in Proc. IEEE Intl. Conf. on Robotics & Autom. 2003

Post-Stabilization for Rigid Body Simulation with Contact and Constraints

Michael B. Cline[*] and Dinesh K. Pai[*†]
[*] University of British Columbia, Vancouver, Canada.
[†] Rutgers, the State University of NJ, Piscataway, NJ
{cline|pai}@cs.ubc.ca

***Abstract*— Rigid body dynamics with contact constraints can be solved locally using linear complementarity techniques. However, these techniques do not impose the original constraints and need stabilization. In this paper we show how constraint stabilization can also be done in a complementarity framework. Our technique effectively eliminates the drift problem for both equality and inequality constraints and requires no parameter tweaking. We describe results from an implemented system, and compare the new technique to the well known Baumgarte stabilization.**

I. Introduction

Dynamic simulation of rigid body contact is central to many aspects of robotics, including manipulation, design, and haptic interaction in virtual environments. Dynamics with contact constraints and friction lead to differential algebraic equations and inequalities, which are difficult to solve. Recently, these contact problems have been effectively formulated as linear complementarity problems (LCPs) [Lot82], [Bar94], [ST96], [AP97]. However, these techniques used "reduced index" formulations of the dynamics [AP98]. They only satisfy the constraints locally and can drift away from the constraint, and must be "stabilized" to continue to satisfy the constraint. In this paper we show how this stabilization can be performed in an LCP framework as well, and ensure that the stabilization steps also satisfy the inequalities due to contact and friction constraints. We provide examples from an implemented system which show that this technique is effective in stabilizing both equality and inequality constraints due to contact.

The remainder of the paper is organized as follows. In Sec. II we describe some related work. Sec. III provides a brief but complete description of LCP-based formulations of rigid body contact mechanics. Sec. IV describes constraint stabilization in general. Sec. V describes our new technique for post-stabilization in an LCP framework. Sec. VI describes the results, and compares the new approach to previous approaches.

II. Related Work

We draw from related work in two areas. The first area is the work on stabilization of ordinary differential equations with invariants. The stabilization method that we will present is based on work by Ascher and Chin [Asc97] [ACR94]. These papers discuss stabilization methods in general and post-stabilization in particular as the favoured method.

The other area of related work is the literature on rigid body dynamics with contact, where the constrained dynamics equations and inequalities are formulated as a linear complementarity problem. The first paper to pose the contact problem as an LCP was published by Lötstedt [Lot82]. Baraff presented a method [Bar94] that used an LCP algorithm by Cottle and Dantzig [CD68] to solve contact and static friction forces at interactive rates.

One of the difficulties with earlier LCP methods was that there is no guarantee of the existence of a solution, in the presence of contact with Coulomb friction. Indeed, there are known configurations of rigid bodies for which no solution exists, such as the Painlevé paradox. Using a model which allows impulsive forces turned out to be the key to avoiding these problems. Stewart and Trinkle [ST96] introduced a time-stepping scheme that combines the acceleration-level LCP with a time step, to obtain an LCP where the variables are velocities and impulses. Anitescu and Potra [AP97], described a modification, which we describe in detail below, that guaranteed solvability regardless of the configuration or number of contacts.

We unify these two areas of research by showing how stabilization for simulations with contact can be done by formulating the post-stabilization problem as linear complementarity problem.

III. Background

A. Notation

We use a translation vector and a quaternion to represent the position and orientation of the rigid body. We append these together in a vector $\mathbf{p}$. The velocity $\mathbf{v}$ of the body is described by a vector containing the linear and angular velocity of the body. The state of the i'th body is described by the vectors $\mathbf{p}_i = (p_{ix} p_{iy} p_{iz} q_{ix} q_{iy} q_{iz} q_{iw})^T$ and $\mathbf{v}_i = (\omega_{ix} \omega_{iy} \omega_{iz} v_{ix} v_{iy} v_{iz})^T$.

We shall write the Newton-Euler equations of motion of body i as

$$\mathbf{f}_i = \mathbf{M}_i \mathbf{a}_i \tag{1}$$

Supported in part by grants from IRIS NCE.

where $\mathbf{f}_i$ is the wrench (torque and force) acting on body i, which includes the Coriolis forces, $\mathbf{M}_i$ is the the spatial inertia matrix of the body, and $\mathbf{a}_i = \dot{\mathbf{v}}_i$. When dealing with a system of many bodies, we shall use the symbols $\mathbf{f}$, $\mathbf{a}$, and $\mathbf{M}$ to indicate vectors and matrices that contain information for many bodies.

B. Constrained Dynamics

Position constraints will be described by a *constraint function* $\mathbf{g}(\mathbf{p})$, which is a function which maps $\mathbf{p}$, the position vector of the rigid bodies, to a point in $\mathbb{R}^n$, where n is the number of degrees of freedom that the constraint removes from the system. If the constraint function returns a zero vector, then the position $\mathbf{p}$ satisfies the constraint.

Position constraints can be divided into *equality constraints* (e.g., joint constraints), where the constraint is $\mathbf{g}(\mathbf{p}) = 0$, and *inequality constraints* (e.g., contact constraints), where $\mathbf{g}(\mathbf{p}) \geq 0$. For the rest of this section, we will just be talking about equality constraints. We will talk about contact constraints separately in the next section.

Constraining the position of an object also constrains its velocity. Velocity constraints are of the form

$$\frac{d\mathbf{g}}{dt} = \mathbf{J}\mathbf{v} + \mathbf{c} = \begin{pmatrix} \mathbf{J}_1 & \mathbf{J}_2 & \dots & \mathbf{J}_n \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} + \mathbf{c} = \mathbf{0}. \quad (2)$$

The matrix $\mathbf{J}$ is called the constraint's *Jacobian matrix*, which we refer to simply as the Jacobian. The Jacobian is a function of the current position of the bodies involved in the constraint. In practice, most constraints will only involve one or two bodies, so most of the matrices $\mathbf{J}_1 \dots \mathbf{J}_n$ will be zero matrices.

A constraint on the acceleration can be found by taking the derivative again. Doing so results in an equation of similar form to Equation 2: $\mathbf{J}\dot{\mathbf{v}} + \mathbf{k} = \mathbf{0}$. The acceleration and velocity constraints use the same Jacobian matrix, but different terms $\mathbf{c}$ and $\mathbf{k}$.

When there are many constraints, we will often write all constraints simultaneously. For a system with n bodies, and m constraints, our velocity constraint equation looks like

$$\frac{d\mathbf{g}}{dt} = \begin{pmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_m \end{pmatrix} = \mathbf{J}\mathbf{v} + \mathbf{c} = \begin{pmatrix} \mathbf{J}_{11} & \mathbf{J}_{12} & \dots & \mathbf{J}_{1n} \\ \mathbf{J}_{21} & \mathbf{J}_{22} & & \mathbf{J}_{2n} \\ \vdots & & \ddots & \vdots \\ \mathbf{J}_{m1} & \mathbf{J}_{m2} & \dots & \mathbf{J}_{mn} \end{pmatrix} \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{pmatrix} + \mathbf{c} = \mathbf{0}. \quad (3)$$

The constraint Jacobian matrix $\mathbf{J}$ has a dual use. In addition to relating the velocities to the rate of change of the constraint function $\mathbf{g}$, the rows of $\mathbf{J}$ act as basis vectors for constraint forces. Thus, when we solve for the constraint forces, we actually just need to solve for the coefficient vector λ (whose components are the *Lagrange multipliers*) that contains the magnitudes of the forces that correspond to each of these basis vectors.

The total force acting on the system is the sum of the external forces $\mathbf{F}_{ext}$ (in which we include Coriolis forces) and the constraint forces $\mathbf{J}^T\lambda$. Combining the Newton-Euler equations,

$$\mathbf{F} = \mathbf{J}^T\lambda + \mathbf{F}_{ext} = \mathbf{M}\dot{\mathbf{v}},$$

with the constraint equations $\mathbf{J}\dot{\mathbf{v}} + \mathbf{k} = 0$, we get the following system

$$\begin{pmatrix} \mathbf{M} & -\mathbf{J}^T \\ \mathbf{J} & 0 \end{pmatrix} \begin{pmatrix} \dot{\mathbf{v}} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{ext} \\ -\mathbf{k} \end{pmatrix}, \quad (4)$$

which we can solve for the acceleration $\dot{\mathbf{v}}$ and the Lagrange multipliers λ. Baraff [Bar96] shows how to solve this system in linear time by exploiting the sparse structure of the matrix on the left hand side of Equation 4. This sparseness exploitation was previously done in the MEXX system [LNPE92].

1) Incorporating Contact Constraints: Contact constraints are different from joint constraints, and require special treatment. There are a few important features that set contact constraints apart from other constraints:

- Contact forces can push bodies apart, but cannot pull bodies towards each other. This leads to inequalities in the constraint equations, whereas other types of constraints are equalities.
- If there are many points of contact between a pair of bodies, there may be many solutions for the contact forces that are plausible.
- In the presence of Coulomb friction, a solution to the dynamics equations may not exist.

There are several methods for handling contact in rigid body simulations. Mirtich [Mir98] gives a summary of several different methods, and explains the advantages and disadvantages of each.

We focus on the *Linear Complementarity Problem* approach, first introduced in the context of constrained mechanical systems by Lötstedt [Lot82], and introduced to the computer graphics community by Baraff [Bar94].

To describe the contact model we use, we introduce the following terminology.

- A *contact* consists of a pair of *contact points*, one point attached to one rigid body, and the other point attached to another rigid body. The contact points are sufficiently close together for our collision detection algorithm to report a collision.
- A *contact normal* is a unit vector that is normal to one or both of the surfaces at the contact points.
- A *contact wrench* $\mathbf{J}^T\lambda$ is a wrench which prevents the two rigid bodies from interpenetrating.
- The *separation distance* of a contact is the normal component of the displacement between the two contact points. It is negative when the bodies are interpenetrating at the contact. The constraint function $\mathbf{g}(\mathbf{p})$ for a contact constraint returns the separation distance. The constraint is satisfied for $\mathbf{g} \geq 0$.
- The *relative normal acceleration* a of a contact is the second derivative of the separation distance with respect to time. The acceleration constraint is satisfied when $a = \mathbf{J}\dot{\mathbf{v}} + \mathbf{k} \geq 0$.

Let $\mathbf{a}$ be a vector containing the relative normal accelerations $\{a_1, ..., a_n\}$ for all contacts, and λ be a vector of contact force multipliers $\{\lambda_1, ..., \lambda_n\}$. The vectors $\mathbf{a}$ and λ are linearly related. Additionally, we have three constraints:

1) The relative normal accelerations must be positive: $\mathbf{a} = \mathbf{J}\dot{\mathbf{v}} + \mathbf{k} \geq \mathbf{0}$.
2) The contact force magnitudes must be positive (so as to push the bodies apart): $\lambda \geq \mathbf{0}$
3) For each contact i, only one of a_i, λ_i can be nonzero.

The problem of finding a solution to a linear equation given such constraints is called a *Linear Complementarity Problem* (LCP), which we discuss in Section III-B.2.

If we let $\mathbf{J}_e$ be the Jacobian of equality constraints, and $\mathbf{J}_c$ be the Jacobian of contact constraints, we can extend Equation 4 to include the contact constraints as follows [ST96]:

$$\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{a}\end{pmatrix} - \begin{pmatrix}\mathbf{M} & -\mathbf{J}_e^T & -\mathbf{J}_c^T\\ \mathbf{J}_e & 0 & 0\\ \mathbf{J}_c & 0 & 0\end{pmatrix}\begin{pmatrix}\dot{\mathbf{v}}\\ \lambda_e\\ \lambda_c\end{pmatrix} = \begin{pmatrix}\mathbf{F}_{ext}\\ -\mathbf{k}_e\\ -\mathbf{k}_c\end{pmatrix}, \qquad \mathbf{a} \geq \mathbf{0}, \lambda_c \geq \mathbf{0}, \mathbf{a}^T\lambda_c = \mathbf{0}. \tag{5}$$

In this form, we have a *mixed LCP*, meaning that only some of the rows have complementarity constraints on them. To obtain a pure LCP, we must eliminate $\dot{\mathbf{v}}$ and λ_e by solving for those in terms of λ_c.

2) Linear Complementarity Problems: A *Linear Complementarity Problem* (LCP) is, given a $n \times n$ matrix $\mathbf{M}$ and a n-vector $\mathbf{q}$, the problem of finding values for the variables $\mathbf{z} = \{z_1, z_2, .., z_n\}$ and $\mathbf{w} = \{w_1, w_2, .., w_n\}$ such that

$$\mathbf{w} = \mathbf{M}\mathbf{z} + \mathbf{q}, \tag{6}$$

and, for all i from 1 to n, $z_i \geq 0$, $w_i \geq 0$ and $z_i w_i = 0$. (These three constraints are called the *complementarity constraints*). We solve the linear complementarity problems using Lemke's algorithm [CPS92], [Mur88].

3) Solvable LCPs for Contact with Friction: It is well known that that when Coulomb friction is added, the acceleration-level dynamics equations (Equation 5) fail to have a solution in certain configurations, even for situations where there is only one contact involved.

Anitescu and Potra [AP97] present a time-stepping method which combines the acceleration-level LCP with an integration step for the velocities, arriving at a method which has velocities and impulses as unknowns, rather than accelerations and forces. Their method is guaranteed to have a solution, regardless of the configuration and number of contacts.

To discretize the system (5), Anitescu and Potra apply a forward Euler step on the velocities: $\dot{\mathbf{v}} \approx (\mathbf{v}_{next} - \mathbf{v}_{current})/h$, where $\mathbf{v}_{current}$ and $\mathbf{v}_{next}$ are the velocities at the beginning of the current time step, and the next time step, respectively, and h is the time step size. We then arrive at the mixed LCP

$$\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{a}\end{pmatrix} - \begin{pmatrix}\mathbf{M} & -\mathbf{J}_e^T & -\mathbf{J}_c^T\\ \mathbf{J}_e & 0 & 0\\ \mathbf{J}_c & 0 & 0\end{pmatrix}\begin{pmatrix}\mathbf{v}_{next}\\ \lambda_e\\ \lambda_c\end{pmatrix} = \begin{pmatrix}\mathbf{M}\mathbf{v}_{current} + h\mathbf{F}_{ext}\\ -\mathbf{k}_e\\ -\mathbf{k}_c\end{pmatrix}, \qquad \mathbf{a} \geq \mathbf{0}, \lambda_c \geq \mathbf{0}, \mathbf{a}^T\lambda_c = \mathbf{0}. \tag{7}$$

Coulomb friction is introduced by adding some additional forces and constraints to the LCP (7).

Essentially, these complementarity conditions guarantee the following properties that we expect from Coulomb friction:

- If the contact impulse lies inside the friction cone (but not on its surface), then the bodies are not exhibiting relative tangential motion.
- If the bodies are in relative tangential motion, then the contact impulse lies on the surface of the friction cone, and exhibits negative work.
- As the approximation of the friction cone becomes closer to the ideal circular friction cone, the friction becomes closer to directly opposing the relative tangential motion.

Extending Equation 7 to include these friction constraints, we get the following linear complementarity problem

$$\begin{pmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{a}\\ \sigma\\ \zeta\end{pmatrix} - \begin{pmatrix}\mathbf{M} & -\mathbf{J}_e^T & -\mathbf{J}_c^T & -\mathbf{J}_f^T & 0\\ \mathbf{J}_e & 0 & 0 & 0 & 0\\ \mathbf{J}_c & 0 & 0 & 0 & 0\\ \mathbf{J}_f & 0 & 0 & 0 & \mathbf{E}\\ 0 & 0 & \mu & -\mathbf{E}^T & 0\end{pmatrix}\begin{pmatrix}\mathbf{v}_{next}\\ \lambda_e\\ \lambda_c\\ \lambda_f\\ \gamma\end{pmatrix} = \begin{pmatrix}\mathbf{M}\mathbf{v}_{current} + h\mathbf{F}_{ext}\\ -\mathbf{k}_e\\ -\mathbf{k}_c\\ 0\\ 0\end{pmatrix},$$

$$\begin{pmatrix}\mathbf{a}\\ \sigma\\ \zeta\end{pmatrix} \geq \mathbf{0}, \qquad \begin{pmatrix}\lambda_c\\ \lambda_f\\ \gamma\end{pmatrix} \geq \mathbf{0}, \qquad \begin{pmatrix}\mathbf{a}\\ \sigma\\ \zeta\end{pmatrix}^T \begin{pmatrix}\lambda_c\\ \lambda_f\\ \gamma\end{pmatrix} = \mathbf{0}. \tag{8}$$

Due to lack of space, we refer the reader to [Ani97], [Cli02] for full details; here it is sufficient to observe that the general form of the LCP remains the same.

IV. Constraint Stabilization

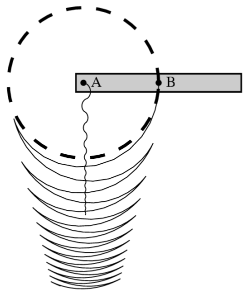

Fig. 1. An unstabilized simulation of a swinging pendulum.

Figure 1 is an example situation that motivates the need for stabilization. The figure shows a simple two-dimensional simulation of a swinging pendulum: one rigid body with one constraint. The only external force is that of gravity. The constraint is a hinge joint constraint that anchors point A to a fixed point. Over time, however, point A actually drifts, due to numerical integration errors. The centre of mass, rather than staying on the circular path that it is supposedly constrained to (shown with a dashed line), continues to fall further from that path as time progresses.

We can lessen this problem by using higher order, more accurate integration scheme (the ones used here are only first-order accurate), or by using smaller time steps. However, even using the best integration methods and small time steps, numerical drift can never be completely eliminated. Furthermore, it may not always be feasible to use the more complicated integrators or decrease the step size and still achieve acceptable performance. Even if we cannot achieve very high numerical accuracy, we still wish to have simulations that are plausible – ones where the constraints are always met. This section describes methods for counteracting constraint drift using stabilization.

Before continuing, we need to define our problem more clearly. *Stability* is a word that is used with a variety of meanings in different contexts in the differential equation literature. By the word *stabilization* here, we mean *stabilization of an ODE with respect to an invariant set*. We shall now explain what this means.

The position constraint equation coupled with the constrained Newton-Euler equations (see Equation 4) together are an example of a *differential-algebraic equation* (DAE). This is a term for a system of equations containing both differential equations and algebraic equations. DAEs are a relatively new area of research, and the methods for solving them directly are somewhat difficult. The usual approach is to convert the DAE into an equivalent ordinary differential equation (ODE) which can be solved by conventional techniques. In our case, we can do this by differentiating the position constraint equations twice, to arrive at a constraint in terms of accelerations, then substituting this acceleration constraint into the Newton-Euler equations. In doing this, we arrive at an ODE. The approach we use is actually slightly different, as we have described in Section III-B.3.[1]

The penalty of this approach is that we lose the constraints on the position and velocity. Upon discretization of the ODE, we introduce numerical errors, and we will experience *drift* away from the *constraint manifold*. The constraint manifold (also known as the *invariant set*) is the subset of the state space in which the constraints are satisfied. To counteract this, we must stabilize the ODE with respect to the invariant set. In other words, we must alter the ODE so that it has the same solutions as the original whenever $\mathbf{g}(\mathbf{p}) = 0$, but whenever $\mathbf{g}(\mathbf{p}) \neq 0$ the solutions is attracted towards the invariant set.

In this section we describe two methods of stabilization, in the context of rigid body simulation. First we will discuss *Baumgarte* stabilization, a method that is very popular because of its simplicity. We will then introduce *Post-Stabilization*, which is based on the work of Ascher et al. [Asc97] [ACR94]

[1]We use *time-stepping* methods, where a numerical integration step is built into the system of equations we solve, and the equations are given in terms of velocities and impulses, rather than forces and accelerations.

A. Baumgarte Stabilization

Baumgarte's stabilization technique [Bau72] is one of the most familiar and commonly used methods, because of its simplicity. The idea here is to replace the acceleration constraint equation $\ddot{\mathbf{g}} = \mathbf{J}\dot{\mathbf{v}} + \mathbf{k} = 0$ with some a linear combination of the acceleration, velocity and position constraint equations:

$$0 = \ddot{\mathbf{g}} + \alpha\dot{\mathbf{g}} + \beta\mathbf{g}, \tag{9}$$

which creates a more stable ODE. If the velocity and position constraints are satisfied, the last two terms on the right hand side vanish, and we are left with the original acceleration constraint equation. A physical interpretation of this method is that we are adding additional correction forces, proportional to the error in the velocity and position constraints, to counteract drift.

Because our implementation uses only velocity-level constraints rather than acceleration constraints (see Section III-B.3), we use an even simpler version of Baumgarte stabilization where we replace the velocity constraint $\dot{\mathbf{g}} = \mathbf{J}\mathbf{v} + \mathbf{c} = 0$ with $\dot{\mathbf{g}} + \alpha\mathbf{g} = \mathbf{J}\mathbf{v} + (\mathbf{c} + \alpha\mathbf{g}) = 0$.

The main difficulty of using Baumgarte stabilization is that it is not always easy to find an appropriate value for the constants α and β.

B. Post-Stabilization

Another approach to stabilization is to follow each integration step with a stabilization step. The stabilization step takes the result of the integration step as input, and gives a correction so that the end result is closer to the constraint manifold. Post-stabilization methods are discussed in detail and compared to other stabilization methods by Ascher et al. [ACR94]. Here we give interpretation of post-stabilization that fits into Ascher's broader definition.

Let $\mathbf{p}$ be the position of a set of rigid bodies after the integration step. Let $\mathbf{g}$ be the constraint function (see Section III-B). In general, due to numerical drift, $\mathbf{g}(\mathbf{p}) \neq \mathbf{0}$. Let $\mathbf{G} = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}$. In our stabilization step, we wish to find some $\mathbf{dp}$ such that $\mathbf{g}(\mathbf{p} + \mathbf{dp}) = 0$. Assuming $\mathbf{dp}$ will be small, we can make the approximation that

$$\mathbf{g}(\mathbf{p} + \mathbf{dp}) \approx \mathbf{g}(\mathbf{p}) + \mathbf{G}(\mathbf{p})\mathbf{dp}. \tag{10}$$

Rearranging this, we see that the stabilization term $\mathbf{dp}$ should satisfy

$$\mathbf{G}\mathbf{dp} = -\mathbf{g}(\mathbf{p}). \tag{11}$$

In general, $\mathbf{G}$ is not square, so $\mathbf{G}^{-1}$ does not exist. One way to solve for $\mathbf{dp}$ is to use the pseudoinverse of $\mathbf{G}$:

$$\mathbf{dp} = -(\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1})\mathbf{g}(\mathbf{p}). \tag{12}$$

This idea works fine as long as $\mathbf{G}\mathbf{G}^T$ is nonsingular, which is usually the case for a system that contains only equality constraints. However, we often run into singularities when contact constraints are involved. This is because the collision detector may find many contact points between a pair of objects, leading to constraints that are redundant. One approach to deal with singularities is to use a pseudoinverse formula based on singular value decomposition of $\mathbf{G}$. By truncating the small (nearly zero) singular values, we can find a pseudoinverse of $\mathbf{G}$ even when $\mathbf{G}\mathbf{G}^T$ is singular.

Using a singular value decomposition in each time step, however, would be expensive, and it does not take the inequality constraints involved in contact into account. Instead, we find it natural to pose the post-stabilization problem as an LCP, just as we do with the dynamics equations. An additional benefit is that we get a more physically meaningful pseudoinverse. We explain this method in the next section.

V. Fitting Post-Stabilization into the Dynamics LCP Framework

As mentioned above, in the absence of contact constraints, we can find the stabilization term using the pseudoinverse of $\mathbf{G}(\mathbf{p})$ (Equation 12). Doing so is equivalent to solving the system

$$-\begin{pmatrix} \mathbf{I} & -\mathbf{G}(\mathbf{p})^T \\ \mathbf{G}(\mathbf{p}) & 0 \end{pmatrix}\begin{pmatrix} \mathbf{dp} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{g}(\mathbf{p}) \end{pmatrix}. \tag{13}$$

In other words, we can express the problem of finding the post-stabilization step as a problem of finding Lagrange multipliers, just as we did with the dynamics equations (see Equation 4). We can make 13 look more like Equation 4 by doing two things:

1) Replace $\mathbf{G}(\mathbf{p})$ with $\mathbf{J}(\mathbf{p})$. These two matrices are both constraint Jacobians. The difference is that $\mathbf{G}$ multiplies with changes in position (7-vectors consisting of a translation, plus a quaternion), whereas $\mathbf{J}$ multiplies with twists, which are 6-vectors. As a result of using $\mathbf{J}$, the post-stabilization step $\mathbf{dp}$ would be a twist and needs to be converted back into our position representation. [2]
2) Replace the identity matrix with the mass matrix $\mathbf{M}$. By doing so, we are no longer using the pseudoinverse $\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}$, but a weighted pseudoinverse, $\mathbf{M}^{-1}\mathbf{G}^T(\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T)^{-1}$. This corresponds to favouring the position change that requires the least amount of energy. This way, for example, a rotation around an axis with low moment of inertia would be favoured over a rotation around an axis with high moment of inertia.

Making these two changes gives us

$$-\begin{pmatrix} \mathbf{M} & -\mathbf{J}(\mathbf{p})^T \\ \mathbf{J}(\mathbf{p}) & 0 \end{pmatrix}\begin{pmatrix} \mathbf{dp} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{g}(\mathbf{p}) \end{pmatrix}. \tag{14}$$

Similar to our dynamics equations for systems with contact, it is natural to place complementarity constraints on the variables having to do with the contact constraints.

[2]Note that in two-dimensional rigid body simulations, $\mathbf{G}$ and $\mathbf{J}$ are identical, so this somewhat confusing distinction can be ignored.

- The post-step should never pull contacting bodies towards each other at the contact points, only push them apart: $\lambda_c \geq \mathbf{0}$
- If contact constraint functions were evaluated after adding the post step, the result must not be negative, but may be positive: $\mathbf{g}_c^+ = (\mathbf{g}_c^- + \mathbf{J}_c\mathbf{dp}) \geq 0$ (we use superscripts "-" and "+" to denote "before" and "after" the post-step. That is, $\mathbf{g}_c^- = \mathbf{g}(\mathbf{p})$, and $\mathbf{g}_c^+$ approximates $\mathbf{g}(\mathbf{p}+\mathbf{dp})$)
- $\lambda_c^T\mathbf{g}_c^+ = \mathbf{0}$: This constraint roughly means that for each contact c, either we are pushing the bodies apart at c or contact c's constraint will be satisfied in the absence of any push at c.

Adding the stabilization for contact constraints, we have the LCP

$$\begin{pmatrix} 0 \\ 0 \\ \mathbf{g}_c^+ \end{pmatrix} - \begin{pmatrix} \mathbf{M} & -\mathbf{J}_e^T & -\mathbf{J}_c^T \\ \mathbf{J}_e & 0 & 0 \\ \mathbf{J}_c & 0 & 0 \end{pmatrix}\begin{pmatrix} \mathbf{dp} \\ \lambda_e \\ \lambda_c \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{g}_e^- \\ \mathbf{g}_c^- \end{pmatrix}, \tag{15}$$
$$\mathbf{g_c}^+ \geq \mathbf{0}, \lambda_c \geq \mathbf{0}, \lambda_c^T\mathbf{g}_c^+ = \mathbf{0}.$$

VI. Results and Discussion

A. 6-Link Chain

In the following test of our stabilization method, we simulate a 6 link chain falling freely under gravity (see figure 2). At time 0, the chain is completely horizontal. We observe the change in constraint error over time using one of three test conditions for comparison: no stabilization, Baumgarte stabilization, or post-stabilization. We evaluate the constraint function $\mathbf{g}(\mathbf{p})$ for each constraint, and use the maximum absolute value as a measure of constraint error. This number roughly the largest joint separation distance. For comparison, each link in the chain is 100mm long. The time step size is 0.001 seconds.

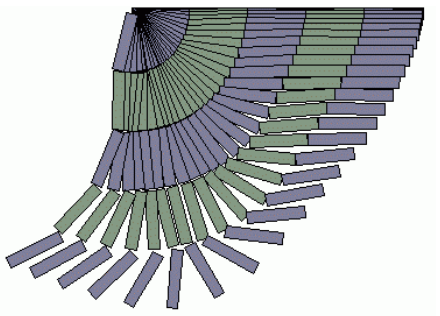
Fig. 2. A simulation of a 6-link chain falling freely under gravity. Screen clearing between frames is turned off to show motion over time. There is no constraint stabilization, so error in the constraints is evident as time progresses (note separation between the lowest two links of the chain).

Figure 3 shows us how the simulation behaves in the absence of any stabilization. The error grows over time as the joints of the chain separate. The graph has stair steps because of the periodic nature of the swinging chain, which behaves something like a pendulum. The error grows rapidly when the chain is moving more quickly.

By the end of 600 time steps (0.6 seconds), the error has climbed to around 20mm.

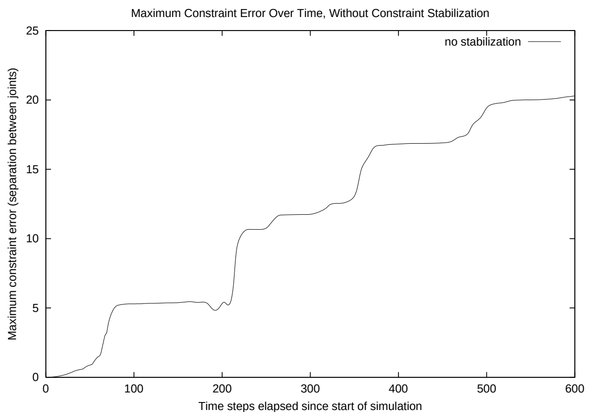

Fig. 3. Maximum constraint error over time, for a simulation of a swinging 6-link chain without any stabilization

In figures 4 and 5, we see the effect of Baumgarte stabilization on the error. The choice of constant used in Baumgarte stabilization has a large effect on how well the stabilization works. In figure 4, we show two choices of constant (50 and 200) that seemed to work best for this situation. Below 50, the error became much larger. With constant of 50, the maximum error is around 5mm, and when the constant is 200, we get slightly better results, with the error never exceeding 3mm.

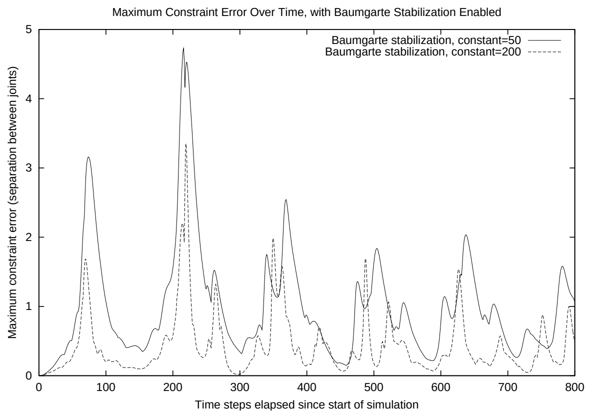

Fig. 4. Maximum constraint error over time, for a simulation of a swinging 6-link chain with Baumgarte stabilization, using a constant of 50 or 200

If the constant is raised higher, the error does not continue to go down. Instead, we start to notice another problem: the chain starts to jiggle unrealistically. This corresponds to the error correction term growing so large that it overshoots its goal in a single time step. Figure 5 shows an example of this, using Baumgarte stabilization with a constant of 2000. When one watches this simulation, one notices the chain bouncing around rapidly due to the large stabilization forces. Correspondingly, the error graph is quite jagged, although quantitatively the error is not much worse than when a constant of 50 is used.

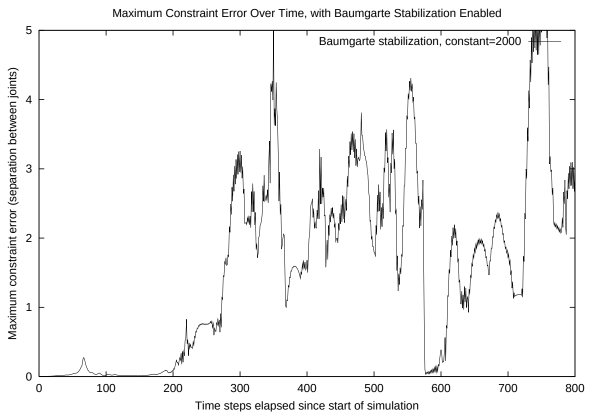

Fig. 5. Maximum constraint error over time, for a simulation of a swinging 6-link chain with Baumgarte stabilization, using a constant of 2000

We show the results of using post-stabilization in figures 6 and 7. Figure 6 shows two curves: one is the constraint error measured in each step *before* the postStabilization step is applied, and the other is the error *after* the postStabilization step. Note that the scale in figure 6 is only one tenth that of the scale in the Baumgarte stabilization graphs above. Even in this scale, the “after postStabilization” curve is barely perceptible. Figure 7 shows just this curve, on an even smaller scale. The constraint error in this curve never goes above 0.01mm.

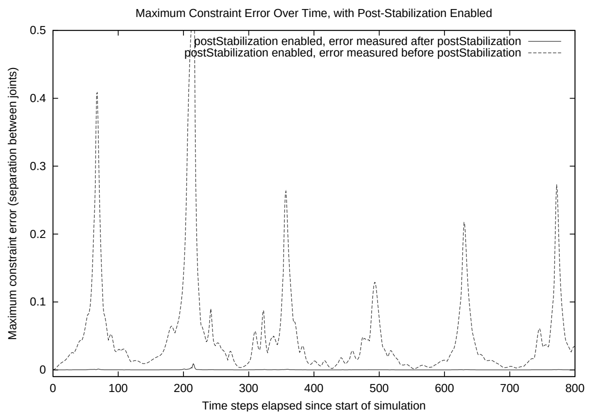

Fig. 6. Maximum constraint error over time, for a simulation of a swinging 6-link chain with post-stabilization enabled. Error is shown both before and after applying the post-stabilization step.

B. Stabilization of Contact Constraints

Contact constraints also have problems with constraint error. A contact constraint is said to have error if the separation distance at the contact is less than the *contact tolerance*. Figure 8 shows contact constraint error over a period of time when a rigid rectangle is pushed around on the screen by the user. Over this time period, the body experiences many different kinds of contact: impacts, resting contacts, sliding contacts and rolling contacts,

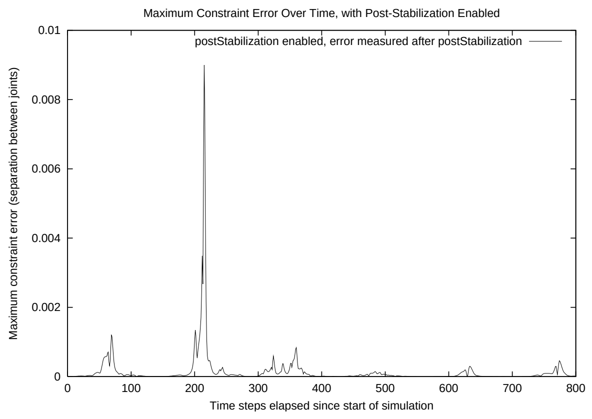

Fig. 7. Maximum constraint error over time, for a simulation of a swinging 6-link chain with post-stabilization enabled. This graph is a close-up of the smaller curve in figure 6

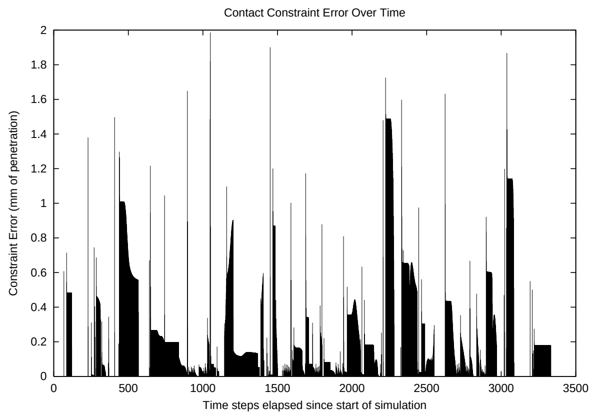

Fig. 8. Contact constraint error over time for a rigid rectangle undergoing user interaction. No stabilization is used.

sometimes with just one point of contact, sometimes with more than one (see figure 9 for some examples of typical motions).

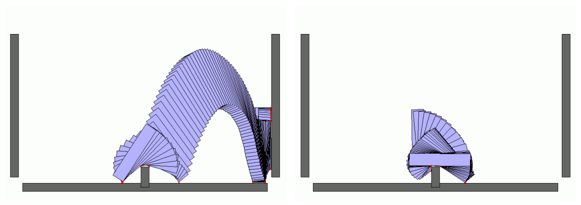

Fig. 9. Two screen captures from a contact simulation with a single moving rigid rectangle. User is pushing the rectangle interactively using repulsive forces at the mouse cursor.

When we run the same simulation with post-stabilization enabled, the constraint errors are eliminated completely. The post-stabilization scales well to simulations with many bodies and many contacts, such as the simulation shown in figure 10.

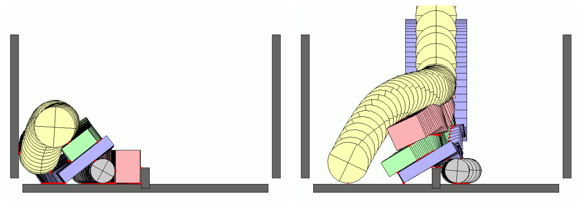

Fig. 10. Two screen captures from a contact simulation with five moving rigid bodies.

C. Advantages and Disadvantages of Post-Stabilization

Comparing Baumgarte stabilization with our post-stabilization method, here are some of the trade-offs:

- Baumgarte stabilization uses special constants that must be carefully set by hand. Post-stabilization does not require any tweaking.
- With post-stabilization, the constraint error usually is eliminated in the same time step that it is created. With Baumgarte stabilization, the constraint error cannot begin to affect the motion until one step later. Usually it takes several time steps for the error to dissipate completely. The amount of time it takes to absorb the constraint error depends on the values for the special constants. If the constants are too small, then the error will dissipate slowly. If the constants are too large, the stabilization will over-correct, causing jerky oscillations of the bodies.
- Baumgarte stabilization is easier to implement, and involves very little extra computation per time step. It only involves measuring the constraint error and adjusting the constant $\mathbf{k}$ in the constraint equation $\mathbf{Jv}+\mathbf{k}=0$. Post-stabilization actually requires formulating and solving another mixed LCP in addition to the dynamics LCP. This is quite a large penalty because solving these LCPs is one of the main bottlenecks in simulation performance.[3]
- Post-stabilization can perform poorly when there are large errors in the constraints. This is because the linear approximation made in Equation 10 becomes a poor approximation. Usually this results in a disturbing crash of the simulation, with the bodies flying off wildly. It is also possible for the bodies to jump into interpenetration during the post-step in some circumstances. This can be helped some by restricting the size of the post-step, and taking multiple post-steps in the same time step, but with the penalties of having to solve more systems, and having to choose thresholds. In our experience, Baumgarte stabilization seems to perform better when there is a lot of constraint error, especially if the constants used are not too large.

[3]Recently, (after the present work was completed) Anitescu and Hart [AH02] have shown how the stabilization described in the paper can be done while solving a modification of the LCP (7). This saves the cost of solving a second LCP.

VII. Conclusion

We described a new technique for performing post-stabilization of contact constraints in a linear complementarity framework. Our approach requires no parameter tweaking and is effective in eliminating the drift problem. We presented examples that show the effectiveness of the technique, and discussed the tradeoffs in comparison to previous techniques. Thus constraint stabilization and dynamics simulation are unified in the same framework, which simplifies the design of contact simulation software.

VIII. References

[ACR94] Uri M. Ascher, Hongsheng Chin, and Sebastian Reich. Stabilization of DAEs and invariant manifolds. *Numerische Mathematik*, 67(2):131–149, 1994.

[AH02] Mihai Anitescu and Gary D. Hart. A constraint-stabilized time-stepping approach for multi-body dynamics with contact and friction. *Reports on computational mathematics, ANL/MCS-P1002-1002, Mathematics and Computer Science division, Argonne National Laboratory,* December 2002.

[Ani97] Mihai Anitescu. *Modeling Rigid Multi Body Dynamics with Contact and Friction*. PhD thesis, University of Iowa, 1997.

[AP97] Mihai Anitescu and Florian Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 1997.

[AP98] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, 1998.

[Asc97] Uri M. Ascher. Stabilization of invariants of discretized differential systems. *Numerical Algorithms*, 14(1–3):1–24, 1997.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. *Computer Graphics*, 28(Annual Conference Series):23–34, 1994.

[Bar96] David Baraff. Linear-time dynamics using Lagrange multipliers. *Computer Graphics*, 30(Annual Conference Series):137–146, 1996.

[Bau72] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, (1):1–16, 1972.

[CD68] R. W. Cottle and G. B. Dantzig. Complementary pivot theory of mathematical programming. *Linear Algebra and Appl.*, (1), 1968.

[Cli02] Michael B. Cline. Rigid body simulation with contact and constraints. Master's thesis, University of British Columbia, July 2002.

[CPS92] R.W. Cottle, J.S. Pang, and R.E. Stone. *The Linear Complementarity Problem*. Academic Press, Inc., 1992.

[LNPE92] C. Lubich, U. Nowak, U. Pohle, and Ch. Engstler. Mexx – numerical software for the integration of constrained mechanical multibody systems. Technical Report SC92-12, Konrad-Zuse-Zentrum fur Informationstechnik, 1992.

[Lot82] P. Lotstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal on Applied Mathematics*, 1982.

[Mir98] B. Mirtich. Rigid body contact: Collision detection to force computation, 1998.

[Mur88] Katta G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*[4]. Heldermann Verlag, Berlin, 1988.

[ST96] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *Internat. J. Numer. Methods Engineering*, 1996.

[4]This book is out of print, but is currently available online at `http://ioe.engin.umich.edu/people/fac/books/murty/linear_complementarity_webbook/`