Extending Open Dynamics Engine for the DARPA Virtual Robotics Challenge

John M. Hsu and Steven C. Peters

Open Source Robotics Foundation, 419 N. Shoreline Blvd, Mountain View, CA 94041, USA
{hsu,scpeters}@osrfoundation.org
http://osrfoundation.org

Abstract. The DARPA Virtual Robotics Challenge (VRC)[1] was a cloud-based robotic simulation competition. Teams competed by writing control software for a humanoid robot to perform disaster response tasks in real-time simulation. Simulating the physics and sensors of a humanoid robot in real-time presented challenges related to the trade-off between simulation accuracy and computational time. The Projected Gauss-Seidel (PGS) iterative solver was chosen for its performance and robustness, but it lacks the accuracy and the fidelity required for reliable simulation of task-level behaviors. This paper presents the modeling decisions and algorithmic improvements made to the Open Dynamics Engine (ODE) physics solver that improved PGS accuracy and fidelity without sacrificing its real-time simulation performance in the VRC. These improvements allowed for stable simulation regardless of user input during the VRC, and supported reliable contact dynamics during VRC tasks without violating the near real-time requirement.

1 Introduction

The DARPA Robotics Challenge (DRC) is a competition with the goal of improving robotic systems for use in disaster response. With sufficient mobility and dexterity, robots may assist technicians and emergency responders in dangerous environments. The DRC will test the ability of robots to complete tasks relevant to disaster response, such as walking on uneven terrain, driving a utility vehicle, opening doors, climbing industrial ladders, and threading a fire hose into a standpipe.

In addition to physical testing in the DRC Trials and DRC Finals, the DRC also included the Virtual Robotics Challenge (VRC), in which teams wrote control software for a simulated humanoid robot. The Open Source Robotics Foundation (OSRF) provided an open-source cloud-based simulator for the disaster response tasks using a model of the Atlas robot from Boston Dynamics [2] and the dexterous Sandia hand [3].

When selecting the simulation tasks for the VRC, it was necessary to consider the difficulty of accurately simulating tasks using Newton-Euler equations of motion with a Coulomb friction approximation. The following high-level tasks were chosen: walk across various terrains, drive a utility vehicle, and thread a fire hose into a standpipe. These tasks require simulation of dynamic balancing, walking on uneven terrain, ingress/egress of a utility vehicle, manipulation of vehicle controls (pedals, gear shift, steering), vehicle dynamics, and object manipulation with a dexterous hand.

D. Brugali et al. (Eds.): SIMPAR 2014, LNAI 8810, pp. 37–48, 2014.

2 Literature Review

There are numerous approaches to modeling rigid body dynamics with frictional contacts. Explicit penalty methods apply restorative forces to ”penalize” collisions between rigid bodies as contacts arise. For example, the Hertz model [4], is an idealized model of material deformation and contact forces for spherical surfaces that is often applied as an explicit penalty force. Constraint-based methods formulated as a Linear Complementarity Problem (LCP) attempt to resolve all contact (and constraint) forces simultaneously when collisions are detected. Examples of constraint-based methods using velocity-impulse formulated as an LCP include [5,6,7,8]. These methods can have performance advantages when compared with pure penalty based methods due to reduced numerical stiffness. This is mostly due to the fact that penalty methods can require very small time steps for stability while simulating dynamic walking and grasping.

The strategies for solving the constraint based LCP problem mainly fall into two categories: iterative and direct (pivoting) methods. This paper presents improvements to the iterative Projected Gauss-Seidel solver that exists within the Open Dynamics Engine (ODE). Note that ODE also includes a direct method that is based on Lemke’s algorithm [9], which is an extension of Dantzig’s algorithm [10].

An important aspect to consider when formulating the equations for articulated rigid body dynamics is the internal state representation. In maximal coordinate formulations, such as ODE, each rigid body has six degrees of freedom. Articulation is encoded as equality constraints on the dynamic states. This approach yields convenient sparse matrix structures, such as a block diagonal mass matrix and a sparse constraint Jacobian. Additionally, the approach treats contact and articulation constraints in a uniform manner. In contrast, formulations based on internal (or generalized) coordinates (often using Featherstone’s algorithm [11,12]) consider articulation implicitly by adding the system degrees of freedom with each articulation joint. This yields more accurate kinematics and a smaller mass matrix, though the mass matrix structure is no longer sparse. There was some discussion prior to the VRC regarding the relative merit of maximal and generalized coordinates [13], though a robust comparison is not presented here.

3 Modeling and Fidelity Considerations

This section presents the modeling decisions that were made to improve real-time performance while maintaining sufficient physical accuracy.

The Atlas robot is a humanoid robot manufactured by Boston Dynamics Inc. (BDI), with 28 hydraulically actuated degrees of freedom [2]. The kinematics, rigid body inertias, 3D mesh collision shapes, maximum joint angles, velocities, and torques were provided by BDI. In the absence of details about the hydraulic systems, each joint was modeled as a pin joint with torque control subject to position, velocity, and torque limits. The torque and speed limits were not coupled with a torque-speed curve. Static joint friction was not modeled, though viscous damping (proportional to joint velocity) was applied at the joints in a heuristic manner to improve solver stability. Contact friction was modeled as Coulomb friction using a friction pyramid (see Section 4.6).

Although a full set of 3D concave meshes was provided for the Atlas robot, each collision shape was approximated by a union of convex sphere, box, and cylinder collision primitives. It was found that the collision primitives exhibited faster collision detection and more robust contact resolution than the 3D meshes [14].

Some physical interactions were approximated due to insufficient fidelity at the required level of real-time performance. Threading a fire hose, for example, involves contact between millimeter-scale features. Instead of modeling the fine contact geometry, a screw joint was dynamically created when the fire hose coupling was sufficiently aligned with the standpipe. The coupling could be rotated in one direction to connect or in the opposite direction to release the coupling. Sitting in the seat of a utility vehicle was another challenge, as the seat was modeled as a rigid body. This caused difficulty in finding a stable seating position. To remedy this, a viscous damping field was created on the surface of the seat. This partially mitigated the problem, though stable sitting in the vehicle proved a continual challenge in the VRC [15].

4 Open Dynamics Engine

This section presents the algorithms used by Open Dynamics Engine (ODE) [16]. ODE represents rigid body states with maximal coordinates, in which each rigid body has six degrees of freedom, and articulation and contact constraints are enforced by adding constraint equations. Please see ODE's User Manual [17] for general documentation.

4.1 Unconstrained Rigid Body Dynamics

The notation for maximal coordinates is borrowed from [18], and it is assumed that all vectors are expressed in the world frame unless otherwise specified. Each rigid body has an associated coordinate frame with center of gravity (c.g.) position $\bar{x}$ and orientation quaternion $\bar{q}$. The coordinate frames evolve in time according to

$$\bar{x}_t = \dot{\bar{x}}, \bar{q}_t = (1/2)\bar{\omega}\bar{q} \tag{1}$$

with $\dot{\bar{x}}$ and $\bar{\omega}$ representing linear and angular velocity. The velocities evolve according to the Newton-Euler equations of motion, which are expressed as

$$m\dot{\bar{x}}_t = \bar{f}, \bar{L}_t = \bar{\tau}$$

where m is the mass; $\bar{f}$ and $\bar{\tau}$ are the net force and torque; $\bar{L} = \bar{\bar{I}}\bar{\omega}$ is the angular momentum with inertia tensor $\bar{\bar{I}} = \bar{\bar{R}}\bar{\bar{D}}\bar{\bar{R}}^T$, rotation matrix $\bar{\bar{R}}(\bar{q})$, and body-frame inertia tensor $\bar{\bar{D}}$. For a single unconstrained rigid body, this can be re-written as:

$$\begin{bmatrix} m\bar{\bar{\delta}} & \bar{\bar{0}} \\ \bar{\bar{0}} & \bar{\bar{I}} \end{bmatrix} \begin{bmatrix} \ddot{\bar{x}} \\ \dot{\bar{\omega}} \end{bmatrix} = \begin{bmatrix} \bar{f} \\ \bar{\tau} - \bar{\omega} \times \bar{\bar{I}}\bar{\omega} \end{bmatrix} \tag{2}$$

where $\bar{\bar{\delta}}$ is the identity matrix.

For a system with multiple rigid bodies, augmented variables are defined for each body b: velocity vector $\bar{v}_b = [\dot{\bar{x}}_b^T, \bar{\omega}_b^T]^T$; block diagonal mass matrix $\bar{\bar{m}}_b = diag(m_b\bar{\bar{\delta}}, \bar{\bar{I}}_b)$;

and effort vector $\bar{e}_b = [\bar{f}_b^T, (\bar{\tau}_b - \bar{\omega}_b \times \bar{\bar{I}}_b \bar{\omega})^T]^T$. The dynamics in equation 2 can then be expressed as:

$$\bar{\bar{m}}_b \dot{\bar{v}}_b = \bar{e}_b \tag{3}$$

For a system of N rigid bodies, system variables are defined as the velocity states $\bar{v} = [\bar{v}_1^T, \bar{v}_2^T, ...\bar{v}_N^T]^T$; the block diagonal system mass matrix $\bar{\bar{M}} = diag(\bar{\bar{m}}_1, \bar{\bar{m}}_2, ..., \bar{\bar{m}}_N)$; and the system effort vector $\bar{E} = [\bar{e}_1^T, \bar{e}_2^T, ...\bar{e}_N^T]^T$. The unconstrained system dynamics from 3 are then given as:

$$\bar{\bar{M}} \dot{\bar{v}} = \bar{E} \tag{4}$$

4.2 Articulation and Contact Constraints

Articulation and contact are encoded through a set of nonlinear constraints on rigid body position and orientation, with articulation using equality constraints $h_e(\bar{x}, \bar{q}) = 0$, and contact using inequality constraints $h_i(\bar{x}, \bar{q}) \geq 0$.

To avoid nonlinearities, the position constraints are differentiated to yield linear velocity constraints of the form

$$\bar{\bar{J}} \bar{v} = \bar{c} \tag{5}$$

where $\bar{\bar{J}}$ is the constraint Jacobian matrix and $c_e = 0, c_i >= 0$, for articulation equality constraints and contact inequality constraints, respectively. The velocity constraints are adjoined to the equations of motion using a vector of Lagrange multipliers $\bar{\lambda}$ as:

$$\bar{\bar{M}} \dot{\bar{v}} = \bar{E} + \bar{\bar{J}}^T \bar{\lambda} \tag{6}$$

4.3 Discretization

The dynamics in equation 6 are discretized over a time interval Δt using a first-order Euler method as $\dot{\bar{v}} \Delta t = \bar{v}^{n+1} - \bar{v}^n$, and rearranged to yield the difference equation for constrained rigid body dynamics in matrix form as

$$\begin{bmatrix} (1/\Delta t)\bar{\bar{M}} & -\bar{\bar{J}}^T \\ \bar{\bar{J}} & 0 \end{bmatrix} \begin{bmatrix} \bar{v}^{n+1} \\ \bar{\lambda} \end{bmatrix} = \begin{bmatrix} (1/\Delta t)\bar{\bar{M}} \bar{v}^n + \bar{E} \\ \bar{c} \end{bmatrix} \tag{7}$$

Assuming that the constraints are satisfied implicitly at the next time step $(t + \Delta t)$, $\bar{\bar{J}} \bar{v}^{n+1} = \bar{c}$, the Lagrange multipliers $\bar{\lambda}$ are computed by left multiplying 7 by $\bar{\bar{J}} \bar{\bar{M}}^{-1}$ as

$$[\bar{\bar{J}} \bar{\bar{M}}^{-1} \bar{\bar{J}}^T] \bar{\lambda} = \frac{\bar{c}}{\Delta t} - \bar{\bar{J}} [\frac{\bar{v}^n}{\Delta t} + \bar{\bar{M}}^{-1} \bar{E}] \tag{8}$$

Solving 8 yields the necessary constraint forces $\bar{\lambda}$ for forward dynamics. This equation is solved using an iterative Projected Gauss Seidel algorithm, omitted here for brevity.

Given $\bar{\lambda}$, the rigid body velocities $\bar{v}^{n+1}$ are computed from equation 7. The positions x^{n+1} and orientations q^{n+1} are computed by integrating equation 1 using the velocity value v^{n+1} to give semi-implicit stability.

Note that the Lagrange multipliers for inequality constraints are initially computed by solving 8 but are afterwards projected into their proper domains.

4.4 Constraint Error Correction

Rigid body dynamics solvers with fixed time stepping schemes will encounter instances where two rigid bodies intersect. Constraint violation can also be caused by position drift from numerical errors during integration and unconverged iterative solvers.

One approach is to backup simulation and take smaller time steps until a non-penetrating contact has been made. Physics engines such as Simbody [19] uses this variable time stepping approach.

On the other hand, ODE adds a position constraint correction term. The position constraint error is evaluated for each constraint and expressed at timestep n as $\bar{h}^n$. It is added to the velocity constraint equation with coefficient β (also known as error reduction parameter **ERP** inside ODE).

$$\bar{\bar{J}}\bar{v} + \frac{\beta}{\Delta t}\bar{h}^n = \bar{c} \tag{9}$$

This term can be considered a form of Baumgarte stabilization [20]. It is used to restore constraint error to zero when $\beta = 1$ with a first-order Euler integrator, though values less than 1 are used in practice.

The constraint error correction term can be added into 8 as

$$[\bar{\bar{J}}\bar{\bar{M}}^{-1}\bar{\bar{J}}^T]\bar{\lambda} = \frac{\bar{c}}{\Delta t} - \frac{\beta}{\Delta t^2}\bar{h}^n - \bar{\bar{J}}[\frac{\bar{v}^n}{\Delta t} + \bar{\bar{M}}^{-1}\bar{E}] \tag{10}$$

4.5 Constraint Force Mixing and Spring-Damper

An interesting concept called Constraint Force Mixing (**CFM**) was applied by Smith in ODE to stabilize the pivoting Lemke's solver. It was also implemented in the standard PGS algorithm in ODE. The approach adds a term $(1/\Delta t)\bar{\bar{C}}\bar{\lambda}$ to equation 10, where $\bar{\bar{C}}$ is a diagonal positive semidefinite matrix composed of *CFM* parameters. With the right-hand side of 10 abbreviated as *rhs*, the equation is rewritten as follows:

$$\left[\bar{\bar{J}}\bar{\bar{M}}^{-1}\bar{\bar{J}}^T + \tfrac{1}{\Delta t}\bar{\bar{C}}\right]\bar{\lambda} = rhs \tag{11}$$

An extremely useful application of the **CFM** and **ERP** parameters is that they map a constraint directly to an equivalent spring and damper system with stiffness k_p and viscous dissipation k_d properties:

$$ERP = \frac{k_p\Delta t}{k_p\Delta t + k_d} CFM = \frac{1}{kp\Delta t + k_d} \tag{12}$$

See Catto [21] for a derivation of equivalence between these parameters.

Effectively, any spring damper system can be implemented using **CFM** and **ERP**. Most importantly, the spring damper system solution is obtained implicitly as part of the overall LCP system, without the numerical stiffness problems experienced by explicit spring dampers.

4.6 Coulomb's Friction Approximation Constraints

Given a frictional constraint with contact normal $\bar{f}_{cn}$ and corresponding frictional force $\bar{f}_{\mu}$ along the direction satisfying the maximum dissipation principle [22], the governing equations can be posed as velocity constraints:

$$\bar{J}_{fric}\bar{v} = \bar{c}_{fric} \tag{13}$$

The corresponding $\bar{\lambda}_{\mu}$ is projected into a corresponding solution space based on Coulomb's law: $\|\bar{f}_{\mu}\| \leq \mu \bar{f}_{cn}$. Solving equation 13 yields a solution for frictional contact based on Coulomb's friction cone if the direction of maximum dissipation $\bar{j}_{\mu}$ is determined.

To avoid computing the direction of maximum dissipation, the frictional constraints can be split into two or more spanning vectors on the contact surface manifold [22]. This approximates the friction cone as a pyramid. The corresponding constraint equations become:

$$\bar{J}_{fric}\bar{v} = \begin{bmatrix} \cdots & \bar{j}_{cn} & \cdots & \bar{j}_{\mu_1} & \cdots \\ \cdots & \bar{j}_{cn} & \cdots & \cdots & \bar{j}_{\mu_2} \end{bmatrix} \begin{bmatrix} \cdots & \bar{v}_{cn}^T & \cdots & \bar{v}_{\mu_1}^T & \bar{v}_{\mu_2}^T \end{bmatrix}^T = \bar{c}_{fric} \tag{14}$$

with corresponding unknowns $\bar{\lambda}_{\mu_1}$ and $\bar{\lambda}_{\mu_2}$, to be projected into their corresponding solution spaces at each iteration based on

$$\|\bar{f}_{\mu_1}\| \leq \mu_1 \bar{f}_{cn}, \;\; \|\bar{f}_{\mu_2}\| \leq \mu_2 \bar{f}_{cn}. \tag{15}$$

This approach yields anisotropic friction, but is much faster than the friction cone.

5 Modifications to Projected Gauss-Seidel Solver within Open Dynamics Engine

ODE is a robust solver that provides an excellent starting point for a fast dynamics simulator with tunable accuracy. There were some inadequacies, however, that needed to be addressed before using it in the VRC. This section will discuss some of the work done to improve the dynamics solver in ODE.

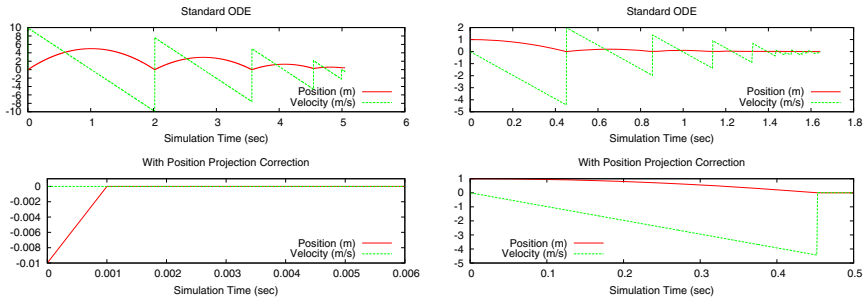
5.1 Contact Constraint Correction with Position Projection (Split Impulse)

The problem with the existing correction method in ODE (described in section 4.4) is that the **ERP** term adds non-physical energy to the system every time an interpenetration occurs. Alternatively, if **ERP** is zero, the solution will not correct for interpenetration and the rigid bodies in contact may *drift* into deeper constraint violations. A method similar to the Split Impulse method was introduced in ODE to cope with position errors caused by fixed time step interpenetration, unconverged iterative PGS residual and numerical integration errors.

For this method, the LCP equations 7 are solved twice, with β to yield $\bar{v}_{\beta}^{n+1}$ and without β to yield $\bar{v}^{n+1}$. Note the two equations can be solved in parallel. The velocity vector without β ($\bar{v}^{n+1}$) is used as the next velocity vector, while the velocity vector with β ($\bar{v}_{\beta}^{n+1}$) is integrated in 1 to yield the next position $\bar{x}^{n+1}$ and orientation $\bar{q}^{n+1}$.

The proposed position projection correction approach effectively *teleports* the overlapping objects away from each other without introducing excess energy into the system. At the velocity level, contact constraints are seen as non-penetrating inelastic impacts if PGS converges fully.

For example, a box on the ground plane with initial collision interpenetration of 1 centimeter will correct its position until resting contact is achieved without gaining energy if position projection is turned on. Figure 1a shows box position trajectory and velocity with and without position projection correction.



(a) Initial box-ground plane overlap of 1 meter

(b) Box impacting ground plane

Fig. 1. Trajectory of box model interactions with ground plane with and without position correction (**ERP** = 1)

In addition to more stable interpenetration correction, position projection correction is ideal for modeling completely inelastic impacts as demonstrated in figure 1b.

5.2 Convergence Acceleration by Static and Dynamics Invariant Inertia Ratio Reduction

The large-mass-ratio problem [23] is a well known issue for iterative LCP solvers. As the solution is updated via row-sweep, the impulse flux propagating across constraints is effectively throttled by the smallest Eigenvalue of the global constraint matrix $\bar{\bar{J}}\bar{\bar{M}}^{-1}\bar{\bar{J}}^{T}$.

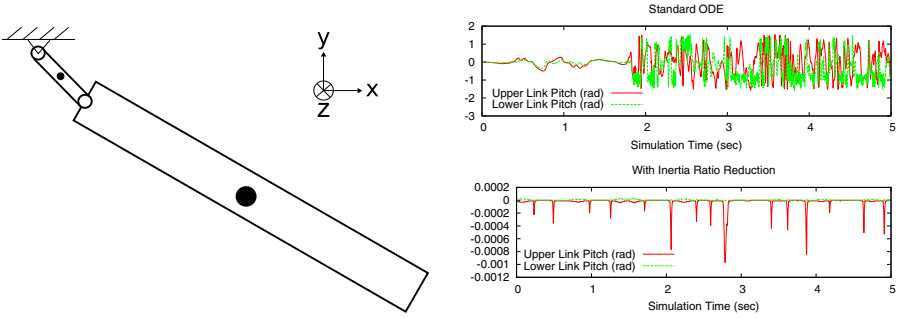
The Atlas and Sandia hand models contain large inertia ratios across some of the constraints. The Atlas model has an inertia ratio of ~ 9017 across the ankle joint causing PGS to converge slowly. With a limited number of PGS inner iterations, this causes joint constraint violations, noise in the dynamics solution and an inability to dynamically control articulated bodies.

During the VRC, issues with PGS convergence were raised (see discussion [24]), and various constraint stabilization methods (e.g. [25]) were suggested. While constraint stabilization is a promising future research direction, an intermediate solution was found which stabilizes the solution for the Atlas model by inertia ratio reduction.

The method for modifying the Atlas model inertias to stabilize simulation is summarized below, with detailed documentation available in Bitbucket pull request [26].

Given two bodies with bilateral constraint Jacobian $\bar{J}_b$, the bodies' moments of inertia in constrained directions can be re-distributed to reduce the inertia ratio. The moment of inertia of a body along an arbitrary unit line vector $\bar{S}$ can be computed by $\bar{\bar{I}}^L = \bar{S}^T \bar{\bar{I}} \bar{S} \cdot \bar{S}\bar{S}^T$. Redistributing the moments of inertia components about the line vector for the two connected bodies is done by modifying the original moment of inertias for both bodies $\bar{\bar{I}}_i^{new} = \bar{\bar{I}}_i + \alpha(\bar{\bar{I}}_j^L - \bar{\bar{I}}_i^L)$, where $\alpha \in [0,1]$ controls the distribution ratio. This inertia redistribution is recomputed on every simulation update step.

To illustrate the effect of inertia ratio reduction, a double pendulum with an large inertia ratio is considered (shown in Fig. 2a). The pendulum links are modeled as uniform boxes of equal density and different size. This size difference leads to an inertia ratio over 6000. Gravity is applied along the y-axis $g_y = -9.81$ and also along the z-axis $g_z = 1$ (into the page in Fig. 2a). The rotations of each rigid body about the x-axis should be zero, but the non-zero g_z component causes constraint errors. With a time step of 1 ms and 50 iterations, the standard algorithm goes unstable with full revolutions about the x-axis (labeled as pitch in top of Fig. 2b). With inertia ratio reduction enabled, the constraint error is held to within 0.001 rad (bottom of Fig. 2b).



(a) Schematics for Inertia Ratio Reduction pendulum example

(b) X-axis orientation error

Fig. 2. Constraint error in rotations about the x-axis for a double pendulum with high inertia ratio, before and after inertia ratio reduction

Since reducing the inertia ratio modifies the inertia of rigid bodies, it may change the behavior of bodies with large angular velocity due to gyroscopic effects. It was not observed to be a problem during the VRC.

5.3 Implicit Joint Spring Dampers

In lieu of simulating friction, viscous joint damping is extremely useful in stabilizing overall dynamics of the simulation. For the VRC, implicit joint damping was applied by adding a constraint to the joint degree of freedom, and adjusting the **ERP** and **CFM** values to set its spring stiffness k_p to zero and its viscous damping coefficient k_d to an estimated value[1]. Equations 12 and 12 were used to construct a joint damping constraint, and enforce joint damping implicitly through the constraint.

[1] A back of the envelope estimate of realistic joint viscous damping coefficients.

Another requirement for making VRC manipulation tasks solvable was indefinite grasp-hold of an object by the Sandia hand in simulation. Contact chatter and vibration have been a major obstacle in stable and robust contact dynamics in simulated grasping. Many high fidelity approaches exist [27,28], but the VRC required an approach compatible with real-time speeds. In one example demonstrated in figure 4a, the Atlas robot with the Sandia hand is seen holding an object of significantly larger uniform density mass (5.39 kg) than the individual finger links (~ 0.01 kg and principle moments of inertia on the order of $\sim 1e-6\,\mathrm{kg}\cdot\mathrm{m}^2$) in contact with the object. The robot eventually loses grasp of the object because: the object drifts due to high frequency chatter of the finger links and insufficient contact force is transmitted from the arm in motion to the grasped object through the finger links due to lack of PGS convergence. With sufficient joint viscous damping, however, high frequency chatter is reduced while additional contact forces are transmitted from the finger joints to the finger-object contact constraints to make the grasp stable. In the limit where viscous damping coefficient approaches infinity, the finger joint becomes effectively a fixed joint.

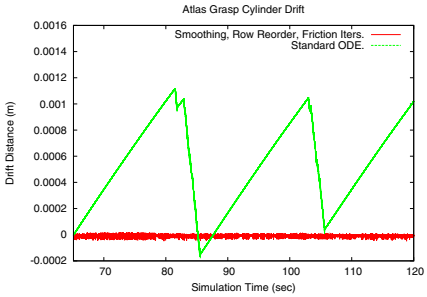
This approach helps overall simulation stability, but errors due to PGS non-convergence are still visible in finger contact simulation due to the fact that PGS has not fully converged.

5.4 PGS Row Ordering and Residual Smoothing

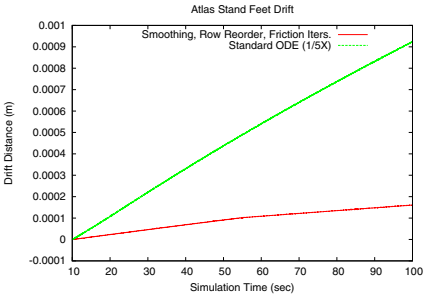
Contact friction drift was a major challenge for the Atlas robot while standing with a dynamically controlled balancing behavior. Approaches to achieving drift-free dynamic standing and grasping using ODE with $\lesssim 50$ inner iterations are being investigated. This would allow real-time simulation of Atlas with Sandia hands on a typical Intel i7 CPU architecture. One possible solution for improving ODE accuracy was to increase the number of inner iterations to above ~ 250, which severely degraded simulation performance to $\lesssim 0.4\times$ real-time. An alternative is to optimize the order of the constraints in the Jacobian matrix as described in equation 10. The constraint rows were arranged in the following order: bilateral, contact normal, and contact friction. At the very end, the friction contraints received an additional 10 iterations. This appeared to speed up convergence and reduce contact chatter.

In addition to solving the frictional directions last in PGS, high frequency oscillations in the frictional contact solutions are reduced by smoothing the contact normal and frictional (non-bilateral) constraints via an exponential smoothing filter $\lambda_{k+1}^{n+1} = (1-\varepsilon)\lambda_k^{n+1} + \varepsilon\lambda^n$, where k denotes the PGS inner iteration number within a time step and ε was hard coded to 0.01 for the VRC.

Figure 3a is an example of Atlas grasping a cylinder indefinitely after row reordering and residual smoothing. In this example, all contact normals are perpendicular to gravity, so friction forces are keeping the cylindrical object from falling out of the grasp. Figure 3b shows the frictional drift of the Atlas feet with and without row reordering and residual smoothing.



(a) Position of grasped cylinder in Sandia hand



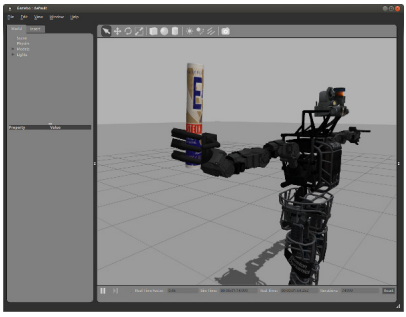
(b) Atlas robot's left foot CG location while standing

Fig. 3. Absolute position drift with and without row reorder and residual smoothing

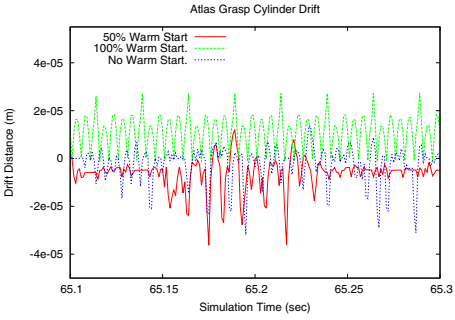
5.5 Warm Starting PGS

Warm starting Gauss-Seidel iterations can help accelerate PGS convergence by starting inner iterations for each time step with the solution from the previous time step (i.e. use $\bar{\lambda}^n$ as the initial guess for $\bar{\lambda}^{n+1}$ in equation 10). Even for a very simple system without contact constraints, however, warm starting PGS must be applied with care as shown in the following example.

Potential pitfalls of warm starting PGS can be demonstrated with a pendulum attached to the inertial frame via a revolute joint constraint. Figure 4a) depicts Atlas robot holding an uniform density 5.39 kilogram cylindrical shaped object. As demonstrated in figure 4b, $\sim 50\%$ warm starting, i.e. $\bar{\lambda}^{n+1} = \beta\bar{\lambda}^n$ where $\beta = 0.5$, does show some reduction in solution chatter over both 0% and 100% warm starts. Note that from our experience, 100% warm start with the solution from the previous time step can result in non-convergence.



(a) Atlas with Sandia hand holding a 5.39 kg cylindrical object



(b) Effect of warm start values on cylinder vertical position drift

Fig. 4. Effect of warm start values on contact chatter and slip during grasping

6 Conclusions

This paper presented the solver algorithm used by the Open Dynamics Engine as well as numerous modifications made during the VRC to stabilize the constrained rigid body dynamics solution and accelerate convergence of the iterative solver.

Without sacrificing real-time dynamics performance, we were able to improve PGS solution for stable contact during dynamic standing and grasping behaviors, maintain overall system stability and prevent divergence in physics solutions. These achievements made the VRC possible from a physics perspective.

A benchmark of the Atlas dynamic walking behavior is shown in figure 5, the resulting simulator performance and constraint violation errors are compared against the number of PGS inner iterations performed at every time step. Figure 5 indicates that at 50 PGS iterations, we were near an optimal trade-off point between constraint error and simulation speed.

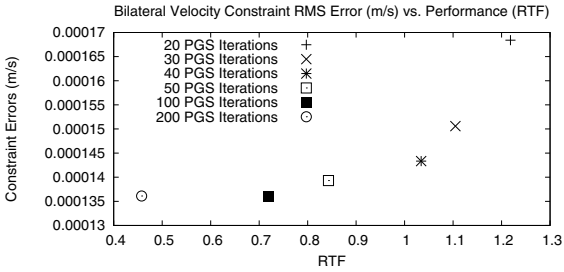


Fig. 5. Plot of Gazebo real-time factor (RTF) vs. bilateral constraint error with varying iterations for Atlas robot with hands performing dynamic walk

Acknowledgment. This work is not possible without the collaborative contributions from the DARPA support team, Boston Dynamics team and the Open Source Robotics Foundation (OSRF) team. This work was supported in part by DARPA contract HR0011-12-C-0111.

References

1. Defense Advanced Research Project Agency (DARPA), DARPA Robotics Challenge (DRC) and Virtual Robotics Challenge (VRC), `http://theroboticschallenge.org/about`
2. Boston Dynamics, Boston Dynamics Atlas Robot, `http://www.bostondynamics.com/robot_Atlas.html`
3. Open Source Robotics Foundation, Sandia Hand Project Webpage, `https://bitbucket.org/osrf/sandia-hand`
4. Stronge, W.: Rigid body collisions with friction. Proceedings of the Royal Society of 431(1881), 169–181 (1990)

5. Brogliato, B., ten Dam, A., Paoli, L., Genot, F., Abadie, M.: Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. Applied Mechanics Reviews 55(2), 107 (2002)
6. Anitescu, M., Potra, F.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. Nonlinear Dynamics 14, 231–247 (1997)
7. Trinkle, D., Stewart, D.E., Trinkle, J.C.: An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. International Journal for Numerical Methods in Engineering 39(15), 2673–2691 (1996)
8. Pang, J., Trinkle, J.: Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction. Mathematical Programming 73(2), 199–226 (1995)
9. Cottle, R., Pang, J.-S., Stone, R.E.: The Linear Complementarity Problem (2009)
10. Cottle, R., Dantzig, G.: Complementary pivot theory of mathematical programming. Linear Algebra and its Applications, 103–125 (1968)
11. Featherstone, R.: A short course on Spatial Vector Algebra the easy way to do rigid body dynamics (2005)
12. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer US, Boston (2008)
13. Smith, J., Gerkey, B.: Drcsim issue #378, `https://bitbucket.org/osrf/drcsim/issue/378`
14. Bridson, R., Fedkiw, R., Anderson, J.: Robust Treatment of Collisions, Contact and Friction for Cloth Animation (1994)
15. Koolen, T., Hsu, J., et al.: Drcsim issue #320, `https://bitbucket.org/osrf/drcsim/issue/320`
16. Smith, R.: Open Dynamics Engine ODE. Multibody Dynamics Simulation Software, `http://www.ode.org`
17. Smith, R.: Open Dynamics Engine ODE. User's Manual, `http://opende.sourceforge.net/wiki/index.php/Manual`
18. Weinstein, R., Teran, J., Fedkiw, R.: Dynamic simulation of articulated rigid bodies with contact and collision. IEEE Trans. Visualization and Computer Graphics 12(3), 365–374 (2006)
19. Sherman, M.A., Seth, A., Delp, S.L.: Simbody: multibody dynamics for biomedical research. In: Procedia IUTAM, vol. 2, pp. 241–261 (January 2011)
20. Baumgarte, J.: Stabilization of constraints and integrals of motion in dynamical systems. Computer Methods in Applied Mechanics and Engineering 1(1), 1–16 (1972)
21. Catto, E.: Soft Constraints reinventing the spring. In: Game Developer Conference (2011)
22. Stewart, D.E.: Rigid-Body Dynamics with Friction and Impact. SIAM Review 42(1), 3 (2000)
23. Silcowitz, M., Niebe, S., Erleben, K.: Nonsmooth Newton Method for Fischer Function Reformulation of Contact Force Problems for Interactive Rigid Body Simulation (2009)
24. Atkeson et al.: Simulating hands is killing performance, `http://answers.gazebosim.org/question/1427`
25. Atkeson et al.: Constraint stabilization methods discussion, `http://www.cs.cmu.edu/~cga/drc/constraint-stabilization.html`
26. Inertia tweaks, `https://bitbucket.org/osrf/drcsim/pull-request/157`
27. Zhang, L., Betz, J., Trinkle, J.: Comparison of simulated and experimental grasping actions in the plane. In: First International Multibody Dynamics... (2010)
28. Drumwright, E., Shell, D.: A robust and tractable contact model for dynamic robotic simulation. In: Proc. ACM symposium on Applied Computing, pp. 1176–1180. ACM (2009)