [\mathbf{S}^T \mathbf{M} \boldsymbol{\gamma} - \mathbf{S}^T \mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u})]. \tag{16}$$

The resulting overall system of ODEs may be expressed in state-space form as

$$\dot{\mathbf{x}}_{(2n-m) \times 1} = \begin{bmatrix} \dot{\mathbf{q}}_{n \times 1} \\ \dot{\mathbf{v}}_{(n-m) \times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{S} \mathbf{v} + \boldsymbol{\eta} \\ -[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} [\mathbf{S}^T \mathbf{M} \boldsymbol{\gamma} - \mathbf{S}^T \mathbf{f}(\mathbf{q}, \mathbf{v}, t, \mathbf{u})] \end{bmatrix}. \tag{17}$$

While this method would work for any projection that is independent of the existing constraints, the particular adopted projection (wherein the independent velocities are selected as a proper subset of the original set of dependent velocities) yields a simple and robust formulation.

### 5. Distributed forward dynamics for a four-bar linkage

#### 5.1. Lagrangian modeling

The modeling process for the four-bar linkage considered here and the selected parameters are similar to Wang et al. [7], and they are summarized here for clarity. The four-bar linkage (Fig. 1) consists of three moving links (input, coupler link and output links) of lengths  $l_1$ ,  $l_2$ , and  $l_3$ , respectively, whose orientation with respect to the horizontal are denoted by the absolute angles of  $\theta$ ,  $\alpha$  and  $\phi$ . For  $i = 1, 2, 3$ , the masses of each moving link  $l_i$  is  $m_i$ , and the moment of inertia of the moving links about the axis through the center of the mass and perpendicular to the plane of its motion is  $I_i$ . The mass centers of each link are situated at a distance  $l_{ci}$  from the proximal joint of each link. The rest of the numerical parameters are summarized in Table 1.

The EOM for the overall system are derived by treating the four-bar linkage as being composed of two chains: (a) Chain A is the 2 DOF left chain (combination of link  $l_1$  and  $l_2$ ) and (b) Chain B is the 1 DOF right chain (only link  $l_3$ ), as shown in Fig. 1. The EOM of each (unconstrained) sub-chain are derived independently by the Lagrange’s method, where

$$\mathbf{M}_i(\mathbf{q}_i) \ddot{\mathbf{q}}_i + \mathbf{V}_i(\mathbf{q}_i, \dot{\mathbf{q}}_i) + \mathbf{G}_i(\mathbf{q}_i) = \mathbf{E}_i(\mathbf{q}_i) \mathbf{u}_i, \quad i = A, B. \tag{18}$$

The subscripts  $A$  and  $B$  and indicate the different quantities for Chain  $A$  and  $B$ , respectively. Hence:

For 2 DOF Chain  $A$ :

$$\mathbf{q}_A = \begin{bmatrix} \theta \\ \alpha \end{bmatrix}, \quad \mathbf{u}_A = \begin{bmatrix} \tau_\theta \\ \tau_\alpha \end{bmatrix},$$

$$\mathbf{M}_A = \begin{bmatrix} m_1 l_{c1}^2 + I_1 + m_2 l_1^2 & m_2 l_1 l_{c2} \cos(\alpha - \theta) \\ m_2 l_1 l_{c2} \cos(\alpha - \theta) & I_2 + m_2 l_{c2}^2 \end{bmatrix}, \quad \mathbf{V}_A = \begin{bmatrix} -m_2 l_1 l_{c2} \sin(\alpha - \theta) \dot{\alpha}^2 \\ m_2 l_1 l_{c2} \sin(\alpha - \theta) \dot{\theta}^2 \end{bmatrix},$$

$$\mathbf{G}_A = \begin{bmatrix} m_1 g l_{c1} \cos \theta + m_2 g l_1 \cos \theta \\ m_2 g l_{c2} \cos \alpha \end{bmatrix}, \quad \mathbf{E}_A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Table 1  
Relevant numerical parameters for the four-bar linkage under consideration (given by Wang et al. [7])

Link lengths	$l_0 = 3.0$ m, $l_1 = 1.0$ m, $l_2 = 4.0$ m, $l_3 = 2.5$ m
Distance of mass centers	$l_{ci} = l_i/2$ , $i = 1, 2, 3$
Link masses	$m_i = 1.0$ kg, $i = 1, 2, 3$
Moment of inertias	$I_i = m_i l_i^2/12$ , $i = 1, 2, 3$
Initial configuration $\mathbf{q}(0)$	$\theta(0) = 1.5708$ rad, $\alpha(0) = 0.3533$ rad, $\phi(0) = 1.2649$ rad
Initial velocity $\dot{\mathbf{q}}(0)$	$\dot{\theta}(0) = \dot{\alpha}(0) = \dot{\phi}(0) = 0$ rad/s
Torque input $\mathbf{u}$	$\tau_\theta = 6.0$ N/m, $\tau_\alpha = \tau_\phi = 0.0$ N/m
Gravitational acceleration	$g = 9.81$ m/s <sup>2</sup>

For 1 DOF Chain B:

$$\begin{aligned} \mathbf{q}_B &= \phi, \quad \mathbf{u}_B = \tau_\phi, \\ \mathbf{M}_B &= m_3 l_{c3}^2 + I_3, \quad \mathbf{V}_B = 0, \\ \mathbf{G}_B &= m_3 g l_{c3} \cos \phi, \quad \mathbf{E}_B = 1. \end{aligned}$$

The constraint equations are then obtained from the requirement that Chain *A* and *B* to stay connected at the cut joint, which can be expressed in matrix form as

$$\mathbf{C}(\mathbf{q}) = \begin{bmatrix} c_1(\mathbf{q}) \\ c_2(\mathbf{q}) \end{bmatrix} = \begin{bmatrix} -l_1 \cos \theta - l_2 \cos \alpha + l_0 + l_3 \cos \phi \\ -l_1 \sin \theta - l_2 \sin \alpha + l_3 \sin \phi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad (19)$$

We can obtain the Jacobian matrix of the constraint  $\mathbf{C}(\mathbf{q})$  as

$$\mathbf{A}(\mathbf{q}) = \frac{\partial \mathbf{C}(\mathbf{q})}{\partial \mathbf{q}} = \begin{bmatrix} l_1 \sin \theta & l_2 \sin \alpha & -l_3 \sin \phi \\ -l_1 \cos \theta & -l_2 \cos \alpha & l_3 \cos \phi \end{bmatrix}. \quad (20)$$

The EOM of the combined overall system can finally be modularly constructed as an index-3 DAEs as

$$\begin{aligned} \dot{\mathbf{q}} &= \mathbf{v}, \\ \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) &= \mathbf{E}(\mathbf{q})\mathbf{u} - \mathbf{A}^T(\mathbf{q})\boldsymbol{\lambda}, \\ \mathbf{C}(\mathbf{q}) &= \mathbf{0}, \end{aligned} \quad (21)$$

where

$$\begin{aligned} \mathbf{q} &= \begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_B \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_A \\ \mathbf{u}_B \end{bmatrix}, \\ \mathbf{M}(\mathbf{q}) &= \begin{bmatrix} \mathbf{M}_A & \mathbf{0}_{2 \times 1} \\ \mathbf{0}_{1 \times 2} & \mathbf{M}_B \end{bmatrix}, \quad \mathbf{V}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} \mathbf{V}_A \\ \mathbf{V}_B \end{bmatrix}, \quad \mathbf{G}(\mathbf{q}) = \begin{bmatrix} \mathbf{G}_A \\ \mathbf{G}_B \end{bmatrix}, \quad \mathbf{E}(\mathbf{q}) = \begin{bmatrix} \mathbf{E}_A & \mathbf{0}_{2 \times 1} \\ \mathbf{0}_{1 \times 2} & \mathbf{E}_B \end{bmatrix}. \end{aligned}$$

In particular, the constraints are incorporated by differentiating the position-level (holonomic) constraints to obtain the velocity-level constraint Jacobian matrix  $\mathbf{A}$  and adjoined with Lagrange multipliers to obtain the constrained dynamics. The rows of the constraint Jacobian matrix or correspondingly the columns of  $\mathbf{A}^T$  span the feasible motion directions, while the Lagrange multipliers (grouped in vector  $\boldsymbol{\lambda}$ ), correspond to the magnitude of the required constraint correction forces.

Without the loss of generality, the above procedures can easily be extended to handle a more general cases, for instance, composite systems with more subsystems or more number of states. This emphasizes the modular development of the reuse of the (existing) smaller (open-chain) components to construct a larger (closed-loop) system.

## 5.2. Compliance-based method (Method A)

In compliance-based method (Method A), let  $\mathbf{q}_A$  and  $\mathbf{q}_B$  has, respectively,  $a$  and  $b$  number of state variables, the distributed model of a two-subsystem case (without the loss of generality) can be obtained in the state-space form from Eq. (10) as

$$\begin{aligned} [\dot{\mathbf{x}}_A]_{2a \times 1} &= \begin{bmatrix} \dot{\mathbf{q}}_A \\ \ddot{\mathbf{q}}_A \end{bmatrix} = \begin{bmatrix} \mathbf{v}_A \\ \mathbf{M}_A^{-1} \{ \mathbf{E}_A \mathbf{u}_A - \mathbf{V}_A - \mathbf{G}_A - \mathbf{A}_A^T [\mathbf{K}\mathbf{C}] \} \end{bmatrix}, \\ [\dot{\mathbf{x}}_B]_{2b \times 1} &= \begin{bmatrix} \dot{\mathbf{q}}_B \\ \ddot{\mathbf{q}}_B \end{bmatrix} = \begin{bmatrix} \mathbf{v}_B \\ \mathbf{M}_B^{-1} \{ \mathbf{E}_B \mathbf{u}_B - \mathbf{V}_B - \mathbf{G}_B - \mathbf{A}_B^T [\mathbf{K}\mathbf{C}] \} \end{bmatrix}, \end{aligned} \quad (22)$$

where  $\mathbf{A}_{m \times (a+b)} = [ [\mathbf{A}_A]_{m \times a} \quad [\mathbf{A}_B]_{m \times b} ]$  and  $\mathbf{K}_{m \times m} = \text{diag}[k_i]$  for  $m$  number of constraints. The two subsystems can be simulated in a distributed manner if at every time step: (a) either the information pertaining to  $\mathbf{C}$  is made available explicitly; or (b) computed by exchanging state information between the two subsystems. This

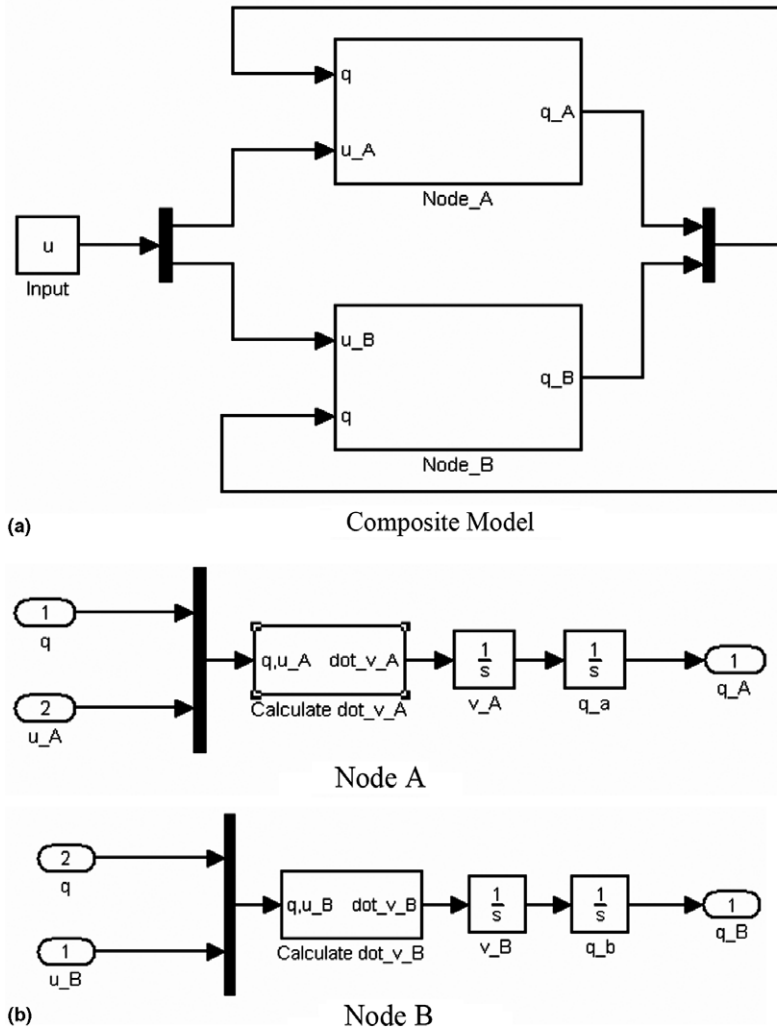


Fig. 2. Distributed computation model of a four-bar linkage for compliance-based method (Method A).

suggests a way to distribute the computational load between two processors, as shown graphically in Fig. 2, wherein each independent sub-part can now be numerically integrated on a different processor. The sole coupling between the two sub-parts is due to the Lagrange multipliers, which are now explicitly calculated using the virtual spring. While the above formulation is shown for a “two-part system”, the process can be generalized easily for an “*n*-part” system.

In the four-bar linkage case under consideration, the elements of the “stiffness matrix”  $\mathbf{K} = \text{diag}[k_i]$  is taken as  $k_i = k, i = 1, 2$ , i.e. equal stiffness in every constraint violation, and the partitioned Jacobian matrices are:

$$\mathbf{A}_A = \begin{bmatrix} l_1 \sin \theta & l_2 \sin \alpha \\ -l_1 \cos \theta & -l_2 \cos \alpha \end{bmatrix}, \quad \mathbf{A}_B = \begin{bmatrix} -l_3 \sin \phi \\ l_3 \cos \phi \end{bmatrix}. \tag{23}$$

### 5.3. Projection-based method (Method B)

In projection-based method (Method B), we first manipulate Eq. (17) to write it in a distributed two-subsystem form. We first distribute the equation  $\mathbf{q} = \mathbf{S}\mathbf{v} + \boldsymbol{\eta}$  into

$$\begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_B \end{bmatrix} = \begin{bmatrix} [\mathbf{S}_A]_{a \times (n-m)} \\ [\mathbf{S}_B]_{b \times (n-m)} \end{bmatrix} v_{(n-m) \times 1} + \begin{bmatrix} [\eta_A]_{a \times 1} \\ [\eta_B]_{b \times 1} \end{bmatrix}, \quad (24)$$

or

$$\begin{aligned} \dot{\mathbf{q}}_A &= \mathbf{S}_A v + \eta_A, \\ \dot{\mathbf{q}}_B &= \mathbf{S}_B v + \eta_B. \end{aligned} \quad (25)$$

We then distribute the equation  $\dot{v} = -[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} [\mathbf{S}^T (\mathbf{M} \gamma + \mathbf{V} + \mathbf{G} - \mathbf{E} \mathbf{u})]$  into

$$\begin{aligned} \dot{v} &= -[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} \left[ [\mathbf{S}_A^T \quad \mathbf{S}_B^T] \left( \begin{bmatrix} \mathbf{M}_A & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_B \end{bmatrix} \begin{bmatrix} \gamma_A \\ \gamma_B \end{bmatrix} + \begin{bmatrix} \mathbf{V}_A \\ \mathbf{V}_B \end{bmatrix} + \begin{bmatrix} \mathbf{G}_A \\ \mathbf{G}_B \end{bmatrix} - \begin{bmatrix} \mathbf{E}_A & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_B \end{bmatrix} \begin{bmatrix} \mathbf{u}_A \\ \mathbf{u}_B \end{bmatrix} \right) \right] \\ &= \underbrace{-[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} [\mathbf{S}_A^T (\mathbf{M}_A \gamma_A + \mathbf{V}_A + \mathbf{G}_A - \mathbf{E}_A \mathbf{u}_A)]}_{\dot{v}_A} - \underbrace{[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} [\mathbf{S}_B^T (\mathbf{M}_B \gamma_B + \mathbf{V}_B + \mathbf{G}_B - \mathbf{E}_B \mathbf{u}_B)]}_{\dot{v}_B}. \end{aligned} \quad (26)$$

Hence, the full distributed projected dynamic equations in the state-space form are

$$\begin{aligned} [\dot{\mathbf{x}}_A]_{(a+n-m) \times 1} &= \begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{v}_A \end{bmatrix} = \begin{bmatrix} \mathbf{S}_A v + \eta_A \\ -[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} [\mathbf{S}_A^T (\mathbf{M}_A \gamma_A + \mathbf{V}_A + \mathbf{G}_A - \mathbf{E}_A \mathbf{u}_A)] \end{bmatrix}, \\ [\dot{\mathbf{x}}_B]_{(b+n-m) \times 1} &= \begin{bmatrix} \dot{\mathbf{q}}_B \\ \dot{v}_B \end{bmatrix} = \begin{bmatrix} \mathbf{S}_B v + \eta_B \\ -[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1} [\mathbf{S}_B^T (\mathbf{M}_B \gamma_B + \mathbf{V}_B + \mathbf{G}_B - \mathbf{E}_B \mathbf{u}_B)] \end{bmatrix}. \end{aligned} \quad (27)$$

By examining Eq. (27), we note that the overall system can be evaluated in a distributed manner if states  $\mathbf{q}_i$  and  $v_i$ ,  $i = A, B$ , are made available. This suggests a way to split the calculation of the dynamic equations, as depicted in Fig. 3. Each independent sub-part can now be numerically integrated on a different processor thereby permitting the distribution of the load. At each time-instant, the complete state of the system needs to be exchanged between the sub-parts. The coupling between the various sub-parts is due to the existence of the term  $[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1}$ . In the arrangement shown in Fig. 3, this matrix inverse needs to be computed on each and every processor (although we note that the explicit calculation of the inverse is typically avoided by using an optimal equation solver). Alternatively, state information from the slave processors could be collected by a master processor at each time instant, the term  $[\mathbf{S}^T \mathbf{M} \mathbf{S}]^{-1}$  could be computed once, and the quantity  $v$  can be summed from the contribution of  $v_A$  and  $v_B$  from each processor, and the results subsequently propagated back to the slave processors where the actual numerical integration is performed. Similarly, the above procedures can be generalized to the case with more number of subsystems.

## 6. Implementation

### 6.1. Computation environment

All computations were implemented using RT-LAB,<sup>2</sup> a commercial-off-the-shelf system for distributing the computation and a series of tests were performed to evaluate the performance of both methods of simulation. In each case, the model was converted into a suitable state-space form for use in conjunction with various numerical time-stepping algorithms, henceforth called ODE Suite [27] to generate the forward-dynamics simulation results. MATLAB/Simulink<sup>3</sup> offered convenient access an extensive set of numerical time-stepping algorithms in a user friendly format and hence served as the primary implementation tool. Together with a toolbox called Real-Time Workshop,<sup>4</sup> the high-level block-diagrammatic algorithm was used to generate C-code and compiled into a real-time executable. Finally, in conjunction with RT-LAB, the executable can be deployed in a distributed manner on PCs connected together by either ethernet or firewire interconnects to form a computational cluster. In particular, RT-LAB provided the necessary glue to run the models in a distributed manner, while meeting strict deterministic performance and synchronization requirements.

<sup>2</sup> RT-LAB is the trademark of Opal-RT Technologies, Inc.

<sup>3</sup> MATLAB and Simulink are the trademarks of The MathWorks, Inc.

<sup>4</sup> Real-Time Workshop is the trademark of The MathWorks, Inc.

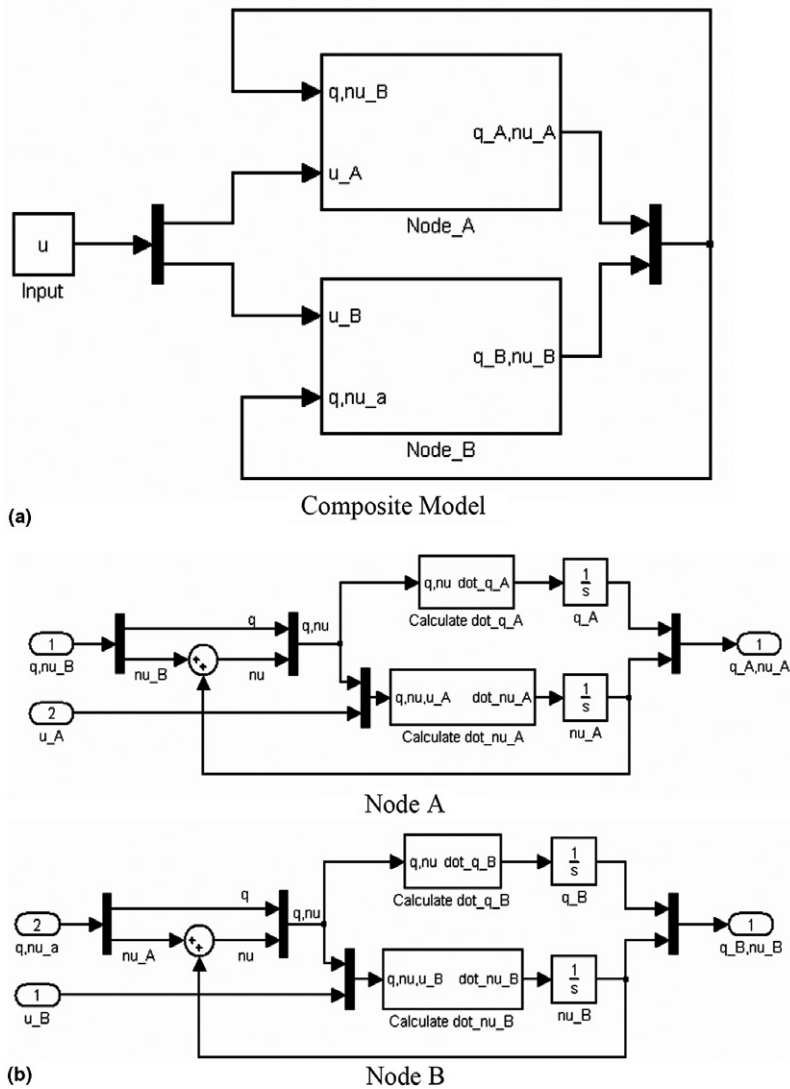


Fig. 3. Distributed computation model of a four-bar linkage for projection-based method (Method B).

6.2. Evaluation criteria

We focus our attention on the following principal factors in the comparative studies (the results of when will be presented in greater detail in Section 7).

6.2.1. Absolute vs. relative comparisons

Absolute comparisons of the simulated results with respect to the benchmark solution, e.g. the joint-angle time-histories across different formulations, would be the most obvious (and pertinent) approach. However, benchmark analytic solutions tend to be difficult/impossible to compute for more complex mechanical systems creating a need for the alternate class of relative comparisons. A good example of such relative-comparisons would be to measure the extent of constraint violation within the system, without explicitly comparing with external/benchmark results. When applicable, one could additionally examine the number of iterations (representative computational cost) required to simulate a fixed simulation duration for each method.

### 6.2.2. Role of time-stepping schemes

Two classes of numerical integration schemes were employed for the simulation: (a) adaptive time-stepping, where an estimate of the integration error is made, and the time-step is adapted to keep this error below a specific tolerance level; and (b) fixed time-stepping schemes, where the user can specify the size of the time step.

Fixed-time stepping schemes are typically employed for real-time parallel/distributed processing from the viewpoint of synchronization of information between multiple processors, which should occur at the end of each time-step. Adaptive time-stepping schemes are typically avoided in real-time and/or distributed implementations owing to the time-uncertainties entailed in the iterative convergence to the desired tolerances. Nonetheless, testing the models with adaptive time-stepping methods can give insight into the overall characteristics of a formulation, including formulation stiffness and computational complexity of implementation (as measured by the number of iterations or the total time taken to simulate a fixed simulation time). Hence, in what follows, all results are computed using a distributed algorithm formulation of Section 5 for the four-bar linkage. In the fixed time-stepping cases, the distributed systems were simulated on multiple real-time processors synchronizing information at the deterministic time steps. The adaptive time-stepping cases were simulated in non-real-time mode with all distributed subsystems implemented on a single processor to eliminate the need for explicit synchronization.

### 6.2.3. Role of parameters

Each of methods developed in Section 5 has one independent parameter that could potentially affect the performance of the method – the stiffness of the virtual spring,  $\mathbf{K} = \text{diag}[k_i]$ ,  $k = k_i$ ,  $i = 1, 2$ , in the compliance-based approach, and the Baumgarte convergence factor  $\sigma$  in the case of the projection-based method. Typically, we vary the value of  $k$  between  $10^2$  N/m and  $10^6$  N/m while  $\sigma$  will be varied between  $10$  s<sup>-1</sup> and  $60$  s<sup>-1</sup>. In the plots of results, we examine the role of this independent parameter on the constraint error; and the effect of the independent parameter on the number of time-steps required to simulate a fixed simulation duration.

## 7. Results

### 7.1. Absolute comparisons with the Benchmark solution

The benchmark solution was created by direct elimination of the surplus variables (in our case  $\alpha$  and  $\phi$ ) and their time-derivatives from the Lagrangian, which can now be expressed completely in terms of  $\theta$  (and its time-derivatives). Lagrange's method can then be applied to generate the governing ODEs in terms of the (minimal) generalized coordinates ( $\theta$ ). The resulting Lagrange's equations of the second kind can then be integrated using standard ODE methods without worrying about the stability issues. However, it is worth noting, that obtaining the governing ODE can be extremely complicated even in the simple four-bar case and may be impossible for more complex cases [21].

The numerical parameters for the four-bar linkage shown in Table 1 were used to generate the benchmark solution. The system was simulated using ODE5 scheme with a fixed time-step of  $1e-4$  s for a 10 s duration and the position time-histories of the three-joints are plotted in Fig. 4. We see that the constant torque input causes acceleration within the system which is reflected in the graphs.

Figs. 5 and 6 show the comparison of the compliance-based method (Method A) and the projection-based method (Method B) with respect to the benchmark solution, respectively. In each case, the subplots depict the time-histories of the logarithm of the errors in the three joint-angles. In Fig. 5, it is readily evident that the error in all three angles reduces as the value of  $k$  is increased. However, the magnitude of this error cannot be reduced enough ( $<1e-3$  s) unless extremely high values of  $k$  ( $>10^6$  N/m) are chosen. This has the unfortunate effect of creating a stiff-numerical system that cannot be simulated using moderately small fixed time-step. For example, the time-step needs to be reduced from  $1e-3$  s to  $1e-4$  s in order to successfully simulate the system with a  $k$  value of  $1e-6$  N/m.

On the other hand, the projection-based method (Method B) performs very well when compared to the benchmark solution. The absolute errors remain well below  $1e-3$  m for almost every value of  $\sigma$  as seen in Fig. 6. However, there is clearly some benefit to increasing the value for  $\sigma$  not only from the perspective of keeping the absolute errors small but also from the perspective of faster from disturbance recovery.

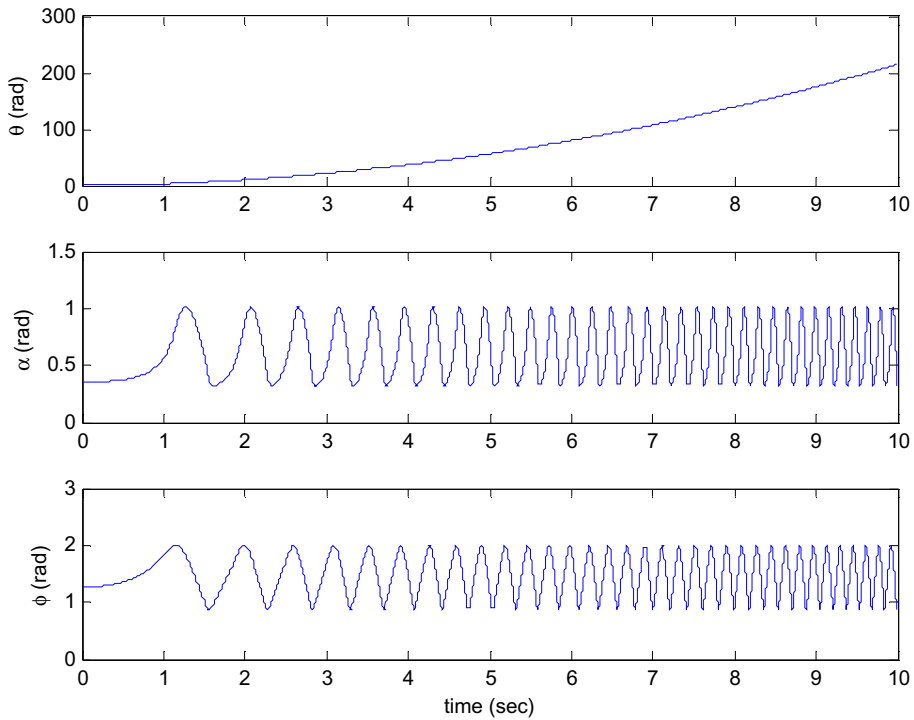


Fig. 4. Benchmark solution of the time evolution of the three angles ( $\theta$ ,  $\alpha$ , and  $\phi$ ) for the dynamic of the four-bar linkage under consideration in 10 s.

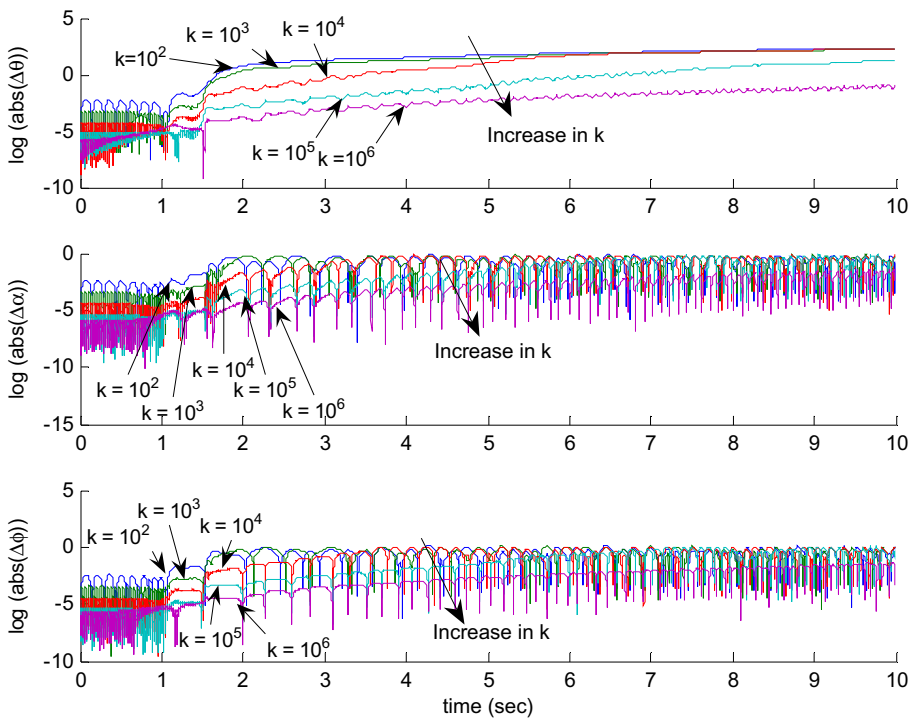


Fig. 5. Absolute error of the time evolution of the three angles ( $\theta$ ,  $\alpha$ , and  $\phi$ ) using the compliance-based method (Method A) with respect to the benchmark solution.

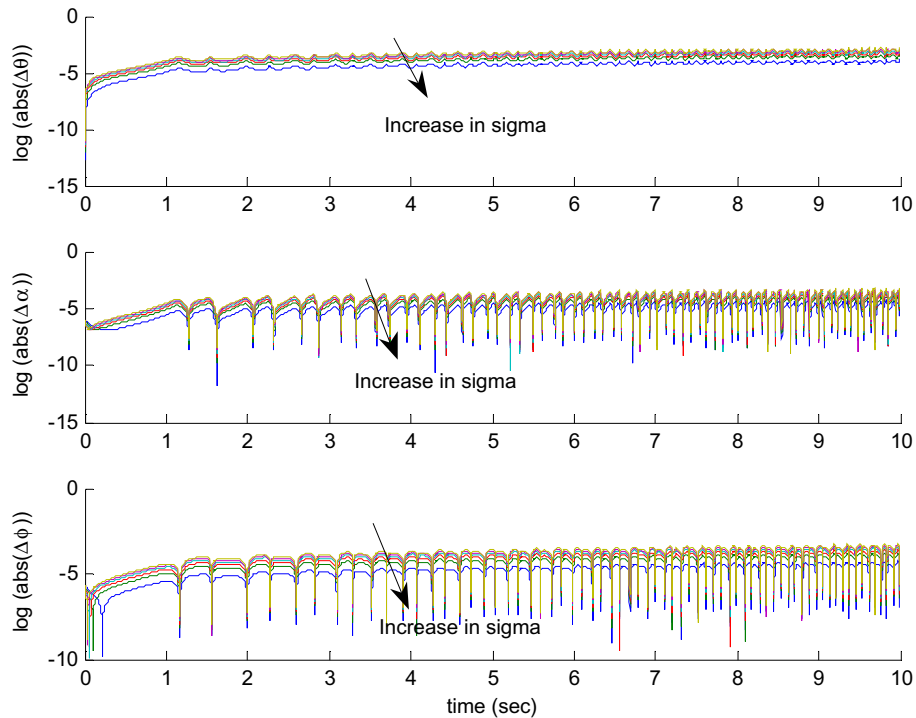


Fig. 6. Absolute error of the time evolution of the three angles ( $\theta$ ,  $\alpha$ , and  $\phi$ ) using the projection-based method (Method B) with respect to the benchmark solution.

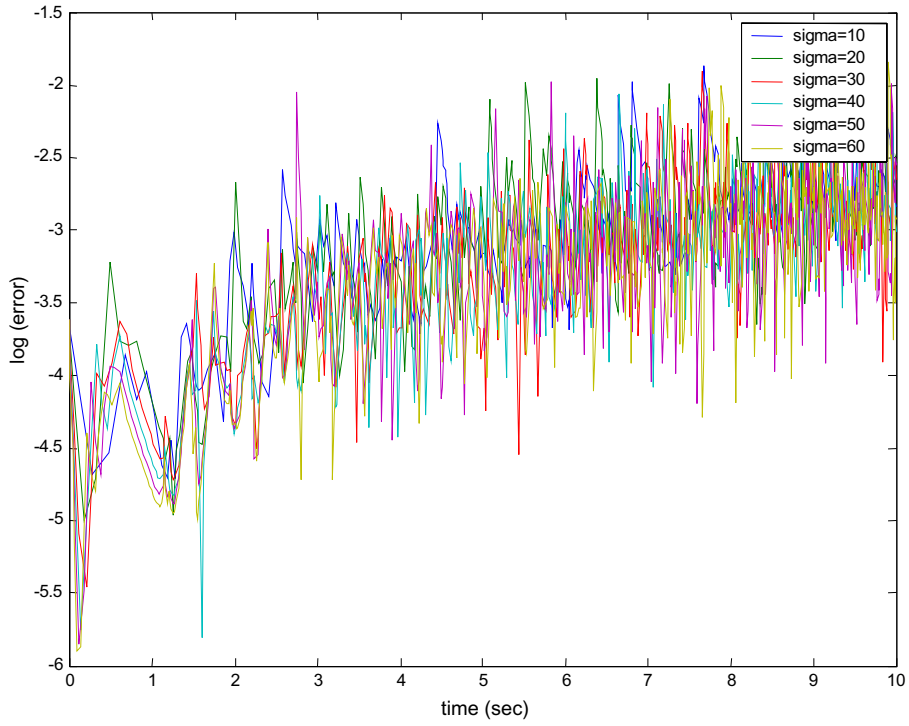
### 7.2. Relative comparisons with adaptive time-stepping

In this scenario, an adaptive time-stepping scheme is used for the simulation with the relative tolerance varied in the orders of magnitude from  $1e-3$  s to  $1e-6$  s. In the results, we examine the role of the independent parameter on the constraint error; and the effect of the independent parameter on the number of time-steps required for a 10 s simulation.

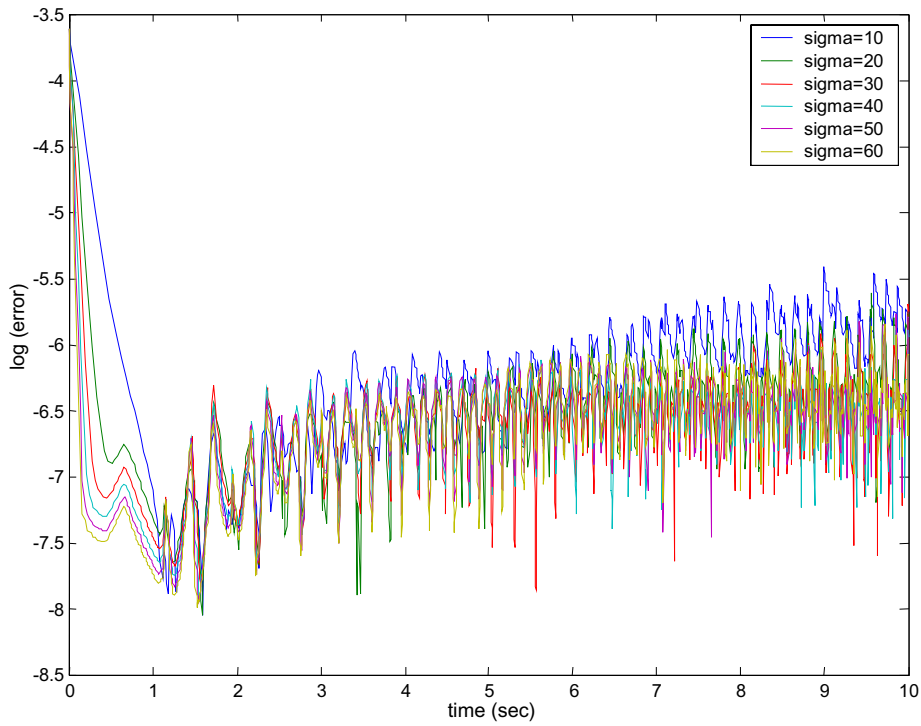
Fig. 7 depicts the constraint error vs. time for implementing the projection-based approach with different values of convergence factor  $\sigma$  for two sets of maximum tolerances ( $1e-3$  s and  $1e-6$  s) using the ODE45 Dormand-Prince adaptive time-step solver. As can be seen from the results, the convergence factor  $\sigma$  does not significantly affect the constraint errors and that the resulting relative constraint errors are representative of the selected relative tolerance settings.

Fig. 8 shows the resulting constraint errors that result from implementing the compliance-based approach with different values of the spring constant  $k$  (varied in orders of magnitude from  $10^1$  N/m to  $10^6$  N/m). These were carried out for various settings of the relative tolerances of the adaptive time step solver but only shown here for two cases ( $1e-3$  s and  $1e-6$  s) in Fig. 8(a) and (b) respectively. Qualitatively speaking, we note that decreasing the relative tolerance by three orders of magnitude did not affect the relative constraint errors, as can be seen by comparing Fig. 8(a) and (b). However, within each graph, we note that when the spring stiffness is increased, the constraint error drops commensurately but never attaining the performance levels of the projection-based method in Fig. 7.

To further compare the two methods in the adaptive time-stepping scenario, we examine the computational time required as measured in the number of discrete time-steps required to simulate 10 s. Fig. 9 depicts the number of discrete time-steps of the adaptive time-stepping algorithm required to keep the constraint errors within the specified tolerances for different values of the parameters in each of the two methods. Two important features are to be noted: *First*, for increasing values of the parameter ( $\sigma$  or  $k$ ) both methods show distinct increases in the number of time-steps required to maintain a desired tolerance. However, while the rate of increase is linear in the case of the projection-based approach but tends to increase exponentially with



(a)



(b)

Fig. 7. Constraint error in the projection-based method (Method B) using adaptive time-stepping for two sets of relative tolerances: (a)  $1e-3$  s and (b)  $1e-6$  s.

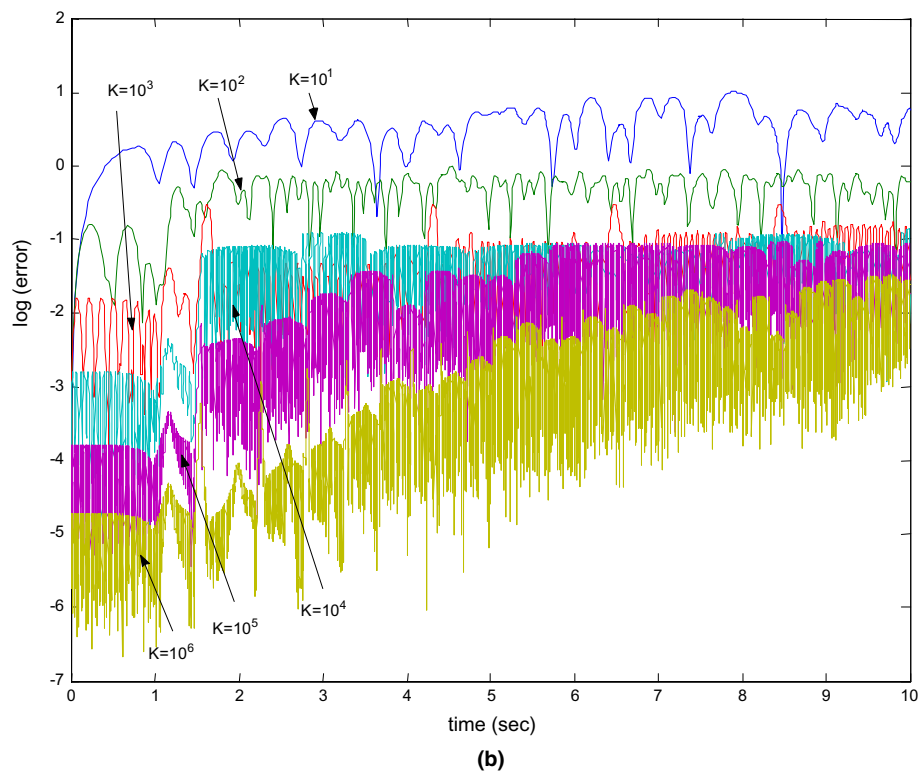
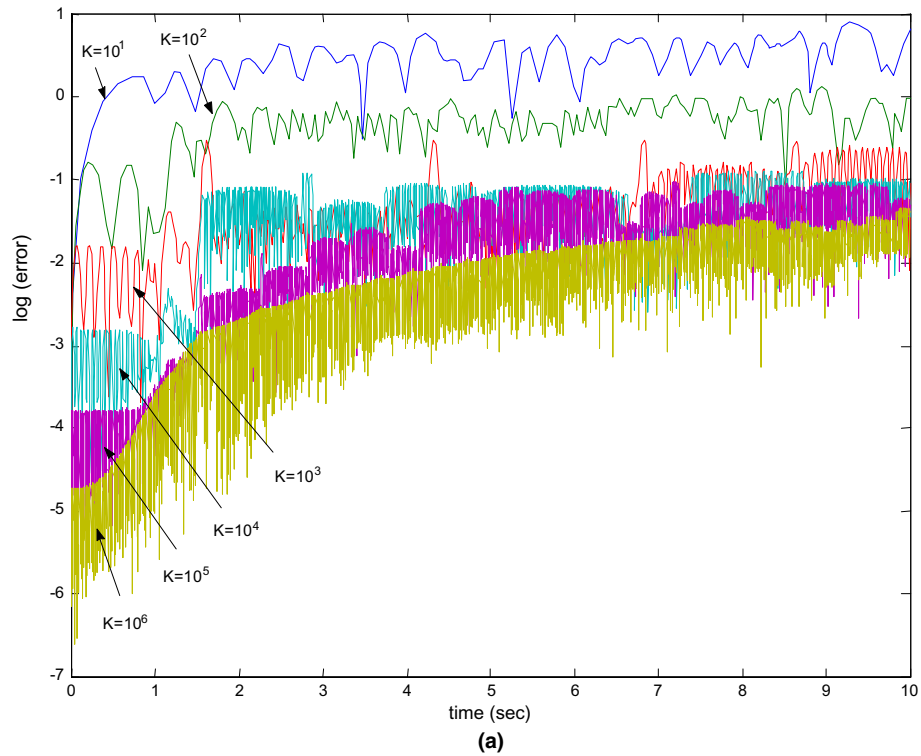


Fig. 8. Constraint error in the compliance-based method (Method A) using adaptive time-stepping for two sets of relative tolerances: (a)  $1e-3$  s and (b)  $1e-6$  s.

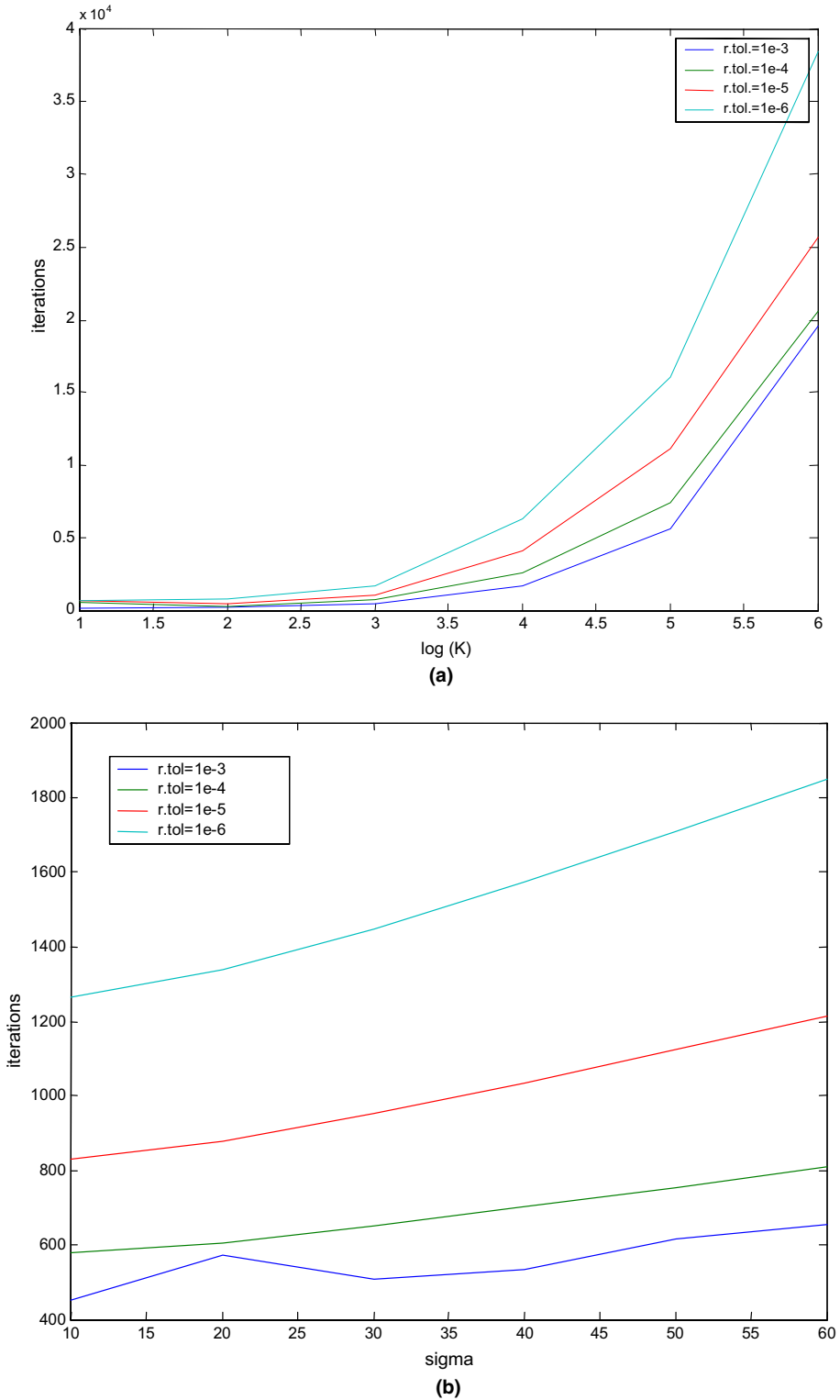


Fig. 9. Number of iterations vs. independent parameter for numerical integration of a 10 s duration with variable time-step for: (a) compliance-based (Method A) and (b) projection-based method (Method B).

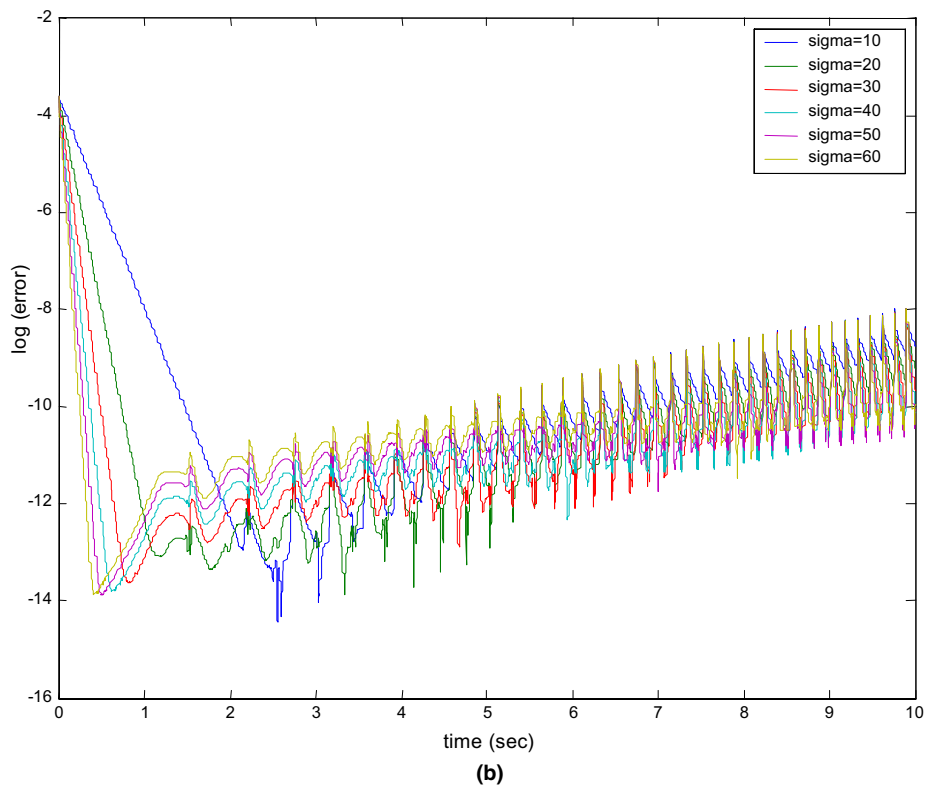
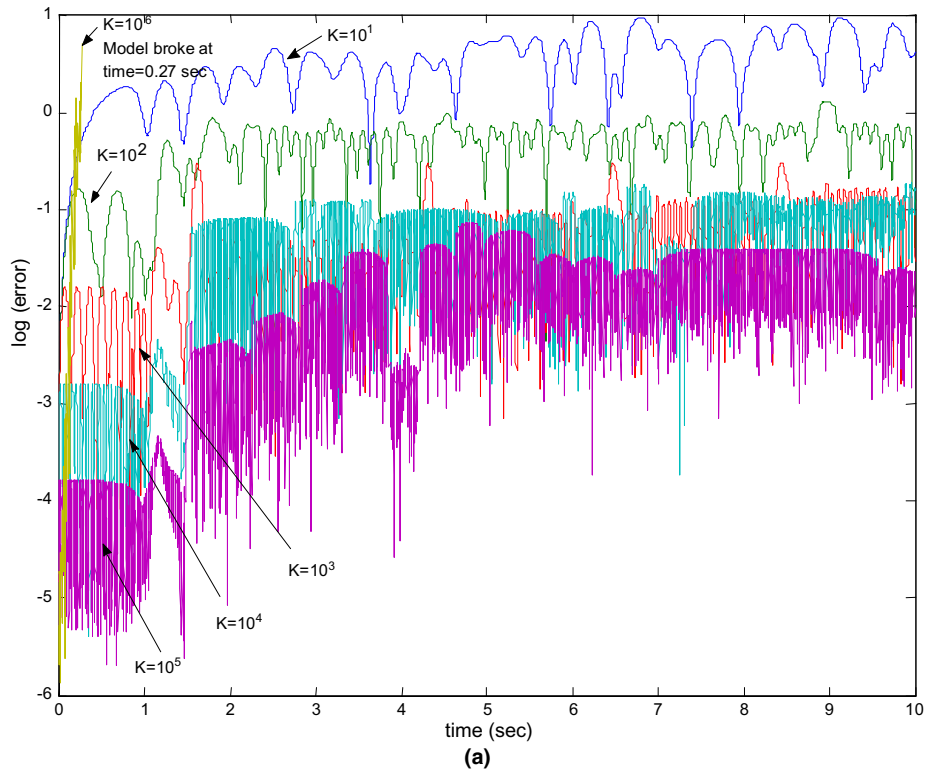


Fig. 10. Constraint error for numerical integration with fixed time-step ( $1e-3$  s) for: (a) compliance-based (Method A) and (b) projection-based method (Method B).

increasing values of the spring stiffness in the compliance-based approach. *Secondly*, for a fixed value of the independent parameter, an exponential increase in the number of iterations can be seen with decreasing values of the relative tolerance in the compliance-based approach.

### 7.3. Relative comparisons with fixed time-stepping

This scenario is more suited for real-time simulation and hardware-in-the-loop type simulation applications where deterministic time-stepping of the simulation is desirable. Here, in addition to the actual algorithm from the ODE Suite, the designer plays an important role in selecting the step size of fixed time step. This selection has critical implications in that an order of magnitude reduction in step-size increases the number of iterations by the same order of magnitude.

A number of simulations with different values for fixed time-steps (ranging from  $1e-3$  s to  $1e-6$  s) were performed. However, only the resulting constraint errors from running the two simulation (compliance- and projection-based) schemes for a fixed step size of  $1e-3$  s are shown in Fig. 10. In Fig. 10(b), we note that the selection of the value of the independent parameter  $\sigma$  only plays a minor role since regardless of the selected value the constraint error remains near about  $1e-12$  m. In contrast, in Fig. 10 we see that for small values of the spring stiffness, considerable constraint error results which decreases as  $k$  is increased. While this constraint error reduces to the order of  $1e-3$  m as the spring stiffness is increased to  $10^6$  N/m, this improvement is not comparable to the error magnitudes observed for the projection-based method.

## 8. Discussion

Schiehlen et al. [28] performed a similar comparison by numerical simulation of a non-minimal description of a mechanical system obtained by coupling two or more minimal local subsystems with explicitly or implicitly stated holonomic constraints, with a variety of examples. They examined the effects of numerical integration with two similar classes of solution approaches, i.e. a so-called *force coupling approach*, which approximate the constraint forces by force laws proportional to the violation of coupling conditions; and the more traditional *differential algebraic approach*, using various integrators using projections on the position and velocity constraint manifold for stabilization.

As they note in their discussion, force coupled systems are especially sensitive to initial violations of the assembly conditions which may cause the coupling forces to create additional disturbances that excite the system. Further, such force-coupled systems can cause the dynamic behavior to be severely changed and even resulting in instability of the whole system. In contrast, they note that DAE description combined with integrators that use projection methods offer an efficient way of coupling local subsystems on the level of their EOM or for closed kinematic loops. Our results developed in the context of distributed computation of similar types of constrained dynamical systems match these observations.

In addition to the “natural” decoupling noted for the compliance-based formulation, several other advantages have been reported in the literature. These include the fact that appearance and disappearance of the constraints can be treated automatically and that the method performs robustly near kinematic singularity positions [2]. However, as noted by one of the reviewers, the Lagrange multipliers only form a part of the complete picture regarding the constraint forces. They represent the magnitude-type contribution, while the other (and perhaps the most important part) is the directional information that is embedded in the constraint Jacobian matrix. The imperfect approximation of the Lagrange multipliers coupled with the (artificial) relaxation of the constraints can over time lead to alternate configurations, thereby indirectly affecting the directions of constraint vectors. Hence, not withstanding the small magnitudes of the constraint violations, the incorrect projection of the Lagrange multipliers would: yield seemingly correct but non-physical results; and additionally act as a continuous source of disturbance.

## 9. Summary

In this paper, we examined aspects of both the development and performance-evaluation of two alternate methods for distributed forward dynamics simulations of constrained mechanical systems exemplified by the

four-bar linkage. Similar situations may also be encountered in other arenas where the governing equations take the form of sets of ODEs coupled together by algebraic constraints and solution of the combined system of DAEs needs to be found. We exploited the natural spatial parallelism of closed-chain manipulators initially for the modular development of overall dynamics, and subsequently for the distributed numerical simulation of the dynamics. Traditionally, the numerical simulation problem has been treated as being composed of two more-or-less independent stages: an initial algorithm development stage followed by a numerical integration stage. While the coupling of the two stages may not be as relevant for unconstrained mechanical systems, it plays a significant role for the simulation of constrained mechanical systems. Our preliminary results examined in the context of four-bar linkage indicated that a global unified view of the evaluation of the computational complexity of the simulation is advisable. Specifically, at an algorithmic development level, the compliance-based approach provides a seemingly natural method for decoupling and distributing the computation and has roughly one-third of the computational complexity of the projection-based approach. However, while the projection-based approach is computationally more expensive per time-step, fewer time-steps are required to maintain a desired relative constraint error tolerance leading to faster simulations.

### Acknowledgements

We would like to acknowledge the support of the Natural Sciences and Engineering Research Council (NSERC) and the support of the National Science Foundation CAREER Award (IIS-0347653) for this research effort. We would also like to acknowledge the anonymous reviewers for their constructive comments.

### References

- [1] U. Ascher, L. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, PA, 1998.
- [2] J. García de Jalón, E. Bayo, *Kinematic and Dynamic Simulation of Multibody Systems: The Real-Time Challenge*, Springer-Verlag, New York, NY, 1994.
- [3] E.J. Haug, *Computer-Aided Kinematics and Dynamics of Mechanical Systems*, Allyn & Bacon, Needham Heights, MA, 1989.
- [4] W. Schiehlen, *Multibody Systems Handbook*, Springer-Verlag, Berlin, 1990.
- [5] A.A. Shabana, *Dynamics of Multibody Systems*, third ed., Cambridge University Press, Cambridge, NY, 2005.
- [6] S. McMillan, *Computational dynamics for robotic systems on land and under water*, PhD Thesis, Department of Electrical Engineering, The Ohio State University, Columbus, OH, 1994.
- [7] J. Wang, C.M. Gosselin, L. Cheng, Modeling and simulation of robotic systems with closed kinematic chains using the virtual spring approach, *Multibody System Dynamics* 7 (2) (2002) 145–170.
- [8] X. Yun, N. Sarkar, Unified formulation of robotic systems with holonomic and nonholonomic constraints, *IEEE Transactions on Robotics and Automation* 14 (4) (1998) 640–650.
- [9] D. Henrich, T. Höniger, Parallel processing approaches in robotics, in: *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE'97)*, Guimarães, Portugal, 1997.
- [10] V. Chaudhary, J.K. Aggarwal, Parallelism in computer vision: a review, in: V. Kumar, P.S. Gopalakrishnan, L.N. Kanal (Eds.), *Parallel Algorithms for Machine Intelligence and Vision*, Springer-Verlag, New York, NY, 1990, pp. 271–309.
- [11] A. Fijany, A. Bejczy, *Parallel Computation Systems for Robotics: Algorithms and Architectures*, World Scientific, Singapore, River Edge, NJ, 1992.
- [12] A.Y. Zomaya, *Modelling and Simulation of Robot Manipulators: A Parallel Processing Approach*, World Scientific, River Edge, NJ, 1993.
- [13] C.S.G. Lee, P.R. Chang, Efficient parallel algorithms for robot inverse dynamics computation, *IEEE Transactions on Systems, Man and Cybernetics* 16 (4) (1986) 532–542.
- [14] C.S.G. Lee, P.R. Chang, Efficient parallel algorithms for robot forward dynamics computation, *IEEE Transactions on Systems, Man and Cybernetics* 18 (2) (1988) 238–251.
- [15] P. Sadayappan, Y.-L.C. Ling, K.W. Olson, D.E. Orin, A restructurable VLSI robotics vector processor architecture for real-time control, *IEEE Transactions on Robotics and Automation* 5 (5) (1989) 583–599.
- [16] M.W. Walker, D.E. Orin, Efficient dynamic computer simulation of robotic mechanisms, *ASME Journal of Dynamic Systems, Measurement and Control* 104 (3) (1982) 205–211.
- [17] R. Featherstone, The calculation of robot dynamics using articulated-body inertias, *International Journal of Robotics Research* 2 (1) (1983) 13–30.
- [18] U.M. Ascher, D.K. Pai, B.P. Cloutier, Forward dynamics, elimination methods, and formulation stiffness in robot simulation, *International Journal of Robotics Research* 16 (6) (1997) 749–758.

- [19] R. Featherstone, *Robot Dynamics Algorithms*, Kluwer Academic, Boston, MA, 1987.
- [20] V.I. Arnold, *Mathematical Methods of Classical Mechanics*, second ed., Springer-Verlag, New York, NY, 1989.
- [21] A. Kecskeméthy, T. Krupp, M. Hiller, Symbolic processing of multi-loop mechanism dynamics using closed form kinematic solutions, *Multibody System Dynamics* 1 (1) (1997) 23–45.
- [22] R.M. Murray, Z. Li, S.S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press LLC, Boca Raton, 1994.
- [23] J.W. Baumgarte, A new method of stabilization for holonomic constraints, *ASME, Transactions, Journal of Applied Mechanics* 50 (1983) 869–870.
- [24] U. Ascher, H. Chin, L. Petzold, S. Reich, Stabilization of constrained mechanical systems with DAEs and invariant manifolds, *Journal of Mechanical Structures & Machines* 23 (1995) 135–158.
- [25] W. Blajer, D. Bestle, W. Schiehlen, An orthogonal complement matrix formulation for constrained multibody systems, *ASME Journal of Mechanical Design* 116 (2) (1994) 423–428.
- [26] M.A. Serna, R. Avilés, J. García de Jalón, Dynamic analysis of plane mechanisms with lower-pairs in basic coordinates, *Mechanism and Machine Theory* 17 (1982) 397–403.
- [27] L.F. Shampine, M.W. Reichelt, The MATLAB ODE suite, *SIAM Journal on Scientific Computing* 18 (1) (1997) 1–22.
- [28] W. Schiehlen, A. Rügauer, T. Schirle, Force coupling versus differential algebraic description of constrained multibody systems, *Multibody System Dynamics* 4 (4) (2000) 317–340.