Fast Multi-Body Simulations of Robots Controlled with Error Feedback

John Shepherd, Samuel Zapolsky, and Evan M. Drumwright

***Abstract*—Roboticists modeling control of manipulator and legged robots often assume force/torque-based control using an open-loop model of voltage, hydraulic pressure, or pneumatic pressure to actuator torques. This paper shows that such force/torque-based models can lead to stiff differential equations, which are computationally inefficient to solve: relatively small integration steps are necessary to ensure stability.**

We have investigated two approaches which appear to mitigate this problem: incorporating transmission modeling (applicable only to electromagnetic actuators at present) and inverse dynamics control. Both of these approaches require increased computation per integration step, but this work will demonstrate that the larger integration steps can still yield considerably higher simulation throughput.

This paper will also identify research challenges with using inverse dynamics control within multi-rigid body simulations. In particular, we examine the state of the art approach for integrating the simulation subject to contact and inverse dynamics constraints and identify algorithmic challenges, both theoretical and practical.

Experimental virtual robot platforms include a UR10 manipulator arm and a locomoting quadrupedal robot.

I. Introduction

Roboticists often wish to simulate controlled systems rapidly while prototyping control schemes and hardware designs. In these cases, reducing the duration of the "edit-compile-test" cycle is more important than reducing numerical solution error, which roboticists might do after obtaining some confidence in their approach.

As an anecdotal example of where such an approach might be useful, the last author recently tasked students in a course with producing gait generation and balance stabilization software for a simulated legged robot. The underactuated nature of the robot limited the utility of strictly kinematic simulation. And the inclusion of error feedback controllers, which requires careful tuning to balance dynamic performance with simulation stability, made debugging both more challenging (e.g., Did the robot fall because control was not sufficiently accurate?) and slower than desirable—with the inclusion of control, inverse kinematics, and planning code, simulations would run several times slower than real-time.

The present work continues a previous investigation [16] into means to accelerate this process. That work found that exponential energy dissipation can be used to increase simulation stability. However, it should be clear that excessive dissipation will lead to artifacts (e.g., the robot acts as if it is moving through molasses), and we have found it challenging to balance numerical stability and physical fidelity with that approach. The present study started from the observation that simulating robots with few degrees of freedom and controlled via error feedback could admit large integration steps. These large integration steps can yield much faster simulations at the expense of lower numerical accuracy. Accordingly, this paper tests the following hypotheses:

Hypothesis 1: Driving a multi-body system (e.g., a robot interacting through contact with one or more rigid bodies) through inverse dynamics control can yield greater numerical stability than tuned PD/PID control can offer.

Hypothesis 2: Integrating a multi-body system that accounts for both contact and inverse dynamics constraints yields greater numerical stability than feeding the output from an inverse dynamics controller into the simulation's integrator.

Hypothesis 3: Incorporating transmission models for electromagnetic actuators can increase numerical stability for simulations of robotic systems driven by PD/PID control.

II. Related work

Related work spans literature in "stiff" differential equations (§II-A), multi-body systems with large mass ratios (§II-B), "motors" in *Open Dynamics Engine* (§II-C), inverse dynamics computation with simultaneous contact constraints (§II-D), and kinematic simulations (§II-E).

A. "Stiff" systems

We are unaware of an accepted formal definition of a "stiff" dynamical system. An informal characterization of such systems is that their solution is smooth, yet uninformed numerical approaches to them can require extremely small integration steps. Mechanical systems with springs and dampers are the archetype of stiff systems, and error feedback controllers applied to mechanical systems without springs and dampers make them act like stiff systems. As a result, rapid prototyping of systems driven with roboticist's typical control techniques has proven challenging; simulations may run one or more orders of magnitude more slowly than useful in such preliminary testing.

B. High mass ratios

Multi-rigid body systems with high mass ratios between links can be viewed as a type of stiff system [2]. However, Featherstone found that the condition number of the joint space inertia matrix increases quartically with the length of a kinematic chain [6], which points to another possible source of system stiffness. The robotic systems used in the experiments within this present work were modeled from CAD data and do not contain significant disparities in masses, but the inertia

matrix condition numbers (on the order of 10^8) can still be problematic.

C. "Motors" in Open Dynamics Engine

The *Open Dynamics Engine* (*ODE*) manual describes "motors", which implement the technique described in this present work, in the following way.

> ... [A]pplying forces directly [to joints] is often not a good approach and can lead to severe stability problems if it is not done carefully.
>
> Consider the case of applying a force to a body to achieve a desired velocity. To calculate this force F you use information about the current velocity, something like this:
>
> $$F = k(\text{desired speed} - \text{current speed}) \quad (1)$$
>
> This has several problems. First, the parameter k must be tuned by hand. If it is too low the body will take a long time to come up to speed. If it is too high the simulation will become unstable. Second, even if k is chosen well the body will still take a few time steps to come up to speed. Third, if any other "external" forces are being applied to the body, the desired velocity may never even be reached (a more complicated force equation would be needed, which would have extra parameters and its own problems).
>
> Joint motors solve all these problems ... They can effectively see one time step into the future to work out the correct force. This makes joint motors more computationally expensive than computing the forces yourself, but they are much more robust and stable, and far less time consuming to design with. This is especially true with larger rigid body systems.

This text indicates that incorporating inverse dynamics into the constraint equations is convenient for the user, but it does not elaborate upon the difficulty of attaining fast simulation performance even with well-tuned error feedback control. We are unaware of a general writeup of this technique, including incorporation into the differential variational inequality formulation.

D. Inverse dynamics with contact

In past work, we have described algorithms for computing inverse dynamics forces, without limiting motor forces/torques, while simultaneously predicting contact forces [15, 17, 14]. Our work has recently shown that a solution always exists as long as the desired velocities are consistent with the contact constraints [17], a finding that contrasted with claims in [13]. Note that we say "desired velocities" rather than "desired accelerations" due to inconsistencies in the rigid contact model with Coulomb friction [12]; velocity-level, *differential variational inequality* [12] formulations of the dynamics problems with contact are provably able to avoid such inconsistencies.

E. Kinematic simulations

The Institute for Human and Machine Cognition's (IHMC) robotics group has adopted a similar approach to that described in this paper:[1] desired joint and floating base accelerations are double integrated to yield kinematically driven simulations of legged robots for rapid testing. This approach apparently works well for robots interacting with static environments, but has been tested little on robotic manipulation tasks. The differences between IHMC's approach and the inverse dynamics approaches described in this paper follow: (1) inverse dynamics retains floating base underactuation, allowing a legged robot to trip, for example (IHMC's approach would not capture this behavior); (2) inverse dynamics incorporates non-interpenetration and torque constraints, among other constraints, precluding dynamically infeasible motions (to the first-order accuracy provided by the differential variational inequality-based approach); and (3) IHMC's approach does not have to constrain the motion by solving optimization or mathematical programming problems, meaning that it will generally operate far faster.

III. Multi-body dynamics simulation with contact and inverse dynamics

We now describe the formulation of the multi-body dynamics problem with contact and inverse dynamics constraints. Joint limits and bilateral constraints can be incorporated using straightforward extensions and are not discussed here to streamline the presentation. This *mixed linear complementarity problem formulation* [3] is used in *ODE* and other software.

$$\begin{bmatrix} \mathbf{M} & -\mathbf{P}^\mathsf{T} & -\mathbf{N}^\mathsf{T} & -\mathbf{F}^\mathsf{T} & \mathbf{0} \\ \mathbf{P} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{F} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{E} \\ \mathbf{0} & \mathbf{0} & \mu & -\mathbf{E}^\mathsf{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}^+ \\ \boldsymbol{\tau} \\ \boldsymbol{f}_N \\ \boldsymbol{f}_F \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{\kappa} \\ -\dot{\boldsymbol{q}}_{\text{des}} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{w}_{\boldsymbol{\tau}}^+ - \boldsymbol{w}_{\boldsymbol{\tau}}^- \\ \boldsymbol{w}_{\boldsymbol{N}} \\ \boldsymbol{w}_{\boldsymbol{F}} \\ \boldsymbol{w}_{\boldsymbol{\lambda}} \end{bmatrix} \quad (2)$$

$$\boldsymbol{\tau} - \boldsymbol{\tau}^- \geq \mathbf{0}, \boldsymbol{w}_{\boldsymbol{\tau}^+} \geq \mathbf{0}, (\boldsymbol{\tau} - \boldsymbol{\tau}^-)^\mathsf{T} \boldsymbol{w}_{\boldsymbol{\tau}^+} = 0 \quad (3)$$

$$\boldsymbol{\tau}^+ - \boldsymbol{\tau} \geq \mathbf{0}, \boldsymbol{w}_{\boldsymbol{\tau}^-} \geq \mathbf{0}, (\boldsymbol{\tau}^+ - \boldsymbol{\tau})^\mathsf{T} \boldsymbol{w}_{\boldsymbol{\tau}^-} = 0 \quad (4)$$

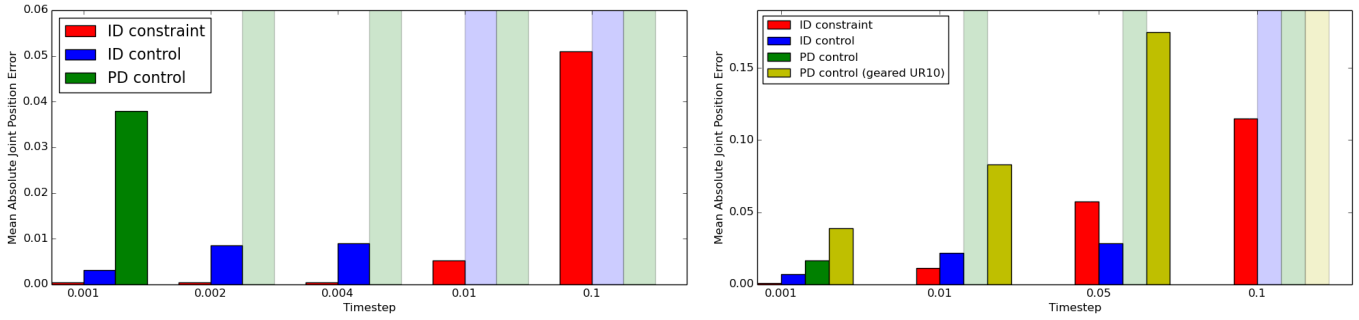
$$\boldsymbol{f}_N \geq \mathbf{0}, \boldsymbol{w}_N \geq \mathbf{0}, \boldsymbol{f}_N{}^\mathsf{T} \boldsymbol{w}_N = 0 \quad (5)$$

$$\boldsymbol{f}_F \geq \mathbf{0}, \boldsymbol{w}_F \geq \mathbf{0}, \boldsymbol{f}_F{}^\mathsf{T} \boldsymbol{w}_F = 0 \quad (6)$$

$$\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{w}_\lambda \geq \mathbf{0}, \lambda^\mathsf{T} \boldsymbol{w}_\lambda = 0 \quad (7)$$

This problem formulation matches that in [17] almost exactly, and the reader is referred to that work for greater insight into inverse dynamics subject to unilateral constraints. Briefly, there are m generalized velocities (r of which are actuated), n points of contact, and k line segments in polygonal approximations to friction cones; problem inputs are $\mathbf{M} \in \mathbb{R}^{m \times m}$ (generalized inertia matrix), $\boldsymbol{v} \in \mathbb{R}^m$ (generalized velocity vector), $\mathbf{P} \in \mathbb{R}^{r \times m}$ is a binary selection matrix (the identity matrix if the controlled system is fully actuated), $\mathbf{N} \in \mathbb{R}^{n \times m}$ (contact normals Jacobian matrix), $\mathbf{F} \in \mathbb{R}^{nk \times m}$ (contact tangents Jacobian matrix), $\mu \in \mathbb{R}^{n \times n}$ (diagonal matrix of friction coefficients), $\mathbf{E} \in \mathbb{R}^{nk \times n}$ (binary matrix described in [1]), $\dot{\boldsymbol{q}}_{\text{des}} \in \mathbb{R}^r$ (desired joint velocities), and $\boldsymbol{\kappa} \equiv \mathbf{M}\boldsymbol{v} + \Delta t \boldsymbol{f}$

[1] Personal communication with Jerry Pratt.



(a) (b)

Fig. 1. The mean absolute joint position error for the ID constraint, ID control, and PD control for the quadruped at various timesteps (a). The mean absolute joint position error for the ID constraint, ID control, PD control and geared PD control for the UR10 at various timesteps (b). A transparent bar indicates instability for the control at the given timestep.

($\boldsymbol{f} \in \mathbb{R}^m$ are all non-contact and non-inverse dynamics forces on the system, Δt).

Key solution variables are $\boldsymbol{v}^+ \in \mathbb{R}^m$ (velocities at $t + \Delta t$, i.e., after integration), $\boldsymbol{\tau} \in \mathbb{R}^r$ (inverse dynamics forces/torques), $\boldsymbol{f}_N \in \mathbb{R}^n$ (contact normal forces), $\boldsymbol{f}_F \in \mathbb{R}^{nk}$ (contact friction forces), $\boldsymbol{\lambda} \in \mathbb{R}^n$ (roughly equivalent to tangent contact velocities at $t + \Delta t$, see [1]).

A. Drawback of the MLCP formulation

The difference between [17] and *ODE*'s approach is that in the latter, $\boldsymbol{\tau}$ is constrained to lie in $[\boldsymbol{\tau}^-, \boldsymbol{\tau}^+]$, which means that the inverse dynamics constraints might not be perfectly satisfied (hence the presence of the term $\boldsymbol{w}_{\boldsymbol{\tau}}^+ - \boldsymbol{w}_{\boldsymbol{\tau}}^-$). These new equations and variables specify that if the inverse dynamics constraint can be satisfied exactly, then $\boldsymbol{w}_{\boldsymbol{\tau}}^+ - \boldsymbol{w}_{\boldsymbol{\tau}}^- = \boldsymbol{0}$. Otherwise, the joint torque acts against the "slack" in the constraint. In other words, if the velocity at $\boldsymbol{v}^+$ for the i^{th} joint is greater than that desired, then the torque applied at that joint must lie at the lower limit; similarly, the torque applied at that joint must lie at the upper limit if the velocity at $\boldsymbol{v}^-$ for the i^{th} joint is lesser than that desired. This problem setup is reasonable in many but not all cases: it is possible that applying a torque at an actuator's lower torque limit could result in greater divergence from the desired velocity at that joint than if no torques were applied (depending on other variable settings). Neither does this problem setup appear to minimize any norm over the difference between desired and resulting velocity.

B. Solvability of the MLCP formulation

[17] showed that this problem can be solved, including determining whether the desired velocities are consistent with the other constraints, for $\boldsymbol{\tau}^- \equiv -\infty, \boldsymbol{\tau}^+ \equiv \infty$ and in expected polynomial time by first converting it to a "pure" linear complementarity problem. For finite torque limits, the mixed linear complementarity problem cannot be converted to a pure LCP, so an algorithm for solving mixed LCPs of this form must be applied. Lemke's Algorithm can be modified to handle lower and upper variable limits (as described in [10]) but is only provably able to solve MLCPs with positive semi-definite matrices. *The result is that the MLCP in Equations 2–7, which results in a copositive matrix [11, 3], is not generally capable of being solved in polynomial time using existing algorithms.* In fact, when the desired velocities are inconsistent with the other constraints, the MLCP is unsolvable even without torque limits. Accordingly, *ODE* uses regularization to solve a "nearby" MLCP. All problem constraints—including non-interpenetration, Coulomb friction, joint limits, and inverse dynamics—will be violated by the degree of regularization. The ramifications of such violations are generally unknown, though one of our experiments in §IV-B describes one outcome.

IV. Experiments

Simulation experiments were conducted using the multi-rigid body dynamics library *Moby*, which uses pivoting solvers in place of the matrix splitting method solvers often employed to speed simulations; these solvers do not provably converge [8], so we avoid them to simplify our experiments, but we remark that they run very fast and often perform acceptably. The robots used in the experiments were the UR10 arm (sourced from an existing open source ROS package) and a floating base quadruped model described in existing work [16]. The UR10 arm possesses eight degrees-of-freedom (DoF), all controllable. The quadruped model possesses 18 DoF, 12 of which are controllable.

The UR10 was directed to follow a sinusoidal motion at each joint, while the quadruped was commanded to follow a sinusoidal pattern that resembled a trot. Each simulation was run for five seconds of virtual time.

When PD controllers were used in the following experiments, gains were tuned by a two-part process consisting of manual tuning followed by nonlinear optimization. The optimization routine minimized the ℓ_2-norm over the sum of all squared joint position errors. The gains were tuned in this way to eliminate human bias to the greatest extent possible; otherwise, the experimenter could subconsciously reduce gains to prioritize simulation stability over tracking accuracy.

Where applicable, our experiments use inverse dynamics-based algorithms that *can* account for torque limits—the ones

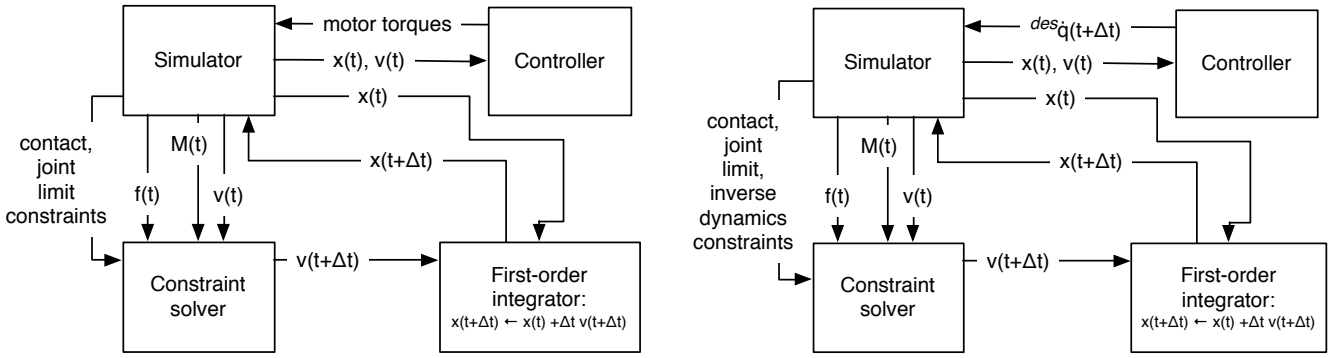


Fig. 2. The difference between inverse dynamics forces fed into the simulator (a) and the simulator computing the next velocity of the multi-rigid body dynamics simulation while accounting for the inverse dynamics constraints (b). $\boldsymbol{x}$, $\boldsymbol{v}$, $\mathbf{M}$, and $\boldsymbol{f}$ are the generalized positions, velocities, inertias, and forces respectively. The architecture on the right is faster, because that on the left solves essentially the same problem twice: once in the controller—the contact forces must be accounted for in the inverse dynamics forces, but the former are then discarded—and then again in the simulator.

described in this paper, not [17]—even though torque limits are set to $\pm\infty$ (i.e., unused). This decision allowed us to study the computational properties of these methods without focusing on whether our virtual robots would possess the actuator forces necessary to complete tasks. Integration steps were limited, albeit arbitrarily, to a maximum of 0.1s. The initial integration step size tested for each model was 0.001, a value that we confirmed produced stable simulations of each model readily. Step sizes were doubled until instability resulted—at which point bisection search was conducted to identify an approximate maximum stable step size—or the maximum value of 0.1s was reached.

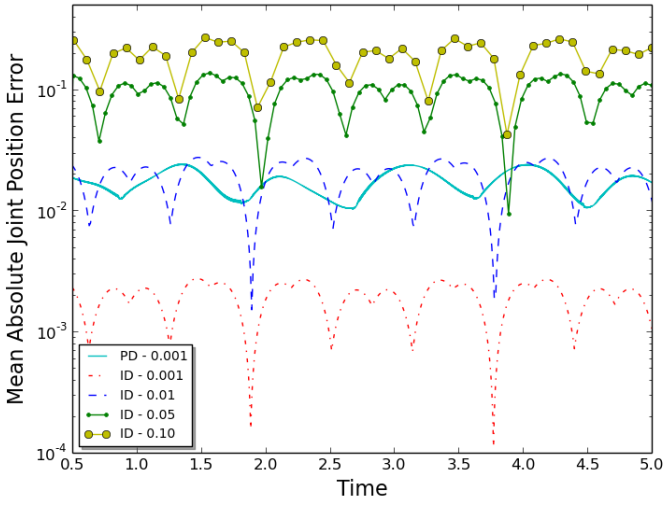


Fig. 3. The mean absolute position error of each joint on the UR10 arm as it executes a sinusoidal motion on each joint at various timesteps. ID constraint was used to achieve the maximum 0.1 timestep.

A. Testing Hypothesis 1: incorporating inverse dynamics control leads to more stable simulations than error feedback control

A PD-controlled UR10 arm served as one experimental control for Hypothesis 1. The maximum step size we attained using this controller without the simulation becoming unstable was 0.001. On the other hand, we found that we could simulate the UR10 arm stably without any controls applied (i.e., the robot falls under the influence of gravity) with a step size of 0.1 (the maximum tested). *This result indicates that the PD control introduces stiffness into the differential equations*; this result is not surprising given that (1) PD control can be viewed as a virtual spring damper and (2) the mass-spring system is the canonical example of a stiff ordinary differential equation [7]. For the experimental variable, we used the standard recursive Newton-Euler Algorithm [5] to generate inverse dynamics torques.

An experimental control in a second experiment used the quadrupedal robot model driven by PD control. We used the non-complementarity-based inverse dynamics control approach described in [17], which was necessary since one or more links of the robot remained in contact with the environment, as our experimental variable. Tables II and I list the results from the experiments with both robots: inverse dynamics control allows the simulations to run 540% and 66% faster, respectively.

In both experiments, the ID control was able to achieve a higher timestep than the PD control. Figure 1 (a) shows that during the quadruped experiment, the ID control was able to achieve a maximum stable timestep of 0.004. The PD control, on the other hand was only able to achieve a maximum timestep of 0.001. Part (b) shows the ID control being able to achieve a timestep of 0.05 when the sinusoidal motion was run on the UR10 arm. PD control was still only able to achieve a maximum step size of 0.001. The large increase of timestep with regards to the ID control from the quadruped experiment to the UR10 experiment is likely due to the fact that the quadruped actually experiences contact in its controlled motion, while the UR10 experiences no contact.

B. Testing Hypothesis 2: incorporating inverse dynamics constraints leads to more stable simulations than using inverse dynamics control

Although it is possible that inverse dynamics torques fed into a simulator yield exactly the desired velocity at the next time step[2], this result is not guaranteed (as the data from the previous section show). Hypothesis 2 arose from our observations about the split nature of the control-simulation process (see Figure 2): we speculated that solving for the next velocity subject to all constraints would yield higher greater simulation stability than feeding the inverse dynamics torques into the simulator's constraint solver (i.e., its mixed or pure linear complementarity problem solver).

We used the inverse dynamics controllers employed as the experimental variables in our tests of Hypothesis 1 as the experimental controls in our tests of Hypothesis 2. For the variables in this experiment, we tested the UR10 arm and the quadrupedal robot using the MLCP-based inverse dynamics formulation described in Section III. We also tested the quadrupedal robot using an experimental, optimization-based constraint solver. We will only provide an outline of the technical approach as we identified a key problem with the approach (to be described below). The solver first computes a feasible point that satisfies both the contact normal velocity constraints ($\mathbf{N}\boldsymbol{v}^+ \geq \mathbf{0}$, from Equation 2) and the inverse dynamics velocity constraints ($\boldsymbol{P}\boldsymbol{v}^+ - \dot{\boldsymbol{q}}_{\text{des}} = \mathbf{0}$). Quadratic programming is then used to attempt to find frictional forces that maximally dissipate kinetic energy without violating these constraints (in the spirit of [4]).

Figure 3 depicts the speedup achieved from the use of constraint-based inverse dynamics. At a timestep of 0.01, the use of inverse dynamics constraints is able to achieve around the same order of accuracy as the PD control at a timestep of 0.001. Incorporating ID constraints into the UR10 simulation process executed in 9.11s at a 0.01 timestep, while the PD controlled arm required 357.02s at a 0.001 timestep. ID constraints achieved a 39x speedup with essentially the same accuracy.

The results for the quadrupedal robot in Table I require explanation. The existing, MLCP-based approach caused the simulation to become unstable at any step size. Regularizing the MLCP (via Tikhonov regularization) did not help: such large regularization—it was necessary to add values on the order of 1.0 to MLCP matrix to attain a solution—that the result was no longer a solution to a "nearby" problem. Note that applying the constraint solver to the individual problems of inverse dynamics without contact (by temporarily de-activating gravitational forces) and contact without inverse dynamics (i.e., just using PD control) works fine; problems only arise when the constraints are considered simultaneously. We used the Dantzig solver [9], which is also used by *ODE*, to solve the mixed linear complementarity problem.

[2]We use desired velocity in place of desired acceleration for reasons described in [17]. Popular open source multi-rigid body dynamics libraries used for robotics (e.g., *ODE*, *Bullet*, *DART*) employ a first-order approximation to velocity, thus supporting our choice.

On the other hand, our experimental approach outlined above was capable of generating an accurate solution—and, as with the UR10 model—the simulation remained perfectly stable for large step sizes. However, our results do not capture an important artifact: the quadruped appeared to be skating as if on ice when it should have been trotting. Examination of the constraint solver indicated that the solution method would have had to slightly violate the inverse dynamics constraints to incorporate frictional forces; our experimental approach is flawed and thus illustrates what can happen when constraints may be violated arbitrarily (refer back to §III-B). Nevertheless, Table I does hint at the possibility of a 226% speedup.

C. Testing Hypothesis 3: incorporating transmission models increases the stability of robots driven by error feedback control

Claude Lacoursière suggested in personal communication that adding gearing to a robot model might reduce the stiffness in the differential equations. Accordingly, we compared the PD controlled UR10 used to test Hypothesis 1 to a PD controlled UR10 with a virtual transmission modeled at each revolute joint; the gains were re-tuned for this modified model. Gearing was not added to our quadrupedal model because significant architectural modifications would be necessary in our robot's locomotion software to accommodate gearing. Table II and Figure 3 illustrate that the gearing does dramatically increase the maximum stable step size, at a clear cost of tracking accuracy.

V. Discussion

We have demonstrated the capability of inverse dynamics to dramatically speed multi-rigid body simulations with contact. Each inverse dynamics constraint (i.e., specification of a joint velocity) increases the size of the mixed LCP (when accounting for force/torque limits) or pure LCP (without force/torque limits) to be solved; in the latter case, a variable is added to a linear system to be solved independently of the LCP, as described in [17]. We note that this is exactly the same procedure as that required to account for a gear constraint: the computational demands are identical. Though the computational demands to solve these problems are larger than that required for error feedback controlled robots, the maximum stable integration step sizes are tens or hundreds of times larger, thereby permitting much faster simulations.

Aside from the wasted computational effort of solving the same problem twice—once for the controller to compute inverse dynamics subject to contact and joint limit constraints and once for the simulation's solver to compute contact and joint limit forces subject to the control forces/torques, see Figure 2—we have shown that incorporating the constraints into the constraint solver is less likely to cause simulation instability. We believe that the explanation for the stability decrease by feeding the inverse dynamics forces/torques into the simulation is due to very slight discrepancies in inputs; for example, the friction pyramids used for the Coulomb friction approximations between the two constraint solvers can use

Control method	Max step	Accuracy (MAE) at max step	Running time
PD control	0.001	3.80×10^{-2}	24.84s
Inverse dynamics (control)	0.004	9.05×10^{-3}	4.97s
Inverse dynamics (constraint, MLCP approach)	—	—	—
Inverse dynamics (constraint, experimental solver)	0.10	5.10×10^{-2}	2.54s

TABLE I
MEAN OF ABSOLUTE ERROR (MAE) JOINT POSITION TRACKING ACCURACY ON THE SIMULATED QUADRUPEDAL ROBOT

Control method	Max step	Accuracy (MAE) at max step	Running time
PD control	0.001	1.01×10^{-2}	357s
PD control (geared robot)	0.05	1.74×10^{-1}	24.6s
Inverse dynamics (control)	0.05	2.85×10^{-2}	6.51s
Inverse dynamics (constraint)	0.10	1.15×10^{-1}	2.85s

TABLE II
MEAN OF ABSOLUTE ERROR (MAE) JOINT POSITION TRACKING ACCURACY ON THE SIMULATED UR10 MANIPULATOR

different principal directions (which are selected arbitrarily). Our prior work [17], which demonstrates high, albeit imperfect tracking accuracy for inverse dynamics of simulated robots, hints at this phenomenon.

It is clear that important work still remains, particularly in finding a computationally tractable model that produces reasonably accurate contact forces and satisfies inverse dynamics constraints as well as force/torque limits allow. We believe the complementarity-free contact model we described in [4] will provide such a foundation, but further research is necessary. In the meantime, adding gearing to robots with electromagnetic actuators can provide the requisite simulation stability necessary for high frequency (realtime and above) simulation, albeit with far lower tracking accuracy.

ACKNOWLEDGEMENTS

This work was supported by ARO grant W911NF-16-1-0118 and by a GWU SEAS SUPER fellowship to John Shepherd.

REFERENCES

[1] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997.

[2] M. Anitescu and F. A. Potra. A time-stepping method for stiff multi-rigid-body dynamics with contact and friction. *Intl. Journal for Numerical Methods in Engineering*, 55:753–784, 2002.

[3] R. W. Cottle, J.-S. Pang, and R. Stone. *The Linear Complementarity Problem*. Academic Press, Boston, 1992.

[4] E. Drumwright and D. A. Shell. Modeling contact friction and joint friction in dynamic robotic simulation using the principle of maximum dissipation. In *Proc. of Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2010.

[5] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer, 1987.

[6] R. Featherstone. An empirical study of the joint space inertia matrix. *The Intl. J. of Robotics Research*, 23(9):859–871, September 2004.

[7] E. Hairer and G. Wanner. *Solving ordinary differential equations II: stiff and differential-algebraic problems, 2nd. ed*. Springer Verlag, Berlin, 1996.

[8] C. Lacoursière. Splitting methods for dry frictional contact problems in rigid multibody systems: Preliminary performance results. In M. Ollila, editor, *Proc. of SIGRAD*, pages 11–16, Nov 2003.

[9] C. Lacoursière. *Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with Dry Frictional Contacts*. PhD thesis, Umeå University, 2007.

[10] R. W. H. Sargent. An efficient implementation of the Lemke Algorithm and its extension to deal with upper and lower bounds. *Mathematical Programming Study*, 7:36–54, 1978.

[11] D. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, San Francisco, CA, April 2000.

[12] D. E. Stewart. Rigid-body dynamics with friction and impact. *SIAM Review*, 42(1):3–39, Mar 2000.

[13] E. Todorov. Analytically-invertible dynamics with contacts and constraints: theory and implementation in MuJoCo. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*, 2014.

[14] S. Zapolsky and E. Drumwright. Quadratic programming-based inverse dynamics control for legged robots with sticking and slipping frictional contacts. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*, 2014.

[15] S. Zapolsky, E. Drumwright, I. Havoutis, J. Buchli, and C. Semini. Inverse dynamics for a quadruped robot locomoting on slippery surfaces. In *Proc. Intl. Conf. Climbing Walking Robots (CLAWAR)*, Sydney, Australia, 2013.

[16] S. Zapolsky and E. M. Drumwright. Adaptive integration for controlling speed vs. accuracy in multi-rigid body simulation. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots & Systems (IROS)*, 2015.

[17] S. Zapolsky and E. M. Drumwright. Inverse dynamics with rigid contact and friction. *Auton. Robots*, (under review).