MULTIBODY DYNAMICS 2009, ECCOMAS Thematic Conference
K. Arczewski, J. Frączek, M. Wojtyra (eds.)
Warsaw, Poland, 29 June – 2 July 2009

# LAGRANGE MULTIPLIERS BASED DIVIDE AND CONQUER ALGORITHM FOR DYNAMICS OF GENERAL MULTIBODY SYSTEMS

**Paweł Malczyk, Janusz Frączek***

* Institute of Aeronautics and Applied Mechanics
Faculty of Power and Aeronautical Engineering
Warsaw University of Technology, Nowowiejska 24, 00-665 Warsaw, Poland
e-mail: `pmalczyk@meil.pw.edu.pl`, `jfraczek@meil.pw.edu.pl`

**Keywords:** Divide and Conquer Algorithm, Parallel computing, Augmented Lagrangian Method, Mass-Orthogonal Projections

**Abstract.** *This paper presents a new parallel algorithm for dynamics simulation of general multibody systems. The proposed method uses a divide and conquer formulation, both for open and closed loop multibody systems. The algorithm is formulated by using Lagrange multipliers approach for constraint imposition and absolute coordinates for the system state description. Penalty and augmented Lagrangian methods are used to prevent from constraint violation errors and to cope with possible Jacobian matrix rank deficiencies. The proposed algorithm generates additional computational costs compared to basic algorithm without stabilization, however it does distribute them in a divide and conquer manner and enables to exploit parallel computing techniques. The proposed parallel algorithm has advantages of being robust under singular configurations, topology changes (not covered in this work) and in the presence of redundant constraints .*

## 1 INTRODUCTION

Computational efficiency of the dynamics of large constrained multibody systems (MBS) is essential in many areas of computer aided engineering and design. There are many examples of such systems including vehicles, biomechanical models, robots and multidisciplinary applications. Computations can be carried out by means of different types of formulations. To meet requirements for high–fidelity performance and accurate dynamics simulations of complex systems, it has become a practice to apply efficient, low order algorithms designed both for sequential and parallel computations.

Computational efficiency of the MBS dynamics simulations has been receiving increasing attention from researchers since seventies and eighties, mainly from the robotics field. The earliest recursive sequential algorithms come from works [1], [2], [3], [4], [5]. Useful order $n$ (where $n$ is the number of bodies in the system) algorithms begin to appear following the Featherstone's work [6], in which the author used an idea of articulated body inertias. After that, many other researchers adopted this method and formulated their own recursive algorithms, e.g. [7], [8], [9]. The survey, which presents many of them in the spatial operator algebra formalism can be found in [10].

The recursive, order $O(n)$ methods for dealing with closed loop systems are developed independently by Bae and Haug [11] and Featherstone [12]. The works are extensions of the previous research, done in [6], [7]. In these methods, equations of motion are formulated by using Lagrange multipliers and generally, constraint forces associated with certain connections between the bodies. Specified joints are cut to open the system's loops and to define a tree structure. Although these formulations are called $O(n)$, they are actually at least of the order $O(n+m^3)$, where $m$ is the number of constraint equations. The mentioned linear complexity is valid only for large number of bodies and small number of constraint equations presented in the system. There are other strategies [13], [14], [15], [16], which offer improved performance relative to the traditional $O(n)$ algorithms. These methods enable to exploit fully recursive treatment of the loop constraints, and particularly to obtain solutions for dependent coordinate velocities and accelerations in terms of independent ones.

As parallel computing resources became more available, researchers began to parallelize the existing formulations or design completely new algorithms, suitable for parallel computing. The strategies enabled to decrease the turnaround time associated with computer simulations. Kasahara [17] was the first, who contributed to parallel computations in robot simulations. The author parallelized Walker and Orin's [3], [4] recursive algorithms and proposed two scheduling strategies, which could be applied to concurrent robot simulations on multiprocessors. Subsequently, Bae and Haug [18] applied their $O(n)$ algorithms [7], [11] to perform dynamic simulation of an off-road vehicle on a shared memory multiprocessors. The exploited parallel strategies, which considered parallel computing along independent branches of the system, were strongly dependent on the topology of the MBS. Lee and Chang [19] investigated the Walker and Orin's algorithm to produce their own parallel algorithm with parallel inverse dynamics procedures developed by Lathrop [20]. Much research was done on achieving real-time simulations by the use of parallel processing. The efforts can be found in Ref. [21], [22], [23].

First algorithm, which achieved logarithmic $O(\log n)$ complexity on $O(n^2)$ processors was Fijany and Bejczy's formulation [24]. The method however was limited to forward dynamics of chain systems. In 1995, Fijany, Sharf, D'Eleuterio [25], developed both the first time $O(\log n)$ and processor $O(n)$ optimal parallel algorithm called Constraint Force Algo-

rithm (CFA) for forward dynamics of a manipulator. It was derived from $O(n)$ algorithm through the intelligent decomposition and Schur's factorization of the mass matrix of a robot. It was later shown that CFA includes inconsistent expressions, which was investigated and improved in [26]. Fisette and Peterkenne [27] and independently Anderson and Duan [28] adopted the idea of a decomposition of MBS into subchains. Within each of the subsystems they employed fast recursive $O(n)$ sequential dynamics analysis procedures and subchains interacted with one another through unknown constraint loads. In 1999, Featherstone [29], [30] developed the first $O(\log n)$ divide and conquer algorithm (DCA) for dynamics of general MBS. The idea behind the formulation lies in a recursive binary assembly and disassembly of articulated bodies. The extension to closed loop systems were obtained through the application of constraint stabilization methods and constraint forces. In 2004, Critchley and Anderson [31] explored ideas of recursive coordinate reduction (RCR) [15], [16] and presented parallel multibody algorithm with optimal $O(\log n)$ time complexity on $O(n)$ processors for general multibody systems applicability. The formulation however was not free from problems arising from the fact that some matrices lose ranks (i.e. matrices relating dependent state variables to independent ones). In 2006, Mukherjee and Anderson [32] presented exact and non-iterative divide and conquer algorithm for forward dynamics of MBS with single and coupled loops. The formulation incorporated neither coordinate reductions nor Lagrange multipliers. The constraint equations were imposed at the acceleration level through kinematic relations involving orthogonal complement of the joint motion subspace. The algorithm indicated good constraint fulfillment and could easily handle systems in singular configurations.

Another issue, which may arise in multibody dynamics simulations is a problem of controlling the accumulation of constraint errors. In general, the constraint equations are imposed at the acceleration level (or possibly at the velocity level [16]). The acceleration constraint equations is known to be unstable. The numerical errors that appear during the integration of equations of motion may cause significant constraint violation errors. Several constraint stabilization methods have been used by multibody researchers over the last years. The widely used techniques are not limited to procedures presented in works [33]-[40].

This paper presents a new parallel algorithm for dynamics simulation of general multibody systems. The proposed method uses a divide and conquer formulation, which is similar to that presented in Ref. [29] and [30]. The algorithm is formulated by using Lagrange multipliers approach for constraint imposition and absolute coordinates for the system state description. Penalty and augmented Lagrangian methods [35], [36], [37] are used to prevent from constraint violation errors. The adopted iterative refinement procedure generates additional computational costs compared to basic algorithm without stabilization, which is exact and non-iterative, however it does distribute them in a divide and conquer manner and enables to exploit parallel computing techniques. The parallel algorithm has advantages of being robust under singular configurations, and in the presence of redundant constraints, where the Jacobian matrix becomes rank deficient.

## 2 BASIC ALGORITHM OVERVIEW

This section presents analytical preliminaries for the divide and conquer algorithm in absolute coordinates for open loop chains. The method used here is similar to that proposed by Featherstone [29] and [30], however the underlying computations are different, because of the type of coordinates employed. The basic algorithm is composed of only two passes, the main pass and the back-substitution pass, without the need of bodies' position and velocity calculations, which are apparent in the original Featherstone's formulation in joint coordinates.

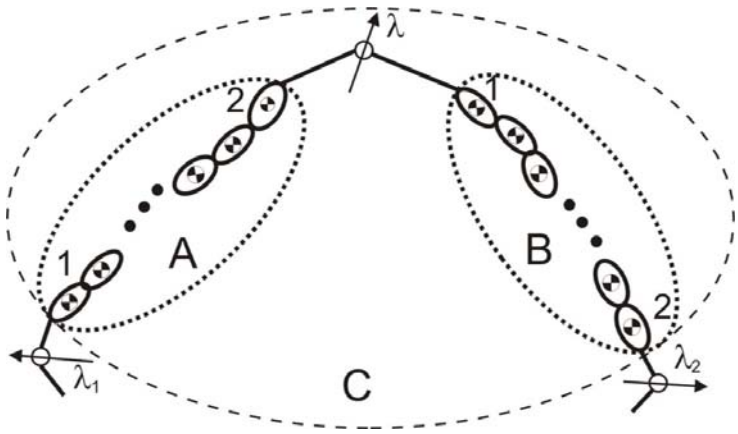


Figure 1: Two articulated bodies.

Consider a system of rigid bodies $A$ and $B$ interconnected by joints (Fig. 1). They are interconnected by joints. The system is under influence of various acceleration-independent forces and its state is known. Three Lagrange multipliers associated with joints are specified. Those with subscript 1 and 2 are used to communicate with other subsystems in the MBS and the multipliers between the articulated bodies $A$ and $B$ serve as those, which will be eliminated during the computations. The specified points, where joint coordinate frames are located, refer as handles, i.e. an interfaces between subassemblies.

Using the spatial equations of motion with Lagrange multipliers, one can write the articulated-body equations of motion as follows:

$$\ddot{q}_1^A = \alpha_{11}^A \Phi^1_{q_1^A}{}^T \lambda_1 + \alpha_{12}^A \Phi^T_{q_2^A} \lambda + \beta_1^A \tag{1}$$

$$\ddot{q}_2^A = \alpha_{21}^A \Phi^1_{q_1^A}{}^T \lambda_1 + \alpha_{22}^A \Phi^T_{q_2^A} \lambda + \beta_2^A \tag{2}$$

$$\ddot{q}_1^B = \alpha_{11}^B \Phi^T_{q_1^B} \lambda + \alpha_{12}^B \Phi^2_{q_2^B}{}^T \lambda_2 + \beta_1^B \tag{3}$$

$$\ddot{q}_2^B = \alpha_{21}^B \Phi^T_{q_1^B} \lambda + \alpha_{22}^B \Phi^2_{q_2^B}{}^T \lambda_2 + \beta_2^B \tag{4}$$

The objective of the algorithm is to obtain articulated-body equations of motion of the set $C$.

$$\ddot{q}_1^C = \alpha_{11}^C \Phi^1_{q_1^C}{}^T \lambda_1 + \alpha_{12}^C \Phi^2_{q_2^C}{}^T \lambda_2 + \beta_1^C \tag{5}$$

$$\ddot{q}_2^C = \alpha_{21}^C \Phi^1_{q_1^C}{}^T \lambda_1 + \alpha_{22}^C \Phi^2_{q_2^C}{}^T \lambda_2 + \beta_2^C \tag{6}$$

Values $\lambda_1$, $\lambda$, $\lambda_2$ are unknown Lagrange multipliers, associated with constraint forces, depicted in Fig. 1. The $\ddot{q}$ quantities are spatial $6\times1$ accelerations (linear and angular) of the body mass center, defined as $\ddot{q} = \left[\dot{v}^T \quad \dot{\omega}^T\right]^T$. Later on, this vector will be redefined to obtain $7\times1$ quantity, which includes second derivatives of Euler parameters. Jacobian matrix associated with constraint equations is given by matrix $\Phi_q$. The size of the Jacobian matrix is

$m \times 6$, where $m$ is the number of constraint equations for a specified joint. This matrix is used to express acceleration level constraint equations between sets of subsystems as

$$\ddot{\Phi} \equiv \Phi_{q_2^A} \ddot{q}_2^A + \Phi_{q_1^B} \ddot{q}_1^B + \gamma_{12} = 0 \tag{7}$$

The coefficients $\alpha_{ij}$ for $i = 1,2$, $j = 1,2$ are inertia coupling terms, and the terms $\beta_i$ for $i = 1,2$ include state dependent expressions (e.g. forces), apart from those coming from inertia. These quantities are used to solve a problem for accelerations. The first step in deriving divide and conquer algorithm in absolute coordinates is to substitute Eq. (1) and (4) into Eq. (7), to obtain

$$\begin{aligned} &-(\Phi_{q_2^A} \alpha_2^A \Phi_{q_2^A}^T + \Phi_{q_1^B} \alpha_1^B \Phi_{q_1^B}^T)\lambda = \\ &= \Phi_{q_2^A} \alpha_{21}^A \Phi_{q_1^A}^{1\ T} \lambda_1 + \Phi_{q_1^B} \alpha_{12}^B \Phi_{q_2^B}^{2\ T} \lambda_2 + \Phi_{q_2^A} \beta_2^A + \Phi_{q_1^B} \beta_1^B + \gamma_{12} \end{aligned} \tag{8}$$

Let us introduce the term:

$$C = -(\Phi_{q_2^A} \alpha_2^A \Phi_{q_2^A}^T + \Phi_{q_1^B} \alpha_1^B \Phi_{q_1^B}^T)^{-1} \tag{9}$$

The matrix inversion in (9) exists if Jacobian matrices have full row ranks. This holds true for tree topology mechanisms. However, in multibody dynamics simulations there are circumstances in which constraint Jacobian matrices lose their ranks. The problem may appear either permanently as in case of redundant constraints or incidentally at particular situations regarding singular configurations or varying number of constraints. The possibilities affect the existence of matrix inversion defined in (9). In section 3, a new method will be presented for unified handling singular and nonsingular case. Let us assume that matrix in (9) is invertible. From (8) and (9), the Lagrange multipliers between set of bodies $A$ and $B$ can be expressed as

$$\lambda = C\Phi_{q_2^A} \alpha_{21}^A \Phi_{q_1^A}^{1\ T} \lambda_1 + C\Phi_{q_1^B} \alpha_{12}^B \Phi_{q_2^B}^{2\ T} \lambda_2 + C(\Phi_{q_2^A} \beta_2^A + \Phi_{q_1^B} \beta_1^B + \gamma_{12}) \tag{10}$$

Substituting relation (10) to (2) and (3) yields

$$\begin{aligned} \ddot{q}_1^A &= [\alpha_{11}^A + \alpha_{12}^A \Phi_{q_2^A}^T C \Phi_{q_2^A} \alpha_{21}^A] \Phi_{q_1^A}^{1\ T} \lambda_1 + [\alpha_{12}^A \Phi_{q_2^A}^T C \Phi_{q_1^B} \alpha_{12}^B] \Phi_{q_2^B}^{2\ T} \lambda_2 + \\ &+ [\alpha_{12}^A \Phi_{q_2^A}^T C(\Phi_{q_2^A} \beta_2^A + \Phi_{q_1^B} \beta_1^B + \gamma_{12}) + \beta_1^A] \end{aligned} \tag{11}$$

$$\begin{aligned} \ddot{q}_2^B &= [\alpha_{21}^B \Phi_{q_1^B}^T C \Phi_{q_2^A} \alpha_{21}^A] \Phi_{q_1^A}^{1\ T} \lambda_1 + [\alpha_{22}^B + \alpha_{21}^B \Phi_{q_1^B}^T C \Phi_{q_1^B} \alpha_{12}^B] \Phi_{q_2^B}^{2\ T} \lambda_2 + \\ &+ [\alpha_{21}^B \Phi_{q_1^B}^T C(\Phi_{q_2^A} \beta_2^A + \Phi_{q_1^B} \beta_1^B + \gamma_{12}) + \beta_2^B] \end{aligned} \tag{12}$$

Relations (11) and (12) are equations of motion for articulated body $C$. The absolute accelerations for external bodies are expressed as a function of Lagrange multipliers $\lambda_1$ and $\lambda_2$, which is the objective of presented algebraic manipulations. Comparing Eq. (11) and (12) with (5) and (6), we obtain recursive formulas for coefficients of the equations of motion for set $C$.

$$\alpha_{11}^C = \alpha_{11}^A + \alpha_{12}^A \Phi_{q_2^A}^T C \Phi_{q_2^A} \alpha_{21}^A \tag{13}$$

$$\alpha_{12}^{C} = \alpha_{12}^{A}\Phi_{q_2^A}^{T} C \Phi_{q_1^B}\alpha_{12}^{B} \tag{14}$$

$$\alpha_{21}^{C} = \alpha_{21}^{B}\Phi_{q_1^B}^{T} C \Phi_{q_2^A}\alpha_{21}^{A} \equiv (\alpha_{12}^{C})^{T} \tag{15}$$

$$\alpha_{22}^{C} = \alpha_{22}^{B} + \alpha_{21}^{B}\Phi_{q_1^B}^{T} C \Phi_{q_1^B}\alpha_{12}^{B} \tag{16}$$

and

$$\beta_{1}^{C} = \alpha_{12}^{A}\Phi_{q_2^A}^{T} C(\Phi_{q_2^A}\beta_{2}^{A} + \Phi_{q_1^B}\beta_{1}^{B} + \gamma_{12}) + \beta_{1}^{A} \tag{17}$$

$$\beta_{2}^{C} = \alpha_{21}^{B}\Phi_{q_1^B}^{T} C(\Phi_{q_2^A}\beta_{2}^{A} + \Phi_{q_1^B}\beta_{1}^{B} + \gamma_{12}) + \beta_{2}^{B} \tag{18}$$

As mentioned earlier, the divide and conquer scheme in absolute coordinates consists of two passes, the main pass, and the back-substitution pass. The presented algebraic manipulations (13)–(18) enable to construct equations of motion of a MBS by the process of recursive binary assembly and disassembly, as depicted in Fig. 2. The process starts with equations of motion for each body in the system. Subassemblies, which correspond to nodes in the graph, are constructed by traversing the binary tree. Finally, the entire MBS is obtained, which is indicated as a root node in Fig. 2. The main pass finishes at this stage. Now, taking into account a connection of a chain to fixed base and free floating terminal body, the back-substitution phase is started. Traversing the tree from root node to leaves, all Lagrange multipliers and bodies' accelerations are computed in the system and the evaluated kinematic quantities may be efficiently integrated.

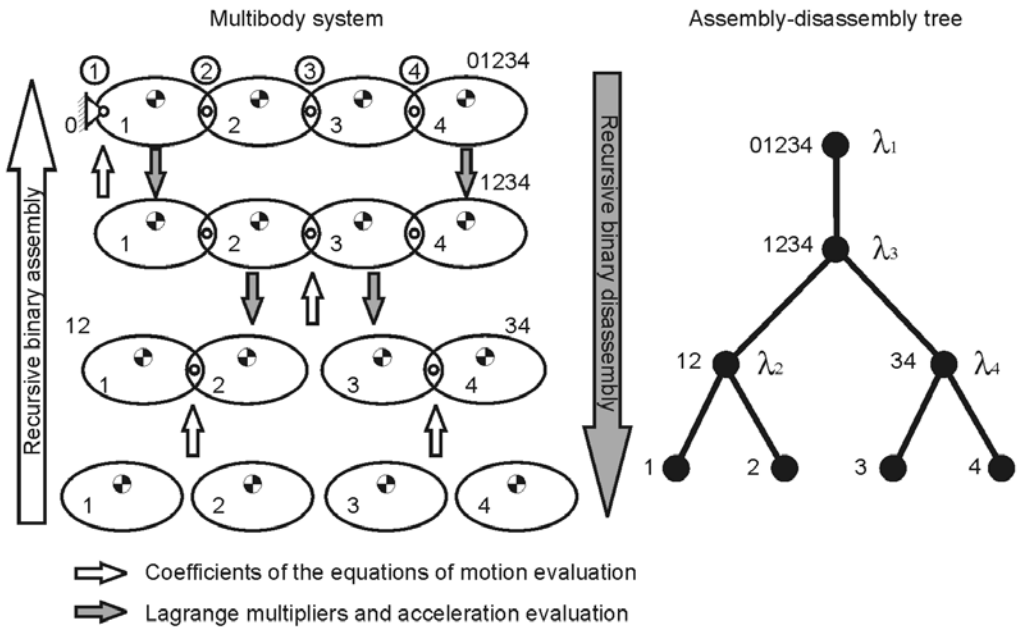


Figure 2: Recursive binary assembly and disassembly of a four-link multibody system.

Although scheme presented here is similar to that proposed by Featherstone, the associated computations appear to be different. First of all, absolute coordinates are employed, which imply two main stages of computations necessary. There is no need to evaluate position and velocity of bodies by means of recursive approach, because these quantities are available just

after the integration. The structure of the algorithm is well suited to parallel computations and features $O(\log_2(n))$ on $O(n)$ processors, where $n$ is the number of bodies in the system. Moreover, the proposed method can be regarded also as a serial algorithm in absolute coordinates, which exploits the structure of multibody dynamics equations and achieves $O(n)$ time complexity.

Nevertheless, the method at this form features several factors, which make general, multibody dynamics simulation rather difficult. The presented algorithm exploits redundant set of coordinates for a system state description. Lagrange multipliers associated with constraint equations are introduced. It is well known, that this kind of equations suffer from constraint violation errors, which can grow exponentially during the simulation time. Moreover, the algorithm fails in case of rank deficiency of constraint Jacobian matrices. These limitations will be addressed in the next section.

## 3 DIVIDE AND CONQUER ALGORITHM FOR GENERAL MULTIBODY SYSTEMS

### 3.1 Augmented Lagrangian Formulation (ALF)

This section provides a derivation for a parallel algorithm that can be used for general MBS simulation. The formulation treats both open and closed loop systems in a unified manner. These features are ensured by the proper combination of the augmented Lagrangian method [35], [36], [37], [38] and the divide and conquer scheme in absolute coordinates.

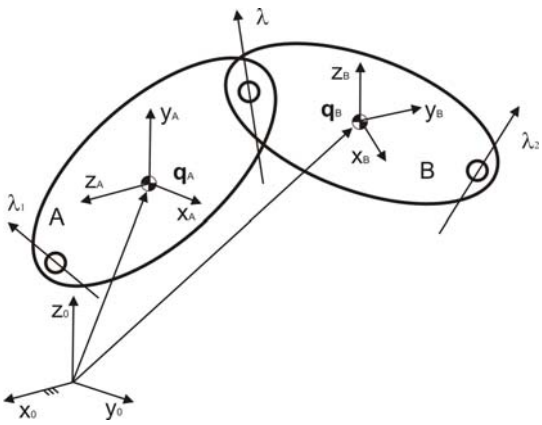


Figure 3: System of two rigid bodies.

Let us consider a system of two rigid bodies connected by a joint, as depicted in Fig. 3. Each body is characterized by generalized coordinates $q$, which are in the form of $7\times 1$ matrix of position and orientation described in Euler parameters, $q=\begin{bmatrix} r^T & p^T \end{bmatrix}^T$. The quantities are associated with the mass center of a body. Redefinition of vector $q$ is motivated by unification of bodies' description at position, velocity and acceleration levels, as will be evident later in this section. Equations of motion for bodies $A$ and $B$ can be written using constrained spatial Newton-Euler formulation:

$$M^A\ddot{q}^A + \Phi^{1\,T}_{q^A}\lambda_1 + \Phi^T_{q^A}\lambda + \Phi^{N\,T}_{q^A}\lambda^N_A = Q^A \tag{19}$$

$$M^B\ddot{q}^B + \Phi^T_{q^B}\lambda + \Phi^{2\,T}_{q^B}\lambda_2 + \Phi^{N\,T}_{q^B}\lambda^N_B = Q^B \tag{20}$$

The matrices $M^A$ and $M^B$ are $7\times 7$ quantities, which are known to be singular [41], [42]. Vectors $Q^A$, $Q^B$ are applied and centrifugal forces, and Lagrange multipliers $\lambda_A^N$, $\lambda_B^N$ are associated with Euler parameters normalization constraints. Acceleration constraint equations between pair of bodies are expressed in the form

$$\ddot{\Phi} \equiv \Phi_{q^A}\ddot{q}^A + \Phi_{q^B}\ddot{q}^B + \gamma^{AB} = 0 \tag{21}$$

where $-\gamma^{AB}$ is a vector of right hand sides of acceleration constraint equations. Normalization constraints at the acceleration level can be expressed as

$$\ddot{\Phi}_A^N \equiv \Phi_{q^A}^N \ddot{q}^A + \gamma_N^A = 0 \tag{22a}$$

$$\ddot{\Phi}_B^N \equiv \Phi_{q^B}^N \ddot{q}^B + \gamma_N^B = 0 \tag{22b}$$

The algebraic manipulations presented in section 2 cannot be directly applied to the equations of motion (19) and (20), since equations (1) to (4) assume that the mass matrices are invertible. This is not the case for Eq. (19) and (20). To overcome this difficulty augmented Lagrangian formulation is employed, which leads to the following Lagrange multipliers iterative approximation [36]:

$$\lambda = \lambda^i + \alpha(\Phi_{q^A}\ddot{q}^A + \Phi_{q^B}\ddot{q}^B + \underbrace{\gamma^{AB} + 2\Omega\mu\dot{\Phi} + \Omega^2\Phi}_{\bar{\gamma}^{AB}}) \tag{23a}$$

$$\lambda_A^N = \lambda_A^{N\,i} + \alpha(\Phi_{q^A}^N\ddot{q}^A + \underbrace{\gamma_N^A + 2\Omega\mu\dot{\Phi}_A^N + \Omega^2\Phi_A^N}_{\bar{\gamma}_N^A}) \tag{23b}$$

$$\lambda_B^N = \lambda_B^{N\,i} + \alpha(\Phi_{q^B}^N\ddot{q}^B + \underbrace{\gamma_N^B + 2\Omega\mu\dot{\Phi}_B^N + \Omega^2\Phi_B^N}_{\bar{\gamma}_N^B}) \tag{23c}$$

Here, $\lambda^i$, $\lambda_A^{N\,i}$ and $\lambda_B^{N\,i}$ are Lagrange multipliers obtained from previous iteration. The quantity $\alpha$ is a diagonal matrix that contain the values of the penalty numbers. Substituting relations (23) to (19) and (20), the following final result is obtained

$$\begin{aligned}&(M^A + \Phi_{q^A}^T\alpha\Phi_{q^A} + \Phi_{q^A}^{N\,T}\alpha\Phi_{q^A}^N)\ddot{q}^A + \Phi_{q^A}^T\alpha\Phi_{q^B}\ddot{q}^B + \Phi_{q^A}^{1\,T}\lambda_1 = \\ &= Q^A - \Phi_{q^A}^T(\lambda^i + \alpha\bar{\gamma}^{AB}) - \Phi_{q^A}^{N\,T}(\lambda_{q^A}^{N\,i} + \alpha\bar{\gamma}_N^A)\end{aligned} \tag{24}$$

$$\begin{aligned}&\Phi_{q^B}^T\alpha\Phi_{q^A}\ddot{q}^A + (M^B + \Phi_{q^B}^T\alpha\Phi_{q^B} + \Phi_{q^B}^{N\,T}\alpha\Phi_{q^B}^N)\ddot{q}^B + \Phi_{q^B}^{2\,T}\lambda_2 = \\ &= Q^B - \Phi_{q^B}^T(\lambda^i + \alpha\bar{\gamma}^{AB}) - \Phi_{q^B}^{N\,T}(\lambda_{q^B}^{N\,i} + \alpha\bar{\gamma}_N^B)\end{aligned} \tag{25}$$

Equations (24) and (25) form a basis for further derivations. They are starting point for the main pass phase. In contrast to equations (19) and (20), relations (24) and (25) can be solved for $\ddot{q}^A$ and $\ddot{q}^B$ respectively. The encountered coefficient matrices are positive definite, even in situations, when constraint Jacobians lose their ranks [36].

Regarding Fig. 1, and expressions (24) and (25), let us consider the system of relations

$$M_{11}^{A}\ddot{q}_1^A + M_{12}^{A}\ddot{q}_2^A + \Phi^{1}_{q_1^A}{}^{T}\lambda_1 = Q_1^A \tag{26}$$

$$M_{21}^{A}\ddot{q}_1^A + M_{22}^{A}\ddot{q}_2^A + \Phi^{T}_{q_2^A}\lambda = Q_2^A \tag{27}$$

$$M_{11}^{B}\ddot{q}_1^B + M_{12}^{B}\ddot{q}_2^B + \Phi^{T}_{q_1^B}\lambda = Q_1^B \tag{28}$$

$$M_{21}^{B}\ddot{q}_1^B + M_{22}^{B}\ddot{q}_2^B + \Phi^{2}_{q_2^B}{}^{T}\lambda_2 = Q_2^B \tag{29}$$

The objective of the following algebraic manipulations is to obtain equations for the set C, in the form

$$M_{11}^{C}\ddot{q}_1^A + M_{12}^{C}\ddot{q}_2^B + \Phi^{1}_{q_1^A}{}^{T}\lambda_1 = Q_1^C \tag{30}$$

$$M_{21}^{C}\ddot{q}_1^A + M_{22}^{C}\ddot{q}_2^B + \Phi^{2}_{q_2^B}{}^{T}\lambda_2 = Q_2^C \tag{31}$$

Unknown Lagrange multipliers can be found through the iterative process

$$\lambda = \lambda^i + \alpha(\Phi_{q_2^A}\ddot{q}_2^A + \Phi_{q_1^B}\ddot{q}_1^B + \bar{\gamma}^{AB}) \tag{32}$$

Lagrange multipliers $\lambda^0 = 0$ for the first iteration. Inserting relation (32) to (27) and (28) yields

$$\overline{M}_{22}^{A}\ddot{q}_2^A = \overline{Q}_2^A - \Phi^{T}_{q_2^A}\alpha\Phi_{q_1^B}\ddot{q}_1^B - M_{21}^{A}\ddot{q}_1^A \tag{33}$$

$$\overline{M}_{11}^{B}\ddot{q}_1^B = \overline{Q}_1^B - \Phi^{T}_{q_1^B}\alpha\Phi_{q_2^A}\ddot{q}_2^A - M_{12}^{B}\ddot{q}_2^B \tag{34}$$

where

$$\overline{Q}_2^A = Q_2^A - \Phi^{T}_{q_2^A}(\lambda^i + \alpha\bar{\gamma}^{AB}) \tag{35}$$

$$\overline{Q}_1^B = Q_1^B - \Phi^{T}_{q_1^B}(\lambda^i + \alpha\bar{\gamma}^{AB}) \tag{36}$$

$$\overline{M}_{22}^{A} = M_{22}^{A} + \Phi^{T}_{q_2^A}\alpha\Phi_{q_2^A} \tag{37}$$

$$\overline{M}_{11}^{B} = M_{11}^{B} + \Phi^{T}_{q_1^B}\alpha\Phi_{q_1^B} \tag{38}$$

Substituting Eq. (34) into Eq. (33):

$$\overline{\overline{M}}_{22}^{A}\ddot{q}_2^A = \overline{\overline{Q}}_{21}^{AB} - M_{21}^{A}\ddot{q}_1^A + \Phi^{T}_{q_2^A}\alpha\Phi_{q_1^B}(\overline{M}_{11}^{B})^{-1}M_{12}^{B}\ddot{q}_2^B \tag{39}$$

where

$$\overline{\overline{M}}_{22}^{A} = \overline{M}_{22}^{A} - \Phi^{T}_{q_2^A}\alpha\Phi_{q_1^B}(\overline{M}_{11}^{B})^{-1}\Phi^{T}_{q_1^B}\alpha\Phi_{q_2^A} \tag{40}$$

$$\overline{\overline{Q}}_{21}^{AB} = \overline{Q}_2^A - \Phi^T_{q_2^A} \alpha \Phi_{q_1^B} (\overline{M}_{11}^B)^{-1} \overline{Q}_1^B \tag{41}$$

And analogously inserting Eq. (33) into Eq. (34) gives

$$\overline{\overline{M}}_{11}^B \ddot{q}_1^B = \overline{\overline{Q}}_{12}^{BA} - M_{12}^B \ddot{q}_2^B + \Phi^T_{q_1^B} \alpha \Phi_{q_2^A} (\overline{M}_{22}^A)^{-1} M_{21}^A \ddot{q}_1^A \tag{42}$$

with

$$\overline{\overline{M}}_{11}^B = \overline{M}_{11}^B - \Phi^T_{q_1^B} \alpha \Phi_{q_2^A} (\overline{M}_{22}^A)^{-1} \Phi^T_{q_2^A} \alpha \Phi_{q_1^B} \tag{43}$$

$$\overline{\overline{Q}}_{12}^{BA} = \overline{Q}_1^B - \Phi^T_{q_1^B} \alpha \Phi_{q_2^A} (\overline{M}_{22}^A)^{-1} \overline{Q}_2^A \tag{44}$$

The final step is to insert Eq. (39) into Eq. (26), and analogously Eq. (42) into Eq. (29), which gives

$$\begin{aligned} &(M_{11}^A - M_{12}^A (\overline{\overline{M}}_{22}^A)^{-1} M_{21}^A) \ddot{q}_1^A + (M_{12}^A (\overline{\overline{M}}_{22}^A)^{-1} \Phi^T_{q_2^A} \alpha \Phi_{q_1^B} (\overline{M}_{11}^B)^{-1} M_{12}^B) \ddot{q}_2^B + \Phi^{1\ T}_{q_1^A} \lambda_1 = \\ &= Q_1^A - M_{12}^A (\overline{\overline{M}}_{22}^A)^{-1} \overline{\overline{Q}}_{21}^{AB} \end{aligned} \tag{45}$$

$$\begin{aligned} &(M_{21}^B (\overline{\overline{M}}_{11}^B)^{-1} \Phi^T_{q_1^B} \alpha \Phi_{q_2^A} (\overline{M}_{22}^A)^{-1} M_{21}^A) \ddot{q}_1^A + (M_{22}^B - M_{21}^B (\overline{\overline{M}}_{11}^B)^{-1} M_{12}^B) \ddot{q}_2^B + \Phi^{2\ T}_{q_2^B} \lambda_2 = \\ &= Q_2^B - M_{21}^B (\overline{\overline{M}}_{11}^B)^{-1} \overline{\overline{Q}}_{12}^{BA} \end{aligned} \tag{46}$$

By gathering appropriate matrix coefficients, and taking into account Eq. (30) and (31), the following matrix formulas are obtained

$$M_{11}^C = M_{11}^A - M_{12}^A (\overline{\overline{M}}_{22}^A)^{-1} M_{21}^A \tag{47}$$

$$M_{12}^C = M_{12}^A (\overline{\overline{M}}_{22}^A)^{-1} \Phi^T_{q_2^A} \alpha \Phi_{q_1^B} (\overline{M}_{11}^B)^{-1} M_{12}^B \tag{48}$$

$$M_{21}^C = M_{21}^B (\overline{\overline{M}}_{11}^B)^{-1} \Phi^T_{q_1^B} \alpha \Phi_{q_2^A} (\overline{M}_{22}^A)^{-1} M_{21}^A = (M_{12}^C)^T \tag{49}$$

$$M_{22}^C = M_{22}^B - M_{21}^B (\overline{\overline{M}}_{11}^B)^{-1} M_{12}^B \tag{50}$$

$$Q_1^C = Q_1^A - M_{12}^A (\overline{\overline{M}}_{22}^A)^{-1} \overline{\overline{Q}}_{21}^{AB} \tag{51}$$

$$Q_2^C = Q_2^B - M_{21}^B (\overline{\overline{M}}_{11}^B)^{-1} \overline{\overline{Q}}_{12}^{BA} \tag{52}$$

These equations define the recursive relations for the object achieved after the binary assembly of child-objects. When the root node is reached, the following equations are obtained

$$M_{11}^C \ddot{q}_1^A + M_{12}^C \ddot{q}_2^B + \Phi^{1\ T}_{q_1^A} \lambda_1 = Q_1^C \tag{53}$$

$$M_{21}^C \ddot{q}_1^A + M_{22}^C \ddot{q}_2^B + \Phi^{2\ T}_{q_2^B} \lambda_2 = Q_2^C \tag{54}$$

For open chain system $\lambda_2 = 0$, which indicate free floating terminal body. In turn, Lagrange multipliers associated with fixed base body can be approximated as

$$\lambda_1 = \lambda_1^i + \alpha(\Phi^1_{q_1^A}\ddot{q}_1^A + \bar{\gamma}_1^A) \tag{55}$$

Substituting (55) into (53) yields

$$(M_{11}^C + \Phi^1_{q_1^A}{}^T \alpha \Phi^1_{q_1^A})\ddot{q}_1^A + M_{12}^C \ddot{q}_2^B = Q_1^C - \Phi^1_{q_1^A}{}^T(\lambda_1^i + \alpha\bar{\gamma}_1^A) \tag{56}$$

Relations (54) and (56) form a linear system in two unknowns $\ddot{q}_1^A$ and $\ddot{q}_2^B$. Starting with evaluated quantities, all accelerations in the system can be computed and then efficiently integrated to provide state variables for the next time instant.

Derived parallel algorithm posses the same features as original augmented Lagrangian formulation does. It shows good overall characteristics and robustness in case of general multibody dynamics analyses. However the formulation does not enforce simultaneously the errors in positions, velocities and acceleration constraints to the desired level. In order to overcome these drawbacks, augmented Lagrangian formulation with projections [37] is employed for further considerations.

### 3.2 ALF with Projections in Positions

During the integration process, absolute position coordinates may not satisfy constraint equations. In order to correct bodies' positions, mass-orthogonal projections of the solution to the constraint manifold are performed. Consider again the system of two rigid bodies connected by a joint, as depicted in Fig. 3. Minimization of a functional ($\min_q \frac{1}{2}(q-q_*)^T M(q-q_*)$) subjected to constraint equations ($\Phi(q,t)=0$) and subsequently Taylor series expansion yield the following relations at the position level

$$M^A \Delta q^A + \Phi^1_{q^A}{}^T \lambda_1 + \Phi^T_{q^A}\lambda + \Phi^N_{q^A}{}^T \lambda_A^N = -M^A(q_i^A - q_*^A) \tag{57}$$

$$M^B \Delta q^B + \Phi^T_{q^B}\lambda + \Phi^2_{q^B}{}^T \lambda_2 + \Phi^N_{q^B}{}^T \lambda_B^N = -M^B(q_i^B - q_*^B) \tag{58}$$

where $\lambda$, $\lambda_1$, $\lambda_2$ are Lagrange multipliers associated with functional minimization, and $q_*^A$ and $q_*^B$ are perturbed values obtained from the numerical integration that does not completely satisfy constraint equations (59)

$$\Phi(q^A, q^B, t) = 0 \tag{59a}$$

$$\Phi_A^N(q^A) = 0 \tag{59b}$$

$$\Phi_B^N(q^B) = 0 \tag{59c}$$

The increments $\Delta q^A$ are defined as follows

$$\Delta q^A = q^A - q_i^A \tag{60a}$$

$$\Delta q^B = q^B - q_i^B \tag{60b}$$

where the subscript indicates the iteration number. For the first iteration we have $q_i^A = q_*^A$, $q_i^B = q_*^B$, and $\lambda^i = 0$, $\lambda_A^{N^i} = 0$, $\lambda_B^{N^i} = 0$. The Lagrange multipliers approximation, taken from the Taylor series expansion

$$\lambda = \lambda^i + \alpha\Phi(q^A, q^B, t) \cong \lambda^i + \alpha(\Phi(q_i^A, q_i^B, t) + \Phi_{q^A}\Delta q^A + \Phi_{q^B}\Delta q^B) \tag{61a}$$

$$\lambda_A^N = \lambda_A^{N^i} + \alpha\Phi_A^N(q^A) \cong \lambda_A^{N^i} + \alpha(\Phi_A^N(q_i^A) + \Phi_{q^A}^N\Delta q^A) \tag{61b}$$

$$\lambda_B^N = \lambda_B^{N^i} + \alpha\Phi_B^N(q^B) \cong \lambda_B^{N^i} + \alpha(\Phi_B^N(q_i^B) + \Phi_{q^B}^N\Delta q^B) \tag{61c}$$

Substituting Eq. (61) into Eq. (57) and (58) we obtain

$$\begin{aligned}&(M^A + \Phi_{q^A}^T\alpha\Phi_{q^A} + \Phi_{q^A}^{N\,T}\alpha\Phi_{q^A}^N)\Delta q^A + \Phi_{q^A}^T\alpha\Phi_{q^B}\Delta q^B + \Phi_{q^A}^{1\,T}\lambda_1 = \\ &= -M^A(q_i^A - q_*^A) - \Phi_{q^A}^T(\lambda^i + \alpha\Phi(q_i^A, q_i^B, t)) - \Phi_{q^A}^{N\,T}(\lambda_A^{N^i} + \alpha\Phi_A^N(q_i^A))\end{aligned} \tag{62}$$

$$\begin{aligned}&\Phi_{q^B}^T\alpha\Phi_{q^A}\Delta q^A + (M^B + \Phi_{q^B}^T\alpha\Phi_{q^B} + \Phi_{q^B}^{N\,T}\alpha\Phi_{q^B}^N)\Delta q^B + \Phi_{q^B}^{2\,T}\lambda_2 = \\ &= -M^B(q_i^B - q_*^B) - \Phi_{q^B}^T(\lambda^i + \alpha\Phi(q_i^A, q_i^B, t)) - \Phi_{q^B}^{N\,T}(\lambda_B^{N^i} + \alpha\Phi_B^N(q_i^B))\end{aligned} \tag{63}$$

The equations (62) and (63) form a basis for the divide and conquer algorithm at the position level. Unknown increments, defined in (60), are quantities that should be found during the iterative process, until $\|\Delta q\| < \varepsilon_p$, where $\varepsilon_p$ is a user specified tolerance. We can write the following generalized relations, which allow to minimize constraint errors in positions.

$$M_{11}^A\Delta q_1^A + M_{12}^A\Delta q_2^A + \Phi_{q_1^A}^{1\,T}\lambda_1 = Q_1^A \tag{64}$$

$$M_{21}^A\Delta q_1^A + M_{22}^A\Delta q_2^A + \Phi_{q_2^A}^T\lambda = Q_2^A \tag{65}$$

$$M_{11}^B\Delta q_1^B + M_{12}^B\Delta q_2^B + \Phi_{q_1^B}^T\lambda = Q_1^B \tag{66}$$

$$M_{21}^B\Delta q_1^B + M_{22}^B\Delta q_2^B + \Phi_{q_2^B}^{2\,T}\lambda_2 = Q_2^B \tag{67}$$

The aim is to find relations describing articulated body C (Fig. 1) in the form

$$M_{11}^C\Delta q_1^A + M_{12}^C\Delta q_2^B + \Phi_{q_1^A}^{1\,T}\lambda_1 = Q_1^C \tag{68}$$

$$M_{21}^C\Delta q_1^A + M_{22}^C\Delta q_2^B + \Phi_{q_2^B}^{2\,T}\lambda_2 = Q_2^C \tag{69}$$

where Lagrange multipliers approximation can expressed as

$$\lambda = \lambda^i + \alpha(\Phi_{q_2^A}\Delta q_2^A + \Phi_{q_1^B}\Delta q_1^B + \Phi^i(q_2^A, q_1^B, t)) \tag{70}$$

Equations (64) to (70) are identical in the form as equations (26) to (32), presented in previous section. The formulas for recursive binary assembly are defined in Eq. (47) to (52). It is important to note that matrix coefficients $M_{ij}^A$ and $M_{ij}^B$ ($i, j = 1,2$) are the same as in case of Eq.

(26) to (31). During the iteration process, these quantities are computed only once, and the only cost associated with stabilization at the position level is to evaluate $Q_i^A$ and $Q_i^B$.

### 3.3 ALF with Projections in Velocities

Similarly, during the time course of the dynamic simulation, the numerical integration procedure gives velocities $\dot{q}_*$, which does not completely satisfy velocity constraint equations. The idea is to perform mass-orthogonal projections of the solution to the velocity constraint manifold. Minimization of a functional ($\min_{\dot{q}} \frac{1}{2}(\dot{q}-\dot{q}_*)^T M(\dot{q}-\dot{q}_*)$) subjected to constraint equations ($\dot{\Phi}(q,\dot{q},t)=0$). Consider again pair of articulated bodies, depicted in Fig. 3. The following relations at the velocity level can be obtained

$$M^A\dot{q}^A + \Phi^{1\,T}_{q^A}\lambda_1 + \Phi^T_{q^A}\lambda + \Phi^{N\,T}_{q^A}\lambda^N_A = M^A\dot{q}^A_* \tag{71}$$

$$M^B\dot{q}^B + \Phi^T_{q^B}\lambda + \Phi^{2\,T}_{q^B}\lambda_2 + \Phi^{N\,T}_{q^B}\lambda^N_B = M^B\dot{q}^B_* \tag{72}$$

where $\lambda$, $\lambda_1$, $\lambda_2$ are Lagrange multipliers associated with functional minimization with respect to perturbed velocities $\dot{q}^A_*$ and $\dot{q}^B_*$.

$$\Phi_{q^A}\dot{q}^A + \Phi_{q^B}\dot{q}^B + \Phi_t = 0 \tag{73a}$$

$$\dot{\Phi}^N_A(q^A,\dot{q}^A) = 0 \tag{73b}$$

$$\dot{\Phi}^N_B(q^B,\dot{q}^B) = 0 \tag{73c}$$

The Lagrange multipliers at the velocity level can be estimated as

$$\lambda = \lambda^i + \alpha(\Phi_{q^A}\dot{q}^A + \Phi_{q^B}\dot{q}^B + \Phi_t) \tag{74a}$$

$$\lambda^N_A = \lambda^{N\,i}_A + \alpha\Phi^N_{q^A}\dot{q}^A \tag{74b}$$

$$\lambda^N_B = \lambda^{N\,i}_B + \alpha\Phi^N_{q^B}\dot{q}^B \tag{74c}$$

Substituting relations (74) to (71) and (72), the following final result is obtained

$$\begin{aligned}&(M^A + \Phi^T_{q^A}\alpha\Phi_{q^A} + \Phi^{N\,T}_{q^A}\alpha\Phi^N_{q^A})\dot{q}^A + \Phi^T_{q^A}\alpha\Phi_{q^B}\dot{q}^B + \Phi^{1\,T}_{q^A}\lambda_1 = \\ &= M^A\dot{q}^A_* - \Phi^T_{q^A}(\lambda^i + \alpha\Phi_t) - \Phi^{N\,T}_{q^A}\lambda^{N\,i}_{q^A}\end{aligned} \tag{75}$$

$$\begin{aligned}&\Phi^T_{q^B}\alpha\Phi_{q^A}\dot{q}^A + (M^B + \Phi^T_{q^B}\alpha\Phi_{q^B} + \Phi^{N\,T}_{q^B}\alpha\Phi^N_{q^B})\dot{q}^B + \Phi^{2\,T}_{q^B}\lambda_2 = \\ &= M^B\dot{q}^B_* - \Phi^T_{q^B}(\lambda^i + \alpha\Phi_t) - \Phi^{N\,T}_{q^B}\lambda^{N\,i}_{q^B}\end{aligned} \tag{76}$$

The equations (75) and (76) form a basis for the divide and conquer algorithm at the velocity level. Unknown improved velocities can be found by the iterative process, until

$\|\dot{q}-\dot{q}_i\|<\varepsilon_v$. The recursions are started by setting $\dot{q}_i^A=\dot{q}_*^A$, $\dot{q}_i^B=\dot{q}_*^B$, and $\lambda^i=0$, $\lambda_A^{N^i}=0$, $\lambda_B^{N^i}=0$. We can write the following generalized relations, which allows to minimize velocity constraint errors as

$$M_{11}^A\dot{q}_1^A+M_{12}^A\dot{q}_2^A+\Phi^{1\,T}_{q_1^A}\lambda_1=Q_1^A \tag{77}$$

$$M_{21}^A\dot{q}_1^A+M_{22}^A\dot{q}_2^A+\Phi^T_{q_2^A}\lambda=Q_2^A \tag{78}$$

$$M_{11}^B\dot{q}_1^B+M_{12}^B\dot{q}_2^B+\Phi^T_{q_1^B}\lambda=Q_1^B \tag{79}$$

$$M_{21}^B\dot{q}_1^B+M_{22}^B\dot{q}_2^B+\Phi^{2\,T}_{q_2^B}\lambda_2=Q_2^B \tag{80}$$

The objective is to find relations describing articulated body C (Fig. 1) in the form

$$M_{11}^C\dot{q}_1^A+M_{12}^C\dot{q}_2^B+\Phi^{1\,T}_{q_1^A}\lambda_1=Q_1^C \tag{81}$$

$$M_{21}^C\dot{q}_1^A+M_{22}^C\dot{q}_2^B+\Phi^{2\,T}_{q_2^B}\lambda_2=Q_2^C \tag{82}$$

where Lagrange multipliers at the velocity level can be estimated as

$$\lambda=\lambda^i+\alpha(\Phi_{q_2^A}\dot{q}_2^A+\Phi_{q_1^B}q_1^B+\Phi_t^i) \tag{83}$$

The recursive binary assembly process is defined in Eq. (47) to (52), and can be applied to Eq. (77) to (80). Again the matrix coefficients $M_{ij}^A$ and $M_{ij}^B$ ($i,j=1,2$) are the same as in case of Eq. (26) to (31) and (64) to (69). They must be evaluated once, after the mass-orthogonal projection in positions is performed.

### 3.4 ALF with Projections in Accelerations

The same procedure as in velocity projections can be applied to accelerations. Mass-orthogonal projections of accelerations onto acceleration constraint manifold are performed. Minimization of a functional ($\min_{\ddot{q}}\frac{1}{2}(\ddot{q}-\ddot{q}_*)^T M(\ddot{q}-\ddot{q}_*)$) subjected to constraint equations ($\ddot{\Phi}(q,\dot{q},\ddot{q},t)=0$) yields the following relations at the acceleration level (see Fig. 3)

$$M^A\ddot{q}^A+\Phi^{1\,T}_{q^A}\lambda_1+\Phi^T_{q^A}\lambda+\Phi^{N\,T}_{q^A}\lambda_A^N=M^A\ddot{q}_*^A \tag{84}$$

$$M^B\ddot{q}^B+\Phi^T_{q^B}\lambda+\Phi^{2\,T}_{q^B}\lambda_2+\Phi^{N\,T}_{q^B}\lambda_B^N=M^B\ddot{q}_*^B \tag{85}$$

where $\lambda$, $\lambda_1$, $\lambda_2$ are Lagrange multipliers associated with functional minimization with respect to perturbed accelerations $\ddot{q}_*^A$ and $\ddot{q}_*^B$. The Lagrange multipliers can be estimated as in equations (23), except for the fact, that constraint equations at the position and velocity level are satisfied, therefore the following terms can be written as $\bar{\gamma}^{AB}=\gamma^{AB}$, $\bar{\gamma}_N^A=\gamma_N^A$, $\bar{\gamma}_N^B=\gamma_N^B$. Equations (24) and (25) are valid with quantities defined as $Q^A=M^Aq_*^A$, $Q^B=M^Bq_*^B$. The same relations as in (26) to (32) can be written for projections in accelerations. The recursive binary assembly process is obtained through gathering matrix coefficients as in equations (47)

to (52). The recursions are started by setting $\ddot{q}_i^A = \ddot{q}_*^A$, $\ddot{q}_i^B = \ddot{q}_*^B$, and $\lambda^i = 0$, $\lambda_A^{N^i} = 0$, $\lambda_B^{N^i} = 0$ and continued until $\|\ddot{q} - \ddot{q}_i\| < \varepsilon_a$. The coefficients $M_{ij}^A$ and $M_{ij}^B$ $(i, j = 1,2)$ used in the process of projections in accelerations are the same as in case of velocity projections. They don't need to be reevaluated at this stage.

## 4 NUMERICAL EXPERIMENTS

This section presents some results of the numerical experiments, which are performed to prove the correctness of the algorithm as well as indicate some of its features. Several test cases are taken into consideration. The first one is a four bar linkage, which enables to investigate robustness of the formulation in case of singular configurations, and the second one is an open loop chain to indicate overall performance of the algorithm. All joints in sample test cases are revolute with rotation axis perpendicular to the plane of figures. The algorithms are implemented in Matlab using $ode45$ numerical integration routine (Runge-Kutta (4,5) formula). The absolute and relative tolerances are set to $10^{-6}$. The simulation time is $20.0$ seconds.

### 4.1 Singular Configurations

Figure 4 shows a four bar mechanism, which will serve as an example to demonstrate some features of the derived algorithms. The length $l = 1m$, and mass and moment of inertia of each body in the system equal to unity. The considered simple MBS presents a single loop case and includes redundant constraints, which make impossible to apply basic algorithm (section 2) or other standard formulations available in the literature. Moreover when angle $\varphi$ is close to zero, the mechanism may undergo singular configurations. In these situations, constraint Jacobian matrices become rank deficient and the system of equations can't be solved without additional interventions.

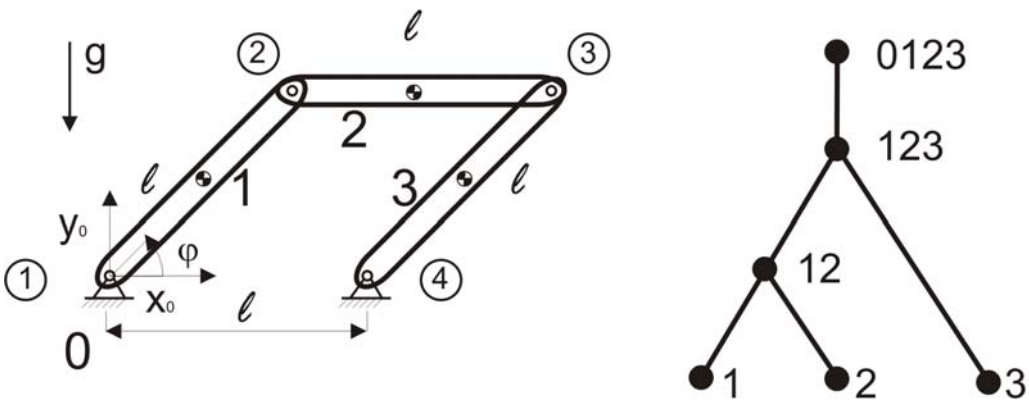


Figure 4: Four bar mechanism and binary assembly-disassembly tree.

At initial instant, the mechanism is in the configuration depicted in Fig. 4, where $\varphi = (\pi/4)rad$. The system is released from this position under gravity forces. The procedure of applying DCA with ALF and DCA with projections follows a binary tree presented in Fig. 4. Three equations at the form of (26), (27) and (28) are obtained as a starting point for binary assembly at acceleration level.

$$
\begin{aligned}
&(M_1 + \Phi_{q_1}^T \alpha \Phi_{q_1} + \Phi_{q_1}^{N\,T} \alpha \Phi_{q_1}^N)\ddot{q}_1 + \Phi_{q_1}^T \alpha \Phi_{q_2} \ddot{q}_2 + \Phi_{q_1}^{1\,T} \lambda_1 = \\
&= Q_1 - \Phi_{q_1}^{2\,T} \alpha \bar{\gamma}_{12} - \Phi_{q_1}^{N\,T} \alpha \bar{\gamma}_1^N
\end{aligned}
\tag{86}
$$

$$
\begin{aligned}
&\Phi_{q_2}^T \alpha \Phi_{q_1} \ddot{q}_1 + (M_2 + \Phi_{q_2}^T \alpha \Phi_{q_2} + \Phi_{q_2}^{N\,T} \alpha \Phi_{q_2}^N)\ddot{q}_2 + \Phi_{q_2}^{3\,T} \lambda_3 = \\
&= Q_2 - \Phi_{q_2}^{2\,T} \alpha \bar{\gamma}_{12} - \Phi_{q_2}^{N\,T} \alpha \bar{\gamma}_2^N
\end{aligned}
\tag{87}
$$

$$
(M_3 + \Phi_{q_3}^{4\,T} \alpha \Phi_{q_3}^4 + \Phi_{q_3}^{N\,T} \alpha \Phi_{q_3}^N)\ddot{q}_3 + \Phi_{q_3}^{3\,T} \lambda_3 = Q_3 - \Phi_{q_3}^{4\,T} \alpha \bar{\gamma}_{12} - \Phi_{q_2}^{N\,T} \alpha \bar{\gamma}_2^N \tag{88}
$$

Applying the procedures described in sections 3.1, we obtain a system of linear equations for body 1 accelerations, which corresponds to the root of the binary tree representation, from which other kinematical quantities may be found, traversing binary tree to leaves. Figure 5 shows the simulations results, where the algorithm with augmented Lagrangian formulation and with projections are set comparatively. User specified tolerances at position, velocity and acceleration level are chosen to $\varepsilon_p = 10^{-12}$, $\varepsilon_v = 10^{-10}$, $\varepsilon_a = 10^{-8}$ respectively, and the maximal number of iterations is limited to $10$. The penalty factor is set to $\alpha = 10^6$.

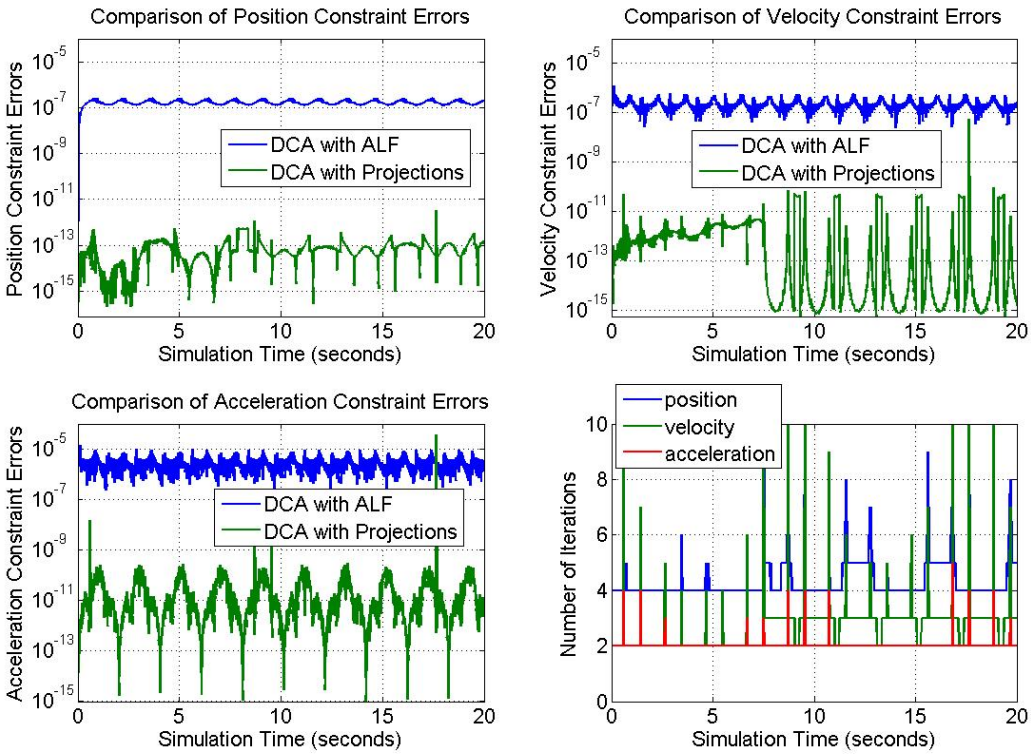


Figure 5: Constraint Violation Errors of four-bar mechanism.

Figure 5 presents the results that indicate good overall accuracy as well as constraint fulfillment for considered test case. Application of the the divide and conquer algorithm with projections reduces significantly constraint errors compared to DCA with augmented Lagrangian formulation. Both procedures do not fail near singular configurations, as basic algorithm does. As may be observed from Fig. 5, the DCA with projections requires more than few ite-

rations, to converge the solutions. This effect may be caused by the properties of presented algorithms, such as conditioning of associated matrices as well as singular positions encountered during the simulations and may affect abrupt changes of Lagrange multipliers and accelerations [38], [40].

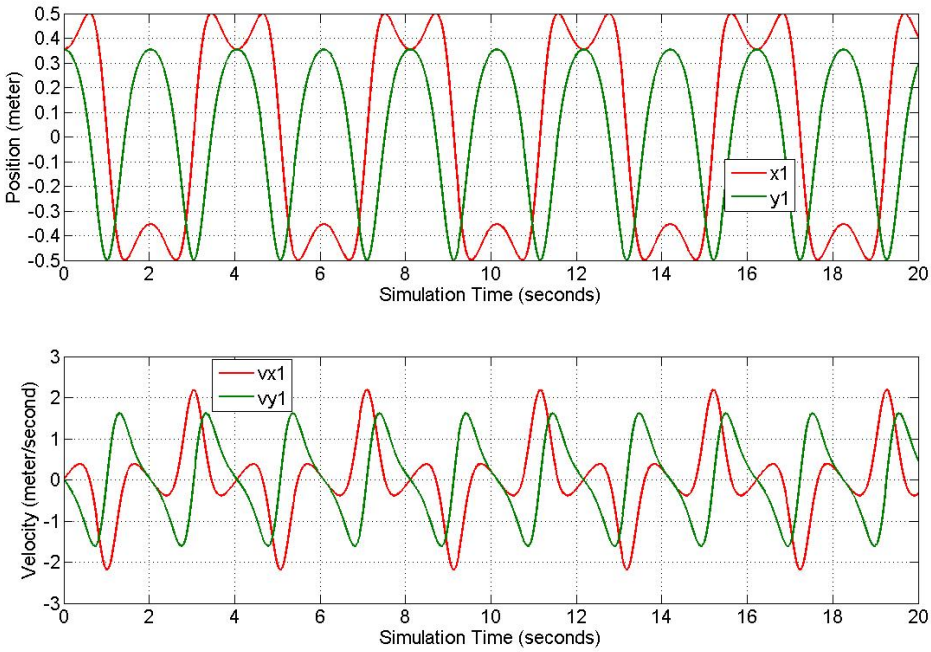


Figure 6: Results for Body 1 of the four bar mechanism.

Figure 6 presents selected kinematical quantities obtained from the dynamical analysis. The plot shows position $x_1$, $y_1$ and $\dot{x}_1$, $\dot{y}_1$ of the first body of the four bar mechanism. During the simulation course, the MBS undergoes 20 times through singular configuration. These motions are performed smoothly without difficulty and sudden changes in position and velocity values.

## 4.2 Open chain system

In order to investigate constraint fulfillment of the derived algorithms, a test mechanical system with sixteen degrees of freedom have been created, as shown in figure 7. Each body in the open chain is connected to each other by simple revolute joints. All axes of revolution are parallel to the global axis $z$. The numbering of the bodies increases successively from the non–movable base body $0$ to the terminal body $16$. The characteristic points and body mass centres were located on the sine function, as indicated in Fig. 7. The properties of each body in the chain are: mass $m_i = 1.0kg$, product of inertia expressed as a $3\times3$ diagonal matrix $I_i = diag(1.0)kgm^2$ with respect to the reference frames fixed at each body. At initial instant, absolute coordinates of bodies are nonzero and absolute velocities are set to zero. Distance between the reference points and the body mass center projected onto the global $x_o$ axis is constant and equals to $\pi/16$. The MBS is released from non-equilibrium state under gravity forces.

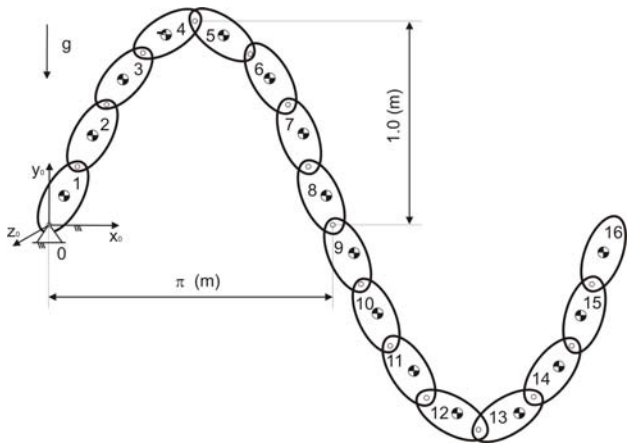


Figure 7: A 16 body open chain system.

Three algorithms are taken into account to compare the accuracy of formulations derived in this paper. The time histories of position, velocity and acceleration constraint errors are reported in Fig. 8, where the basic algorithm in absolute coordinates without stabilization (section 2), DCA with augmented Lagrangian formulation with one iteration (section 3.1) and DCA with mass-orthogonal projections (sections 3.2, 3.3, 3.4) are combined. The penalty coefficients $\alpha$ are set to $10^6$ globally. User specified tolerances at position, velocity and acceleration level are chosen to $\varepsilon_p = 10^{-12}$, $\varepsilon_v = 10^{-10}$, $\varepsilon_a = 10^{-8}$ and the number of iterations is limited to $10$.

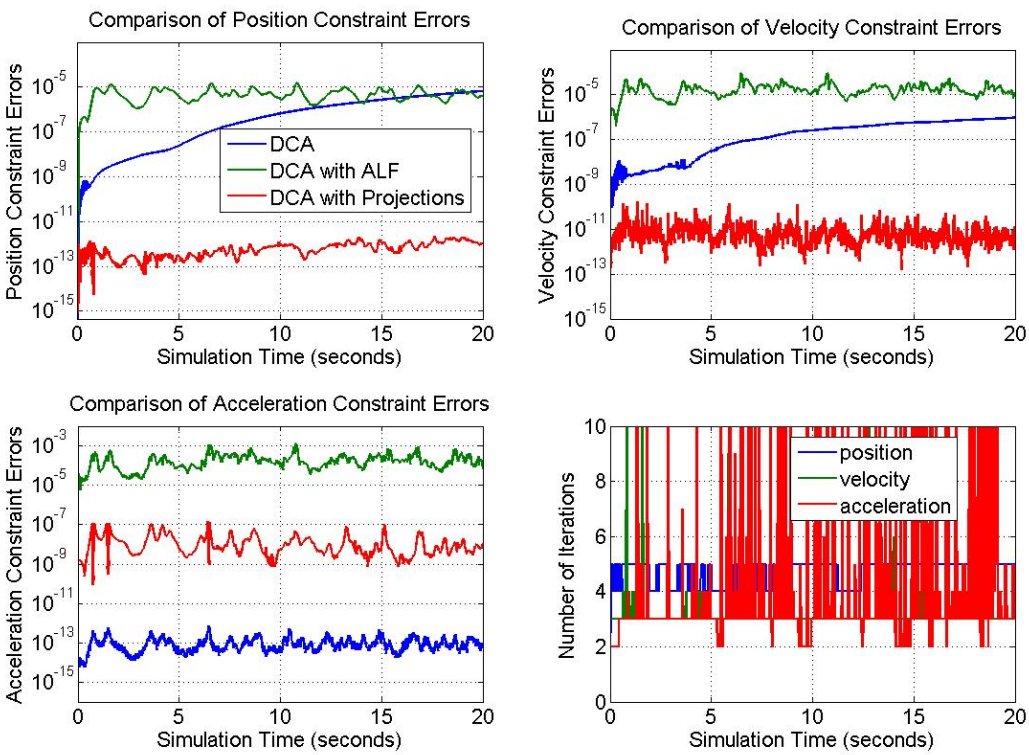


Figure 8: Constraint Violation Errors of open chain system.

From the results in Fig. 8, it may be observed that the basic algorithm, which is a kind of formulation with differential index 1, leads to significant constraint violation errors at the position and velocity levels. Explicitly imposed acceleration constraint are fulfilled during the simulation course. The algorithms with constraint refinement procedures allows to reduce constraint violation errors and obtain much improvement in accuracy. Figures 8 shows also improvement in accuracy of the DCA with mass-orthogonal projections over the DCA with ALF formulation. Few iterations at position and velocity levels are needed to converge solutions, however lot of expense is required to converge accelerations.

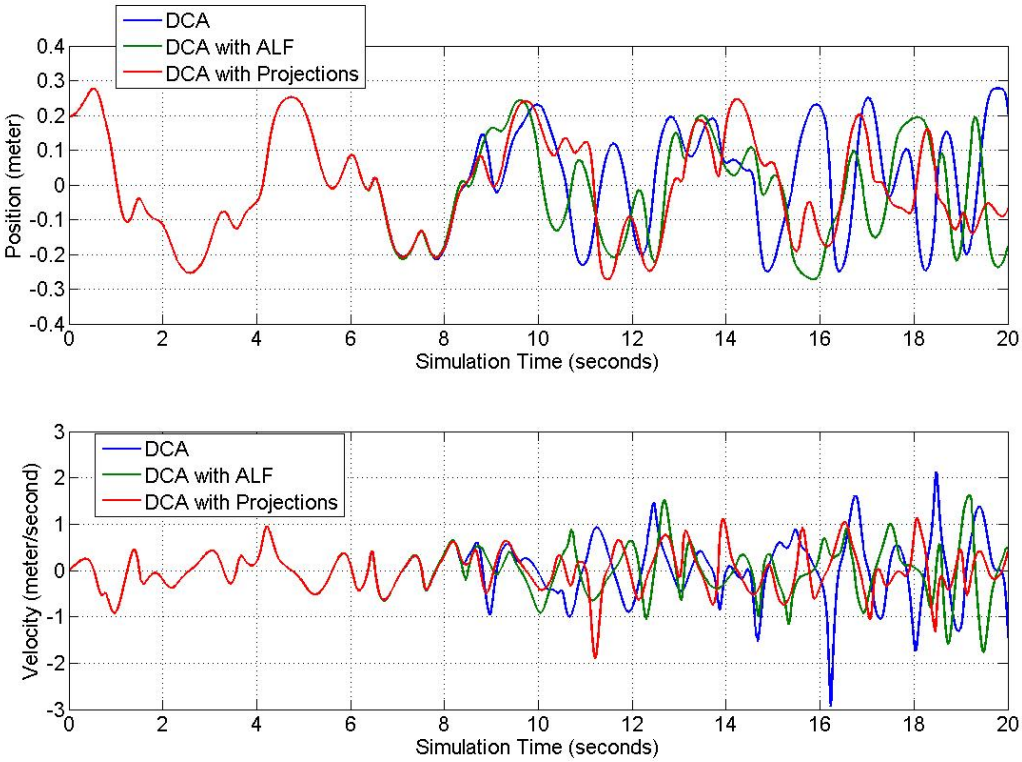


Figure 9: Position $x_1$ and velocity $\dot{x}_1$ for body 1 of open chain system.

Figure 9 shows time histories of the position $x_1$ and velocity $\dot{x}_1$ of the first body in the open chain system. It may be observed that for short simulation time, the curves are overlapped but they differs as the simulation proceeds towards end time, which results from the type of the formulation used.

### 4.3 Preliminary parallel efficiency results

Another issue, which should be considered is an efficiency of the derived algorithms. This preliminary study includes only the performance results of the basic divide and conquer algorithm, presented in section 2. Parallel performance measures of the derived algorithms with stabilizations, especially procedures with projections are areas of current research of the authors.

For demonstration purposes, the basic divide and conquer algorithm is implemented in Fortran programming language, in double precision arithmetic. The multibody code is parallelized by using OpenMP [43], [44], [45] compiler directives, chosen with respect to their simplicity and portability across shared memory architectures. The structure of the algorithm

allows easily to use parallel loop constructs that specify iterations of one or more loops to be executed in parallel by threads in the team. The basic DCA is implemented on a shared memory parallel computer (Sun Server) equipped with two-socket motherboard, in which two quad-core processors are installed. Each processor is a Quad-Core AMD Opteron Processor 2356 (2.3GHz) with 512kB cache L2 per core and four 2GB ECC DDR-667 memory modules are set in the motherboard.

A test mechanical system used for performance evaluation was described in section 4.2. The only change is that the system has adjustable number of bodies. When more bodies are added to the system, the chain is getting longer but the links maintain the same dimension. The simulation time is 10.0 seconds and the integration time step is set to 0.01 second. The equations of motion are integrated using fixed step size integrator Runge–Kutta of the fourth order without error check. Figure 10 presents performance characteristic obtained from the numerical experiments for $256$, $512$, $1024$, $2048$ and $4096$ bodies.

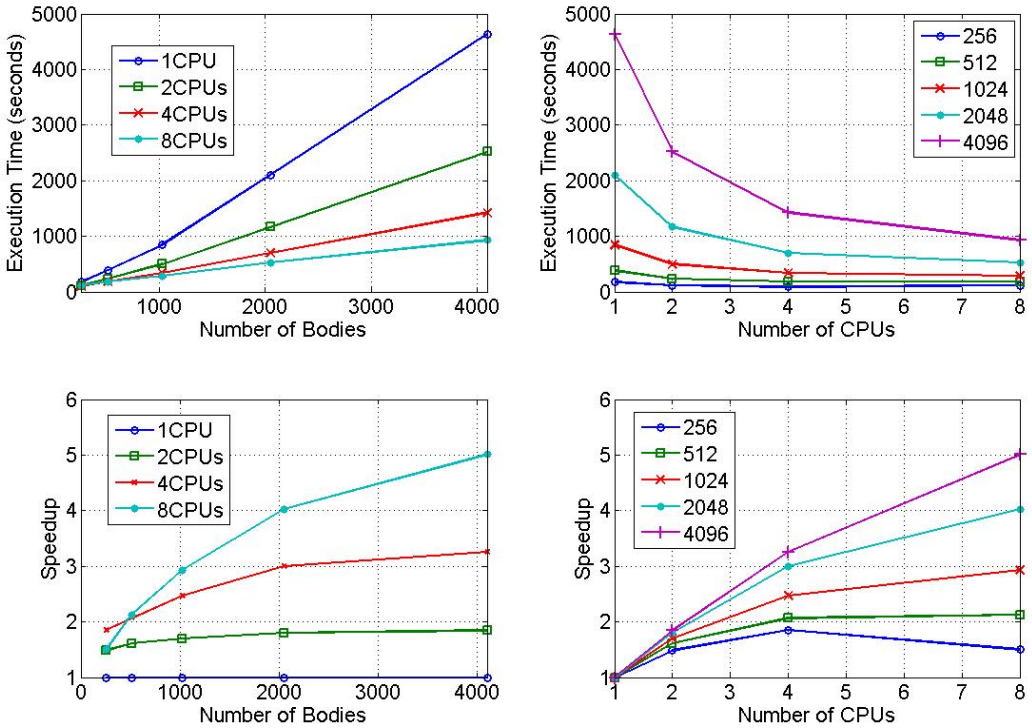


Figure 10: Parallel performance results for open chain system.

Figure 10 presents plots of execution times and speedups with respect to number of bodies and number of processing units used for computations. The results indicate that for the simple open chain system, the basic DCA algorithm offers good overall parallel efficiency. For small MBS timing data does not gain much benefits expected from parallel computations, especially when $8$ processing units are considered. The parallelization is more effective in case of large number of bodies presented in the system, where higher increase in computational load/thread may be observed.

## 5 CONCLUSIONS

In this paper, a new parallel procedures for application to general multibody system dynamics has been developed, verified and compared. The absolute coordinates divide and conquer algorithm is linear, when considered sequentially and logarithmic complexity for parallel implementation. The basic algorithm is exact. Due to the formulation properties, some form of stabilization techniques have been added to the basic procedure, obtaining parallel iterative algorithms useful in analyzing closed loop systems, especially those with redundant constraints and in singular positions encountered. The results from the experiments indicate good accuracy performance dependent on the expense put in the iterative refinement of constraint equations. Parallel implementation with OpenMP compiler directives for application on shared memory computers demonstrated useful speed increases that may be exploited in dynamics simulations of large multibody systems.

## ACKNOWLEDGEMENTS

This work has been supported by the European Union in the framework of European Social Fund through the Warsaw University of Technology Development Programme.

The work is co-financed by the European Regional Development Fund within the framework of the 1. priority axis of the Innovative Economy Operational Programme, 2007-2013 - PROTEUS under grant number PO IG.01.02.01-00-014/08-00.

## REFERENCES

[1] A. F. Vereshchagin. Computer simulation of the dynamics of complicated mechanisms of robot manipulators. *Engineering Cybernetics*, **6**, pp. 65-70, 1974.

[2] W. W. Armstrong. Recursive solution to the equations of motion of an n-link manipulator. In *Proceedings of the 5th World Congress on Theory of Machines and Mechanisms*, pp. 1343–1346, 1979.

[3] J. Luh, M. W. Walker, R. Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, **102**, pp. 69–76, 1980.

[4] M. W. Walker, D. E. Orin. Efficient Dynamic Computer Simulation of Robotic Mechanisms. *ASME Journal of Dynamic Systems, Measurements, and Control*, **104**(3), pp. 205–211, 1982.

[5] D. E. Rosenthal, M. A. Sherman. High Performance Multibody Simulations via Symbolic Equation Manipulation and Kane's Method. *The Journal of the Astronautical Sciences*, **34**(3), pp. 223–239, 1986.

[6] R. Featherstone. The calculation of robot dynamics using articulated–body inertias, *International Journal of Robotics Research*, **2**, pp. 13–30, 1983.

[7] D. S. Bae, E. J. Haug. A recursive formulation for constrained mechanical system dynamics: Part I: Open Loop Systems. *Mechanics of Structures and Machines*, **15**, pp. 359–382, 1987.

[8] D. Rosenthal. An order n formulation for robotic systems. *The Journal of the Astronautical Sciences*, **38**, pp. 511–529, 1990.

[9] G. Rodriguez. *Kalman Filtering, Smoothing and Recursive Robot Arm Forward and Inverse Dynamics*. IEEE Journal of Robotics and Automation, **3**(6), 1987.

[10] A. Jain, Unified formulation of dynamics for serial rigid multibody systems. *Journal of Guidance, Control, and Dynamics*, **14**, pp. 531–542, 1991.

[11] D. S. Bae, E. J. Haug, A recursive formulation for constrained mechanical system dynamics: Part II: Closed Loop Systems. Mechanics of Structures and Machines, **15**, pp. 481–506, 1987–88.

[12] R. Featherstone. Robot Dynamics Algorithms. Kluwer Academic Publishers, 1987.

[13] V. Stejskal, M. Valasek. *Kinematics and Dynamics of Machinery*. Marcel Dekker, NY, 1996.

[14] K. S. Saha, W. Schiehlen. *Recursive Kinematics and Dynamics for Parallel Structured Closed-Loop Multibody Systems*. Mechanics of Structures and Machines, **29**(2), pp. 143–175, 2001.

[15] K. S. Anderson, J. H. Critchley. *Improved 'Order-N' Performance Algorithm for the Simulation of Constrained Multi–Rigid–Body Dynamic Systems*. Multibody System Dynamics, **9**, pp. 185–225, 2003.

[16] J. H. Critchley, K. S. Anderson. *A generalized recursive coordinate reduction method for multibody system dynamics*. International Journal for Multiscale Computational Engineering, **1**, pp. 181–200, 2003.

[17] H. Kasahara, H. Fujii, M. Iwata. *Parallel processing of robot motion simulation*. In Proceedings IFAC World Congress, Munich, 1987.

[18] D. S. Bae, J. G. Kuhl, E. J. Haug. A recursive formulation for constrained mechanical system dynamics. Part III. Parallel Processor Implementation. *Mechanics of Structures and Machines*, **16**, pp. 249–269, 1988.

[19] C.S.G. Lee, P.R. Chang. Efficient Parallel Algorithms For Robot Forward Dynamics Computation. *IEEE Transactions on Systems, Man, and Cybernetics*, **18**, pp. 238–251, 1988.

[20] R. Lathrop. Parallelism in Manipulator Dynamics. *Technical Report 754, MIT Artificial Intelligence* Laboratory, 1984.

[21] R. S. Hwang , D. S. Bae, J. G. Kuhl, E. J. Haug. *Parallel Processing for Real-Time Dynamic System Simulation*. Journal of Mechanical Design, **112**, pp. 520–528, 1990.

[22] S. Chung, E. J. Haug. Real-time Simulation of Multibody Dynamics on Shared Memory Multiprocessors. *Journal of Dynamic Systems, Measurement, and Control*, **115**, pp. 627–637, 1993.

[23] A. Avello, J. M. Jimenez, E. Bayo, J. G. Jalon. A simple and highly parallelizable method for real–time dynamic simulation based on velocity transformations. *Computer Methods in Applied Mechanics and Engineering,* **107**, pp. 313–339, 1993.

[24] A. Fijany, A. K. Bejczy. Techniques for Parallel Computation of Mechanical Manipulator Dynamics. Part II: Forward Dynamics. *Control and Dynamics Systems*, **40**, pp. 357–410, 1991.

[25] A. Fijany, I. Sharf, G. D'Eleuterio. Parallel $O(\log n)$ Algorithms for Computation of Manipulator Forward Dynamics. IEEE Transactions on Robotics and Automation, **11**, pp. 389–400, 1995.

[26] R. Featherstone, A. Fijany. A Technique for Analyzing Constrained Rigid–Body Systems, and its Application to the Constraint Force Algorithm. *IEEE Transactions on Robotics and Automation*, **15**(6), pp. 1140–1144, 1999.

[27] P. Fisette, J. M. Peterkenne, Contribution to parallel and vector computation in multibody dynamics. *Parallel Computing*, **24**, pp. 717–728, 1998.

[28] K. S. Anderson, S. Duan. Highly Parallelizable Low–Order Dynamics Simulation Algorithm for Multi–Rigid–Body Systems. *Journal of Guidance, Control, and Dynamics*, **23**(2), pp. 355–364, 2000.

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

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

[31] J.H. Critchley, K.S. Anderson. A Parallel Logarithmic Order Algorithm for General Multibody System Dynamics. *Multibody System Dynamics*, **12**, pp. 75–93, 2004.

[32] R. Mukherjee, K. Anderson. Orthogonal Complement Based Divide-and-Conquer Algorithm for constrained multibody systems. *Nonlinear Dynamics*, **48**, pp. 199-215, 2007.

[33] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, **1**, pp. 1–16, 1972.

[34] K. C. Park, J. C. Chiou. Stabilization of computational procedures for constrained dynamical systems. *Journal of Guidance, Control, and Dynamics*, **11**, pp. 365-370, 1988.

[35] E. Bayo, J.G. de Jalon, M.A. Serna. A modified Lagrangian formulation for the dynamic analysis of constrained mechanical systems. *Computer Methods in Applied Mechanics and Engineering,* **71**, pp. 183–195, 1988.

[36] E. Bayo, A. Avello. Singularity-Free Augmented Lagrangian Algorithms for Constrained Multibody Dynamics. *Nonlinear Dynamics*, **5**, pp. 209-231, 1994.

[37] E. Bayo, R. Ledesma. Augmented lagrangian and mass-orthogonal projection methods for constrained multibody dynamics. *Nonlinear Dynamics*, **9**, pp. 113-130, 1996.

[38] J. Cuadrado, J. Cardenal, E. Bayo. Modeling and Solution Methods for Efficient Real-Time Simulation of Multibody Dynamics. Multibody System Dynamics, **1**, 259-280, 1997.

[39] K. S. Anderson. An order-n formulation for the motion simulation of general multi-rigid-body constrained systems. *Computers and Structures*, **43**, pp. 565-579, 1992.

[40] W. Blajer. Augmented Lagrangian Formulation: Geometrical Interpretation and Application to Systems with Singularities and Redundancy, *Multibody System Dynamics*, Vol. 8, 141–159, 2002.

[41] E. J.Haug. Computer Aided Kinematics and Dynamics of Mechanical Systems. *Allyn and Bacon*, 1989.

[42] P. Nikravesh. Computer-Aided Analysis of Mechanical Systems. *Prentice Hall*, 1988.

[43] A. Grama, A. Gupta, G. Karypis, V. Kumar. Introduction to Parallel Computing. *Addison Wesley*, 2003.

[44] T. G. Mattson, B. A. Sanders, B. L. Massingill. Patterns for Parallel Programming, *Addison-Wesley*, Boston, 2005.

[45] OpenMP Application Program Interface, Specification for Parallel Programming, Version 3.0, http://www.openmp.org, May 2008.

[46] N. J. Higham, Accuracy and Stability of Numerical Algorithms, *SIAM*, 2002.