Comparing time-stepping and event-based simulation for nonsmooth dynamics with frictional impacts

Chiheb Boussema
Robotics Institute
Carnegie Mellon University
chiheb@cmu.edu

***Abstract*—Nonsmooth dynamics that experience frictional impact events and frictional contacts are notoriously difficult to simulate. The two extreme approaches to doing this are time-stepping, also known as contact-implicit, and event-based, or hybrid formulation (along with a spectrum of in-betweens). While usually slow, recent promising advances in contact-implicit formulation make it worthwhile revisiting the merits of both approaches. We make progress towards this comparison in this work and discuss that while time-stepping may be slower, it may be possible to remediate to this in the near future. Meanwhile, event-based approaches are characterized by their low computational time but increased coding complexity and sensitivity to parameters.**

***Index Terms*—frictional impacts, simulation, time stepping, event-based**

I. Introduction

Simulation of physics and dynamics is an essential part of engineering and science. From detailed computational fluid dynamics (CFD) to robotics simulation, it is important to have the appropriate physics engine that matches one's needs. And while CFD simulations may take hours if not more to complete, simulators for robotic applications are required to run close to real time, if not faster, at the potential cost of lower fidelity (only to some degree) [1]. Given the importance of simulation and modeling, there has been a lot of effort in developing models of particular laws, such as impact and friction laws, in simulation approaches, and in building physics engines. Some of the most popular include MuJoCo [3], PyBullet [5], Dart [6], and Gazebo [4]. Several of these simulation environments use what is called an time-stepping approach to simulation, which does not resolve impact time but solves for the impact forces or impulses. Some, like MuJoCo, also exclusively use explicit integration schemes (e.g., Euler, RK4).

Particularly important to robotic applications, being able to simulate dynamics with contacts and discrete impact events in an efficient way is of primary interest to the robotics community, and to the work presented here. In order to do this, modeling and simulation efforts abound. In [7], the authors review some of the most widely used impact models that include treatment of friction for single-point contacts. They also introduce their own model which is a mix of inelastic frictionless collision and a perfectly sticking inelastic collision. In [8], the authors extend single-point impacts to multiple simultaneous contacts. Their extension is inspired from and analyzed by the treatment of nonsmooth dynamics with frictional impacts in the field of differential inclusions [10]. The field allows recasting a certain notion of continuity to such nonsmooth dynamics, allowing for theoretical analyses and statements [9]. Practical algorithmic implementations had however been lacking. The algorithm presented in [8] compensates for that: they propose sampling a normal impact impulse, from which tangential impulses are recovered by way of an optimization that satisfies friction cone constraints, then the velocity of the object(s) being simulated is updated through an instantaneous kinematic relationship. The process repeats itself until the velocity is no longer impacting (i.e., it no longer leads to interpenetration).

Despite a high interest in the simulation of dynamics with impacts, a recent review of the different approaches that can be taken is missing. In particular, while usually described as slow, recent advances in time-stepping or contact-implicit problem formulations [2] make it worthwhile revisiting the relative merits of both approaches.

In this work, we make progress towards comparing event-based and time-stepping approaches to the simulation of nonsmooth dynamics with frictional collisions. We will compare the computational time, coding difficulty, and sensitivity to parameters.

The report is organized as follows. We review some preliminaries in Sec. II, in which we briefly describe time-stepping and event-based approaches. In Sec. III, we introduce the details of implementation of the hybrid dynamics in both approaches. We present results in Sec. IV, followed by a brief discussion in Sec. V.

II. Preliminaries

A. Implicit midpoint variational integrator

The variational integrator relies on the discrete Euler-Lagrange equation (DEL). Making use of the discrete Legendre transform (DLT), the DEL can be reinterpreted as a momentum balance between the states q_{k-1} and q_k on one side, and q_k and q_{k+1} on the other. The DLT is just the corresponding momentum, which is computed based on the midpoint velocity between the corresponding pair of knot points (see [11] for a more complete overview).

B. Time-stepping

The time-stepping or contact-implicit formulation consists in solving a constrained optimization at every time step. The constraints enforce no interpenetration between objects by solving for contact forces (additional constraints are imposed on the nature of the forces as fits the situation).

C. Event-based

The event-based or hybrid formulation uses a guard function (e.g., $\phi \geq 0$) that detects when a discrete event happens (impact). Often, the guard function will be violated, making it necessary to backtrack to resolve the exact time of impact. At impact, a jump map that models the discontinuity is executed.

III. Hybrid dynamics with frictional impacts

Physical systems often experience a sudden change in their behavior that can be characterized by a near-discrete or discontinuous jump in the state. This can be observed in several situations, from near-instantaneous jumps in thermodynamic states (e.g., pressure) due to the passage of a shock wave, to the sudden velocity reversal of an object bouncing on the floor, to the stopping of a legged animal's or robot's foot as it hits the ground when walking. Additionally, these events often involve a change in the dynamics followed by the physical system. For example, a free-falling object follows ballistic dynamics while in the air, until it hits the ground where it follows dynamics governed by friction forces. Such nonsmooth dynamics are therefore often characterized by sets of dynamic models that are valid in different regimes, and by transitions that are described by discrete jump maps (hence the appellation of hybrid dynamics). In this work, we focus on nonsmooth dynamics that experience impacts with dry friction.

A. Impacts

When an object makes a collision with a fixed object (like the ground), its velocity experiences a sudden change. Because no interpenetration is allowed, the component of the velocity of the impacting object that is normal to the collision surface must obey the following constraint:

$$\nabla f \cdot v^+ \geq 0,$$

where $f = 0$ represents the collision surface, and v^+ is the post-impact velocity of the contact. For an inelastic shock, the constraint becomes $\nabla f \cdot v^+ = 0$, meaning that $v_n^+ = 0$, where the subscript n is used to denote the normal component of a vector. We will use the subscript t to denote the tangential components of a vector. Additionally, the superscript + will be used to denote post-impact velocities, while the superscript - will denote pre-impact velocities. For a purely inelastic shock therefore, the object does not rebound. If an elastic shock is considered, the post-impact velocity may take a value of the form $v_n^+ = -\alpha v_n^-$, where α is a coefficient of restitution. A natural way to link the velocity change to an impulse is through the impulse-momentum relationship $P_n = m\Delta v_n = m(v_n^+ - v_n^-)$ (where m is the mass of the object). For a single-point contact, and if the mass matrix M is diagonal, then this relationship may be enough. However, in cases where M is not diagonal, recovering the collisional impulse may not be as simple. For example, consider the case of a rigid cube impacting the ground at an angle. Because the cube is not a simple point mass, impact forces, which are applied on the corners, generate a torque about the center of mass and affect velocity in more than one direction. A simple relationship of the type $P = M(v^+ - v^-)$ (3D velocities) may result in unphysical behavior, making the object exhibit unnatural tangential motions and vertical jumps. Nondiagonal mass matrix and the presence of tangential (friction) forces therefore require special attention, which we treat in the next paragraph.

B. Impacts with friction

When tangential frictional forces are allowed, a friction model must be chosen and be consistent with the impact law. A common friction model is the Coulomb friction model which states that $||\lambda_t||_2 \leq \mu\lambda_n$, where μ is the friction coefficient, λ is the contact force, and $\lambda_n \geq 0$. This inequality describes what is called a friction cone.
If the simulation method involves defining collisional impulses manually, one of the simplest ways of combining the impact and friction laws is to set [7]:

$$\begin{aligned}
\lambda_n &= -m(1+\alpha_n)v_n^- \\
\lambda_t &= -\frac{v_t^-}{||v_t^-||_2}\min\{\mu\lambda_n, (1+\alpha_t)m||v_t^-||_2\}
\end{aligned} \tag{1}$$

In this case, if the tangential impulse that is proportional to the change in tangential velocity lies within the friction cone, then it takes that values, and the resulting behavior is a sticking (including direction reversal due to a potential nonzero restitution coefficient). If it doesn't respect the friction cone however, slippage occurs and the tangential impulse lies on the edge of the friction cone.
For a nondiagonal mass matrix or when the simulation method doesn't allow prescribing impact impulses, one should write an optimization with appropriate constraints to find physically-consistent forces.

1) Time stepping: In time stepping, both the state and the contact forces (including impact) are the result of an optimization. To remain physically consistent, the following constraints must be observed: (i) equations of motion are satisfied at all times, (ii) forces are only allowed at non-separating contacts, (iii) no interpenetration, (iv) tangential friction forces are maximally-dissipative while (v) lying within the friction cone. We recall that the velocity of the object, v, is related to a given contact point's velocity, $\dot{p}$, by the kinematic relation $\dot{p} = K(q)v$, where q is the state and K is the contact point jacobian. Given a normal force or impulse $\lambda_n \geq 0$, the principle of maximum dissipation can then be stated as:

$$\begin{aligned} \min_{\lambda_t} \ & (K_t v) \cdot \lambda_t \\ \text{s.t. } & ||\lambda_t||_2 \leq \mu \lambda_n \end{aligned} \tag{2}$$

where K_t is the tangential contact jacobian. As (2) is based on an inequality constraint, we can formulate the lagrangian $L = (K_t v) \cdot \lambda_t - l \cdot (\mu \lambda_n - ||\lambda_t||_2)$. The KKT conditions then give:

$$\begin{aligned} & K_t v = -l \frac{\lambda_t}{||\lambda_t||_2} \\ & \mu \lambda_n - ||\lambda_t||_2 \geq 0 \\ & l \geq 0 \\ & l \odot (\mu \lambda_n - ||\lambda_t||_2) = 0 \end{aligned} \tag{3}$$

where l is a lagrange multiplier. The last condition of (3) can be relaxed by introducing a non-negative slack variable, s, such that we have $s - l \odot (\mu \lambda_n - ||\lambda_t||_2) \geq 0$. Now in order to include the friction force calculation with that of the normal force and the state, we do not directly set up an optimization for the maximum dissipation principle on its own, but rather include the relevant constraints from the KKT conditions above with the rest of the optimization. The whole optimization then becomes, for each time step of the simulation:

$$\begin{aligned} \min_{q_{k+1}, s_1, s_2, l, \lambda_k} \ & \sum_i s_{1,i} + \sum_j s_{2,j} \\ \text{s.t. } & \text{DEL}(q_{k-1}, q_k, q_{k+1}, \lambda_k) = 0 \\ & \text{norm(quaternion)} - 1 = 0 \\ & \phi(q_k) \geq 0 \\ & s_1 - \lambda_{n,k} \odot \phi(q_k) \geq 0 \\ & K_t(q_k) v_k + l_k \frac{\lambda_{t,k}}{||\lambda_{t,k}||_2} = 0 \\ & \mu \lambda_n - ||\lambda_{t,k}||_2 \geq 0 \\ & s_2 - l \odot (\mu \lambda_{n,k} - ||\lambda_{t,k}||_2) \geq 0 \\ & s_1, s_2, l, \lambda_{n,k} \geq 0 \end{aligned} \tag{4}$$

where DEL is the discrete Euler-Lagrange equation which represents the dynamics. Satisfaction of the dynamics here is a root-finding problem, hence the implicit nature of this integration scheme. $\phi(q)$ is the distance of the corners to the ground.

Note on friction cone constraint: The second order cone described by $\mu \lambda_n - ||\lambda_t||_2 \geq 0$ can be difficult to optimize over. Taking derivatives of the $||\lambda_t||_2$ term induces a division by the norm of λ_t, which introduces a singularity as $||\lambda_t||_2 \to 0$. One workaround this is to modify the norm $||\cdot||_2$ by a smoothed version computed as $smooth\_norm(x) = \sqrt{x^T x + \epsilon^2} - \epsilon$. However, in practice I observed the solver still found it very hard to converge even with the smooth norm. For increased computational speed, the friction cone is often linearized as $||\lambda_t||_1 \leq \mu \lambda_n$. Let d_k be unit vectors that discretize the cone's surface of revolution. As the linearized cone is a convex polytope, a vector that lies within it can be expressed as convex combination of its edges. In other words, $\lambda_t = \sum_k \alpha_k d_k$ with $\alpha_k \geq 0$ represents tangential forces that comply with the linearized friction cone. If the unit vectors d_k are concatenated in a matrix D and the α_k's in a vector λ_D, then $\lambda_t = D \lambda_D$. The linearized friction cone constraint can be explicitated as:

$$\begin{aligned} ||\lambda_t||_2 = ||\sum_k \lambda_{D,k} d_k||_2 \leq \sum_k ||\lambda_{D,k} d_k||_2 = \sum_k \lambda_{D,k} \leq \mu \lambda_n \\ \lambda_{D,k} \geq 0, \ ||d_k||_2 = 1 \end{aligned} \tag{5}$$

We will be using the linearized friction cone for both time stepping and event-based simulation.

2) Event-based: In contrast with time-stepping where impact impulses are necessarily integrated over a time step (which alleviates the dirac delta nature of the event), the impact in event-based simulation has to be introduced as an instantaneous change in velocity, but not in configuration. As the velocity change happens at the level of the collision point(s), the task is to find the corresponding impulses, then apply an instantaneous kinematic relationship to the velocity of the center of mass of the object. For reasons explained later, the details of implementation differ slightly between a variational midpoint integrator and an explicit (e.g., RK4) integrator. The fundamentals however remain the same. We present these fundamentals here, then dedicate a paragraph for the implementation details related to an variational implicit versus explicit integrator.

As mentioned previously, systems with non-diagonal mass matrix make it difficult to analytically write down physically-consistent impact impulses. In this work, we write an optimization program to solve for the impact impulses.

This optimization is solved every time an impact is detected and is simpler than the one performed for time stepping as no dynamics or interpenetration constraints are required. The principle of maximum dissipation with (linearized) friction cone are two of the three constraints used. The third constraint is for the normal component of the post-impact velocity of the impacting point to be equal to a desired value (zero in our case for an inelastic shock). To introduce this new constraint, we first consider the dynamics equation and let the time step Δt tend to 0. All that remains then is

$$v^+ = v^- + \bar{M}^{-1} K^T \lambda \tag{6}$$

where $\bar{M}$ is the body inertia matrix, K is the contact jacobian (dependence on the state q is omitted) and λ is the collision impulse. By multiplying on the right by K_n, the normal contact jacobian, we get:

$$\dot{p}_n^+ = \dot{p}_n^- + K_n \bar{M}^{-1} K^T \lambda \tag{7}$$

where all components are known except λ. The full optimization to resolve impact impulses in an event-based simulation then is:

$$\min_{s,l,\lambda} \sum s \tag{8}$$

$$\text{s.t.} \quad \dot{p}_n^+ - (\dot{p}_n^- + K_n \bar{M}^{-1} K^T \lambda) = 0 \tag{9}$$

$$D^T \dot{p}_t^- + \mathbf{1}^T l = 0 \tag{10}$$

$$s - l \odot (\mu \lambda_n - \sum_k \lambda_{D,k}) \geq 0 \tag{11}$$

$$\sum_k \lambda_{D,k} \leq \mu \lambda_n \tag{12}$$

$$s, l, \lambda_{D,k}, \lambda_n \geq 0 \tag{13}$$

where (10)–(12) represent the maximum dissipation and linearized friction cone constraints.

As mentioned above, the exact implementation details differ slightly between an explicit (RK4) and an implicit variational (midpoint) integrator. We cover these differences in the next two paragraphs.

a) Explicit integrator: Here, as the state contains the velocity vector, the latter (but not configuration) gets simply updated after resolving impact impulse as $v^+ = v^- + \bar{M}^{-1} K^T \lambda$. Then the dynamics are integrated starting from this new state and for what remains of the nominal step size.

b) Variational implicit integrator: As a reminder, because of backtracking to resolve impact time and because a variational midpoint integrator requires 3 knot points (2 are known, the third one is the one we look for), the time step that separates these 3 points at each iteration may not be the same (in case an impact is detected). So if impact is detected at q_{k+1} and time is backtracked to the impact time and state, then we will refer to this impacting state as q_{k+1} by abuse of notation, but it won't be recorded in the trajectory history. Let δt be the time step that separates it from q_k. The recorded knot point q_{k+1} will be the one obtained using q_k and q_{k+1} (with notation abuse) with a step size δt and using a step size of $h - \delta t$ between the impacting q_{k+1} (with notation abuse) and the recorded q_{k+1}.
In short, when there is an impact event, the DEL is re-called on the knot point before impact (q_k), the impact state (q_{k+1} - not recorded), and the next knot point (q_{k+1} - recorded), with different time steps between each pair of states. So while the DEL call after which an impact event was detected used knot point $q_1 \leftarrow q_{k-1}$ and $q_2 \leftarrow q_k$, the DEL call once impact has been resolved uses $q_1 \leftarrow q_k$ and $q_2 \leftarrow impact\_point$.

Another remark is that in this implicit integrator, the state, q, does not contain velocity. Instead, the midpoint velocity between 2 consecutive knot points is computed, and on the basis of which the discrete Euler-Lagrange equation makes sure the left and right momenta (between q_{k-1} and q_k on one side and q_k and q_{k+1} on the other) are consistent with the force being applied between q_k and q_{k+1}. Now because of the discrete impact event that happens at q_k, the DEL cannot rely on a momentum calculated based on the midpoint velocity between q_{k-1} and q_k. The latter would not be able to reflect the discrete change that happened to the velocity of q_k due to impact. It is therefore important to make sure that the left momentum is based on the post-impact velocity of knot point k. The right momentum can still be computed based on the midpoint velocity between q_k and q_{k+1}.

IV. Results

To evaluate and compare an event-based and time-stepping simulation for nonsmooth contacts with frictional impacts, we investigate 3 points of interest: computational time, difficulty of coding, and sensitivity to the integration time step (which may represent some notion of simulation accuracy). All optimization solves were carried out using IPOPT [12]. The friction coefficient considered here was $\mu = 0.3$, and no coefficients of restitution were used (inelastic shocks). The simulation is that of a rigid cube tossed in the air with some initial 6D velocity.

A. Computational time

When it comes to robotics simulation, it is imperative simulators run around real time to improve research efficientcy and, in case of RL, decrease training time. We therefore compare how long it takes for the simulation per iteration for both time stepping (TS) and event-based (EB). For a step size $h = 0.05$s (20Hz), each iteration (finding next state and forces) took an average of 3s. In the case of EB however, explicit RK4 integration took an average of 0.17s per iteration, while implicit midpoint integration took 0.052s per iteration. The difference between RK4 and implicit midpoint in EB may be explained by the fact that explicit integration is more sensitive to step size and can start producing unphysical behaviors at moderate step sizes. This may result in a higher number of detected impacts, hence a higher number of backtracking operations and optimization solves.
We also implemented TS and EB with a step size $h = 0.01s$ (100Hz). In this case, TS took an average of 2.2s per iteration, whilst EB with RK4 took 0.03s, and EB with implicit midpoint took 0.032s per iteration.
In all cases, in terms of computational time, EB compares favorably against TS with 1 to 2 orders of magnitude difference.

B. Ease of coding

As discussed throughout this report, event-based simulation involves a number of additional steps, each of which can be tricky to implement and may result in unphysical behavior if not carefully implemented. These include impact detection and impact time resolution, which we perform through bisection but can also be resolved more accurately through root-finding. The additional steps also include a careful separation of impacting (or equivalently penetrating) versus non-impacting but non-separating contacts, and assigning or solving for the appropriate forces/impulses. Midpoint integration in EB is even more tricky as it involves 'rolling' the knot points and possibly using two different time steps for each pair of knot points (as discussed in III-B2b).
By contrast, the time-stepping method is a unified approach

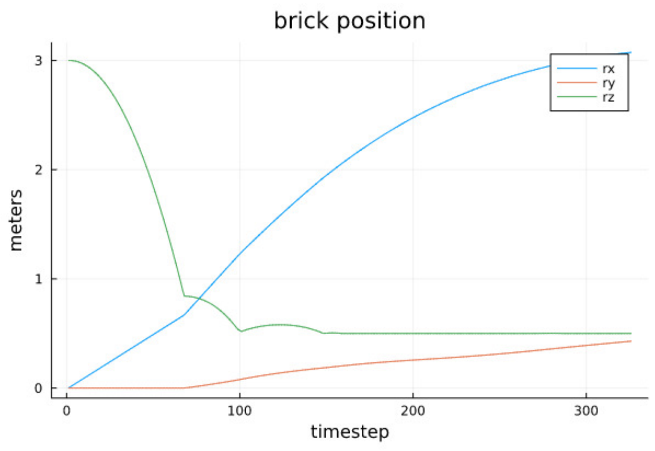

Fig. 1. Time-stepping simulation of a falling cube with a time step size $h = 0.01$s.

with a clean formulation, even though the optimization problem is larger with more constraints. However, many of the details that need to be explicitly taken care of in an event-based simulation are implicitly dealt with and do not require special care.

C. Sensitivity to integration time step

Another interesting aspect to look at is how confident one can be about the results of the simulation. This can be assessed qualitatively by using increasing step sizes. If the behavior remains stable, then one can be relatively confident that numerical artifacts are not being introduced. We simulated both TS and EB with $h = 0.01$s and $h = 0.05$s. For EB, we again further compare RK4 to the variational midpoint integrator.
As shown in Figs. 1-2, TS with $h = 0.01$s and $h = 0.05$s exhibit a largely similar outcome. The cube initially follows a ballistic trajectory as it is being released mid-air, then makes several impacts with the ground that reorient it, then keeps contact with the ground while seeing its tangential velocities decrease over time due to friction. The normal component of the contact forces resolved by the optimization are shown in Fig. 3, which clearly shows impact-related impulses to mitigate impacts, beside supporting the object's weight to avoid sinking when contacts are non-separating.

By contrast, EB simulation shows qualitatively different behaviors at 100 vs 20Hz, the most egregious of which occurs for RK4. For the coarse time step, the cube start sinking and sliding after it came to rest, which is unphysical. The midpoint integrator fairs qualitatively better, although with a smaller time step it appears to come to rest sooner and slides less. For EB at 100Hz, both RK4 and midpoint appear to be consistent with each other, although the behavior is slightly different from the one obtained by time-stepping (see Figs. 4–7). The difference may be explained by the difference in how the linearized friction cone is enforced. In time stepping, implementing a constraint of the type $\sum \lambda_{D,k} \leq \mu\lambda_n$ resulted in a dead impact whereby the cube impacted the ground on one corner and remained in that position, which is implausible. To remediate to this, the friction cone constraint was changed to $|\lambda_{D,k}| \leq \mu\lambda_n/N$, where N is the number of directions with which the friction cone was discretized. This results in a conservative approximation, explaining why the cube appears to slide further.

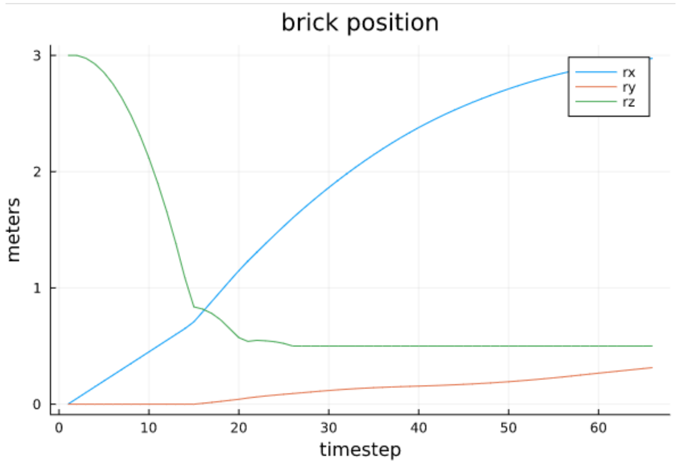

Fig. 2. Time-stepping simulation of a falling cube with a time step size $h = 0.05$s.

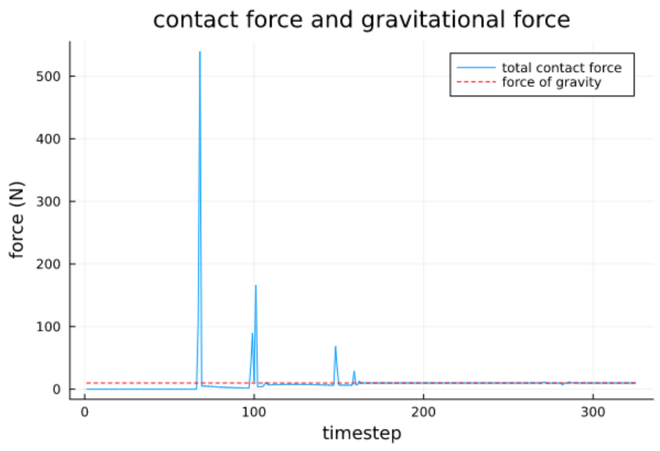

Fig. 3. Normal contact forces recovered in the time-stepping simulation of a falling cube with a time step size $h = 0.01$s.

V. DISCUSSION

In this work, we tried comparing the relative benefits and drawbacks of simulating nonsmooth dynamics with frictional impact events with a time-stepping (TS) approach versus an event-based (EB) approach. We have seen that TS takes considerably more computational time than EB, which may help explain why it is not a widely-used approach in popular physics engines. That being said, the current results rely on the use of IPOPT to do the heavy lifting and solve the optimization. A more specialized and carefully-written solver that exploits the structure of the equations might vastly improve computational time [2]. Additionally, in the EB's impact time resolution, we were only able to use simple bisection to coarsly

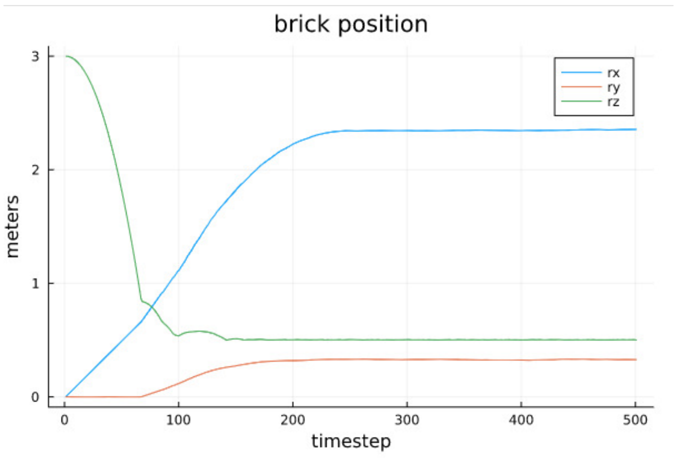

Fig. 4. Event-based simulation of a falling cube with RK4 and time step size $h = 0.01$s.

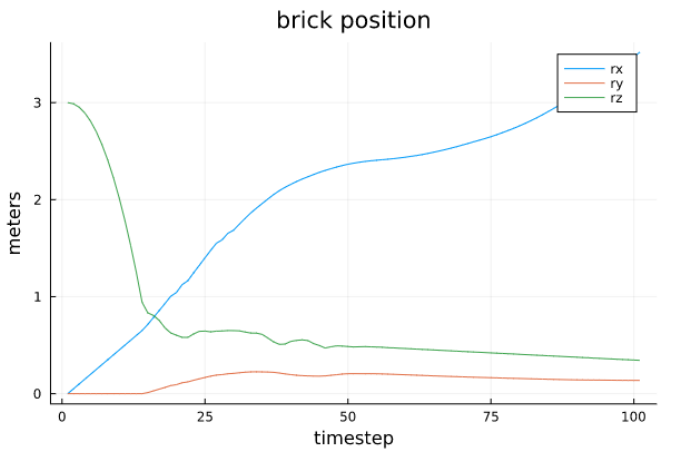

Fig. 5. Event-based simulation of a falling cube with RK4 and time step size $h = 0.05$s.

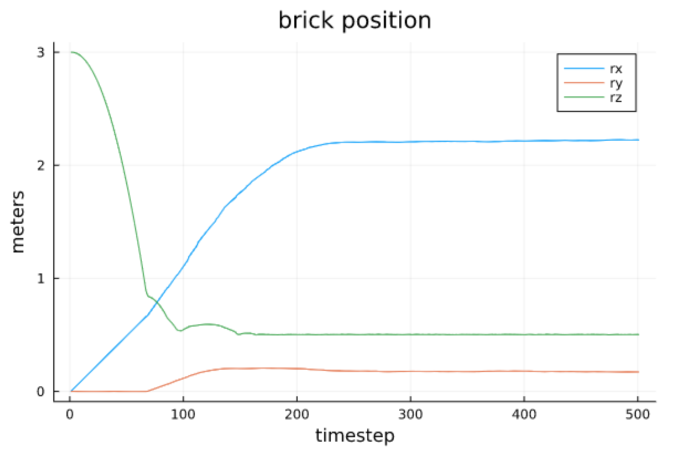

Fig. 6. Event-based simulation of a falling cube with variational implicit midpoint and time step size $h = 0.01$s.

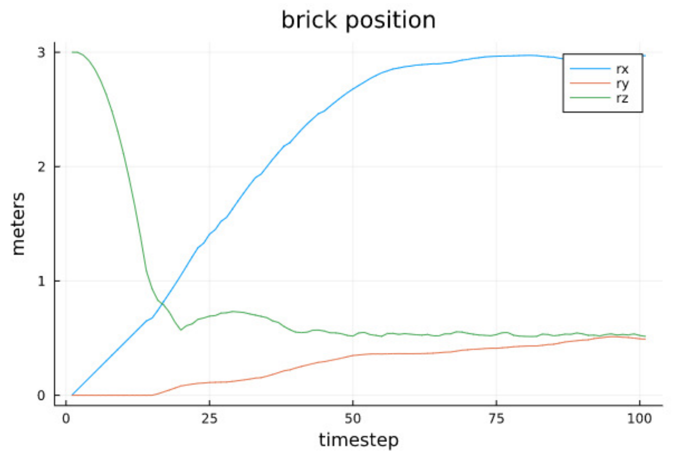

Fig. 7. Event-based simulation of a falling cube with variational implicit midpoint and time step size $h = 0.05$s.

resolve impact time. We would have liked to complement the coarse search with a fine-grained one by casting the time resolution as a root-finding problem. The solution would have converged to tight tolerances, with an additional computational cost which might have made the comparison with TS fairer.

While computational time remains an issue for TS, EB in turn suffers from increased complexity to keep track of impact events, nonseparating contact points, and making sure impact impulses and contact forces are physically-consistent. The difficulty is even greater when using an implicit variational integrator. A programmer can make mistakes in several parts, increasing coding time and uncertainty about the outcome. Moreover, the time step should be chosen sufficiently small to avoid unphysical behaviors, whereas TS seems to be less sensitive to an increase in step size (which in turn reduces computational time).

VI. Conclusion

Comparing the use of a time-stepping approach versus an event-based approach to simulate dynamics with frictional impacts and contacts, it is not obvious which is better. While computational time is an issue for time-stepping, a specially-crafted solver may be able to mitigate the issue. Meanwhile, while EB is faster, it is also more complicated and more sensitive to parameters like the step size (or others used in identifying contact modes), making it potentially less reliable. Future work may include simulating more complex objects (such as an egg), and exploring writing a specialized solver to speed up solve times for TS.

References

[1] Krber, M., Lange, J., Rediske, S., Steinmann, S., & Glck, R. (2021). Comparing Popular Simulation Environments in the Scope of Robotics and Reinforcement Learning. arXiv preprint arXiv:2103.04616.
[2] Cleac'h, S. L., Howell, T., Schwager, M., & Manchester, Z. (2021). Fast Contact-Implicit Model-Predictive Control. arXiv preprint arXiv:2107.05616.
[3] Todorov, E., Erez, T., & Tassa, Y. (2012, October). Mujoco: A physics engine for model-based control. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 5026-5033). IEEE.

[4] Koenig, N., & Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566) (Vol. 3, pp. 2149-2154). IEEE.

[5] Coumans, E., & Bai, Y. (2016). Pybullet, a python module for physics simulation for games, robotics and machine learning.

[6] Lee et al., (2018). DART: Dynamic Animation and Robotics Toolkit. Journal of Open Source Software, 3(22), 500, https://doi.org/10.21105/joss.00500

[7] Chatterjee, A., & Ruina, A. (1998). A new algebraic rigid-body collision law based on impulse space considerations.

[8] Halm, M., & Posa, M. (2021). Set-Valued Rigid Body Dynamics for Simultaneous Frictional Impact. arXiv preprint arXiv:2103.15714.

[9] Marques, M. (2013). Differential inclusions in nonsmooth mechanical problems: Shocks and dry friction (Vol. 9). Birkhuser.

[10] Aubin, J. P., & Cellina, A. (2012). Differential inclusions: set-valued maps and viability theory (Vol. 264). Springer Science & Business Media.

[11] Manchester, Z., & Kuindersma, S. (2020). Variational contact-implicit trajectory optimization. In Robotics Research (pp. 985-1000). Springer, Cham.

[12] Wchter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical programming, 106(1), 25-57.