Trajectory Optimization and Model Predictive Control (MPC) for an Inertia Wheel Cube

*This is a final project for 6.832 (Underactauted Robotics) taught by Professor Russ Tedrake and TAs Greg Izatt and Pete Florence.

Ethan Weber
MIT EECS
Cambridge, MA
ejweber@mit.edu

***Abstract*—Inertially-actuated robots pose interesting problems for underactuated control. Here we present trajectory optimization and model predictive control (MPC) to control a 2d cube with a torque-limited flywheel mounted inside. Inspired by the Cubli [2] and M-Blocks [3], we use contact-implicit trajectory optimization to generate trajectories for an inertia cube. Optimization techniques applied are guided by earlier work [1].**

In this paper, we use one actuator (the flywheel) to control the 8 states of the cube over time. Using Sparse Nonlinear OPTimizer (SNOPT) in Drake (https://drake.mit.edu/), the cube can be described as a floating body with contact-implicit dynamics via linear complementarity constraints [1]. This leads to elegant optimal trajectory control for the cube. After successful generation of trajectories, MPC was successfully used to control about unstable fixed points. MPC worked better than expected and solved very quickly at each timestep due to the convex optimization formulation.

***Index Terms*—trajectory optimization, underactuated, inertia, contact-implicit, model predictive control**

I. Introduction

The cube is particularly interesting both due to its extreme underactuation and its simplicity. After seeing the results of M-Blocks [3], it's clear that inertially-actuated cubes have potential to be used in a variety of applications. The possibilities of movements are immense, as seen in this video: https://www.youtube.com/watch?v=Nns0qzd8Noo. M-Blocks foreshadow the future or transformable robots. Furthermore, the Cubli presents very nice control in stabilization to resist movements (https://www.youtube.com/watch?v=n_6p-1J551Y). By combining prior work with ideas from optimal control and trajectory optimization, the cube has potential for much more complex behavior. With elegant algorithms, we may be able to help robots explore extraterrestrial space [6] or create configurable robots like the nanobots in the movie *Big Hero 6*. We are interested in applying our best underactuated control algorithms to this system, and we present our results and future goals in this paper.

II. Defining the Model

In this section, we describe the cube in floating-body coordinates with a few simplifying assumptions. The dynamics work out nicely when described in this way. We can then proceed to optimize over the state trajectory, input torque, and external contact forces over time.

A. State Space Explanation

Fig. 1 is a diagram depicting the states of the cube. The cube is defined in floating-base coordinates, meaning there is no notion of the ground in the state description. Rather, there are 8 contact forces (0-8), with 2 on each corner (A-D).

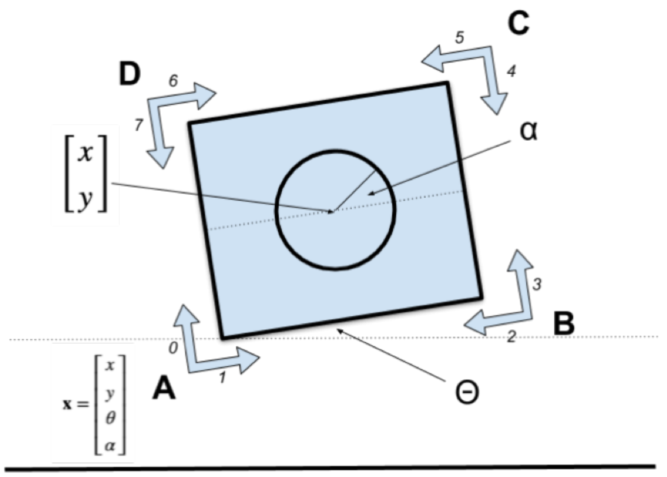

Fig. 1. The states of the floating cube. The derivatives of these states are also included in the total state vector **x** but are not shown here for convenience.

Following from the cube diagram, the full states can be described in the following way:

$$\mathbf{x} = \begin{bmatrix} x & y & \theta & \alpha & \dot{x} & \dot{y} & \dot{\theta} & \dot{\alpha} \end{bmatrix}^T \quad (1)$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} & \dot{\alpha} & \ddot{x} & \ddot{y} & \ddot{\theta} & \ddot{\alpha} \end{bmatrix}^T \quad (2)$$

The state vector **x** includes the positions, angles, and their velocities. This is standard state-space notation. Following from Fig. 1, the dynamics are shown below. These dynamics are informed by prior work done at Chalmers University of Technology [4] but modified for the free-body coordinates used here. We do not assume any pin joint dynamics (fixing the corner in place) [4].

$$\ddot{x} = (f_1 - f_2 + f_6 - f_5)\cos\theta \tag{3}$$
$$- (f_0 + f_3 - f_4 - f_7)\sin\theta \tag{4}$$
$$\ddot{y} = (f_1 - f_2 + f_6 - f_5)\sin\theta \tag{5}$$
$$+ (f_0 + f_3 - f_4 - f_7)\cos\theta - g(m_c + m_w) \tag{6}$$
$$\ddot{\theta} = \frac{-u + b_w\dot{\alpha} - b_c\dot{\theta}}{I_c} + \frac{1}{2}(\sum_{n\in 1,3,5,7} f_n - \sum_{n\in 0,2,4,6} f_n) \tag{7}$$
$$\ddot{\alpha} = \frac{u(I_c + I_w) + b_c I_w\dot{\theta} - b_w\frac{I_c+I_w}{2}\dot{\alpha}}{I_w I_c} \tag{8}$$

Here we explain the notation used in the dynamics and some simplifying assumptions.

- f_n is the force acting on the corner based on Fig. 1.
- u is the torque on the flywheel.
- The cube is unit sized, meaning its dimensions are 1 for each edge length.
- b_w and b_c are the friction coefficients for the wheel and cube respectively.
- I_w and I_c are the moments of inertia for the wheel and cube respectively.
- m_w and m_c are the masses for the wheel and cube respectively.

In this paper, the following parameters were used in the trajectory optimization and MPC solutions. We chose not to spend much time on choosing parameters and instead focused on the optimizations. The purpose of this project is to use contact-implicit trajectory optimization and MPC, which is described later in the paper. In the future we hope to prove robustness of trajectories with parameter perturbations. Ideally solved trajectories will work for cubes with a tolerance in frictions, moment of inertia, and mass estimates.

- $b_w = b_c = 0.5$
- $I_w = I_c = 0.5$
- $m_w = m_c = 0.5$
- $g = 9.81$

III. Trajectory Optimization Formulation

In this section, we explain the linear complementarity constraint optimization formulation to solve for a cube trajectory through space. The motions of the cube involve contact with the ground, which is why we are choosing to use contact-implicit trajectory optimization to avoid dealing with many contact modes of the system [1]. Therefore, instead of modeling the hybrid dynamics between every mode (and the dynamics associated with all modes), we can define the cube in floating-base coordinates and restrict contact forces to only be active when touching the ground. We can also ensure that the contact forces are within the friction cones. This is done with the following optimization problem:

$$\begin{aligned}
\underset{x[0:N],u[0:N],f[0:N]}{\text{find}} \quad & \\
\text{subject to} \quad & x[n+1] = x[n] + f(x[n], u[n], f[n])dt, \\
& n \in [0, N-1], \\
& -u_{max} < u < u_{max}, \\
& 0 \le f[n][i] \le f_{max}, i \in [0,7], \\
& f[n \in 0,2,4,6] \cdot \phi[:] = 0.0, \\
& f[n \in 1,3,5,7] \cdot \phi[:] = 0.0, \\
& -f_z[n]\mu \le f_{xy}[n] \le f_z[n]\mu, \\
& \text{min time} \le (N-1)dt \le \text{max time}, \\
& x[0] = x_{initial}, \\
& x[N] = x_{final},
\end{aligned} \tag{9}$$

Eq. (9) is shown without an objective function. Rather, this formulation only needs to find a viable solution. This formulation is only meant to represent the structure of the optimization formulation used. We used different objective functions (costs) depending on the problem at hand. Here is a description of the values included in this optimization:

- $f(x[n], u[n], f[n])$ represents $\dot{\boldsymbol{x}}$ at time n.
- dt is the timestep duration, which is a decision variable.
- N is the number of knot points used.
- $0 \le f[n][i] \le f_{max}, i \in [0,7]$ ensures that contact forces only push away from the ground. The forces cannot pull the cube down to the ground.
- $\phi[i]$ is the distance from the ith corner to the ground. This is assuming that the ground is the only other object in the world that can apply forces to the cube. Corners (A-D) are indexed (0-4).

A. The Swing-Up Problem

The swing-up is the action of getting the cube to stand on exactly one corner in a steady position when starting with one face on the ground. By using contact-implicit trajectory optimization, this is possible. We formulate the optimization problem with direct transcription and a linear complementarity constraint, shown in Eq. (9) as the dot product between the ϕ and f vectors. There can either be distance **or** force, so both cannot have non-zero values at the same time. The means force can only be acting on corners in contact with the ground. We found that this constraint is often too strict for the solver to find a solution, so it's important to add some slack to the constraint.

1) Swing-Up Specific Formulation: It's important to add some slack to the problem formulation to achieve good results. In particular, the linear complementarity constraint should be adjusted to allow for some force when near the ground, not just touching the ground. This results in better gradients for SNOPT as the corners approach contact. There were also other tweaks used to make this problem work. Here is a list of the modifications used to achieve a swing-up trajectory:

- **Floor penetration.** We allow for floor penetration up to 0.1 units into the ground.

- **Stay on ground.** The center of mass cannot go higher than it's max height while still maintaining contact. From empirical evidence, this helped to restrict the search space and resulted in quicker solution convergence.
- **Fix corner.** We constrain corner A to be in contact with the ground at all knot points.
- **Dynamics error tolerance.** We set a 0.01 tolerance on dynamically error. This is because our model may not be perfectly accurate.
- **Friction.** $\mu = 0.1$ seemed to have good results.
- **Initial and Final States.** We set the initial state and final state as described above. However, we do not control for the α position of the flywheel. This would be too unrealistic.
- **Objective Function.** No objective function is used, which is why it takes long to reach the final state. This is shown in Fig. 2. It takes nearly 15 seconds.
- **Thresholds.** Max torque of 1000. Max time of 15 sec. Min time of 0.5 sec. Max contact force of 100.

2) Swing-Up Results: The computed trajectory values for the problem specified above are shown in Fig. 2, 3, and 4. The trajectory in these figures start on the ground and stand on corner A. See http://ethanweber.me/cube.html for videos of these trajectories on a simulated cube.

Upon analysis of Fig. 4, the contact forces look as expected. Before moving, the cube is held above the ground by nearly constant normal forces acting on vertical forces 0 and 3. During the swing-up, corner B leaves the ground and force 3 goes to 0 magnitude. When this occurs, force 1 becomes active and equates force 0 in the upright state. This shows that the contact forces on corner A are perfectly balanced when at the upright while all other corner forces are at 0.

B. The Limit Cycle Problem

With success of the swing-up trajectory, we make slight modifications to find the optimal limit cycle for moving the cube horizontally by fixing the same corner (A) and rotating θ a positive 90 degrees. This requires a similar torque trajectory to the swing-up trajectory but now it must swing all the way around onto the next cube face. To solve for this problem, we did not put any restrictions on starting and final velocities. Rather, they only had to equate each other. Furthermore, the objective functions tested were to minimize total torque, minimize time, or maximize speed. We found the most interesting results by minimizing torque, which is shown in Fig. 5.

The most interesting region of the torque is found in 10 to 15 sec. The force increases, and then quickly brakes to make the cube jump. As the cube approaches its next face contact, it cranks down on the torque once again to brake it as it falls. A video is shown at http://ethanweber.me/cube.html. The static time until 10 seconds is rather strange, but it is likely due to numerical errors and could be excluded from the limit cycle. This would result in a velocity of about $\frac{1}{5}$ unit/sec.

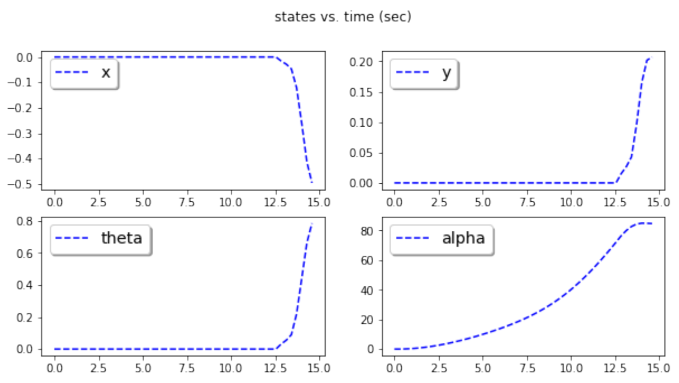

Fig. 2. Swing-up trajectory optimization states.

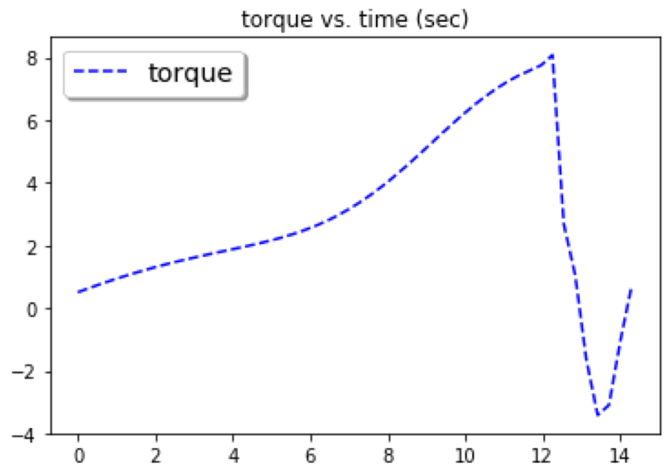

Fig. 3. Swing-up trajectory optimization torque input.

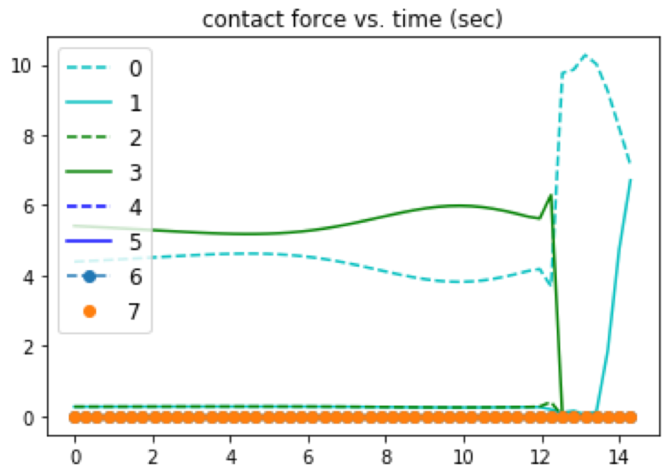

Fig. 4. Swing-up trajectory optimization ground forces.

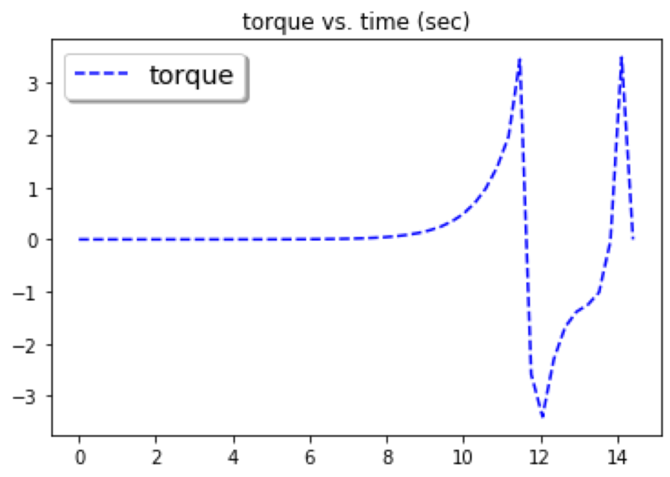

Fig. 5. This is the torque trajectory of the limit cycle with corner A fixed and moving to the left with 90 degree turns. The cube starts flat on the ground at the origin. Video available at http://ethanweber.me/cube.html

IV. MPC Stabilization

This section explains the success of MPC for stabilizing about the upright. We also show surprising swing-up results with solely MPC and no trajectory optimization.

A. MPC about the Upright

With success of the swing-up trajectory, we stabilize the upright with a proportional and derivative (PD) and quadratic programming (QP) controller. This is referred to as our model predictive control (MPC) in this paper. The optimization was formulated by wrapping a PD controller around θ to control for torque. The torque was computed by multiplying a proportional gain by the error in θ and adding that to a derivative gain times $\dot{\theta}$. The controller worked by solving at each time step, fixing the torque decision variable to the value computed, and solving implicitly for the contact forces given an objective function. The objective function minimized the L2 norm between the current state and target state (excluding θ and $\dot{\theta}$). Furthermore, the dynamics were linearized at each time step. We also added a quadratic cost to maintain the original and desired corner (A) position. Results from starting near the upright are shown in Fig. 6 and Fig. 7.

$$u = Kp(\theta - \theta_{desired}) + Kd\dot{\theta} \tag{10}$$

Here are some values chosen for our particular controller.

- $\theta_{desired} = \frac{\pi}{4.0}$
- $Kp = 50$
- $Kd = 10$
- $-10 \le u \le 10$

B. MPC for Swing-Up

Surprisingly, it was possible to use the same controller to swing the cube up from the the ground. This was a surprising result, especially considering how fast the swing-up is. However, we hypothesis that the short swing-up time

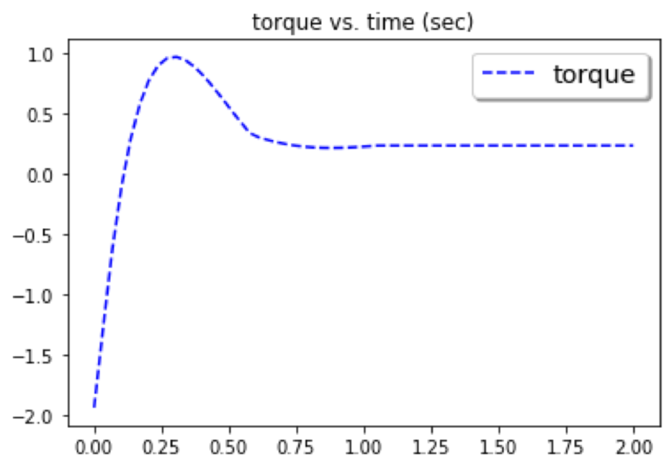

Fig. 6. Torques from MPC starting near the upright.

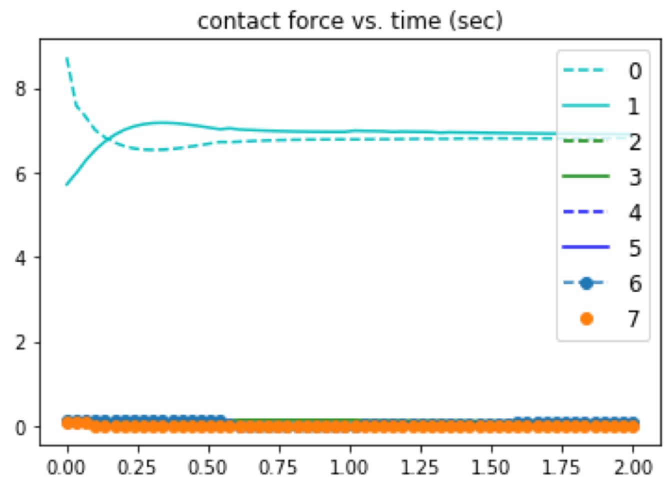

Fig. 7. Contact forces from MPC starting near the upright.

is because the corner position is not strictly maintained. It's much easier for the cube to swing-up when it's corner can slide with friction. Fig. 8 shows the torque and Fig. 9 shows the forces found. It's clear that the corner slides according to force 4 in the diagram. Furthermore, it takes less than two seconds to complete the swing-up given our optimization constraints and torque limits. Simulation video also reveals this at http://ethanweber.me/cube.html.

V. Findings and Future Work

In this section, we present general results, interesting findings, and commonalities between the solutions presented. We also use this section to discuss on-going work related to the cube.

A. Experiments in Higher Dimensional State Space

As we were implementing optimization code with SNOPT in Pydrake, we noticed that some problems would only solve when we added a few extra free variables to the states. These made no change to the dynamics, but this somehow resulted in solution convergence. We found this strange and have begun

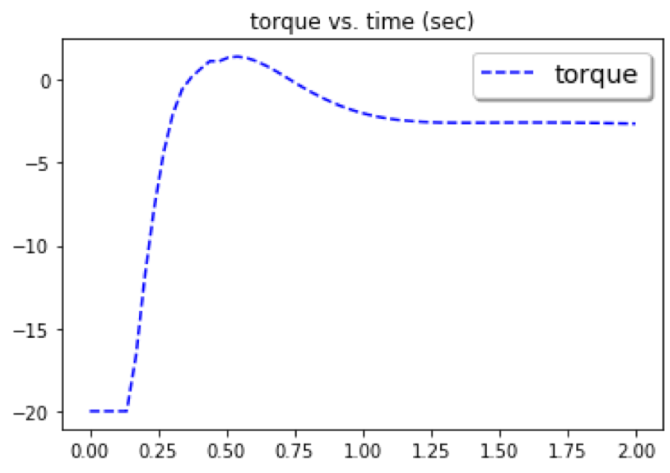

Fig. 8. This is the torque trajectory of the swing-up with MPC.

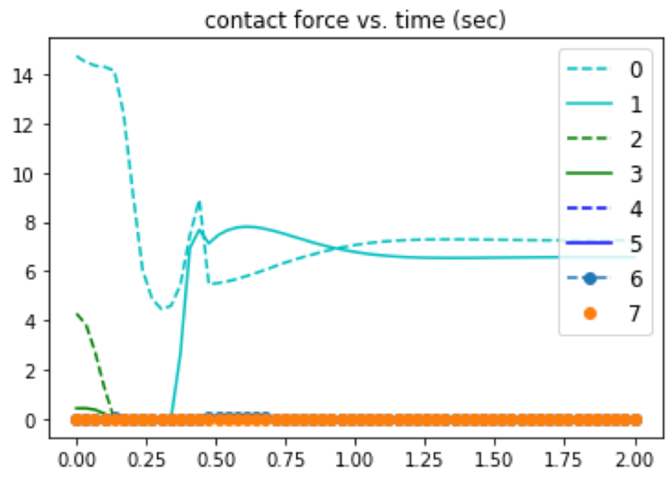

Fig. 9. These are the forces for the swing-up with MPC.

writing code to compute the time to solution with respect to the number of parameters involved. This effort is inspired by prior work and ideas on this topic [5].

The way we noticed this issue was by originally modeling the cube's state space in 3 dimensions (in Drake's simulation environment). This meant that there were 14 states instead of 8, but 6 of the states were free, meaning they did not enter the dynamics. In other words, the cube was restricted to stay in a 2d plane. After deciding to reimplement the state space formulation in 8 states, we could no longer solve the same swing-up problem. Analyzing the solution of the free variables showed very small numbers being computed, but nothing meaningful. After tweaking the trajectory optimization parameters, the solution would solve in 8 states. This finding remains a topic that we hope to address more rigorously in the future.

B. Initial Guesses

In this paper, no initial guesses were used in the optimization problems. We struggled for a long time with how to generate an initial guess given the contact-implicit dynamics. However, we now have plans and ideas to proceed with. The idea is to relax contact force constraints so they can act at much greater distances. This may result in trajectories in which the cube appears to float, but at least a solution will be created. This solution can then as act as the seed (initial guess) to SNOPT with a problem of tighter constraints. This idea is informed by other work on this topic [1] [7]. We have high hopes for its success due to the limitation of other initial guess solutions due to the large search space of forces.

C. Jumping and M-Block Work

Although we haven't attempted much to make the cube jump off the ground, it should be feasible due to the forward integration dynamic constraints. This enforces that the ground contact forces can only be active at time n if at time $n+1$ the corner is in contact with the ground [1]. This means that if the corner leaves the ground on $n+1$, then a force cannot be applied to it a time n. Therefore, the energy that would make the cube jump would be due to the energy from braking the flywheel. However, although this is true, we've had a hard time getting the cube to jump in an expected way. It's been much simpler to keep the cube on the ground when creating trajectories, and we defer this problem to future work.

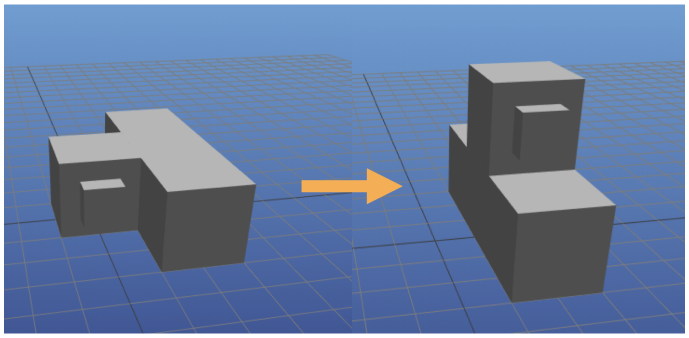

Fig. 10. Here is an example for an M-Block inspired trajectory (start and end state) we wish the achieve. This is image from the visualization environment, meshcat (), that I'm using.

In the future, we hope to get jumping to work, and we also hope to add more contact forces with other cubes. In order to add more contact forces, we will have to replicate the force vectors shown in Fig. 1 with new phi functions representing the corner distance to contact object. This will impose new forces that can activate when in contact with other objects, but this will come at a cost to the optimization. Each new contact will introduce 8 decision variables per timestep to the trajectory optimization method.

Furthermore, we hope to make more realistic models of the M-Block. The breaking mechanism can likely be modeled as a separate actuator for quicker flywheel deceleration. Also, magnetic forces may not be modeled correctly with contact-implicit models [3].

D. Proving Robustness

In the future, we hope to prove robustness of the trajectories computed by trajectory optimization. Our current solutions are guaranteed for point to point movements, but we wish to generalize to regions of space to target point. We also wish to show that a range of cube sizes, weights, friction coefficients, etc. can work with the same trajectory and some controller. We plan to create provably robust controllers that can follow an optimal trajectory and simulate results on real hardware (after success with an accurate simulator in Drake).

VI. Conclusion

With this work, we present success in modeling the inertia cube in floating-base coordinates and using contact-implicit trajectory optimization to control the cube. We also show results with using MPC to solve a convex problem that results in extremely fast control of the cube to the upright. Some key takeaways from the work done is that slack thresholds in optimization is very important, relaxing inequality state constraints to convex constraint (ex. adding a quadratic corner position and height cost instead of strict inequality position constraints eliminates infeasible solutions), and tuning of paramaters is very important. There is a lot potential for the inertia cube, and the methods presented in this paper are one possible means to achieve good results. We plan to continue the work discussed in the previous section. Code is located at https://github.com/ethanweber/cube and will continue to be updated as work continues.

References

[1] M. Posa, C. Cantu, and R. Tedrake. "A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact," in The International Journal of Robotics Research, 2014. http://groups.csail.mit.edu/robotics-center/public_papers/Posa13.pdf

[2] M. Gajamohan, M. Merz, I. Thommen, and R. D'Andrea. "The Cubli: A Cube that can Jump Up and Balance," in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012. https://www.ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Research_DAndrea/Cubli/Cubli_IROS2012.pdf

[3] J. Romaniskin, K. Gilpin, and D. Rus. "M-Blocks: Momentum-driven, Magnetic Modular Robots," in Proc. IROS, IEEE/RSJ, 2013. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.824.7275&rep=rep1&type=pdf

[4] E. Bjerke, B. Pehrsson. "Development of a Nonlinear Mechatronic Cube," Chalmers University of Technology, 2016. http://publications.lib.chalmers.se/records/fulltext/233543/233543.pdf

[5] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," in Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, 2014. https://ganguli-gang.stanford.edu/pdf/14.SaddlePoint.NIPS.pdf

[6] B. Hockman and M. Pavone. "Stochastic Motion Planning for Hopping Rovers on Small Solar System Bodies," in Int. Symp. on Robotics Research, 2017. https://asl.stanford.edu/wp-content/papercite-data/pdf/Hockman.Pavone.ISRR17.pdf

[7] I. Mordatch, E. Todorov, and Z. Popovic . Discovery of complex behav- iors through contact-invariant optimization. To appear in ACM SIGGRAPH, 31(4):43, 2012. https://homes.cs.washington.edu/~todorov/papers/MordatchSIGGRAPH12.pdf