A Numerically Robust LCP Solver for Simulating Articulated Rigid Bodies in Contact

Katsu Yamane
Dept. of Mechano-Informatics
University of Tokyo
Email: yamane@ynl.t.u-tokyo.ac.jp

Yoshihiko Nakamura
Dept. of Mechano-Informatics
University of Tokyo
Email: nakamura@ynl.t.u-tokyo.ac.jp

Abstract— **This paper presents a numerically robust algorithm for solving linear complementarity problems (LCPs), and applies it to simulation of frictional contacts of articulated rigid bodies each modeled as a general polygonal object. We first point out two problems of the popular pivot-based LCP solver called Lemke Algorithm and its extension with lexicographic ordering, due to numerical errors especially for ill-conditioned LCPs. Our new algorithm solves these problems by storing all pivot candidates and searching for a sequence of pivots that leads to a solution. An LCP-based contact dynamics formulation is combined with a forward dynamics algorithm for articulated rigid bodies to perform the whole simulation using a dynamic programming approach. Simulation examples using a humanoid robot show that the Lemke Algorithm (with or without lexicographic ordering) cannot solve complex contact problems, while our algorithm can successfully simulate such situations. We also demonstrate that the simulation results are qualitatively similar to those of hardware experiments.**

I. Introduction

Modeling collisions and contacts has been a long term research issue in both robotics and graphics. Most models can be categorized into penalty- and constraint-based methods. In penalty-based models, contact force at each contact point is modeled as the force exerted by a spring and damper. Although contact forces are easily computed from penetration depths and relative velocities, the approach suffers from numerical instability problem due to impulsive forces. This paper deals with the constraint-based approach, where we determine contact forces such that unilateral constraints on the post-contact relative motion and force are satisfied.

Constraint-based approaches often employ linear complementarity problem (LCP) [1] to formulate the constraints. An n-dimensional LCP is to find a set of vectors $\boldsymbol{w} \in \boldsymbol{R}^n$ and $\boldsymbol{z} \in \boldsymbol{R}^n$ that satisfy

$$\boldsymbol{w} = \boldsymbol{M}\boldsymbol{z} + \boldsymbol{q} \tag{1}$$

$$\boldsymbol{w} \geq 0,\ \boldsymbol{z} \geq 0,\ \boldsymbol{w}^T\boldsymbol{z} = 0 \tag{2}$$

for a given square matrix $\boldsymbol{M} \in \boldsymbol{R}^{n\times n}$ and vector $\boldsymbol{q} \in \boldsymbol{R}^n$. In the rest of the paper, we shall denote condition (2) as

$$\boldsymbol{w} \geq 0 \perp \boldsymbol{z} \geq 0. \tag{3}$$

LCPs can be solved by either iterative or pivot-based approach. Iterative approaches (e.g. [2]) utilize the fact that the solution of an LCP is the equilibrium point of the associated quadratic cost function and employ numerical root-finding techniques such as Newton's method to find the equilibrium. Pivot-based approaches (such as Lemke Algorithm [3]), on the other hand, sequentially pivot a pair of elements of $\boldsymbol{w}$ and $\boldsymbol{z}$ according to specific rules until all elements of $\boldsymbol{q}$ of the pivoted equation become zero or positive. Once such pivot sequence is found, we can obtain the pivoted solution by setting $\boldsymbol{w} = \boldsymbol{q}$, $\boldsymbol{z} = 0$ and then moving the pivoted elements back to the original vectors.

Iterative approaches are generally easier to implement and numerically robust, although convergence is proven only for a limited class of $\boldsymbol{M}$. Pivot-based approaches are theoretically guaranteed to find a solution with finite number of trials (2^n) for general problems, and several systematic procedures are proposed to efficiently find a solution [1]. However, it is known that pivot-based approaches often suffer from numerical problem especially for large-scale and/or ill-conditioned problems.

There have been a body of research on developing efficient and robust methods for solving LCPs in the context of collision/contact modeling. Jourdan et al. [4] applied an iterative LCP solver similar to Gauss-Seidel algorithm to frictional contacts of rigid bodies and proved convergence in most practical cases. Förg et al. [5] utilized the sparsity of $\boldsymbol{M}$ to accelerate an iterative LCP solver. Stewart et al. [6] formulated frictional contacts between rigid bodies as an LCP and applied Lemke Algorithm. Lloyd [7] also utilized the structure of $\boldsymbol{M}$ in rigid-body contact model for reducing the computational cost for Lemke Algorithm. Guendelman et al. [8] combined a number of stabilization techniques to obtain visually plausible simulation results for highly complex scenes. All of these papers address contact dynamics between free rigid bodies, in which case $\boldsymbol{M}$ is generally sparse and the LCP is likely to be relatively easily solved by both iterative and pivot-based approaches.

In contrast, we are interested in modeling frictional contacts between articulated rigid bodies, each represented as a general polygonal object. A possible solution is to treat all links as free rigid bodies and extend the work described above to solve both unilateral (contact) and bilateral (joint) constraints at the same time. With such modeling strategy $\boldsymbol{M}$ would have similar structure as in free rigid-body case. This type of model has been employed in some dynamics engines such as [9]. However, integrating linear and angular accelerations of each rigid body independently occasionally breaks joint constraints, which should be corrected by applying heuristic recovering forces. Weinstein et al. [10] takes a different approach where joint constraints are maintained by sequentially applying impulses to joints, while contact constraints are handled as described in [8]. Although these models can generate visually plausible results for highly complex scenes, they are not suitable for applications that require physical precision.

Another approach is to combine a forward dynamics algorithm for articulated rigid bodies such as [11], [12] with an LCP-based contact formulation. Kokkevis [13] utilized Articulated-Body Algorithm [11] for computing the mass matrix in the LCP formulation and applied an iterative algorithm for solving the LCP. Gayle et al. [14] applied an adaptive forward dynamics algorithm [15] based on Divide-and-Conquer Algorithm (DCA) [12] to a collision and contact model based on Mirtich et al. [16]. Kry and Pai [17] derived an LCP-based contact formulation for simulating interactions between a rigid body and compliant fingers.

The general problem of the latter approach is that the associated LCP tends to be dense and ill-conditioned, in which case iterative methods do not guarantee convergence to a solution. In this paper, we pursue the application of Lemke Algorithm [3] to simulation of articulated rigid bodies under frictional contacts because of its potential generality, although its numerical robustness should be considerably improved to be practically applicable to complex problems. Lemke Algorithm has been successfully applied to contact problems of articulated rigid bodies in Kry and Pai [17], but they only consider one contact point per finger and hand dynamics is represented by finger compliance rather than its inertia.

The main contribution of this paper is improvement of Lemke Algorithm to deal with large-scale and ill-conditioned LCPs derived from frictional contacts between articulated rigid bodies of arbitrary geometry. The contact dynamics is formulated in a similar way as Stewart et al. [6], while the spatial mass matrix of free rigid bodies are replaced by inverse articulated-body inertias (IABI) [11] at all contact points. The contact model is combined with a forward dynamics algorithm called Assembly-Disassembly Algorithm (ADA) [18], which internally uses IABI for resolving the joint constraints and therefore fits well with our contact formulation.

A well-known extension of Lemke Algorithm is *lexicographic ordering* [1], [19] to solve *cycling* problem where the same pivot sequence is infinitely repeated when an inappropriate pivot choice is made. The problem is often encountered in ill-conditioned problems and the extension has been employed in [6], [7], [20].

Although lexicographic ordering can theoretically avoid cycling, we found that it is not enough for solving our contact problem under round-off errors. We will also point out another practical problem of numerical instability that, to our knowledge, has never been described in literature. Our solver addresses both of these problems. The basic idea of the method is to store all possible pivots at each pivot step and, in case a particular choice of pivot sequence resulted in an infinite pivot loop or numerical instability, track back the queue of possible pivots and try other possible pivots. In other words, the method searches for the best pivot sequence that leads to a solution of the LCP.

The rest of the paper is organized as follows. Section II reviews the Lemke Algorithm and point out the problems caused by round-off errors. In section III, we describe our numerically robust algorithm for solving LCPs. Section IV presents the LCP formulation of frictional contacts of articulated rigid bodies, along with several implementation issues. In Section V, we first show that the problems described in Section II actually happen in practical simulations using a simple example, and then demonstrate the robustness of the proposed solver by a number of simulation examples. Finally we conclude the paper in Section VI.

II. Lemke Algorithm

A. Algorithm Outline

We first show the outline of Lemke Algorithm [3] as explained in [7]. In general, pivot-based methods try to find a partition of Eq.(1):

$$\begin{pmatrix} \boldsymbol{w}_{\tilde{\alpha}} \\ \boldsymbol{w}_{\tilde{\beta}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{M}_{\tilde{\alpha}\alpha} & \boldsymbol{M}_{\tilde{\alpha}\beta} \\ \boldsymbol{M}_{\tilde{\beta}\alpha} & \boldsymbol{M}_{\tilde{\beta}\beta} \end{pmatrix} \begin{pmatrix} \boldsymbol{z}_{\alpha} \\ \boldsymbol{z}_{\beta} \end{pmatrix} + \begin{pmatrix} \boldsymbol{q}_{\tilde{\alpha}} \\ \boldsymbol{q}_{\tilde{\beta}} \end{pmatrix} \quad (4)$$

such that the pivoted system

$$\begin{pmatrix} \boldsymbol{z}_{\alpha} \\ \boldsymbol{w}_{\tilde{\beta}} \end{pmatrix} = \boldsymbol{M}' \begin{pmatrix} \boldsymbol{w}_{\tilde{\alpha}} \\ \boldsymbol{z}_{\beta} \end{pmatrix} + \boldsymbol{q}' \quad (5)$$

satisfies the following conditions:

Condition 1: $\boldsymbol{w}_{\tilde{\alpha}}$ and $\boldsymbol{z}_{\alpha}$ contain the same set of indices, and

Condition 2: $\boldsymbol{q}' \geq 0$.

The vectors $(\boldsymbol{z}_{\alpha}^T\ \boldsymbol{w}_{\tilde{\beta}}^T)^T$ and $(\boldsymbol{w}_{\tilde{\alpha}}^T \boldsymbol{z}_{\beta}^T)^T$ are called *basic* and *non-basic* variables, respectively.

$\boldsymbol{M}'$ and $\boldsymbol{q}'$ are computed from the original matrix and vector as follows:

$$\boldsymbol{M}' = \begin{pmatrix} \boldsymbol{M}_{\tilde{\alpha}\alpha}^{-1} & -\boldsymbol{M}_{\tilde{\alpha}\alpha}^{-1}\boldsymbol{M}_{\tilde{\alpha}\beta} \\ \boldsymbol{M}_{\tilde{\beta}\alpha}\boldsymbol{M}_{\tilde{\alpha}\alpha}^{-1} & \boldsymbol{M}_{\tilde{\beta}\beta} - \boldsymbol{M}_{\tilde{\beta}\alpha}\boldsymbol{M}_{\tilde{\alpha}\alpha}^{-1}\boldsymbol{M}_{\tilde{\alpha}\beta} \end{pmatrix} \quad (6)$$

$$\boldsymbol{q}' = \begin{pmatrix} \boldsymbol{q}'_{\tilde{\alpha}} \\ \boldsymbol{q}'_{\tilde{\beta}} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{M}_{\tilde{\alpha}\alpha}^{-1}\boldsymbol{q}_{\tilde{\alpha}} \\ \boldsymbol{q}_{\tilde{\beta}} - \boldsymbol{M}_{\tilde{\beta}\alpha}\boldsymbol{M}_{\tilde{\alpha}\alpha}^{-1}\boldsymbol{q}_{\tilde{\alpha}} \end{pmatrix}. \quad (7)$$

Once such pivot is found, we can easily obtain the solution as $\boldsymbol{w}_{\tilde{\alpha}} = 0, \boldsymbol{w}_{\tilde{\beta}} = \boldsymbol{q}'_{\tilde{\beta}}, \boldsymbol{z}_{\alpha} = \boldsymbol{q}'_{\tilde{\alpha}}, \boldsymbol{z}_{\beta} = 0$.

Lemke Algorithm is one of the systematic methods to efficiently find an appropriate pivot. In Lemke Algorithm, we first introduce an auxiliary variable z_0 and modify the original LCP (1) as follows:

$$\boldsymbol{w} = \bar{\boldsymbol{M}} \begin{pmatrix} \boldsymbol{z} \\ z_0 \end{pmatrix} + \boldsymbol{q} \quad (8)$$

where

$$\begin{aligned} \bar{\boldsymbol{M}} &= \begin{pmatrix} \boldsymbol{M} & \boldsymbol{c} \end{pmatrix} \\ \boldsymbol{c} &= (1\ 1\ \dots\ 1)^T. \end{aligned}$$

The solution of Eq.(8) can be found by the following steps:

Step 0 If $\boldsymbol{q} \geq 0$, stop: $\boldsymbol{w} = \boldsymbol{q}, \boldsymbol{z} = 0$ is the solution. Otherwise, obtain $r = \arg\min q_i/c_i$ and pivot z_0 with w_r. Compute $\bar{\boldsymbol{M}}'$ and $\boldsymbol{q}'$, and set the *driving variable* $y_r = z_r$.

Step 1 Let $\boldsymbol{m}'$ denote the column vector of $\bar{\boldsymbol{M}}'$ corresponding to y_r. If $\boldsymbol{m}' \geq 0$, stop: there is no solution or this algorithm cannot solve the LCP. Otherwise, obtain $s = \arg\min\{-q_i/m'_i : m'_i \leq 0\}$ and let y_s denote the s-th element of the basic variables.

Step 2 Pivot y_s with y_r and update $\bar{\boldsymbol{M}}'$ and $\boldsymbol{q}'$. If $y_s = z_0$, stop: $\boldsymbol{q}'$ gives the solution. Otherwise set y_r to the complement of y_s and return to Step 1.

After Step 0, $\boldsymbol{q}' \geq 0$ holds with the choice of r and the update rule Eq.(7). Similarly, all elements of subsequent $\boldsymbol{q}'$ are always equal to or larger than 0 with the choice of s in Step 1. The second condition above is therefore satisfied at every iteration. In Step 2, the first condition is met by setting the driving variable to the complement of the previously pivoted basic variable y_s, and by terminating when z_0 returns to a non-basic variable.

B. Problems of Lemke Algorithm

A well-known problem of Lemke Algorithm is that the minimum ratio test in Step 1 can result in tie, i.e. $-q_i/m'_i$ can take the same minimum value at multiple i's, in which case the LCP is said to be *degenerate*. This problem often occurs when the LCP includes redundant constraints, such as when there are more than three contact points between a pair of rigid bodies. Inappropriate choice of pivot in such cases can lead to cycling and should be avoided. It is known that an extension of Lemke Algorithm by *lexicographic ordering* [1], [19] (Lexicographic Lemke Algorithm) can resolve the tie by considering additional columns of $\bar{\boldsymbol{M}}'$.

According to our experience, however, this solution still has a problem if the algorithm is implemented and executed on computers. In computer programs, exact tie of floating-point numbers almost never happens due to round-off errors even if two numbers are analytically equal. They would have very small difference and Step 1 would proceed without encountering a tie. However, the choice of pivot in such situations does not have any logical basis and, if the choice was inappropriate, the algorithm would fall into cycles. Alternatively, we could set some small threshold to determine if two values are equal. This approach however imposes another issue of choosing an appropriate threshold because if the threshold is too small the desirable pivot may be discarded due to numerical errors, and if too large even lexicographic ordering may not be able to discriminate the best pivot choice.

Another problem that, to our knowledge, has never been addressed in literature is that $\boldsymbol{M}_{\bar{\alpha}\alpha}$ to be inverted in Eq.(6) may be close to singular with some of the pivots encountered during the process. In such cases, even if a solution is found, it may have large error with respect to the original equation (1). In contact simulation, this problem would result in physically unrealistic behavior such as penetration.

After presenting our new LCP solver and contact model in the following sections, we will demonstrate that these numerical problems actually occur in real simulation in Section V-A.

III. Robust Pivot-Based Solver for LCPs

The idea of our new algorithm is to store all the pivot candidates at every iteration of Steps 1 and 2, and return to them when cycling or numerical problem occurs. We store the i-th row as a pivot candidate at Step 1 if $m'_i < 0$ and the minimum element of $\boldsymbol{q}'$ after pivoting at row i is larger than a user-defined threshold. The threshold is usually chosen as a negative value with small absolute value to allow round-off errors. We define the cost function as a decreasing function of the minimum value of $\boldsymbol{q}'$ to prioritize pivot sequences with smaller errors.

In the algorithm, we construct a search tree composed of nodes each representing one pivot between a pair of basic and non-basic variables. The descendants of a node represent the possible pivots found in Step 1. The goal node is the one that includes z_0 in the pivot pair, and a successful sequence of pivots is reconstructed by tracing the ancestor nodes from the goal. We also construct a queue of nodes in which the nodes are sorted in the ascending order of the cost associated to each node.

Algorithm 1 shows the higher-level search algorithm, where

- Q is a queue of nodes,
- $Q.appendNode(x)$ adds a new node x to the queue,
- $Q.getBest()$ finds and pops the node with the smallest total cost,
- $x.isGoal()$ determines if node x is a goal by checking if z_0 is in the non-basic variables, and
- $Q.addDescendants(x)$ adds all possible descendant nodes of x to Q.

Details of $Q.addDescendants(x)$ is described in Algorithm 2, where

- $x.q\_min(i)$ computes the smallest element of $\boldsymbol{q}'$ after the i-th basic variable is pivoted,
- ϵ is a user-defined positive constant,
- $x.newNode(i)$ creates a new node representing a pivot of the i-th basic variable, and
- $x'.error()$ computes the norm of the current error $\boldsymbol{w}_{x'} - \boldsymbol{M}\boldsymbol{z}_{x'} - z_{0x'}\boldsymbol{c} - \boldsymbol{q}$ where $\boldsymbol{w}_{x'}$, $\boldsymbol{z}_{x'}$ and $z_{0x'}$ are the values of $\boldsymbol{w}$, $\boldsymbol{z}$ and z_0 after performing the pivot x',
- e_{max} is a user-defined permissible numerical error,
- $Q.unique(x')$ returns true if Q does not include the same pivot set as x'.

After updating $\boldsymbol{M}'$ and $\boldsymbol{q}'$ in accordance with the current pivot set, the $addDescendants()$ function checks the minimum element of $\boldsymbol{q}'$ when the i-th basic variable is further pivoted (line 4). The minimum value would ideally be zero when $-q_i/m'_i$ is the minimum and negative otherwise. In our problem, however, there may be multiple i's that yield small negative q_{min} due to round-off errors as mentioned in the previous section. We keep such rows as pivot candidates if $q_{min} > -\epsilon$ ($\epsilon > 0$) (line 5). For each pivot candidate, we

Algorithm 1 Search Pivot Sequence

```
Require: an LCP
1: perform Step 0
2: create initial (dummy) node $x_0$
3: $Q.appendNode(x_0)$
4: while $Q$ not empty do
5:    $x \leftarrow Q.getBest()$
6:    if $x.isGoal()$ then
7:       return $x$
8:    end if
9:    $Q.addDescendants(x)$
10: end while
11: return NULL
```

Algorithm 2 $Q.addDescendants(x)$

```
1: update $\bar{M}'$ and $q'$
2: for $i = 1, 2, \ldots, n$ do
3:    if $m'_i < 0$ then
4:       $q_{min} \leftarrow x.q\_min(i)$
5:       if $q_{min} > -\epsilon$ then
6:          $x' \leftarrow x.newNode(i)$
7:          $x'.totalCost \leftarrow x.totalCost + \exp(-q_{min})$
8:          if $x'.error() \leq e_{max}$ and $Q.unique(x')$ then
9:             $Q.appendNode(x')$
10:         end if
11:      end if
12:   end if
13: end for
```

verify that the error is smaller than the user-defined permissible error and that the same pivot set has never been visited before to avoid cycling (line 8).

The advantage of this method over directly comparing $-q_i/m'_i$ as in Lemke Algorithm is that $|q_{min}|$ has a clear physical meaning: either contact force or relative velocity in the normal direction, and therefore it is much easier to choose the threshold. The cost of each node is computed by $\exp(-q_{min})$ (line 7), which takes the maximum value $\exp(\epsilon)$ when $q_{min} = -\epsilon$. This cost penalizes the pivots with larger error, and as a result the optimal solution would be more physically reasonable.

Choosing ϵ and e_{max} is much easier than it would be with the threshold for determining tie in lexicographic ordering because we only have to make sure that it is sufficiently large not to drop correct pivots from the candidate list. Larger threshold degrades the speed because more candidates are kept in the queue, but it does not harm the result because the pivot sequence with the minimum cost is chosen anyway.

Updating M' and q' (line 1) based on Eqs. (6)(7) requires the inversion of a matrix of the size of pivot number, which can be computationally expensive for large problems. In fact, the update can be performed incrementally with less computational cost by using the M' and q' in the direct ancestor [7].

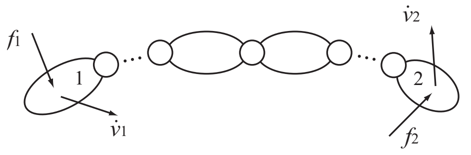

Fig. 1. Inverse articulated-body inertia.

IV. Contact Model for Articulated Rigid Bodies

A. IABI [11]

Inverse articulated-body inertia (IABI) is the inverse of the apparent inertia matrix of articulated bodies. This matrix describes the relationship between a force applied to a link called *handle* and resulting spatial acceleration, at current configuration. Furthermore, we can consider multiple handles, in which case we need m^2 IABIs to describe the relationship between forces and accelerations of m handles. In Fig. 1, for example, suppose links 1 and 2 are handles. The relationship between forces $\boldsymbol{f}_1, \boldsymbol{f}_2$ and accelerations $\dot{\boldsymbol{v}}_1, \dot{\boldsymbol{v}}_2$ is described using IABIs $\boldsymbol{\Phi}_{ij}$ $(i, j = 1, 2)$ as follows:

$$\begin{pmatrix} \dot{\boldsymbol{v}}_1 \\ \dot{\boldsymbol{v}}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Phi}_{11} & \boldsymbol{\Phi}_{12} \\ \boldsymbol{\Phi}_{21} & \boldsymbol{\Phi}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix} + \begin{pmatrix} \boldsymbol{\phi}_1 \\ \boldsymbol{\phi}_2 \end{pmatrix} \tag{9}$$

where $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are the bias accelerations of links 1 and 2, respectively. IABIs can be computed recursively as described in [12].

Forward dynamics algorithms such as DCA [12] and ADA [18] utilize IABI to describe the equation of motion of articulated bodies. We may be able to directly use IABIs in our contact model. The contact model of Kokkevis [13] is based on Articulated-Body Algorithm (ABA) [11], which uses articulated-body inertia (ABI) rather than IABI. In [13], IABIs are computed by applying unit test forces to the contact links and computing link accelerations by ABA. Gayle et al. [14] uses an extension of DCA as the basic forward dynamics engine; however, they apply a method similar to [13] to compute IABIs between contact links. In our implementation, we explicitly specify contact links as handles and let the forward dynamics algorithm compute IABIs between contact links.

B. LCP Formulation of Contacts

We apply the formulation in [7] to articulated rigid bodies, whose dynamics is represented by ABIs instead of spatial inertia matrix of rigid bodies.

Suppose N_L links are mutually in contact at N_C contact points. We first compute IABIs $\hat{\boldsymbol{\Phi}}_{ij}$ $(i, j = 1, 2, \ldots, N_L)$ and bias accelerations $\hat{\boldsymbol{\phi}}_i$ $(i = 1, 2, \ldots, N_L)$ of the N_L links using DCA or ADA. These IABIs describe the relationship between forces applied to contact links and their accelerations as

$$\dot{\hat{\boldsymbol{v}}} = \hat{\boldsymbol{\Phi}} \hat{\boldsymbol{f}} + \hat{\boldsymbol{\phi}} \tag{10}$$

where $\hat{\boldsymbol{v}} \in \boldsymbol{R}^{6N_L}$ and $\hat{\boldsymbol{f}} \in \boldsymbol{R}^{6N_L}$ are vectors composed of the spatial velocities and forces of all links respectively, and

$$\begin{aligned}\hat{\boldsymbol{\Phi}} &= \begin{pmatrix} \hat{\boldsymbol{\Phi}}_{11} & \hat{\boldsymbol{\Phi}}_{12} & \dots & \hat{\boldsymbol{\Phi}}_{1N_L} \\ \hat{\boldsymbol{\Phi}}_{21} & \hat{\boldsymbol{\Phi}}_{22} & \dots & \hat{\boldsymbol{\Phi}}_{2N_L} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\boldsymbol{\Phi}}_{N_L 1} & \hat{\boldsymbol{\Phi}}_{N_L 2} & \dots & \hat{\boldsymbol{\Phi}}_{N_L N_L} \end{pmatrix} \\ \hat{\boldsymbol{\phi}} &= \begin{pmatrix} \hat{\boldsymbol{\phi}}_1^T & \hat{\boldsymbol{\phi}}_2^T & \dots & \hat{\boldsymbol{\phi}}_{N_L}^T \end{pmatrix}^T.\end{aligned}$$

Let $\boldsymbol{f} \in \boldsymbol{R}^{3N_C}$ and $\boldsymbol{v} \in \boldsymbol{R}^{3N_C}$ denote contact forces and relative velocities at contact points, respectively. The relationship between forces and velocities of links and contact points can be described by a Jacobian matrix $\boldsymbol{J}$ as

$$\boldsymbol{v} = \boldsymbol{J}\hat{\boldsymbol{v}} \tag{11}$$

$$\hat{\boldsymbol{f}} = \boldsymbol{J}^T \boldsymbol{f}. \tag{12}$$

Substituting Eqs.(11)(12) into Eq.(10) yields

$$\begin{aligned}\dot{\boldsymbol{v}} &= \boldsymbol{J}\hat{\boldsymbol{\Phi}}\boldsymbol{J}^T\boldsymbol{f} + \boldsymbol{J}\hat{\boldsymbol{\phi}} + \dot{\boldsymbol{J}}\hat{\boldsymbol{v}} \\ &= \boldsymbol{\Phi}\boldsymbol{f} + \boldsymbol{\phi} \qquad (13) \\ \boldsymbol{\Phi} &\stackrel{\triangle}{=} \boldsymbol{J}\hat{\boldsymbol{\Phi}}\boldsymbol{J}^T \\ \boldsymbol{\phi} &\stackrel{\triangle}{=} \boldsymbol{J}\hat{\boldsymbol{\phi}} + \dot{\boldsymbol{J}}\hat{\boldsymbol{v}}\end{aligned}$$

which represents the dynamics at contact points.

We then discretize the equation of motion Eq.(13). Let $\boldsymbol{v}^-$ and $\boldsymbol{v}^+$ denote relative velocities at contact points before and after the current integration. Assuming that we apply Euler integration with time step Δt, we can write $\boldsymbol{v}^+$ as

$$\boldsymbol{v}^+ = \bar{\boldsymbol{\Phi}}\boldsymbol{f} + \bar{\boldsymbol{\phi}} \tag{14}$$

where

$$\bar{\boldsymbol{\Phi}} = \Delta t \boldsymbol{\Phi} \tag{15}$$

$$\bar{\boldsymbol{\phi}} = \boldsymbol{v}^- + \Delta t \boldsymbol{\phi}. \tag{16}$$

We now derive an LCP formulation of unilateral constraints to model the contact, similar to the one used in [7]. The friction cone is approximated by an M-sided polyhedral cone. We also assume that each contact point has the same static and slip friction coefficients. Let $\boldsymbol{n}_i$ denote the normal vector at contact i, and $\boldsymbol{c}_{im}$ $(m = 1, 2, \dots, M)$ the normal vectors of the side faces of the cone projected onto the contact tangential plane and normalized.

We write the contact force at contact point i, $\boldsymbol{f}_i$, as a linear combination of $\boldsymbol{n}_i$ and $\boldsymbol{c}_{im}$ $(m = 1, 2, \dots, M)$ by the non-negative coefficients a_i and b_{ik} $(k = 1, 2, \dots, M)$, i.e.

$$\begin{aligned}\boldsymbol{f}_i &= a_i \boldsymbol{n}_i + \sum_{m=1}^{M} b_{im}\boldsymbol{c}_{im} \\ &= a_i \boldsymbol{n}_i + \boldsymbol{C}_i \boldsymbol{b}_i \qquad (17)\end{aligned}$$

where

$$\boldsymbol{C}_i = \begin{pmatrix} \boldsymbol{c}_{i1} & \boldsymbol{c}_{i2} & \dots & \boldsymbol{c}_{iM} \end{pmatrix} \in \boldsymbol{R}^{3\times M} \tag{18}$$

$$\boldsymbol{b}_i = \begin{pmatrix} b_{i1} & b_{i2} & \dots & b_{iM} \end{pmatrix}^T \in \boldsymbol{R}^M. \tag{19}$$

By combining Eq.(17) at all contact points, we obtain

$$\boldsymbol{f} = \boldsymbol{N}\boldsymbol{a} + \boldsymbol{C}\boldsymbol{b} \tag{20}$$

where

$$\begin{aligned}\boldsymbol{N} &= diag\{\boldsymbol{n}_i\} \in \boldsymbol{R}^{3N_C \times N_C} \\ \boldsymbol{a} &= \begin{pmatrix} a_1 & a_2 & \dots & a_{N_C} \end{pmatrix}^T \\ \boldsymbol{C} &= diag\{\boldsymbol{C}_i\} \in \boldsymbol{R}^{3N_C \times N_C M} \\ \boldsymbol{b} &= \begin{pmatrix} \boldsymbol{b}_1^T & \boldsymbol{b}_2^T & \dots & \boldsymbol{b}_{N_C}^T \end{pmatrix}^T\end{aligned}$$

and $diag\{*\}$ denotes a block diagonal matrix.

The linear complementarity condition for normal directions is described as

$$\boldsymbol{N}^T \boldsymbol{v}^+ \geq 0 \perp \boldsymbol{a} \geq 0. \tag{21}$$

The condition for the friction force and tangential velocity is described as

$$\boldsymbol{\mu}\boldsymbol{a} - \boldsymbol{E}\boldsymbol{b} \geq 0 \perp \boldsymbol{\lambda} \geq 0 \tag{22}$$

$$\boldsymbol{C}^T\boldsymbol{v}^+ + \boldsymbol{E}^T\boldsymbol{\lambda} \geq 0 \perp \boldsymbol{b} \geq 0 \tag{23}$$

where $\boldsymbol{\lambda} \in \boldsymbol{R}^{N_C}$ is a Lagrangian, $\boldsymbol{\mu}$ is a diagonal matrix composed of the friction coefficients at all contact points, and $\boldsymbol{E} \in \boldsymbol{R}^{N_C \times N_C M}$ is a constant block-diagonal matrix defined as

$$\boldsymbol{E} = diag\{\boldsymbol{1}\},\ \boldsymbol{1} = \begin{pmatrix} 1 & \dots & 1 \end{pmatrix} \in \boldsymbol{R}^M. \tag{24}$$

Substituting Eqs.(14)(20) into Eqs.(22)(23), we obtain the whole LCP:

$$\begin{pmatrix} \boldsymbol{N}^T\bar{\boldsymbol{\Phi}}\boldsymbol{N} & \boldsymbol{N}^T\bar{\boldsymbol{\Phi}}\boldsymbol{C} & 0 \\ \boldsymbol{C}^T\bar{\boldsymbol{\Phi}}\boldsymbol{N} & \boldsymbol{C}^T\bar{\boldsymbol{\Phi}}\boldsymbol{C} & \boldsymbol{E}^T \\ \boldsymbol{\mu} & -\boldsymbol{E} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{\lambda} \end{pmatrix} + \begin{pmatrix} \boldsymbol{N}^T\boldsymbol{\phi} \\ \boldsymbol{C}^T\boldsymbol{\phi} \\ 0 \end{pmatrix} = \begin{pmatrix} \boldsymbol{w}_a \\ \boldsymbol{w}_b \\ \boldsymbol{w}_\lambda \end{pmatrix} \tag{25}$$

$$\begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \\ \boldsymbol{\lambda} \end{pmatrix} \geq 0 \perp \begin{pmatrix} \boldsymbol{w}_a \\ \boldsymbol{w}_b \\ \boldsymbol{w}_\lambda \end{pmatrix} \geq 0. \tag{26}$$

C. Implementation Issues

As with many contact models, the most important factor in determining the computational cost is the number of contact points. Because our collision detection library handles general polygonal objects, many contact points are detected for complex objects. We accelerate the computation by removing unnecessary contact points such as those placed within some small distance δ from another point or inside the contact area. Note that there still may be redundant constraints even after removing some contact points. Although in some papers [16], [21] contact points with positive normal velocities are also removed, we found that ignoring these points can cause penetration because contact forces at other points may change their normal velocities negative.

We also applied a two-step solution to further accelerate the computation. We first solve the frictionless version of the contact problem by considering only the normal direction of each contact point to identify which points are likely to be active in the current contact state. We then solve the frictional problem from the initial guess that the normal directions of all active contact points are constrained (zero velocity and positive contact force), while others are unconstrained. We expect that this method greatly reduce the number of additional pivots required to reach a solution.

V. Results

The experiments presented in this section were executed on a workstation with a Pentium Xeon 3.8GHz processor. The code was written in C++ and the compiler was Microsoft Visual Studio .NET 2003 with optimization. We used 4-th order Runge-Kutta integration with 1 ms time step except when otherwise noted. Collision detection between general polygonal objects was performed by a library called PQP [22] with an extension to compute penetration depths and normal vectors [23]. The constants were set as $e_{max} = 1 \times 10^{-3}$, $\epsilon = 1 \times 10^{-3}$, $M = 8$ and $\delta = 1 \times 10^{-3}$ (m), and we employed the incremental update of $\boldsymbol{M}'$ and $\boldsymbol{q}'$ mentioned in the last paragraph of Section III. The search queue is implemented as a binary tree to efficiently find the node with minimum cost.

A. Comparison with Conventional Algorithm

We first compare our LCP solver with Lemke and Lexicographic Lemke algorithms. The example used here is a squat motion of a small 20-joint humanoid robot [24] on a horizontal flat floor. The simulated robot is under high-gain feedback mode, i.e. the joint angle, velocity and acceleration computed from the reference trajectory were directly applied to each joint. The geometry data of the links were extracted from the CAD model.

We used our own implementations of the conventional algorithms with the following details (refer to the algorithm outline in Section II-A):

- *Lemke Algorithm*—Always chooses the row with the minimum $-q_i/m'_i$ for pivot in Step 1. In case of tie (in the sense of floating-point numbers), the row with the minimum index is chosen.
- *Lexicographic Lemke Algorithm*—A row is regarded as tie if its $-q_i/m'_i$ is within a threshold Δ from the minimum, and the lexicographic ordering test is repeated until a unique minimum is identified using the same threshold. If multiple rows are in tie in the last test, the row with the minimum value is chosen. If z_0 is among the tie rows in any test, it is immediately chosen as the pivot variable and hence the algorithm terminates.

The implementations were tested using the examples in [1] that are known to be solvable by Lemke and/or Lexicographic Lemke algorithms.

There are the following three possible failure modes:

1) *no solution* is the case when $\boldsymbol{m}' \geq 0$ occurred in Step 1,
2) *cycle* is emitted when the same set of pivot was already found in one of the previous steps, and
3) *error* is emitted when the error of the solution is larger than e_{max}.

TABLE I
Comparison of the proposed solver, original Lemke Algorithm and Lexicographic Lemke Algorithm with four different thresholds Δ.

	proposed	*Lemke*	*Lexicographic Lemke*			
			0	10^{-8}	10^{-6}	10^{-4}
contacts	15.4	16.1	16.0	15.9	12.1	15.5
active	2.18	3.03	3.02	3.03	2.30	2.55
total frames	999	998	998	998	1000	1000
success	999	506	514	900	990	996
failure	0	492	484	98	10	4
no solution	0	52	27	17	3	3
cycle	0	157	147	18	2	0
error	0	283	310	63	5	1

If the algorithm failed in solving the LCP with friction and the frictionless LCP was successfully solved, the frictionless result is applied to prevent penetration, although it causes slipping. The number of frames where both LCPs failed was at most three and resulting penetration was practically negligible in all simulations.

The results are summarized in Table I. The first two rows represent the average numbers of detected contact points and those identified to be active respectively, and the next three rows represent the number of total frames with contact, successfully terminated frames, and failure frames respectively. The number of frames with each failure mode is shown in the last three rows. We also tested Lexicographic Lemke Algorithm with $\Delta = 10^{-3}$ but the robot fell down due to a slip caused by applying a frictionless solution.

The three algorithms behaved differently even for this relatively static motion. The results obviously show the advantage of our algorithm. Lemke Algorithm could solve only around half of the frames, and Lexicographic Lemke Algorithm did not help much either when the threshold is too small. $\Delta = 10^{-4}$ gave the best result in this example, but still had a few failure frames. In contrast, our algorithm successfully solved the LCP in all frames.

Note that 12 to 16 contact points were detected in average even with the contact point reduction described in Section IV-C, and only 2 to 3 of them were identified to be active. This is physically reasonable because three contact points are enough to constrain the motion of the robot in high-gain feedback mode, and the original contact points would yield highly ill-conditioned LCP. This is probably the reason for failures in conventional algorithms.

Fig. 2 compares real and simulated total vertical forces of left and right feet. The simulation was performed using the proposed algorithm, and a single force plate measured the total contact forces and moments at the feet. The simulated force exhibits qualitatively similar pattern to the real one, but the measured force shows more oscillation. This discrepancy is probably because the real hardware has some elasticity in the links, joints and controller. The constant offset is due to the error in the mass parameters estimated from the CAD model that does not include wires and screws.

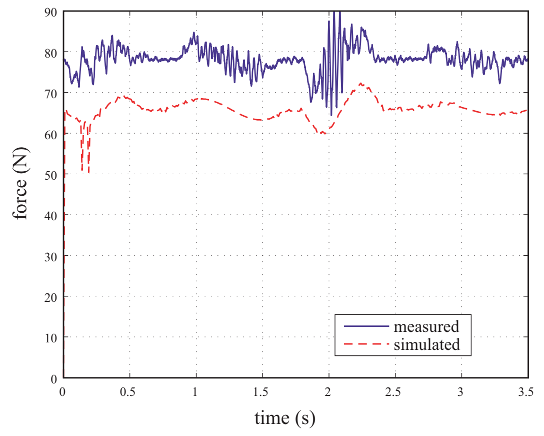

Fig. 2. Measured and simulated vertical forces for the humanoid motion.

TABLE II
COMPARISON OF SUCCESS RATE AND AVERAGE COMPUTATION TIME OF LEXICOGRAPHIC LEMKE AND PROPOSED ALGORITHMS FOR THE HUMANOID EXAMPLE.

	Lemke	*proposed*
contacts	6.89	7.19
active	2.02	1.70
total frames	9656	9670
success	9620	9670
failure	36	0
no solution	13	0
cycle	7	0
error	16	0
LCP solve time (ms)	0.375	0.182
simulation time (ms)	3.49	3.54
total pivots	71102	34335

B. Tap-Dancing of a Humanoid Robot

We perform another comparison including computation time using a tap-dancing motion of the same humanoid robot. This is a more challenging task because it includes frequent collisions as well as static contacts. We applied our solver and Lexicographic Lemke Algorithm. The best Δ was 10^{-10} in this case, which implies the necessity of finding threshold tailored to each task.

The result is shown in Table II, where “LCP solve time” indicates the time for solving the main frictional LCP. The time for solving the frictionless one is included in the total simulation time. Lexicographic Lemke Algorithm still failed to find a solution in about 0.4% of the frames with contact, while the proposed algorithm succeeded in all frames. Our LCP solver is also faster because lexicographic ordering requires more pivot computations than the search in our algorithm as shown in the “total pivots” row, although the total simulation time is longer because of the larger number of contact points that require preprocessing. A visual comparison of simulated and actual motions is shown in Fig. 3. Note the qualitatively similar behaviors such as yaw rotation.

TABLE III
COMPUTATION TIME FOR THE LONG AND CLOSED CHAIN EXAMPLES.

	hoist	*hoist* (10 ms)	*ring*	*net*
duration (s)	10	10	4	2
contacts	34.2	48.4	26.1	25.7
active	20.8	23.0	21.7	16.9
total frames	9357	935	3926	1818
failure	0	0	0	0
LCP solve time (ms)	62.0	96.5	35.0	9.72
simulation time (ms)	120	187	80.6	62.5

C. More Complex Scenarios

Figure 4 shows three simulation examples involving complex collisions and contacts of open and closed articulated rigid bodies. In the *hoist* example, a string-like object modeled as a 25-joint chain is subject to continuous contact with the rod and therefore takes long time for solving the LCP. We also varied the time step for integration. The simulation result with 10 ms timestep was similar to that with 1 ms and the total computation time was about 4.5 times shorter, although each step requires longer computation time because the penetration depth tends to be larger due to integration errors and more contact points are detected.

The *ring* example includes a number of contacts of non-convex objects. Each wire is composed of five spherical joints with two ring-shaped links at the ends. In the *net* example, five cylinders fall onto a net composed of four strings each modeled as a 16-joint chain with both ends fixed to the inertial frame, forming closed loops. Our algorithm still yields realistic results without failure.

VI. CONCLUSION

The conclusions of this paper are summarized by the following three points:

1) We pointed out two numerical issues of Lemke and Lexicographic Lemke algorithms, and proposed a robust algorithm for solving general LCPs. The main idea of the new algorithm is to store all pivot candidates at each step, and back trace the queue in case a numerical problem is found.
2) We modeled frictional contact of articulated rigid bodies as an LCP using inverse articulated-body inertia (IABI) [11], and applied above algorithm to solve the LCP. We combined the formulation with a forward dynamics algorithm called ADA [18] but it can also be combined with DCA [12].
3) Experimental results showed that our algorithm can robustly solve LCPs formulating contact dynamics of articulated bodies, which cannot be solved by Lexicographic Lemke Algorithm.

ACKNOWLEDGEMENTS

The authors thank Professor Tomomichi Sugihara (Kyushu University) and Mr. Kou Yamamoto (University of Tokyo) for their help in the humanoid hardware experment. This work was supported by a MEXT Special Coordination Fund for Promoting Science and Technology.

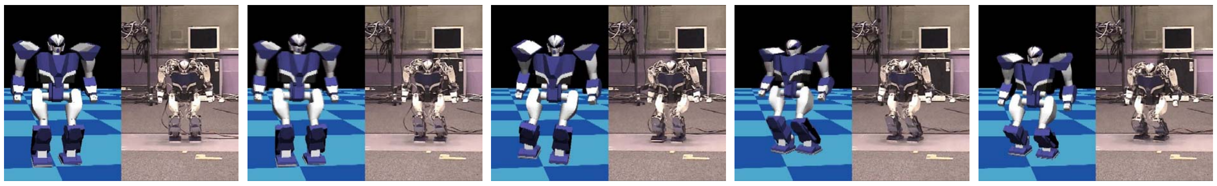

Fig. 3. Comparison of simulated (left) and actual (right) motions of humanoid tap-dancing.

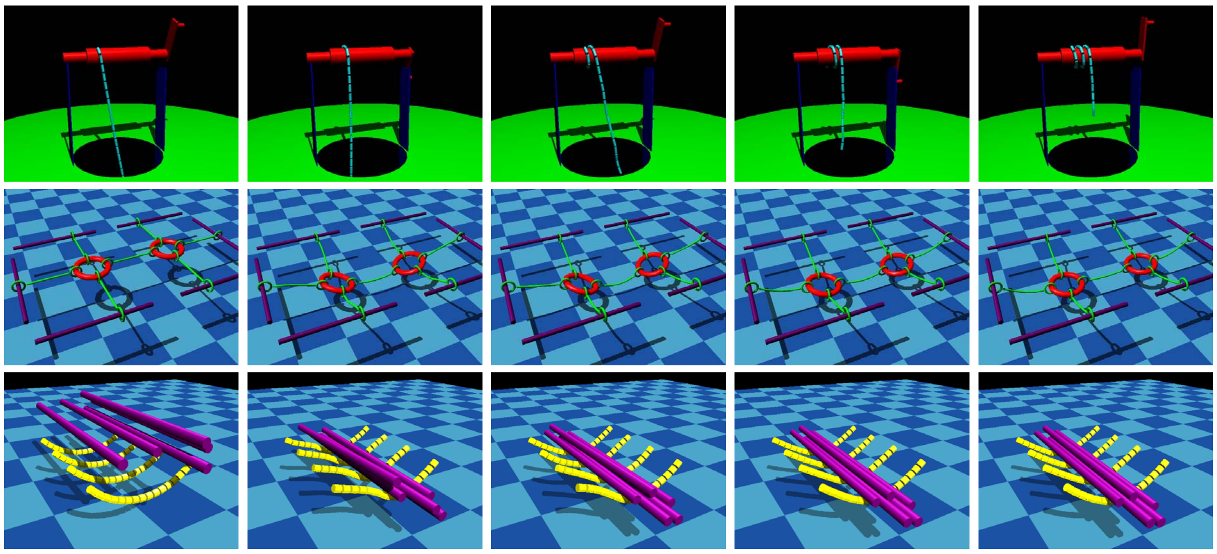

Fig. 4. Examples of contact simulation of articulated rigid bodies; from the top row: *hoist*, *ring*, and *net*.

References

[1] K. Murty, *Linear Complementarity: Linear and Nonlinear Programming*. Lemgo, Germany: Heldermann-Verlag, 1988.

[2] M. Ferris and T. Munson, "Complementarity Problems in GAMS and the PATH Solver," University of Wisconsin–Madison, Tech. Rep. 98-12, September 1998.

[3] C. Lemke and J. Howson, "Equilibrium points of bimatrix games," *SIAM Journal on Applied Mathematics*, vol. 12, pp. 413–423, 1964.

[4] F. Jourdan, P. Alart, and M. Jean, "A Gauss-Seidel Like Algorithm to Solve Frictional Contact Problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 155, pp. 31–47, 1998.

[5] M. Förg, F. Pfeiffer, and H. Ulbrich, "Simulation of unilateral constrained systems with many bodies," *Multibody System Dynamics*, vol. 14, pp. 137–154, 2005.

[6] D. Stewart and J. Trinkle, "An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction," in *Proceedings of IEEE International Conference on Robotics and Automation*, San Francisco, CA, May 2000, pp. 162–169.

[7] J. Lloyd, "Fast Implementation of Lemke's Algorithm for Rigid Body Contact Simulation," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, April 2005, pp. 4538–4543.

[8] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 871–878, 2003.

[9] R. Smith, "Open dynamics engine," http://www.ode.org/, 2007.

[10] R. Weinstein, J. Teran, and R. Fedkiw, "Dynamic simulation of articulated rigid bodies with contact and collision," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 3, pp. 365–374, 2006.

[11] R. Featherstone, *Robot Dynamics Algorithm*. Boston, MA: Kluwer Academic Publishers, 1987.

[12] ——, "A Divide-and-Conquer Articulated-Body Algorithm for Parallel $O(\log(n))$ Calculation of Rigid-Body Dynamics. Part1: Basic Algorithm," *International Journal of Robotics Research*, vol. 18, no. 9, pp. 867–875, September 1999.

[13] E. Kokkevis, "Practical physics for articulated characters," in *Game Developers Conference*, 2004.

[14] R. Gayle, M. Lin, and D. Manocha, "Adaptie dynamics with efficient contact handling for articulated robots," in *Robotics: Science and Systems*, 2006, pp. 231–238.

[15] S. Redon, N. Galoppo, and M. Lin, "Adaptive dynamics of articulated bodies," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 936–945, 2005.

[16] B. Mirtich and J. Canny, "Impulse-based simulation of rigid bodies," in *Proceedings of Symposium on Interactive 3D Graphics*, Monterey, CA, 1995, pp. 181–188.

[17] P. Kry and D. Pai, "Interaction capture and synthesis," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 872–880, 2006.

[18] K. Yamane and Y. Nakamura, "Efficient Parallel Dynamics Computation of Human Figures," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2002, pp. 530–537.

[19] R. Cottle and G. Dantzig, "Complementary pivot theory of mathematical programming," *Linear Algebra and Its Applications*, vol. 1, pp. 103–125, 1968.

[20] V. Acary and Pérignon, "An Introduction to Siconos," INRIA, Tech. Rep. 340, July 2007.

[21] D. Kaufman, T. Edmunds, and D. Pai, "Fast frictional dynamics for rigid bodies," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 946–956, 2005.

[22] UNC Gamma Group, "PQP—A Proximity Query Package," http://www.cs.unc.edu/˜geom/SSV/.

[23] K. Yamane and Y. Nakamura, "Stable penalty-based model of frictional contacts," in *Proceedings of IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006, pp. 1904–1909.

[24] T. Sugihara, K. Yamamoto, and Y. Nakamura, "Architectural design of miniature anthropomorphic robots towards high-mobility," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1083–1088.