Analysis of Grasping Failures in Multi-Rigid Body Simulations

James R. Taylor[1], Evan M. Drumwright[2], and John Hsu[3]

Abstract— **Rigid body simulation libraries are sophisticated software systems that include multiple, tricky to implement numerical algorithms: solving initial value problems, root finding, geometric intersection (collision detection) and contact determination, and solving mathematical programming and optimization problems. How and why such systems fail to produce expected behavior is not readily known and not easy to discern; few of the components described above use reference implementations or are modular, for example, which makes attempting to identify points of failure challenging. Additionally, most libraries present an extraordinary number of parameters to be tuned, which complicates assessment.**

For the field of robotics, such failures are frustrating. One of the most seminal tasks in the domain, grasping of rigid objects, can require considerable parameter tuning. This paper uses a recent development, the support of four open-source physics engines in the GAZEBO simulator, to assess the ability of simulated robots to maintain grasps of rigid objects. We develop a metric that captures grasping performance and run a multitude of experiments to ascertain causes of failure. We have made all of our experimental code and data freely available, which allows others to reproduce our results and the authors of the corresponding physics engines to compete toward maximizing performance on the grasping task.

I. INTRODUCTION

Researchers and software developers periodically release new multi-rigid body simulation libraries with promises of improvements in speed and accuracy over the previous state of the art. And yet, problems with simulations persist: systems "exploding", robots vibrating at high frequency, bodies interpenetrating, and robots generally operating in an unexpected manner. Have simulation libraries actually been getting better, specifically with respect to robotics applications? If so, how much better are these new libraries? This paper represents the beginning of an objective effort to answer these questions.

Every simulation library is the product of a series of design decisions, including direct *vs.* iterative mathematical programing/optimization solver, geometric representation, coordinate type (reduced or absolute), and simulation paradigm (penalty-based, piecewise differential algebraic equation-based, time stepping). Exactly how any of these decisions affects a particular scenario or task or a class of scenarios or tasks has yet to be established, making objective comparison and evaluation of simulators extremely difficult. This paper focuses on a particular scenario and task that is ubiquitous in robotics: grasping of quasi-rigid objects with rigid robots.

[1]George Washington University, Washington, DC 20052, USA jrt@gwu.edu

[2]Toyota Research Institute, Palo Alto, CA 94306, USA drum@tri.global

[3]Open Source Robotics Foundation, Mountain View, CA 94040, USA hsu@osrfoundation.org

Stable quasi-rigid grasping is an anecdotally challenging task to simulate, and potential causes of failure are numerous: constraints are subject to numerical drift; truncation error arising from terminating an iterative method early can cause slip, and "jitter" due to correcting bilateral constraint errors and interpenetration can transmit excessive forces through the multi-body *system* (used henceforth to refer to the combination of the robot and grasped object).

This work uses an open test platform, GAZEBO version 4.0, to directly compare the performance of four existing multi-rigid body simulators on the aforementioned grasping task. GAZEBO's broad adoption and its support for multiple dynamics libraries make it well-suited for the comparison: the experimenter can ensure that the same model is used across all simulators. GAZEBO's ongoing support means that the experiments detailed in this paper can be maintained online and their results updated as software improves (or even regresses). As of this publication, GAZEBO supports the Open Dynamics Engine (ODE), BULLET physics, the Dynamic Animation and Robotics Toolkit (DART), and SIMBODY.

This paper considers two grasping scenarios: (1) a simulated robot arm moving through a trajectory while maintaining a grasp using a parallel gripper; and (2) a simulated fixed parallel gripper attempting to maintain a grasp on multiple objects simultaneously. The arm model used in the former scenario is a hybrid of the Universal Robotics UR10 arm, sourced from an existing ROS package and mated to a model of the Schunk MPG-80 parallel gripper. This configuration represents a reasonably accurate model of an industrial arm with respect to kinematics, geometry, and dynamics. On the other hand, the model used in Scenario (2) is contrived: a pair of boxes represent the parallel jaw gripper. The "robots" used in both scenarios are depicted in Figure 1.

Finally, we note that this paper does *not* try to answer the question of which software library may be superior. All libraries tested are actively developed and performance is subject to change, often dramatically, as bugs are introduced or corrected and new techniques are tested. It is also conceivable that GAZEBO contains bugs in the interfaces to one or more libraries. This paper instead focuses on understanding the factors that can contribute to grasping performance in simulation and providing a metric for evaluating grasp performance in simulation.

II. BACKGROUND ON MULTI-RIGID-BODY SIMULATIONS

Every simulation library is the product of many design decisions. This section discusses the most salient such de-

cisions, as judged by GAZEBO's currently supported multi-rigid body simulators and other available open source simulation software.

A. Simulation paradigm

Brogliato *et al.* [3] broadly categorize rigid body dynamics simulation software into three paradigms: event driven, time stepping, and penalty-based. Penalty methods treat contact using virtual springs and dampers. The resulting system of equations of motion can be integrated forward directly using arbitrary integration schemes, though implicit integrators are often recommended to avoid issues with the "stiff" equations that can result. Event driven methods (i.e., piecewise differential algebraic equation-based approaches) integrate the equations forward in time until an impact ("the event") occurs; the impact is modeled and the simulator resumes integrating the equations forward until further events are detected. Time-stepping methods, which are based around differential variational inequality (DVI) mathematical models, can avoid the need to locate events precisely by collecting all of the events that may occur during a time interval into a single complementarity problem. By solving a single such problem, time-stepping can allow systems with many contacts or impacts happening over a small period of time (e.g., a pool break) to be simulated more quickly.

B. Coordinate type

Articulated bodies can be defined using absolute coordinates (i.e., six coordinates per link) or independent coordinates (i.e., six coordinates per link minus the number of bilateral joint constraint equations). The former is used in several pieces of open source software (including ODE and BULLET), likely because of ease of implementation. However, absolute coordinates require constraint stabilization (e.g., Baumgarte stabilization [2]) to be used to minimize robot links from unrealistically separating at joints.

Constraint stabilization is also used to separate interpenetrating bodies. The bodies may interpenetrate due to non-convergent iterative solves (see §II-C) or inability of the collision detection system (see §II-D) to find precise times of contact. Interpenetration is also an organic byproduct of time stepping simulation processes.

C. Solver type

The bilateral (joint) and unilateral (contact and joint limit) constraint equations must be solved for constraint forces that obey compressive, non-interpenetration, frictional, and complementarity constraints; GAZEBO's supported simulators pose these constrained optimization problems as either nonlinear or linear complementarity problems. If the latter, either a direct (pivoting [5]) or iterative (matrix splitting method [17]) can be used. The only available solver for NCPs at this time is PATH [6]. For LCPs, the iterative Projected Gauss Seidel method seems to work reasonably in practice. However, convergence to a solution is not guaranteed; both Lacoursieré [15] and Drumwright and Shell [8] found these methods yield inaccurate solutions; Lacoursieré showed that for matrix condition numbers greater than 10^7, the method is unusable and that the method also "stagnates".

Direct solvers are limited too. Accuracy is significantly higher than with iterative methods (see [8]), but the linear system solves required of pivoting operations limit the practical number of variables to on the order of 10,000, both due to compuational complexity and available memory. The maximum number of variables is limited even further by the rounding error that accumulates during the pivoting process; the LEMKE [10] library limits the number of pivots to $\min(50n, 1000)$ for this reason.

Some simulation approaches enforce constraints by solving optimization problems rather than complementarity problems: see [9], [25] for examples. While these approaches are being actively investigated, GAZEBO does not currently support any simulators using such techniques, which removes a point of comparison. Our intimate knowledge of these approaches leads us to postulate that grasping performance would be affected little using an optimization based approach, but such a comparison will require future investigation.

D. Collision detection

There are few collision detection libraries that provide contact points and normal data; most such libraries only test whether two shapes are intersecting. There are multiple ways to compute contact points and normals (see, e.g., [12], [7]), which can depend on the geometric representation (primitive types, constructive solid geometry, polyhedron, implicit surface, or triangle mesh).

Finding correct contact data is complicated because two geometries are unlikely to be "kissing" due to floating point arithmetic, and because contact points and normals can only be estimated when bodies are interpenetrating. The estimation can conceivably be prevented by tracking the movement of the contact surface over time, but we are unaware of an existing open-source simulator that has implemented this idea.

E. Quasi-rigid DVI-based contact

Lacoursieré's dissertation [16] described a relationship between the complementarity problem regularization and constraint stabilization parameters and a first-order spring and damper model. This relationship indicates that contact constraints between rigid bodies can be effectively modeled as *quasi-rigid* in place of the fully rigid contact previously considered by DVI-based methods. We use the term quasi-rigidity to indicate that contacts between bodies are treated, at least partially, using spring-and-damper terms. We label those models with deformation along the tangent plane of the contacting surface between two bodies as fully compliant; this term would encompass both traditional finite element models and the penalty type model of [21].

F. Multi-rigid body verification and validation

This work continues our research into correctness of multi-body dynamics simulation using verification and validation [24], [4]. Where *verification* examines the solution

accuracy of our numerical implementation, *validation* seeks to reproduce behavior observed *in situ*. The present work thus focuses on validation because manipulators grasping blocks has certainly been observed *in situ*.

III. FAILURE MODES FOR QUASI-RIGID GRASPING

Objects slipping from a simulated robot's grasp can occur for several reasons, described below.

slip: If the grasping force or friction coefficient is not large enough for force closure. This case is the only one that is not an artifact of the simulation software.

iterative method non-convergence: Iterative methods are often terminated early; non-convergence can cause slip to occur at points of contact, joints to move apart unrealistically (thereby, for example, causing one or more gripper fingers to separate from the grasped object), or interpenetration to occur (which could cause objects to drop or become entangled [20]).

rounding error: Rounding error can occur even with direct solvers, leading to all of the problems described immediately above, though with lesser severity expected in this case.

regularization error: Regularizing the complementarity problem (via, e.g., the "constraint force mixing" parameter in ODE), causes constraint violation, which can lead to all of the issues described under *iterative method non-convergence*.

constraint stabilization: Projection methods (used in RPI-SIM, ODE, BULLET, and DART) for correcting joint constraint errors and interpenetration are prone to adding energy to the multi-body system [22] (which, among other issues, can compromise simulation stability).

imprecise contact information: As §II-D intimates, correct algorithms—and even precise problem specifications—for determining contact data have yet to be established. The general effects of inaccurate points of contact, normals, or both is currently unknown. The effects of approximating by point samples a polygonal or nonlinear manifold corresponding to the locus of intersection between two contacting bodies is also unknown. *Contacting shapes between the bodies considered in the present research are convex polyhedra, meaning that bodies will not interpenetrate if the contact constraints are not violated on the convex hull of the contact manifold.* This fact permits ignoring the effects of point sample approximation in the present work.

tangential drift: Although interpenetration and bilateral constraint errors are addressed by constraint stabilization, we are unaware of any multi-rigid body simulation libraries that stabilize constraint drift in the tangential direction, which would require tracking the evolution of the contact manifold as two bodies move. Without addressing such drift, we expect manipulated objects to gradually slip out of grasp.

We determined the factors above from reasoning about how multi-rigid body simulation software functions. However, teasing apart which of these factors is responsible is not the focus of the present paper. Indeed, evaluating even a single factor independently is impractical with most of the simulators that we evaluated. Instead, this paper applies multiple simulation libraries to test hypotheses (see §V) using the spectrum of available multi-rigid body dynamics software libraries.

IV. EXPERIMENTAL PRELIMINARIES

This section covers the evaluated simulators (§IV-A), the tasks (§IV-B), the performance metric used (§IV-C), and the description of the experimental controls (§IV-D).

A. Simulators

We assessed every simulation using an integration step size of 0.001. We found anecdotally that smaller step sizes produced better grasping performance (as expected), though our focus in this work is on factors that affect grasping performance *independently of integration step*. We conducted this investigation because reducing the step size leads to considerably slower simulations (e.g., simulations integrated at 10^{-4} generally run an order of magnitude more slowly than those running at 10^{-3}).

1) ODE*:* The version of OPEN DYNAMICS ENGINE (ODE) that we tested has been modified by Open Source Robotics Foundation to correct interpenetration by attempting to project bodies to a disjoint configuration (see [14] for details). ODE uses a slightly modified version of the first-order time stepping method described by Stewart and Trinkle [23] and Anitescu and Potra [1] with constraint stabilization procedure following the approach described in [13]. ODE uses absolute coordinates and supports both primitive and triangle mesh geometric types. GAZEBO 4.0's interface to ODE uses an iterative solver only, though "vanilla" ODE supports a direct solver.

2) BULLET*:* The version of BULLET we tested (2.82) also uses a time stepping approach; BULLET supports independent coordinates, though only absolute coordinates are supported in current versions of GAZEBO. Like ODE, BULLET also uses an iterative solver and supports both primitives and meshes; our examination of the source code, interface, and discussion groups indicate that ODE and BULLET function very similarly at the multi-rigid body dynamics level. However, BULLET also includes code for simulating soft bodies, fluids, and particles (none of which are used in the present study).

3) DART*:* We tested DART version *core-4 1.1.0*, which uses an independent coordinate formulation and the Stewart-Trinkle/Anitescu-Potra time stepping approach. DART uses a direct, mixed linear complementarity problem solver (specifically, ODE's Dantiz-Cottle LCP solver) and a triangle mesh geometry representation. DART uses a polygonalized friction cone for contact, thereby following the Stewart-Trinkle/Anitescu-Potra contact model more faithfully than the approximate pyramidal friction model used by both ODE and BULLET. We had to modify GAZEBO 4.0 to allow us to change the Coulomb friction coefficient in DART.

4) SIMBODY*:* SIMBODY version 3.4 uses a considerably different paradigm than the other physics engines in these experiments. SIMBODY uses a quasi-rigid contact model (to permit modeling the human skeleton, muscles, and skin)

that follows the event-driven paradigm in place of the time stepping approaches used by ODE, BULLET, and DART. SIMBODY proved to be too slow to conduct the numerous requisite experiments (simulating a single grasping scenario required on the order of days, i.e., in the same timeframe as FEM-based simulations). This elimination is unfortunate because SIMBODY's focus has been on accuracy, while the other simulations described above have targeted low running time. SIMBODY's performance does make it clear why the present work focuses on rigid body dynamics in place of more accurate FEM-type deformable codes, for which the remainder of the libraries described in this section can simulate the scenarios in minutes: considerably longer simulation times preclude model predictive control, fast edit-compile-test cycles, motion planning, and similar applications of multi-body dynamics simulation to robotics.

5) RPI-SIM*:* The multi-block grasping example was sufficiently simple that we were also able to set up the example in RPI-SIM (Subversion revision 463). RPI-SIM supports pivoting solvers (Lemke's Algorithm); a popular iterative matrix splitting method (Projected Gauss Seidel); and PATH [6], a popular library for solving complementarity problems (which operates using Lemke's Algorithm for LCPs with few variables and via a Newton-based method more generally). RPI-SIM uses the first-order Stewart-Trinkle/Anitescu-Potra method with the stabilization approach described in [13].

B. Tasks

The experiments using both grasping tasks, each of which is described immediately below, begin with the grasped object lodged firmly in the gripper: no grasp planning, pre-grasping, *etc.* were necessary in any experiment.

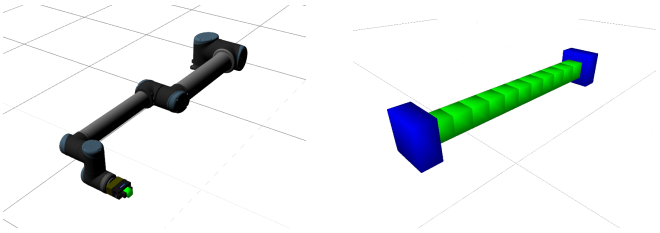
1) Grasp with industrial arm: We acquired a model of the UR10 robotic arm (depicted in Figure 1a) from the ROS Industrial repository and converted it to GAZEBO's SDF format. We attached this model to a model that we constructed of the Schunk MPG-80 hand, using CAD files and technical schematics from the manufacturer. The mated UR10/MPG-80 model yields a 8-DoF system (6-DoF arm + 2-DoF gripper). The inertial properties for the Schunk hand were set by approximating the bounding volume of the palm assembly and the individual fingers with boxes and apportioning the mass of the palm to be 80% of the total mass (described in the hand schematics); each of the two fingers were assigned the remaining 20% of the total mass. The box approximations were used for collision geometries in the geometric primitive experiments and the CAD meshes were used for collision geometries in the tessellated mesh experiments. The gripper prismatic joints were constrained by joint limits derived from the stroke per finger (defined in the schematics). Each gripper is actuated toward closure with 100N of force, which should be sufficient to maintain force closure on even relatively massive objects.

The object to be grasped was modeled as a simple cube with equal dimensions of 100mm. This yields a primitive/primitive representation, which we hypothesized would yield more robust grasping than using primitive/mesh or mesh/mesh-based experiments. A "collision box" was used for the geometric primitive experiments; for the tessellated mesh experiments, a mesh was generated in BLENDER using the same dimensions as the cube. The mass of the object to be grasped was set to 1 kg, and its inertia tensor was derived from a box with given dimensions and mass.

The arm controller was designed to move the arm through randomized sinusoidal trajectories for all joints using PD-control. The amplitude of each sinusoid was tuned to be as large as possible without the arm colliding with either itself or the ground plane. As noted above, the prismatic joints controlling the grippers applied 100N of constant force.

2) Block grasp: The block grasp scenario consists of a simple manipulator with a fixed base and a pair of parallel grippers (see Figure 1b). Each gripper is modeled as a box with dimensions larger than the grasp object and with a mass of 1 kg. Each gripper is actuated toward closure with 100N of force *per block*.

The object to be grasped is the same as used in the industrial arm experiment, i.e. 1/1000m^3 with 1kg mass.



(a) (b)

Fig. 1: The UR10 arm with Schunk hand used in the Grasp with industrial arm experiment (a) and the block gripper with eleven objects to be grasped used in the multi-block experiment (b). In both images, grasped objects are shaded in green and in (b) the grippers are shaded in blue.

C. Metric

Selecting a metric for identifying effective grasping performance proved challenging because our focus is only the qualitative behavior of keeping the object within the gripper. Comparing telemetry data against, e.g., control commands, would not be indicative of simulation performance as the robot might be observed to maintain a perfect grasp while the end-effector deviates from its commanded path. We devised a work-type metric to capture the qualitative behavior we sought. Specifically, we use a first-order approximation to the kinetic energy necessary to restore the grasped object to its *initial pose relative to the grippers*. This formula is described below:

$$T = \frac{1}{2} m \boldsymbol{v}^{\mathsf{T}} \boldsymbol{v} + \frac{1}{2} \boldsymbol{\omega}^{\mathsf{T}} \mathbf{J} \boldsymbol{\omega} \tag{1}$$

$$\boldsymbol{v} \equiv \frac{\boldsymbol{x}^* - \boldsymbol{x}}{\Delta t} + (\dot{\boldsymbol{x}}^* - \dot{\boldsymbol{x}}) \tag{2}$$

$$\boldsymbol{\omega} \equiv 2\mathbf{G}\left(\frac{\boldsymbol{q}^* - \boldsymbol{q}}{\Delta t}\right) + (\dot{\boldsymbol{\theta}}^* - \dot{\boldsymbol{\theta}}) \tag{3}$$

$\boldsymbol{x}^*$ and $\boldsymbol{x}$ in Equation (2) are the desired and current position of the grasped object, $\dot{\boldsymbol{x}}^*$ and $\dot{\boldsymbol{x}}$ in Equation (2) are the desired and current linear velocity of the grasped

object, $\boldsymbol{q}^*$ and $\boldsymbol{q}$ in Equation (3) are the desired and current orientation of the grasped object (in unit quaternions), and $\dot{\boldsymbol{\theta}}^*$ and $\dot{\boldsymbol{\theta}}$ in Equation (3) are the desired and current angular velocity of the grasped object. The difference between the unit quaternions is converted to an angular differential using well known formulas; we use one such formula to determine $\mathbf{G}$, which is defined with respect to $\boldsymbol{q}$ (see, [18], p. 175).

Each desired configuration is defined with respect to the grasped object's initial configuration. Velocities are defined relative to the gripper's velocities. The inertia tensor of the block, $\mathbf{J}'$ is defined relative to the grasped object body frame and is transformed to the world frame by operation $\mathbf{J} = \mathbf{R}\mathbf{J}'\mathbf{R}^{\mathrm{T}}$ where $\mathbf{R}(\boldsymbol{q})$ is the grasped object's orientation matrix. All other variables described in the equations above are defined relative to the global frame.

For each iteration, we compute the metric T of all grasped objects with respect to both the left and right grippers; those values are then averaged to yield a single value. The simulation was terminated if the simulation time reached 100 seconds or the metric exceeded 10^7 units (the simulation configurations lack a ground plane, so an object that slips from a grasp causes the metric to quickly exceed 10^7). Other invalid configurations, such as possible locking of models due to interpenetration (see [20] also) were noted, but otherwise not objectively measured; however, our metric does penalize interpenetration in a stable grasp compared to stable grasps with no interpenetration. Finally, note that $T = 0$ when the relative configuration between the object and the grippers matches the initial relative configuration.

D. Experimental controls

The experimental controls consisted of each of the three simulators (four in the case of the multi-block experiment) listed previously using GAZEBO's default parameters. We do tune one simulator—ODE, which was GAZEBO 4.0's best supported simulator—to show that it is possible to improve on these parameters significantly for this task; however, we also wished to establish a performance baseline. GAZEBO's default parameters have been determined using domain knowledge to maximize performance (speed and accuracy) over numerous simulation models and environments.

The models used in the experimental controls used realistic inertial values for all rigid bodies. Primitive geometric boxes were used to model the grippers and grasped objects. Coulomb friction coefficients were set to 100.0, which is significantly higher than that considered to be natural but ensures that the models contact without slip (in theory, if not implementation). Examination of multiple published friction tables indicates that 1.0 acts as an effective upper limit, though values larger than 1.0 are compatible with Coulomb's friction model.

V. TESTED HYPOTHESES

A. Hypothesis 1: modifying inertia properties could lead to improvements in grasping performance

It is accepted in the multi-body dynamics simulation community that large discrepancies in mass ratios lead to lower simulation performance. Such discrepancies cause ill-conditioning of the manipulator inertia matrix[1], given Featherstone's analysis [11] that showed that the condition number of this matrix grows up to $O(n^4)$ in the number of links in a robot and is also dependent on variations in link inertia (among other factors). This ill-conditioned inertia matrix may be viewed as a source of stiffness in the underlying differential equations [19], thereby affecting simulation stability. However, we guessed that grasping performance might be affected as well.

We tested this hypothesis by comparing performance for the experimental control against a robot model with modified robot arm and gripper link inertias (for the UR10). We assigned the shoulder link a mass of 1kg and inertia matrix of identity. Each subsequent link in the chain would have its inertial properties scaled geometrically (i.e., the second link would be scaled by $1/x$, the third link would be scaled by $1/x^2$, *etc.*) We experimented with scaling factors between 1.0 and 10.0.

B. Hypothesis 2: increasing the friction coefficient to very large values will not reduce grasping performance

While setting the Coulomb friction coefficient to a large value (i.e., larger than 1.0) will not generally yield a realistic simulation, there exists numerous anecdotal accounts of simulation users setting friction coefficients to particularly large values to prevent slipping at contacts. We tested the hypothesis above to see whether particularly large values of μ would reduce the numerical stability of the pivoting LCP solvers; we would expect the condition number of the LCP matrix (see [1] for the matrix's structure) to increase proportionally with μ. For testing the hypothesis, we modified the friction coefficients between contacting models in our experimental group by using a friction value of $\mu = 10^8$.

C. Hypothesis 3: using primitive geometries in place of mesh geometries yields improved grasping performance

Historically, modeling contacts between bodies represented using ubiquitous mesh-based geometries (e.g., triangle mesh) has proven challenging. We wanted to test whether this effect was present in the evaluated simulation software. In cases where multi-rigid body dynamics libraries supported both mesh and polyhedral representations (i.e., ODE and BULLET), we varied the representation of the grippers, of the object, and both. The evaluated groups were then (1) polyhedral grippers, polyhedral object; (2) polyhedral grippers, mesh object; (3) mesh grippers, polyhedral object; and (4) mesh grippers, mesh object.

D. Hypothesis 4: pivoting LCP solvers are more effective at simulating grasping than iterative matrix splitting solvers

This hypothesis follows from past research that has found that pivoting solvers are considerably more effective at solving LCPs (corresponding to multi-rigid body contact problems) with tens of variables to high degrees of accuracy [8].

[1] Although such a matrix is not explicitly constructed within all simulators, the conditioning of that matrix should yield a measure of numerical stability.

We speculated that a scenario with low admissibility for error would be more likely to expose this difference. Figure 1b illustrates this scenario, for which even one block slipping would likely lead to catastrophic failure. The experimental variable for testing this hypothesis was the number of blocks in the grasp, which varied from one to eleven.

VI. RESULTS FROM EXPERIMENTATION

The experiments generated significant data, including numerous plots and videos, which are available at our repository (`https://github.com/PositronicsLab/grasp-data`). Plots of control performance for the industrial arm and multi-block performance are depicted in Figures 2 and 3.

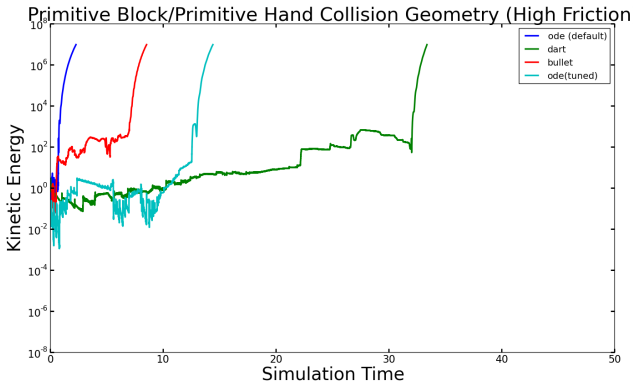


Fig. 2: Metric performance for the industrial arm grasping experimental control over three simulators (ODE, BULLET, and DART). BULLET becomes unstable immediately. ODE is able to maintain a stable grasp for nearly a second of virtual time. DART is able to maintain a stable grasp for nearly 35 seconds.

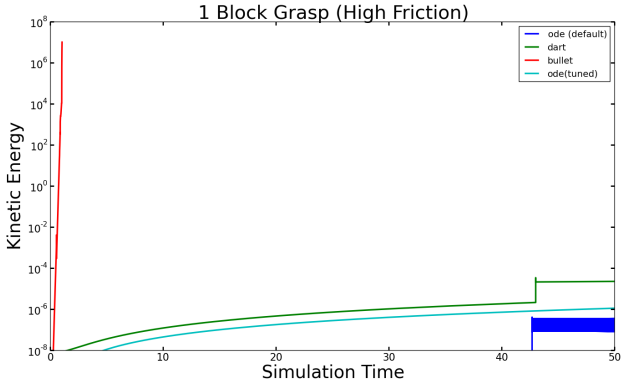


Fig. 3: Metric performance for the multi-block grasping experimental control (using a single block) over three simulators (ODE, BULLET, DART). BULLET becomes unstable immediately. DART (and to a lesser extent, ODE) exhibits a concerning discontinuity in performance, but generally is able to maintain a grasp for the length of the experiment. ODE maintains a stable grasp.

The data indicates the following responses to the hypothesis presented in the previous section.

Does altering inertial values affect grasping performance (Hypothesis 1)?: *Yes*. Unit mass / inertia yields significantly better grasping performance compared to the control group with realistic inertias (as depicted in Figure 4). Any integer scaling factor reduces the performance significantly worse than the control. These results indicate that simulation users must take care in setting inertial values, *even for this task for which simulation stability does not seem to be a factor.*

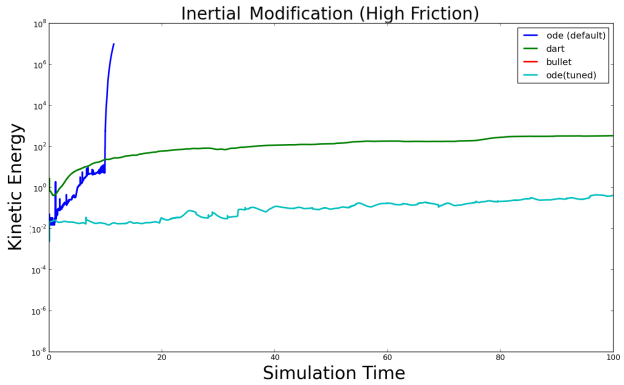


Fig. 4: Metric performance for the industrial arm grasping experiment over three simulators (ODE, BULLET, and DART) using modifications to inertial values. Grasping performance is improved significantly in two simulators compared to the depiction in Figure 2.

Do very high friction coefficients yield less stable simulations (Hypothesis 2)?: *No*. Our experiments (and Figure 5) indicate that a high friction coefficient ($\mu = 10^8$) does not affect the stability of the simulations.

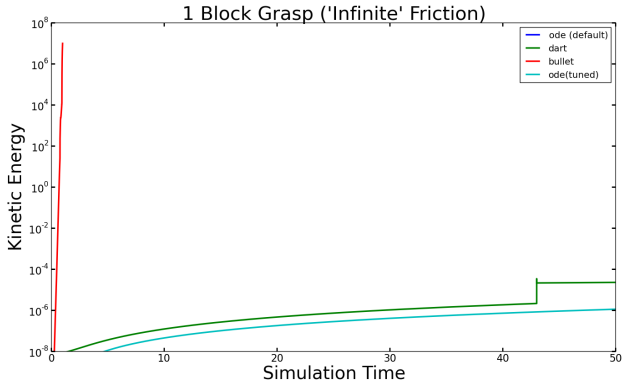


Fig. 5: Metric performance for the multi-block grasping experiment (using a single block) and very high friction value over three simulators (ODE, BULLET, and DART). Performance appears identical to Figure 3.

Does using mesh geometries impact grasping performance (Hypothesis 3)?: *Yes*. Both tested versions of DART and BULLET "segfaulted" with tessellated geometry. ODE drops the grasped object nearly immediately. We recommend using primitive type geometric representations in these simulation libraries whenever possible; aside from maximizing the performance metric in our experiments, the software implementations for primitive geometries have anecdotally proven to be far more robust.

Does the iterative solver fail readily on the grasping tasks (Hypothesis 4)?: *No*. Surprisingly, the lack of convergence proofs for the iterative LCP solver [15] and the much poorer solution performance [8] does not translate to challenges with

grasping. Instead, *we have found that the iterative solver performs better than the pivoting solver on the sensitive grasping scenario that we devised* (see Figure 6).

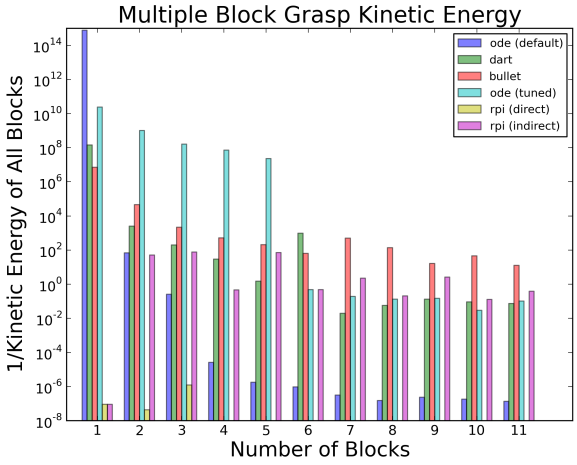


Fig. 6: **Logarithmic inverse** metric performance for the multi-block grasping experiment, illustrating direct vs. iterative solver performance at the end of 0.3s of virtual time. Taller bars indicate better performance. A missing bar (see, e.g. `rpi (direct)` for 11 blocks) indicates failure.

Does a particular simulation library outperform others across the board?: While we prefer not to address this question for reasons described in §I, we understand that readers will ask it. With the caveats in that section in mind, DART generally outperformed ODE under our metric; this performance differential could be due to many factors (pivoting method vs. iterative, accuracy of friction cone approximation, generalized coordinates vs. absolute coordinates). One key observation: ODE was able to simulate all scenarios, even if it does not always yield the highest performance; the library crashed less frequently than others, as the software has remained quite stable. BULLET performed surprisingly poorly, which Erwin Coumins (the lead developer) attributed to the particular version we examined (BULLET undergoes a fairly aggressive release schedule) as well as possible bugs in the GAZEBO-BULLET interface, in personal communication.

VII. CONCLUSIONS

Our experimental results make clear that analyzing quasi-rigid grasp failures in multi-rigid body simulation software is not a trivial problem. Software quality and design decisions can both affect grasping performance; the latter's effect is not necessarily predictable, as, for example, the results from testing our fourth hypothesis indicate. Our results point to new research efforts as well. Why do splitting matrix iterative LCP solution methods work better on the multi-block grasping scenario? What is the mechanism by which "regularizing" the inertia matrix values leads to more stable grasping performance? Can there be an effective middle group between primitive geometric representations (which are fast to evaluate and less challenging to code) of objects and mesh based representations (which are slow and challenging to code but for which formats are ubiquitous and shapes are generalizable)?

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] J. Baumgarte. Stabilization of constraints and integrals of motion in dynamical systems. *Comp. Math. Appl. Mech. Engr.*, 1:1–16, 1972.

[3] B. Brogliato, A. A. ten Dam, L. Paoli, F. Génot, and S. Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. *ASME Appl. Mech. Reviews*, 55(2):107–150, March 2002.

[4] B. Cheng Yi and E. M. Drumwright. Determining contact data for time stepping rigid body simulations with convex polyhedral geometries. In *Proc. Intl. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, San Francisco, CA, 2016.

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

[6] S. P. Dirkse and M. Ferris. The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5:123–156, 1995.

[7] E. Drumwright. A fast and stable penalty method for rigid body simulation. *IEEE Trans. on Visualization and Computer Graphics*, 14(1):231–240, Jan/Feb 2008.

[8] E. Drumwright and D. Shell. Extensive analysis of linear complementarity problem (LCP) solver performance on randomly generated rigid body contact problems. In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS)*, Vilamoura, Algarve, Oct 2012.

[9] E. Drumwright and D. A. Shell. A robust and tractable contact model for dynamic robotic simulation. In *Proc. of ACM Symp. on Applied Computing (SAC)*, pages 1176–1180, 2009.

[10] P. L. Fackler and M. J. Miranda. LEMKE. http://people.sc.fsu.edu/ burkardt/m_src/lemke/lemke.m.

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

[12] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. *ACM Trans. on Graphics*, 22(3):871–878, 2003.

[13] S. Hart, R. Grupen, and D. Jensen. A relational representation for generalized knowledge in robotic tasks. Technical Report 04-101, Computer Science Dept, Univ. of Massachusetts Amherst, 2004.

[14] J. M. Hsu and S. C. Peters. Extending open dynamics engine for the DARPA virtual robotics challenge. In *Proc. Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2014.

[15] 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.

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

[17] K. G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Heldermann Verlag, Berlin, 1988.

[18] P. E. Nikravesh. *Computer-Aided Analysis of Mechanical Systems*. Prentice Hall, 1988.

[19] F. A. Potra, M. Anitescu, B. Gavrea, and J. Trinkle. A linearly implicit trapezoidal method for stiff multibody dynamics with contact, joints, and friction. *Intl. Journal for Numerical Methods in Engineering*, 66(7):1079–1124, 2006.

[20] A. Rocchi, B. Ames, Z. Li, and K. Hauser. Stable simulation of underactuated compliant hands. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*, 2016.

[21] P. Song, M. Yashima, and V. Kumar. Dynamic simulation for grasping and whole arm manipulation. In *Proc. IEEE Intl. Conf. Robot. Autom. (ICRA)*, San Francisco, CA, USA, Apr 2000.

[22] 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.

[23] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction. *Intl. J. Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[24] J. R. Taylor and E. M. Drumwright. State estimation of a wild robot toward validation of rigid body simulation. In *Proc. Intl. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, San Francisco, CA, 2016.

[25] E. Todorov. A convex, smooth and invertible contact model for trajectory optimization. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Shanghai, 2011.