Trajectory Optimization and Value Iteration for an Egg-Flipping Spatula Controller

Ronak Roy
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
ronakroy@mit.edu

***Abstract*—Using a spatula to flip a cracked, partially-cooked egg is a crucial part of making over-easy fried eggs. Since the egg's motion is ruled solely by the contact dynamics, which levies limitations on the forces applied to the egg, this is a prime underactuated system to attempt to control using trajectory optimization and an optimal controller derived using value iteration. The contact dynamics were modelled implicitly, allowing constraints to be added to the trajectory optimization problem that model the equations of motion and the restrictions on the magnitudes and directions of the normal and frictional forces. For the value iteration, a highly simplified model was constructed that allowed contact forces to be determined explicitly, encoding the contact constraints instead in the value iteration's cost function. The trajectory optimization was mostly successful, but high sensitivity to initial guess led to incomplete optimization results. Passing the spatula trajectory into a partial-feedback PID controller demonstrated that the modelled dynamics were sufficiently accurate, and full-state feedback and stabilization through a method like a Finite-Horizon LQR controller could improve the performance. The value iteration approach was unsuccessful, due to the immense size of the search space. Developing an optimal controller could be more successfully if done with a neural fitted approach, which would reduce the size of the space considerably.**

***Index Terms*—underactuated robotics, contact dynamics, implicit dynamics, trajectory optimization, value iteration**

I. Code

The IPython notebook for this work can be found at https://github.com/RonakRoy/EggFlipper.

II. Introduction and Related Work

Of the many ways to cook an egg, the method of making over-easy fried eggs is one that is beloved by many, but somewhat physically challenging. To make an over-easy fried egg, the egg is cracked onto a greased pan, allowed to cook for a few minutes, and then flipped such that the top of the egg may cook for a few seconds. This methodology ensures that the layer of egg white on top of the egg can cook fully, which would otherwise take a much longer amount of time and result in the egg yolk being cooked mostly though. One of the goals of the over-easy egg, though, is to have a runny yolk.

Having a robot perform the flipping operation raises an interesting challenge. A robot, or a human, for that matter, has full control over the spatula. However, the motion of the egg is governed only through the contact forces present between the egg and the spatula–given that the egg is not welded to the spatula, the system here is indeed underactuated, as certain accelerations of the egg are unachievable from any given state. One example is the inability for an egg with the yolk facing upwards to accelerate downwards with a magnitude greater than that of gravity–at best, the egg can only fall. Another more relevant, example of the underactuated nature of the system is how a configuration with the egg and spatula oriented vertically is not capable of being driven with an acceleration that is strictly upward. To generate an unpward frictional force on the egg, the spatula must also generate a positive normal force by "pushing" the egg sideways as it travels upward.

Thus, the goal of this work is to develop a controller that, given full wrench control of a spatula, is capable of taking an egg from resting on top of a spatula on top of a pan to upside down on the pan. The success of the controller is evaluated based on the angle of the egg, as well as its proximity to the origin, when its height above the pan becomes zero. To do this, two control strategies were attempted. The first is a trajectory optimization, which carries the benefit of allowing the contact dynamics to be modelled implicitly. The results of the trajectory optimization were used to develop a highly simplified model to use to attempt to obtain an optimal controller. This controller used a value iteration approach, in which the contact dynamics were enforced using the cost function.

This research project is an application of the controls techniques of trajectory optimization and value iteration, but incorporates the additional challenge of contact dynamics. This work was inspired by Dawson's work on trajectory optimization for pancake flipping [1]. However, this work puts special emphasis onto restricting the dynamics of the trajectory to account for the soft-body nature of an egg. As such, special care was taken to ensure that, over the course of the motion, the egg would never enter a regime in which it is expected to flex. The author is not aware of other work that offers this treatment to task involving contact dynamics.

III. Approach

A. System Dynamics with Implicit Contact

We begin by developing a model of the real-world system. In this model, we consider the motion existing in the vertical

XZ plane, as shown in Fig. 1, because the egg and spatula are symmetric about said XZ plane, and intuition suggests that the only control input that is provided by a human flipping an egg, and thus would be provided by a robot flipping an egg, is vertical force, horizontal force, and torque. We also consider the egg and spatula rigid bodies, for the sake of modelling, and thus give them a time-invariant mass and moment of inertia about the y-axis, as shown in Fig. 2.

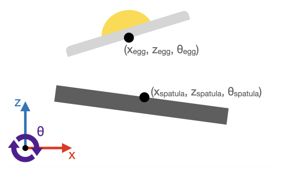

Fig. 1: Planar coordinate system, with origins affixed to the top center of the spatula and bottom center of the egg.

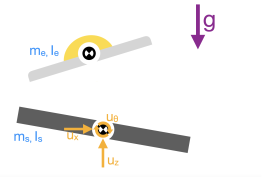

Fig. 2: Contributions to the manipulator equations: masses, moments of inertia, gravity, and control input.

Given this multibody model, we can use the Newton-Euler method to determine the standard coefficient matrices for the the manipulator equations, with generalized positions $\vec{q} = [x_s\ z_s\ \theta_s\ x_e\ z_e\ \theta_e]^T$ and control inputs $\vec{u} = [u_x\ u_z\ u_\theta]^T$, inertial matrix $\mathbf{M}$, Coriolis matrix $\mathbf{C}$, gravity vector $\vec{\tau_g}$, and control input matrix $\mathbf{B}$.

$$\mathbf{M}(\vec{q})\ddot{\vec{q}} + \mathbf{C}(\vec{q},\dot{\vec{q}})\dot{\vec{q}} = \vec{\tau_g}(\vec{q}) + \mathbf{B}\vec{u} \tag{1}$$

for

$$\mathbf{M} = \begin{bmatrix} m_s & 0 & 0 & 0 & 0 & 0 \\ 0 & m_s & 0 & 0 & 0 & 0 \\ 0 & 0 & I_s & 0 & 0 & 0 \\ 0 & 0 & 0 & m_e & 0 & 0 \\ 0 & 0 & 0 & 0 & m_e & 0 \\ 0 & 0 & 0 & 0 & 0 & I_e \end{bmatrix}$$

$$\mathbf{C} = \mathbf{0}$$

$$\vec{\tau_g} = \begin{bmatrix} 0 & -m_s g & 0 & 0 & -m_e g & 0 \end{bmatrix}^T$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

However, the contact dynamics require an additional term in the manipulator equations. Generally, a multibody system with contact can be modelled by considering contact at a finite number of points at which two bodies meet. To algebraically formulate the contact dynamics in a 3D multibody problem, one must define a the normal vector and two tangential vectors [2]. However, in the planar case, the forces involved with contact dynamics only act in the normal direction, i.e. the normal force, and in the tangential direction, i.e. the frictional force. Since we modelled the system as two interacting rigid bodies with flat surfaces that come in to contact with each other, we can abstract the contact interactions as being concentrated on the two corners of the egg, as shown in Fig. 3.

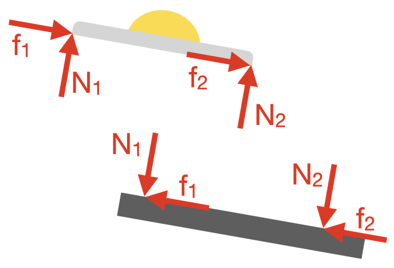

Fig. 3: The normal and frictional forces between the egg and the spatula, modelled as being concentrated at the egg corner points, directed normal to and tangential to the contact surface, respectively. Note the convention of positive normal force being towards the egg/into the spatula, and positive frictional force being to the egg's left/the spatula's right.

The forces present in this model introduce into the system dynamics the sources of the force and moment provided by the spatula onto the egg. In the real system, friction and normal force are distributed along the contact surface; however, in our rigid-body model, this abstraction is accurate, and simplifies calculations. Accounting for the fact that an egg is not a rigid body will happen during the constraints and/or costs for each of the two controller types.

Using the Newton-Euler method to determine the equations of motion now incorporating the contact force vector yields the following structure of the manipulator equations, with contact force vector $\vec{\lambda} = [N_1\ N_2\ f_1\ f_2]^T$. The aforementioned inertial, Coriolis, gravitational, and input terms remain the same.

$$\mathbf{M}(\vec{q})\ddot{\vec{q}} + \mathbf{C}(\vec{q},\dot{\vec{q}})\dot{\vec{q}} = \vec{\tau_g}(\vec{q}) + \mathbf{B}\vec{u} + \mathbf{W}(\vec{q})\vec{\lambda} \tag{2}$$

for

$$\mathbf{W} = \begin{bmatrix} -\sin\theta_s & -\sin\theta_s & -\cos\theta_s & -\cos\theta_s \\ -\cos\theta_s & -\cos\theta_s & -\sin\theta_s & -\sin\theta_s \\ b_1 & b_2 & -h_s/2 & -h_s/2 \\ \sin\theta_s & \sin\theta_s & \cos\theta_s & \cos\theta_s \\ \cos\theta_s & \cos\theta_s & -\sin\theta_s & -\sin\theta_s \\ m_{N1} & m_{N2} & m_{f1} & m_{f2} \end{bmatrix}$$

For the contact matrix $\mathbf{W}$, we introduce constant h_s, the height of the spatula, and variable terms b_1 and b_2 which represent the distance from the center of the spatula to the contact points, as shown in Fig. 4. The terms in the last row of $\mathbf{W}$ come from the moment balance on the egg, which can be derived using vector algebra on the geometry of the system. The complexity comes from the fact that the egg is not required to be parallel to the pan, but the contact forces still act according to the pan, as shown in Fig. 5. We also introduce d, the vertical distance along the height of the egg upon which its center-of-mass lies, and $\mathbf{R}(\theta)$, the 3D rotation matrix about the y-axis.

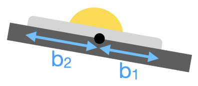

Fig. 4: Definitions of b_i, the contact point positions.

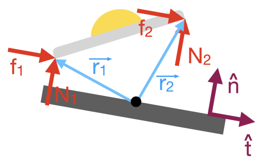

Fig. 5: Vectors used to calculate the m_i terms of $\mathbf{W}$.

Using the aforementioned vector quantities and constants, we can obtain the m_i terms as follows:

$$\begin{aligned} m_{N1} &= \mathbf{R}(\theta_e)\vec{r_1}\hat{n} \\ m_{N2} &= \mathbf{R}(\theta_e)\vec{r_2}\hat{n} \\ m_{f1} &= \mathbf{R}(\theta_e)\vec{r_1}\hat{t} \\ m_{f2} &= \mathbf{R}(\theta_e)\vec{r_1}\hat{t} \end{aligned} \tag{3}$$

B. Trajectory Optimization Formulation

Now, with the resultant manipulator equations, we can construct our trajectory optimization problem. We formulate the problem as a direct transcription trajectory optimization using the implicit Euler method. As such, we consider breakpoints $0, 1, \ldots, T$. Thus, our problem's decision variables are the following, $\forall i \in \{0, \ldots, T-1\}$ and $\forall j \in \{0, \ldots, T\}$:

- h_i. The length of the time interval between breakpoints i and $i+1$.
- $\vec{q_j}, \dot{\vec{q_j}}, \ddot{\vec{q_i}}$. The generalized coordinates and their first and second derivatives. The coordinates here are the x, z, and θ position of both the spatula and the egg. The position and velocity vectors are the state variables at each of the $T+1$ breakpoints, whereas the acceleration is defined along the T transitions.
- $\vec{u_i}$. The control inputs, i.e. the forces and torque applied to the two linear axis and one rotational axis. Like $\ddot{\vec{q_i}}$, it is factored into the system as a transition between states.
- $\vec{f_i}$. The contact forces N_1, N_2, f_1, f_2. Again, they are defined on the intervals between states.

1) Additional Constraints: In addition to adding constraints to represent the implicit system dynamics, we must also add a number of constraints to enforce contact requirements. Right now, the frictional and normal forces are completely unconstrained, and could be set by the optimizer to any arbitrary values to enable the trajectory optimization ends up with the egg upside down, on the pan, as if the egg was glued to the pan.

Non-penetration Constraint. First and foremost, the egg cannot be allowed to pass through the spatula. To achieve this, we define distance function $\vec{\Phi}(q)$ to be the projection of the corner distance vectors onto the pan normal vector, i.e. $\vec{r}_{1/2} \cdot \hat{n}$. We constrain this value to be greater-than-or-equal-to zero, ensuring that the egg stays above or at the surface of the spatula at all times. Similarly, we specify that $z_e \geq 0$, so the optimization finds no trajectories in which the egg attempts to swing under the pan.

Non-stickiness Constraint. We also levy the constraint that $N_{1/2} \geq 0$, since the egg should be allowed to fly off of the spatula. There is no adhesion between the egg and the spatula–again, our egg is not glued to the spatula.

Friction Cone Constraint. Using the standard model of friction, the friction force is variable, able to increase as high as possible in order to keep the two surfaces stationary w.r.t each other, up until the maximum frictional force magnitude, which is the coefficient of friction times the normal force. As such, we constrain the following: $f_{1/2}^2 \leq \mu^2 N_{1/2}^2$.

Distance/Normal Force Complementary Constraint. Our final constraint ensures that the spatula can only influence the egg's motion when it is actually in contact. An egg corner is only in contact with the spatula when the distance $\Phi_{1/2}$ is zero; normal force can only be experienced at a corner when it is in contact with the pan. Thus, we levy the constraint that $\Phi_{1/2} \cdot N_{1/2} = 0$. Since both normal force and distance are previously constrained to be non-negative, this constraint ensure that one of the two must be zero.

On-spatula Constraint. To prevent the program from assuming the spatula is infinitely wide, which would make the solutions less practically applicable, we add the constraint that $b_{1/2}$ lie within the bounds of $[-w_s/2, w_s/2]$, where w_s is the width of the spatula. We also constrain the distance to be less that a millimeter. This is to prevent the egg from flying off of the spatula and re-colliding–requiring an implementation of impulsive collisions combined with a hybrid-dynamic approach. We later discuss why considering breaking and making contact is not necessary, which is why we constrain out the possibility.

Start and End Constraints. Our last constraint is the one that actually enables the flipping. Our start constraints are that, at state 0, all positions and velocities are zero. The end constraint is on the egg at state T: the angle is π, and the z-position is the egg's height, such that the egg is upside, with the top of the egg touching the surface of the pan.

2) Costs: With the constraints specified, we must also add some costs to drive the trajectory optimization to a favorable solution.

Velocity/Input Costs. As "standard" costs, we add a quadratic cost for every state that is proportional to the magnitude of the velocity and the magnitude of the actuation input.

Normal Force Difference Cost. Due to our egg being a softbody, we penalize the square of the difference in N_1 and N_2 at each time step; this is so we prevent situations in which a very high moment is applied to egg, granted that it would likely cause the egg to flex, bend, or redistribute mass, especially if one of the corners was not in contact with the pan.

Height Cost. As a simply practical concern, we penalize the height of the egg at each time step. This cost arose experimentally, as without its presence, trajectories had an affinity for tossing the egg up a few meters into the air, which is not something that a real person would do, or a real robot should do.

Distance-from-origin Cost. Finally, since our pan would presumably be finitely large, we add a cost to the final x-position of the egg.

Now, the setup for our trajectory optimization is complete. All that is left is to set an initial guess and run the optimization problem. We will return to this in the evaluation and discussion section.

C. Value Iteration Formulation

Before moving on to evaluating the trajectory optimization, we first discuss the setup for the optimal control attempt. One of the major caveats of using value iteration for this problem is that the dynamics must be solved explicitly. This is challenging, because an arbitrary multibody system with contact dynamics has multiple dynamic modes. In our situation, these modes are full contact with the spatula, one corner contact, and no contact. The challenge arises in switching between these modes–impulsive collisions cause discontinuities in system state. In the case of the egg-flipper, however, we use our intuition to greatly simplify the problem in a manner that allows us to explicitly calculate contact forces.

When flipping an egg, one tries to move the spatula as to maximize the time the egg is on the spatula. Since the egg is a soft-body, allowing it to leave and then re-collide with the spatula over the course of the motion is unnecessarily complicated. Furthermore, we also do not want the egg to be allowed to "tip" about one of the corners, because it is not a rigid body. It would instead start flexing and bending the moment it stopped having full contact with the pan. Given these considerations, we formulate our model as follows: the egg is constrained to stay completely stationary, centered on the spatula, with both bodies having width w. Furthermore, as depicted in Fig. 6, we consolidate the frictional forces to a single force applied along the flat egg/spatula interface. We can do this because that contact surface is flat, and the frictional force is applied parallel to it. Thus, the same frictional force being applied on any point of that contact surface leads to the same resultant moment.

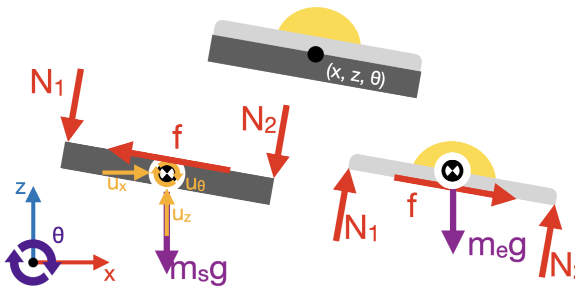

Fig. 6: Free-body diagrams of the system using the simplified value-iteration dynamical model, featuring the consolidated frictional forces.

WIth this model, we are left with the 6-by-6 system of linear equations that can be solved explicitly for $\ddot{x}$. $\ddot{z}$, $\ddot{\theta}$, N_1, N_2, and f.

$$
\begin{aligned}
m_e\ddot{x} &= (N_1+N_2)\sin\theta + f\cos\theta \\
m_e\ddot{z} &= (N_1+N_2)\cos\theta - f\sin\theta - m_e g \\
I_e\ddot{\theta} &= (N_1-N_2)\frac{w}{2} - fd
\end{aligned} \tag{4}
$$

$$
\begin{aligned}
m_s\ddot{x} &= -(N_1+N_2)\sin\theta - f\cos\theta + u_x \\
m_s\ddot{z} &= -(N_1+N_2)\cos\theta + f\sin\theta + u_z - m_s g \\
I_s\ddot{\theta} &= (N_1-N_2)\frac{w}{2} - f\frac{h_s}{2} + u_\theta
\end{aligned} \tag{5}
$$

There is, of course, an obvious oversight. If these system dynamics were directly used by value iteration, the normal forces and frictional force would be completely unconstrained, acting, again, as if the egg was welded to the pan. The contact constraints that we've left out are the crux of the problem; thus, we must incorporate them into our cost function for value iteration.

To do this, we add an exorbitantly high, and thus prohibitive cost, if $N_{1/2} < 0$ or $f^2 > \mu^2(N_1+N_2)^2$. This works because any state transition that would cause the egg to break contact with the spatula would require a negative normal force, and any state transition that would cause the egg to slide off of the spatula would require a friction force that violates the friction cone constraint. Thus, in calculating the transition matrices, our system dynamics would be "allowed" to calculate the accelerations for the "illegal" transitions. However, when value iteration actually runs, the calculated cost-to-go function would not rely on any "illegal" transitions, since the cost of them is so high. Thus, the resultant controller will enforce the egg not slipping or breaking contact with the spatula.

This cost is, of course, in addition to all of the "normal" costs: a cost on the difference between the egg angle and π, a cost on the x-distance from the origin, and a quadratic cost on actuation input. These base two costs drive the value iteration to develop a controller that actually flips the egg.

IV. Evaluation & Discussion

A. Trajectory Optimization Results

To measure the success of the trajectory optimization, we specify the following evaluation protocol:

1) **Check if the optimization was successful.** After running the trajectory optimization, the solver we're using, Snopt, outputs whether or not the optimization was successful. Additionally, in the case of failure, constraints that could not be satisfied can also be output.
2) **Qualitatively evaluate the trajectory.** If the optimization was successful, that means that a trajectory that satisfies all of the constraints was found. However, and this is true even if the optimization fails, we can visualize the trajectory and graph the positions, velocities, and

accelerations, as well as the calculated contact forces to evaluate the feasibility of the optimization

3) **Test the spatula trajectory with the contact solver in Drake.** Using a PID controller operating on the partial state (the positions and velocity of the spatula), we can control the spatula to follow the trajectory and check if the egg behaves in simulation.

In running the optimization, the most prominent issue was the optimization not completing. Removing, adding, and reformulating constraints and costs did not make a large difference in the success of the optimization. Instead, the initial guess was the most important variable that affected the quality of the optimization result. Many different initial guesses were attempted, leading to a range of different solutions (none of which, unfortunately, actually fully finished the optimization). Here, we look at two characteristic guesses, and the solutions the optimizer generated from them.

With this first guess, pictured on the left of Fig. 7, we can see the sequence of positions that resulted from running the optimization on the right of Fig. 7. The general motion here is a fairly constant horizontal velocity superimposed with a counterclockwise rotation that continues throughout throughout the entire motion.

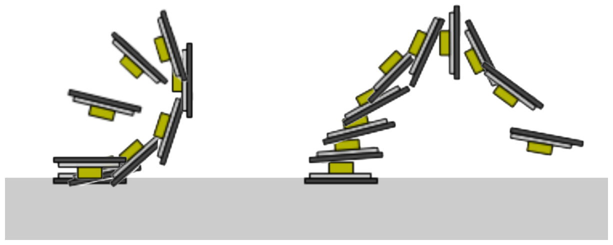
Fig. 7: (left) the first initial guess. (right) the result of the optimization.

However, running the spatula trajectory through the PID controller yields the output in Fig. 8. The egg leaves the spatula as it approaches the apex of the trajectory. This behavior is explained by the contact force graph in Fig. 9.

After 0.25 seconds, the normal forces drop off to zero; however, we see small spikes in frictional forces. These are the constraints that are violated to generate this solution. In this case, they give the egg little impulses to keep it in contact

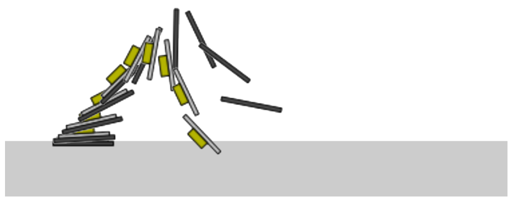
Fig. 8: The motion of the egg and the spatula with the spatula portion of the trajectory, derived from the first guess, commanded into the system with a PID controller.

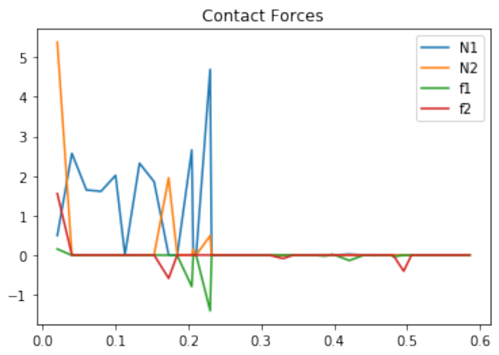

Fig. 9: Plot of the contact forces over time in the optimizer-output solution for the first guess.

with the spatula and to correct the angular velocity. As we can see here, it doesn't rotate enough by the time it hits the pan.

In Fig. 10, we see our second characteristic guess. This yields a similar trajectory (Fig. 11) to the first guess, except it has the feature of tipping the spatula slightly counterclockwise before following through with the counterclockwise flip.

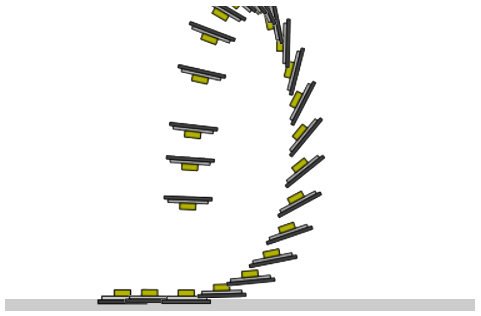
Fig. 10: The second initial guess.

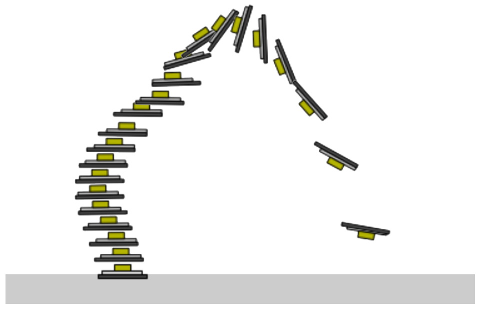
Fig. 11: The trajectory that resulted from running the optimizer on the second inital guess.

When run with the PID controller (as seen in Fig. 12), we see that it still fails to track after the egg departs near the apex, but egg ends up much closer in final angle to π when it approaches the surface of the pan. As we can see in Fig. 13, there aren't nearly as many spikes in frictional force that are not accompanied with normal forces; however, we now see additional forces near the end that do not correspond to actual contact in the PID-controlled trajectory. The forces are likely the one that correct the angle before the egg ends up below the pan.

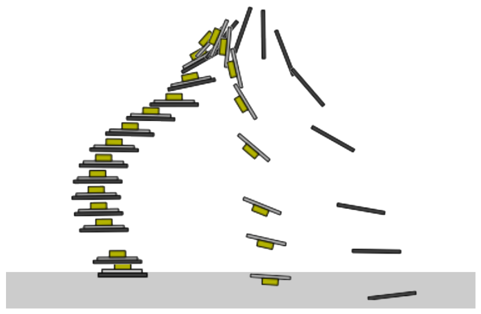

Fig. 12: The motion of the egg and the spatula with the spatula portion of the trajectory, derived from the second guess, commanded into the system with a PID controller.

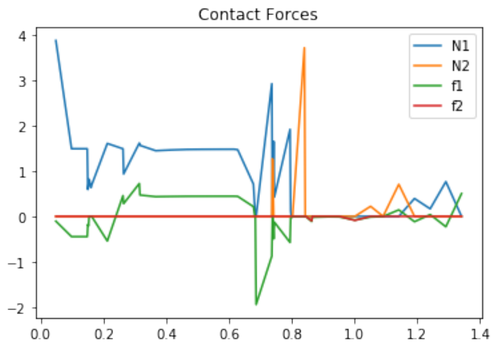

Fig. 13: Plot of the contact forces over time in the optimizer-output solution for the second guess.

The overarching lesson here is the supreme importance of the initial guess. The initial guesses were modelled as cubic hermite splines, with the breakpoints' positions and velocities determined by intuition. However, changing this drastically changes the success of the optimization. Ever so slightly adjusting the breakpoints can take the optimization from having a few instances in which the constraints are broken to having the optimization's output literally being unchanged from the initial guess. And, yes, it is true that using the PID controller does leave room for improvement. A Finite Horizon LQR controller was attempted; unfortunately, Drake does not support FHLQR for systems with contact dynamics.

Even so, the optimizer's best results included a few time steps in which the contact constraints were not appropriately met, or the system dynamics were violated, in order to keep the egg to a trajectory that succeeds. Upon visual inspection, intuition can suggest a small tweak to the trajectory that would make it succeed. However, the actually mathematics of the contact dynamics are deeply nonconvex, with harsh cusps present in many of the functions that we used for our costs. As such, it is incredibly easy for the optimizer to get stuck in local minima, in which the trajectory is very close to one that would be successful. Going from a trajectory where the egg is able to get to the goal state with sparse violations of the constraints to one where there are no violations may require pushing through numerical territory where there are many violations or where the costs momentarily spike due to something like incredibly high input magnitudes–this is something we can see the optimizer unwilling to do.

B. *Value Iteration Attempt*

Unfortunately, the implementation of the value iteration formulation was unsuccessful. The key issue encountered here was the size of the problem space. Even reducing the number of generalized coordinates to 3 still left us with a 6-dimensional state vector and a 3-dimensional input vector. Using the range of positions, velocities, and inputs from the trajectory optimization as a basis for the bounds of the space, the number of knot points needed to get any semblance of precision was simply massive. As such, even reducing the number of knot points in each dimension to 7, which was the minimum that seemed reasonable, the value iteration solver was unable to compute even the transition and cost matrices, let alone run value iteration on the result.

V. Conclusion

Overall, this work succeeded in formulating a trajectory optimization problem, as well as a value iteration problem, that could generate successful trajectories and controllers, respectively, to flip an egg. However, the deep non-convexity of the contact dynamics resulted in trajectory optimizations that were largely incomplete and heavily sensitive to the initial guess, and computational limitations prevented the development of an optimal controller using value iteration. As far as developing an optimal, nonlinear controller goes, a neural fitted approach would help during the actual value iteration algorithm, because it would greatly reduce the dimensionality of the search space.

As for the trajectory optimization, through tweaking the initial guess, reasonable trajectories can be developed. And, when the output trajectory, even if it violates the constraints, is passed into a partial-state feedback PID controller, the egg's final position on the pan can be made approximately correct–after all, the final position of the egg does not need to be perfectly upside down for a qualitatively successful flip. From this, we can conclude that our two-point contact abstraction of the egg was sufficiently accurate. Furthermore, aside from fine-tuning the initial guess, the next step for this would be to implement full-state feedback. Currently, measurement and feedback on the egg state is nonexistent; using a linearized controller with full-state feedback, like a finite horizon LQR, could help account for the loss of contact by "pushing" the spatula against the egg to re-establish normal force.

Acknowledgment

Thank you to Prof. Russ Tedrake, the course instructor of 6.832: Underactuated Robotics, for teaching the class, writing the textbook, and being an invaluable resource on dynamics, algorithms, and Drake throughout the course of the project, as well as Teaching Assistants Lujie Yang and Alexndre Amice for their help with this project.

References

[1] C. Dawson, “Pancake flipping via trajectory optimization,” Underactuated Robotics Final Project, May 2020.

[2] M. Posa, “Optimization for Control and Planning of Multi-contact Dynamic Motion”, Ph.D Thesis, June 2017.