A convex, smooth and invertible contact model for trajectory optimization

Emanuel Todorov
Departments of Applied Mathematics and Computer Science and Engineering
University of Washington

Abstract— **Trajectory optimization is done most efficiently when an inverse dynamics model is available. Here we develop the first model of contact dynamics defined in both the forward and inverse directions. The contact impulse is the solution to a convex optimization problem: minimize kinetic energy in contact space subject to non-penetration and friction-cone constraints. We use a custom interior-point method to make the optimization problem unconstrained; this is key to defining the forward and inverse dynamics in a consistent way. The resulting model has a parameter which sets the amount of contact smoothing, facilitating continuation methods for optimization. We implemented the proposed contact solver in our new physics engine (MuJoCo). A full Newton step of trajectory optimization for a 3D walking gait takes only 160 msec, on a 12-core PC.**

I. Introduction

Optimal control theory provides a powerful set of methods for planning and automatic control. These methods have traditionally been applied to smooth dynamical systems because of the common requirement for differentiability. However many robotic tasks – such as legged locomotion and object manipulation – involve contact dynamics which are inherently discontinuous. Trajectory optimization in the presence of contacts is indeed a difficult computational problem and has rarely been addressed. There are exceptions in the domain of locomotion [8], [10], however the examples we are aware of require the user to provide contact information to the optimizer. This greatly reduces the level of automation. Even if the user is willing to do the extra work, in more complex tasks such as object manipulation (especially with a non-anthropomorphic robotic hand) humans may be unable to guess when and where contacts should occur. Here we aim to automate this process. Our main contribution is a new contact model which is well suited for trajectory optimization.

Before discussing the details, we outline the requirements that such a model should meet. **i.** The output of the computation should be a differentiable function of the input, so as to facilitate numerical optimization. **ii.** Contact interaction forces should decay smoothly with distance (otherwise the result will not be differentiable with respect to the position variables). **iii.** The computation should be efficient, because it will have to be repeated for a large number of states in the course of optimization (in particular to compute derivatives through finite-differencing). These three requirements can be met by extending existing contact models such as the widely used Linear Complementary Problem (LCP) formulation [3], [2], [1]. Indeed in our recent work we have developed such an extension based on a stochastic LCP [4].

There is however one additional requirement which we believe is essential and requires a qualitatively new approach: **iv.** The contact model should be defined for both forward and inverse dynamics. Here the forward dynamics

$$\mathbf{w}_{t+1} = \mathbf{a}\left(\mathbf{q}_t, \mathbf{w}_t, \mathbf{u}_t\right) \tag{1}$$

compute the next-step[1] velocity $\mathbf{w}_{t+1}$ given the current generalized position $\mathbf{q}_t$, velocity $\mathbf{w}_t$ and applied force $\mathbf{u}_t$, while the inverse dynamics

$$\mathbf{u}_t = \mathbf{b}\left(\mathbf{q}_t, \mathbf{w}_t, \mathbf{w}_{t+1}\right) \tag{2}$$

compute the applied force that caused the observed change in velocity. Computing the total force that caused the observed change in velocity is of course straightforward. The hard part is decomposing this total force into a contact force and an applied force. Such decomposition is needed in order to define under-actuation as well as control costs.

One might wonder if inverse contact dynamics can even be defined; for example if both your feet are on the ground and you try to push them apart, friction will prevent movement and so the force you apply will not affect the positions and velocities of your joints. Indeed the contact model developed here will rely on a convex approximation that involves smoothing – in the sense that some contact interaction forces will be exerted from a distance (which is an adjustable parameters). Despite smoothing, however, our model generates sufficient forces to prevent penetration – like LCP models and unlike spring-damper models. Approximations that have an adjusable smoothness parameter are particularly useful in a continuation framework: start with a large amount of smoothing for which the optimal trajectory is easy to find, then gradually reduce smoothing while tracking the optimal solution. In the limit the model will represent hard contact and will no longer be invertible; however the computation will terminate some distance away from the limit where the inverse is well-defined. This idea is reminiscent of interior-point methods for constrained optimization [11], and indeed our model defines the contact interaction force as the output of a custom interior-point solver for conic programming.

A. The advantages of inverse dynamics

To see why we need the inverse dynamics, consider optimizing a finite-horizon trajectory cost in the form

$$J = r_T\left(\mathbf{q}_T, \mathbf{w}_T\right) + \sum\nolimits_{t=1}^{T-1} r_t\left(\mathbf{q}_t, \mathbf{w}_t, \mathbf{u}_t\right) \tag{3}$$

[1] We assume that the dynamics are represented in discrete-time, which is now standard in contact modeling [1].

One approach to such optimization is to treat the position and force sequences $\mathbf{q}_{1 \cdots T}$ and $\mathbf{u}_{1 \cdots T-1}$ as independent variables, define $\mathbf{w}$ from $\mathbf{q}$ via numerical differentiation, and then optimize J subject to the forward dynamics constraints. This approach is common and we have also used it [4]. Nevertheless it has a shortcoming, which is that it effectively doubles the number of variables being optimized and then eliminates the extra degrees of freedom by imposing dynamics equality constraints. Since these constraints are nonlinear, such an approach tends to slow down optimization algorithms. One could treat only $\mathbf{q}_1$ as independent and define the remaining $\mathbf{q}$'s using the forward dynamics, but then the Hessian of J w.r.t. $\mathbf{u}$ becomes dense which again slows down the optimization. The latter "dense" approach has the additional drawback that it only works for finite-horizon optimization and cannot be generalized to, say, limit cycle optimization.

The alternative is to treat the sequence of positions $\mathbf{q}_{1 \cdots T}$ as the only independent variable, define $\mathbf{w}$ via numerical differentiation, and further define $\mathbf{u}$ from the inverse dynamics. This explicitly resolves the above equality constraints, allowing the optimizer to search in a lower-dimensional space and freeing it from the need to enforce so many constraints. Note that the Hessian of J w.r.t. $\mathbf{q}$ is sparse. Indeed some of the most impressive trajectory optimization results in the presence of contact have used this inverse dynamics approach [10], however since an invertible contact model was not available, these authors had to fix the contact states manually outside the optimization loop.

Another significant advantage of the inverse dynamics approach is that it can get away with large timesteps. This is because the dynamics do not need to be integrated forward in time in the course of optimization, and thus stability is not an issue. In contrast, trajectory optimization with respect to the forward dynamics can require very small timesteps to avoid instabily. A recent example of such optimization in the context of locomotion [16] had to use a frame rate of 2400 Hz with the Open Dynamics Engine – which slowed down the dynamics evaluation to almost real time. With the approach developed here, using a frame rate of 100Hz and our efficient new physics engine [6], dynamics evaluation for a similar humanoid model is around 4000 times faster than real time (on a computer with two 6-core processors). This enables us to perform a full Newton step of trajectory optimization, including the necessary finite-differencing and sparse Hessian factorization, in around 160 msec. The details of this application are given in [7]. The engine itself is described in more detail in [6]. In the present paper we focus on the contact model.

In recent work [5] we have developed another contact model which is aslo implemented in the new engine. It has the advantage that it can handle non-linear complementary problems efficiently, and so it can simulate forward dynamics with hard contacts faster and more accurately than LCP solvers. However, like LCP, it is restricted to forward dynamics and does not have a well-defined inverse.

II. Preliminaries

A. Projecting the dynamics in contact coordinates

Let $M(\mathbf{q})$ denote the generalized inertia matrix, $\mathbf{n}(\mathbf{q}, \mathbf{w})$ the vector of Coriolis, centrifugal, gravitational and other passive forces, $J(\mathbf{q})$ the Jacobian of the active contacts (see below), and $\mathbf{f}$ the impulse in contact space. Then the equations of motion in continuous time are

$$M(\mathbf{q})\, d\mathbf{w} = (\mathbf{n}(\mathbf{q}, \mathbf{w}) + \mathbf{u})\, dt + J(\mathbf{q})^{\mathsf{T}} \mathbf{f} \tag{4}$$

Using Euler approximation, these equations can be represented in discrete time as

$$M_t \mathbf{w}_{t+1} = M_t \mathbf{w}_t + (\mathbf{n}_t + \mathbf{u}_t)\, h + J_t^{\mathsf{T}} \mathbf{f}_t \tag{5}$$

where h is the discretization time step. Multiplying by $J_t M_t^{-1}$ we obtain the discrete-time equations of motion in contact space:

$$\mathbf{v}_{t+1} = \mathbf{c}_t + A_t \mathbf{f}_t \tag{6}$$

The coordinate system over contact space will be clarified below. $\mathbf{v}_{t+1} = J_t \mathbf{w}_{t+1}$ is the next-state contact velocity, $A_t = J_t M_t^{-1} J_t^{\mathsf{T}}$ is the inverse inertia in contact space, and

$$\mathbf{c}_t = J_t \mathbf{w}_t + J_t M_t^{-1} (\mathbf{n}_t + \mathbf{u}_t)\, h \tag{7}$$

is the contact velocity which would result if $\mathbf{f}_t = 0$.

We will now focus on equation (6) and drop the time indices for clarity. The goal will be to define the mappings

$$\begin{aligned} (A, \mathbf{c}) &\overset{\text{forward}}{\longrightarrow} (\mathbf{f}, \mathbf{v}) \\ (A, \mathbf{v}) &\overset{\text{inverse}}{\longrightarrow} (\mathbf{f}, \mathbf{c}) \end{aligned} \tag{8}$$

Once these are defined, we can recover the forward and inverse dynamics in generalized coordinates using (5).

B. Constraints on the contact velocities and impulses

Let n_l and n_c denote the number of active joint limits and contacts at the present time step. The vectors $\mathbf{f}, \mathbf{v}, \mathbf{c}$ have dimensionality $n_l + 3n_c$. The first n_l components correspond to the joint limits (we assume that only 1-dof joints like hinges and sliders can have limits), while the remaining components correspond to the contacts. Each contact has a 3D coordinate frame associated with it; we will denote the vector components corresponding to contact number i with $(f_{i,1}, f_{i,2}, f_{i,3})$ where the first two are the tangent components and the last is the normal component.

Let the friction coefficient at contact i be μ_i. The first constraint we impose is that the contact impulses must lie within the friction cones:

$$\mu_i^2 f_{i,3}^2 - f_{i,1}^2 - f_{i,2}^2 \geq 0 \tag{9}$$

Another constraint is that the normal impulses (for both joint limits and contacts) must be non-negative, i.e. they can push the contacting bodies apart but cannot pull them towards each other:

$$N\mathbf{f} \geq 0 \tag{10}$$

Here N is the matrix of 1's an 0 's that extracts the normal components: $N\mathbf{f} = (f_1 \cdots f_{n_l}, f_{1,3} \cdots f_{n_c,3})^{\mathsf{T}}$.

Finally, the normal velocity after the impulse must be such that penetration is avoided:

$$N\mathbf{v} - \mathbf{v}_{\min} \geq 0 \tag{11}$$

$\mathbf{v}_{\min}$ has dimensionality $n_l + n_c$ and encodes the minimal acceptable normal velocity at each limit and contact. There are two reasons why the components of $\mathbf{v}_{\min}$ may be non-zero. If penetration has already occurred, we want to not only prevent further penetration but also fix the current penetration, in which case $\mathbf{v}_{\min}$ is positive. Conversely, if contact has not yet occurred but proximity between two bodies has been detected (by using a safety margin in the collision detection engine), we can set $\mathbf{v}_{\min}$ to a negative number which ensures that penetration will not occur on the next step.

One could impose additional constraints. For example, the LCP formulation involves the complementarity constraint

$$(N\mathbf{f})^{\mathsf{T}} (N\mathbf{v} - \mathbf{v}_{\min}) = 0 \tag{12}$$

However it has been argued [9] that these additional constraints do not necessarily correspond to fundamental physical properties. Indeed it is not clear if complementarity will hold empirically when multiple contacts are active. This points to a larger issue, which is that rigid bodies in contact no longer behave like rigid bodies, and all available contact models are approximations.

Here we only enforce constraints (9, 10, 11), and observe that our new contact model yields seemingly realistic behavior as shown in the accompanying movie. Numerical comparisons to a complementarity solver remain to be done. Complementarity is in fact roughly enforced (as a soft constraint) because our model is defined in terms of kinetic energy minimization, and violating complementarity tends to lead to higher-than-necessary kinetic energy. Attempting to enforce complementarity strictly, however, leads to non-convexity and (as far as we can tell) makes it impossible to formulate a proper inverse model.

III. The new contact model

The idea behind the new model is simple. What do contact forces do? They prevent movement in contact space[2]. It then makes sense to look for the smallest possible contact velocity that can be achieved without violating the constraints. It is natural to measure the magnitude of velocity in terms of kinetic energy, which yields the optimization problem

$$\min_{\mathbf{f},\mathbf{v}} \frac{1}{2}\mathbf{v}^{\mathsf{T}} A^{-1}\mathbf{v} \tag{13}$$
$$\text{subject to } (9, 10, 11)$$

Our model (in the forward dynamics direction) is defined as the solution to the above optimization problem, with some caveats explained later. Note the similarity to the Gauss principle of equality-constrained dynamics [12]. The Gauss principle says that the constraint forces are the forces that minimize a kinetic-energy-like term subject to the constraints. This principle has not been applied to contact dynamics as far as we know, however there is a related principle of maximal dissipation [15] which has been applied, see below.

[2]Throughout the paper we focus on inelastic collisions. Our work is easily extended to elastic collisions via Poisson's hypothesis: compute the impulse for the inelastic phase and then correct it using the coefficient of restitution.

Importantly, the optimization problem (13) is convex, which means that it has a unique solution (i.e. no local minima) and the solution can be found quickly with existing numerical methods. The uniqueness of the solution will play a key role in defining the inverse dynamics later. Since (9) describes a cone, (13) is an instance of conic programming.

A. *Defining the forward dynamics*

We can express the energy function in (13) as

$$\frac{1}{2}(A\mathbf{f} + \mathbf{c})^{\mathsf{T}} A^{-1} (A\mathbf{f} + \mathbf{c}) = \frac{1}{2}\mathbf{f}^{\mathsf{T}} A\mathbf{f} + \mathbf{f}^{\mathsf{T}}\mathbf{c} + const \tag{14}$$

The constant depends on $A, \mathbf{c}$ which are known in the forward dynamics, so we can ignore it. We will further include a regularizing diagonal matrix R with positive elements, which also provides smoothing (see below). Then the energy is

$$\ell(\mathbf{f}) = \frac{1}{2}\mathbf{f}^{\mathsf{T}} (A + R)\mathbf{f} + \mathbf{f}^{\mathsf{T}}\mathbf{c} \tag{15}$$

We want to minimize $\ell(\mathbf{f})$ subject to the above constraints. There are two popular classes of numerical methods for constrained optimization: sequential quadratic programming, and interior point methods. The latter (used here) have gained popularity recently and are considered a revolution in the field of numerical optimization [11]. The idea is to replace the constraints with barrier functions which go to infinity at the constraints, and thereby transform the optimization problem into an unconstrained one. In our case (9, 10, 11) involve a total of $2n_l + 3n_c$ scalar constraints, which can be written in the form $s_j(\mathbf{f}) \geq 0$ using the fact that $\mathbf{v} = A\mathbf{f} + \mathbf{c}$ and $\mathbf{c}$ is known. We thus define the log-barrier function

$$d(\mathbf{f}) = -\kappa \sum\nolimits_{j=1}^{2n_l+3n_c} \log s_j(\mathbf{f}) \tag{16}$$

where $\kappa > 0$ determines the amount of constraint smoothing (in the limit $\kappa \to 0$ the solution becomes exact). The contact impulse is now defined as the unconstrained minimizer of

$$\ell(\mathbf{f}) + d(\mathbf{f}) \tag{17}$$

with κ fixed. Details on how this optimization problem is solved numerically will be given below.

B. *Defining the inverse dynamics*

Now suppose we are given $A, \mathbf{v}$ and need to compute $\mathbf{f}, \mathbf{c}$. Since $\mathbf{f}$ was defined as the unconstrained minimizer of $\ell + d$, it must satisfy

$$\nabla_{\mathbf{f}}\ell + \nabla_{\mathbf{f}}d = 0 \tag{18}$$

Both ℓ and d are convex functions, thus the above equation has a unique solution. We will now use this fact to invert the forward model. We have

$$\nabla_{\mathbf{f}}\ell = (A + R)\mathbf{f} + \mathbf{c} = R\mathbf{f} + \mathbf{v} \tag{19}$$

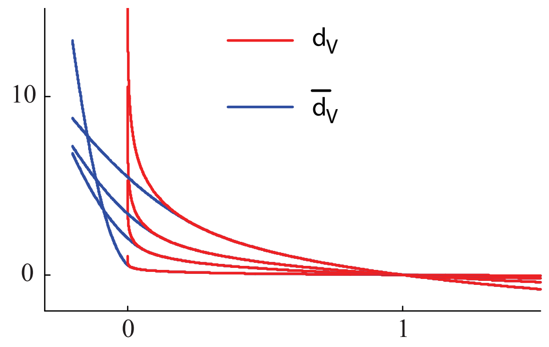

Fig. 1. The functions d_V and $\overline{d}_V$ for $q = 7$ and $\kappa = 0.1, 0.5, 1, 2$.

The log-barrier function d can be decomposed as

$$d = d_F(\mathbf{f}) + d_V(\mathbf{v}) \tag{20}$$

where d_F corresponds to (9, 10) and d_V to (11). Then

$$\nabla_{\mathbf{f}} d = \nabla_{\mathbf{f}} d_F + \frac{\partial \mathbf{f}}{\partial \mathbf{v}} \nabla_{\mathbf{v}} d_V = \nabla_{\mathbf{f}} d_F + A \nabla_{\mathbf{v}} d_V \tag{21}$$

and (18) becomes

$$R\mathbf{f} + \mathbf{v} + \nabla_{\mathbf{f}} d_F + A \nabla_{\mathbf{v}} d_V = 0 \tag{22}$$

It is notable that the latter expression no longer depends explicitly on $\mathbf{c}$ but only on $\mathbf{v}$, which is known in the inverse dynamics computation. Now (18) is satisfied if and only if $\mathbf{f}$ is the unconstrained minimizer of

$$\frac{1}{2}\mathbf{f}^{\mathsf{T}} R\mathbf{f} + d_F(\mathbf{f}) + \mathbf{f}^{\mathsf{T}}(\mathbf{v} + A\nabla_{\mathbf{v}} d_V(\mathbf{v})) \tag{23}$$

This is again a convex function and has no local minima.

In summary, the contact impulse $\mathbf{f}$ is defined as the minimizer of (17) in the forward dynamics and as the minimizer of (23) in the inverse dynamics.

C. Contact smoothing and regularization

The diagonal matrix R serves two related purposes. The obvious one is regularization: larger values on the diagonal of R result in smaller impulses. The second purpose is spatial smoothing: we want some contact forces to be felt from a distance so as to avoid discontinuities which slow down the trajectory optimizer. Let the normal distance (for either a joint limit or a contact) be w. One way to define the corresponding diagonal element(s) of R is

$$\begin{cases} R_{\min} + (R_{\max} - R_{\min})\dfrac{w}{w_{\max}} & \text{if } w \in [0, w_{\max}] \\ R_{\max} \quad \text{if } w > w_{\max} \\ R_{\min} \quad \text{if } w < 0 \end{cases} \tag{24}$$

For large w the elements of R are large, the impulse is penalized more, and thus contact interactions decay with distance. $w_{\max}$ will normally equal the safety margin used by the collision detection engine. R could depend on w in other ways, say exponentially – which we have recently found to have empirical advantages [7].

D. A primal interior-point method with elastic mode

Interior point methods apply Newton's method for a decreasing sequence of κ, and use the solution from each phase to initialize the next phase. The simplest approach is to apply Newton's method directly to (17) or (23). This is known as a primal method going back to the 60's [13]. Nowadays primal-dual methods are more commonly used; they augment the system with Lagrange multipliers and solve a larger problem. The transition from primal to primal-dual methods was motivated by ill-conditioning of the Hessian for small values of κ. Yet recent analysis [11] shows that such ill-conditioning is not actually a problem, and instead the only drawback of primal methods appears to be that the Newton step often lands far outside the feasible region, requiring many steps of backtracking line-search to get back. This analysis however assumes that the feasible region is defined by general nonlinear inequalities that can only be handled numerically. The situation here is different because our constraints are planes and cones. Given the Newton search direction, we can therefore compute the intersection with the constraint surfaces analytically, and remain in the feasible region without expensive backtracking. This yields an efficient primal method.

Another algorithmic issue is the availability of a feasible initial condition. If any constraints are violated the log-barrier function becomes infinite; thus one has to either guarantee a feasible initial condition, or implement an extended method which can handle infeasible points. In our case it is not clear how to find a feasible initial condition quickly. Therefore our solver uses a more elaborate method as follows. Given an initialization for $\mathbf{f}$, we first modify it so as to satisfy (9, 10), by making the normal components positive and scaling down the tangential components if necessary. Then, at the beginning of each iteration, we check if (11) is also satisfied. If so, we proceed with the primal method as described above. Otherwise we modify the function d_V so that it is everywhere finite: for each component s of $N\mathbf{v} - \mathbf{v}_{\min}$, we set

$$\overline{d}_V(s) = \begin{cases} -\kappa \log(s) & \text{if } s \geq s_0 \\ a_0 s^2 + a_1 s + a_2 & \text{if } s < s_0 \end{cases} \tag{25}$$

Here s_0 is chosen so that the slope of $-\kappa \log(s)$ evaluated at s_0 equals a predefined value q: $s_0 = -\kappa/q$. The coefficients a_0, a_1, a_2 are then computed so that the first and second derivatives of the quadratic and the log-barrier functions coincide at s_0. In this way $\overline{d}_V$ has a continuous second derivative. In the limit $\kappa \to 0$ the essential property of the log-barrier function (i.e. forcing the solution within the feasible region) is recovered. The original and modified barrier functions are illustrated in Fig 1. Note that once the solution enters the feasible region, the algorithm automatically switches from $\overline{d}_V$ to d_V and so the solution is guaranteed to remain feasible.

E. Relation to the maximal dissipation principle

For smooth continuous-time dynamics, the rate of change in kinetic energy equals the rate of work. For a scalar pointmass for example, we have

$$\frac{d}{dt}\left(\frac{1}{2}m\dot{x}^2\right) = m\ddot{x}\dot{x} = f\dot{x} \tag{26}$$

One might have thought that the same relation holds for discrete-time impulse dynamics, and in particular that minimizing kinetic energy at the next time step is equivalent to maximizing the rate of negative work (i.e. maximizing dissipation). Indeed maximal dissipation has been used as a variational characterization of Coulomb friction [15]. It turns out however that maximal dissipation does not minimize kinetic energy in the present context. Indeed dissipation is

$$\mathbf{f}^\mathsf{T}\mathbf{v} = \mathbf{f}^\mathsf{T}A\mathbf{f} + \mathbf{f}^\mathsf{T}\mathbf{c} \tag{27}$$

which differs from the above expression for kinetic energy by $\frac{1}{2}\mathbf{f}^\mathsf{T}A\mathbf{f}$. We can make the two results coincide if we define dissipation using $\frac{1}{2}(\mathbf{v}+\mathbf{c})$ instead of $\mathbf{v}$.

Recent work generalized the maximal dissipation principle to 3D and defined the contact impulse as the solution to a pair of coupled convex optimization problems [14]. While this resembles our approach in spirit, the overall optimization problem these authors arrived at was no longer convex. The difference is not actually due to using kinetic energy vs. dissipation but to imposing different sets of constraints.

IV. Simulation Results

We implemented and tested the proposed algorithms in our new physics engine. First we verified numerically that the forward and inverse dynamics are indeed inverses of each other. This is of course guaranteed mathematically but sanity checks never hurt; furthermore, inconsistencies could arise if the optimizer fails to reach the minimum of either cost function. We found that the differences between the forward and inverse dynamics solutions were very small, on the same order as the convergence tolerance of the interior-point method (as one would expect).

Having established that our main goal – namely to define an invertible contact model – has been achieved, we next asked if the forward dynamics are realistic. This is important to test empirically because we are using a novel formulation of contact. The most reliable test for realism is to simulate interesting systems, interact with them in real time, and watch their responses. We worked with the two systems illustrated in Fig 2. The results are shown in the attached movie. For real-time interaction we used a Logitech 3D mouse. In the movies, the red arrow shows the force while the magenta arrow shows the torque applied by the user to the highlighted object. The simulation time step is 10 msec. The algorithm parameters are $\kappa = 0.1$, $q = 10$, $R_{\max} = 100$, $R_{\min} = 0.01$, $w_{\max} = 0.1$. Overall we were quite satisfied with the realism of the forward dynamics.

Next we illustrate the smoothness (over time) of the contact forces computed by the forward dynamics. Fig 3 shows results from dropping a single ball. The initial phase corresponds to gravitational acceleration. The ball then reaches the safety margin above the ground and the contact impulse starts to grow, gradually slowing down the ball

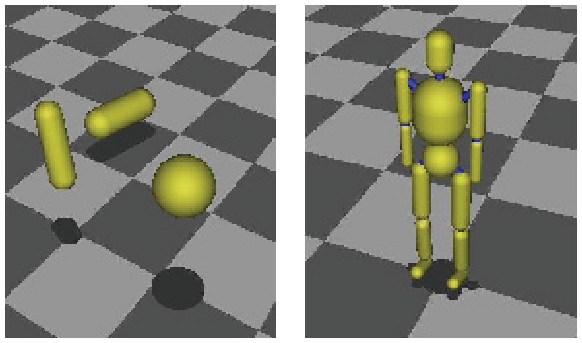

Fig. 2. The two systems we experimented with. See attached movie.

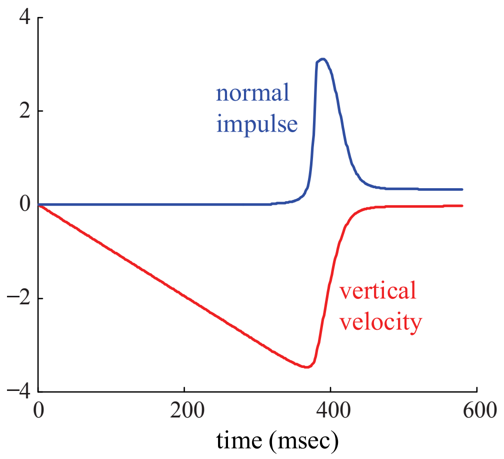

Fig. 3. Results from a ball-drop test. Shown are the velocity and ground reaction impulse in the vertical direction. The time step is 1 msec.

until it comes to a complete stop. Note how the impulse is distributed over time, and yet penetration does not occur. This is a very useful property to have if trajectories are being optimized numerically.

Next we show timing statistics for our inverse solver – which is more relevant than the forward solver since trajectory-optimization applications will spend most of their time in the inverse solver. The data were recorded while playing with the humanoid model. We achieved different numbers of contacts by enabling and disabling gravity, and also by applying vertical forces to different body parts. Fig 4 left shows the number of Newton iterations taken by the interior-point method with default initialization (i.e. no warm starts). We see that the number of iterations is quite small, and also that it grows modestly with number of contacts.

Fig 4 right shows the CPU time per simulation step. This is single-threaded performance on a 3 GHz Intel QX9650 processor running Windows 7. To collect the timing data, we run the inverse computation 1000 times per simulation step (so that it takes long enough to be timed accurately) and used the 1 msec resolution Windows multimedia clock. Note that the CPU time here is only for the inverse solver in contact space, and does not include the time needed to compute the smooth dynamics and project them in contact

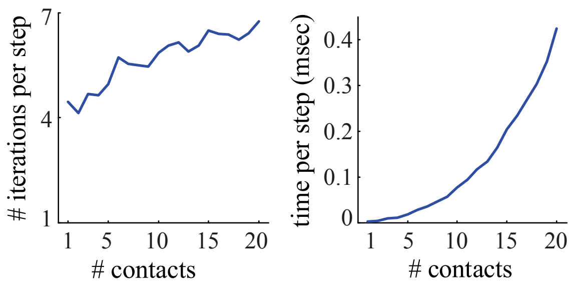

Fig. 4. Number of Netwon iterations taken by the interior-point inverse solver per simulation step, and CPU time, as a function of number of active contacts. The results are obtained from the humanoid simulation. Data for all time steps with the same number of active contacts are averaged.

space (although the latter is faster than the contact solver). As expected, profiling revealed that the bottleneck is in the factorization of the Hessian of the cost function – which is done by our custom Cholesky solver. The curve in the figure is well-fit by a cubic function, which is expected because Cholesky factorization has $O\left(n^3\right)$ complexity. The bottom line is that our algorithm is extremely fast. Table 1 makes the timing results a bit easier to interpret, by showing how many times per second we can run our inverse solver in a single thread.

# contacts	inverse evaluations per second
2	$241,774$
5	$54,190$
10	$12,991$
20	$2,361$

(Table 1)

Finally, note that in a trajectory-optimization application the inverse dynamics computations (which are the bottleneck) can be performed in parallel for all states along the trajectory. Since these computations are independent of each other, performance will scale roughly linearly with the number of available cores. Indeed we have observed such linear scaling when using a multi-threaded version of our new physics engine. We expect even higher performance when the engine is ported to GPUs – which we plan to do in the near future. A version of the engine running on clusters (that support MPI) is also in progress.

V. Summary and future work

We developed the first model of rigid-body contact which is defined for both forward and inverse dynamics. This was possible due to a new formulation of forward dynamics in terms of kinetic energy minimization under non-penetration and friction-cone constraints. The complementarity constraints used in most prior work were excluded. The resulting behavior was still realistic, and furthermore the contact model became convex – which was key to developing a proper inverse model. Both the forward and inverse dynamics were evaluated in simulations using our new physics engine. The tests revealed that the proposed algorithms are fast, accurate and stable.

While a new contact model is useful in itself (especially when it can be evaluated quickly), our main motivation is not so much physical simulation but control optimization. Such optimization can be done more efficiently when an inverse dynamics model is available. Although we achieved very high speed, and have plenty of room for improvement using multi-core processors, such speed is matched by the demands of trajectory optimization for complex systems. This is especially true when trajectory optimization is done in real time, as in model-predictive control settings. The present work (along with our new physics engine) sets the stage for model-predictive control applied to interesting robotics tasks. We suspect that such control schemes will be able to achieve qualitatively better performance than what is currently possible in robotic control.

Acknowledgements

This work was supported by the US National Science Foundation.

References

[1] D. Stewart and J. Trinkle. An implicit time-stepping scheme for rigid-body dynamics with inelastic collisions and Coulomb friction. *International Journal Numerical Methods Engineering* (1996).

[2] M. Anitescu and G. Hart. A fixed-point iteration approach for multibody dynamics with contact and small friction. *Mathematical Programming* (2004).

[3] F. Pfeiffer and C. Glocker. *Multibody dynamics with unilateral constraints*. Wiley Series in Nonlinear Science (2006).

[4] Y. Tassa and E. Todorov. Stochastic complementarity for local control of discontinuous dynamics. *Proceedings of Robotics: Science and Systems* (2010).

[5] E. Todorov. Implicit complementarity: A new approach to contact dynamics. *ICRA* (2010).

[6] E.Todorov. MuJoCo: A fast and accurate physics engine for model-based control. *Online preprint* (2011).

[7] T. Erez, Y. Tassa and E. Todorov. Inverse dynamics optimal control for periodic tasks with contacts. *Online preprint* (2011).

[8] M. Srinivasan and A. Ruina. Computer optimization of a minimal biped model discovers walking and running. *Nature* (2005).

[9] A. Chatterjee and A. Ruina. A new algebraic rigid body collision law based on impulse space considerations. *Journal of Applied Mechanics* (1998).

[10] K. Wampler and Z. Popovic. Optimal gait and form for animal locomotion. *SIGGRAPH* (2009).

[11] A. Forsgren, P. Gill, M. Wright. Interior point methods for nonlinear optimization. *SIAM Review* (2002).

[12] F. Udwadia and R. Kalaba. A new perspective on constrained motion. *Proceedings of the Royal Society* (1992).

[13] A. Fiacco and G. McCormick. *Nonlinear programming* (1968).

[14] D. Kaufman, S. Sueda, D. James and D. Pai. Staggered projections for friction contact in multibody systems. *SIGGRAPH ASIA* (2008).

[15] J. Moreau. On unilateral constraints, friction and plasticity. *New Variational Techniques in Mathematical Physics* (1973)

[16] J. Wang, D. Fleet and A. Hertzman. Optimizing walking controllers for uncertain inputs and environments. *SIGGRAPH* (2010).