Direct Trajectory Optimization of Dynamic Rigid Body Pushing

Kuan-Ting Yu

***Abstract*—In this project, I explore the method of direct trajectory optimization on a dynamic pushing problem, i.e. how to control a pusher to push an object to a target state. Specifically, I focus on letting the optimization solver to schedule contacts without users specifying it manually, and modeling friction dynamics between pusher and slider, as well as slider and the supporting surface. Implementation challenges such as non-differentiable system dynamics at zero velocity, and convergence problem for solvers are discussed. In experiment, I demonstrate several successful plan of pushing.**

I. Introduction

In this underactuated robotics class, we are impressed by how underactuated robots such as acrobots and compass-gait walker can do amazing tasks by exploiting dynamics. Moreover, we found that by considering dynamics of robots and the environment, robots can achieve better energy efficiency, and also move faster, as illustrated with the example of a swimming dead fish. Inspired by the spirit of underactuation, I want to investigate a kind of underactuated manipulation – pushing. In our daily life, we push objects that are too large or too heavy to lift, e.g. refrigerators and shelves. We do so because less force and energy is required. Pushing is also a simpler manipulation, we only need one point of contact, rather than multiple contacts to make force or form closures.

With pushing, we can control either the force that we exert or the acceleration of the end effector in pushing. To plan a trajectory of input control, I apply the direct trajectory optimization technique studied in class. It is easy to apply in general, but we found some issues in the case of kinetic friction in dynamics. The frcitional acceleration is a *sign* function of velocity, which is non differentiable at the zero point. This is bad for optimization. I found that by adding new variables and a complementarity constraint, we can make the constraint about frictions differentiable.

On the other hand, in order to learn the implementation fully and also learn to exploit drake system, I tried different combinations of tools to implement the system. The first one is a full implementation by myself without drake and only depends matlab's *fmincon* function, which results in bad solution when constraints cannot be met. In the second version, I switched from fmincon to SNOPT optimization tool, which gaves me the most reliable result. Third, I tried to derive my system from the drake system. Fourth, I specified the model in URDF, and let the drake do the work. However, this cannot incoporate 2 objects in contact-implicit trajectory optimization.

A. Terminologies

break points
: $t_0, t_1, ..., t_N$, a seires of time step where we specify a trajectory

knot points
: $x_0, x_1, ..., x_N, u_0, u_1, ..., u_N$, the values corresponding to the break points.

II. Related Work

A. Pushing mechanics

Mason [4] provides a qualitative theory on pushing: whether an object will turn right or turn left when pushed on a point. The reason that he does not give an exact motion of pushed object is because finding the total frictional force relies on integrating over an indeterministic pressure distribution between object and surface. Since we can never calculate it correctly, Mason [4] mentioned that approximating the contact using a tripod model is usually sufficient. Thus, Peshkin and Sanderson [5] give bounds on all possible motions resulting from pushing. The procedure is: sample all possible stable supporting tripod contacts, and use each of them to predict the motion. Because the motion uncertainty makes it hard to develop a usable pushing mechanics, Lynch and Mason [3] proposed using a fence as a pusher so that the slider's motion will become deterministic under quasistatic assumption. During trajectory planning, the contact side is allowed to change. To deal with dynamic pushing, Goyal [1] proposed the concept of limit surface through which the applied torque on the object given arbitrary force can be found easily. The limit surface is an approximation of the mapping function using a sphere. In this project, I want to remove the quasistatic assumption and incoporate dynamic pushing, and I adopt the tripod approximation so as to avoid the integral in the pushing dynamics inside trajectory optimization.

B. Trajectory Optimization

There are two main categories in trajectory optimization: *shooting* method and *direct* method. In shooting method, the only decision variables are the finite control plan over discrete time steps. Although the space of decision variable is small, the gradients of the dynamic constraints at every time step with respect to decision variables is a dense lower triangular matrix, causing the time to satisfy constraints to scale quadratically with the number of time step. Moreover, the cost gradient over decision variables is huge for the first time step, and decreasing toward the last time step. This is a bad situation for numerical stability.

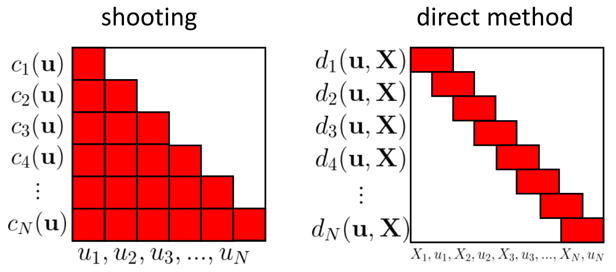

shooting
direct method
$c_1(\mathbf{u})$
$c_2(\mathbf{u})$
$c_3(\mathbf{u})$
$c_4(\mathbf{u})$
$c_N(\mathbf{u})$
$u_1, u_2, u_3, ..., u_N$
$d_1(\mathbf{u},\mathbf{X})$
$d_2(\mathbf{u},\mathbf{X})$
$d_3(\mathbf{u},\mathbf{X})$
$d_4(\mathbf{u},\mathbf{X})$
$d_N(\mathbf{u},\mathbf{X})$
$X_1, u_1, X_2, u_2, X_3, u_3, ..., X_N, u_N$

Fig. 1: Sparsity comparison of gradient matrices for shooting method and direct method: c_i's and d_i's are the constraints for each method.

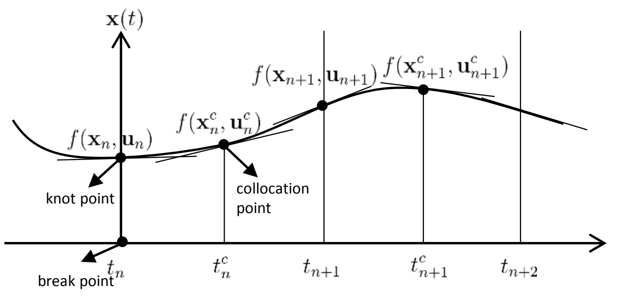

$\mathbf{x}(t)$
$f(\mathbf{x}_{n+1}, \mathbf{u}_{n+1})$
$f(\mathbf{x}^c_{n+1}, \mathbf{u}^c_{n+1})$
$f(\mathbf{x}^c_n, \mathbf{u}^c_n)$
$f(\mathbf{x}_n, \mathbf{u}_n)$
knot point
collocation point
break point
t_n
t^c_n
t_{n+1}
t^c_{n+1}
t_{n+2}

Fig. 2: Direct collocation method constrains the slope of the spline at the mid points to be coherent with system dynamics f.

On the other hand, direct method [2] incoporates system states at break points as variables besides the control variables. In this way, the gradient matrix becomes sparse, as illustrated in Figure 1.

In this project I use direct transcription and direct collocation method [2]. They mainly differ in how they impose system dynamic constraint. In direct transcription, a euler integration method is used: $\mathbf{x}_{n+1} = \mathbf{x}_n + dt \cdot \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n), \forall n$. In direct collocation, the dynamics is imposed on the slope of a cubic spline that approximates the continuous trajectory of $\mathbf{x}$ at the break points and the midpoints between them (aka collocation points), as shown in Figure 2. Constraints at collocation points are specified as

$$\dot{\mathbf{x}}_{\text{spline}}\left(t_n^c\right) = \mathbf{f}\left(\mathbf{x}\left(t_n^c\right), \mathbf{u}\left(t_n^c\right)\right), \forall n$$

$$t_n^c = \left(\frac{t_n + t_{n+1}}{2}\right)$$

C. *Contact implicit trajectory optimization*

Posa et al. [6] propose using nonlinear complementarity constraints to simultaneously resolve constraint forces and optimize the trajectory. They leverage Sequential Quadratic Programming (SQP) to solve the nonlinear program efficiently. This is the approach that I based on to let the optimization solver to schedule the contacts automatically.

III. Problem Formulation

In our system a pusher and a slider on a planar table are modeled. The target is to plan a input trajectory of the pusher $\mathbf{u}(\cdot)$ to push the slider to a goal state. The pusher is a massless and volumeless point of which we can control the acceleration. The pusher is described as its Cartesian position x_p, y_p, so $\mathbf{q}_p = [x_p, y_p]^T$. The slider's state is described as its Cartesian position x_s, y_s and orientation θ_s.

$$\mathbf{q}_s = [x_s, y_s, \theta_s]^T.$$

The whole system state becomes $\mathbf{x} = [\mathbf{q}_s, \mathbf{q}_p, \dot{\mathbf{q}}_s, \dot{\mathbf{q}}_p]^T$. The slider has mass m and rotational inertia I. The slider's shape M is described as a polygon of $|M|$ vertices:

$$M = \begin{bmatrix} v_{x_1} & v_{x_2} & \dots & v_{x_{|M|}} \\ v_{y_1} & v_{y_2} & \dots & v_{y_{|M|}} \end{bmatrix}$$

A. *System dynamics*

The system dynamics is described as a control differential equation of $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. In the output of funtion $\mathbf{f}$, $\ddot{x_s}, \ddot{y_s}, \ddot{\theta_s}$ are the result from the push force and frictional force, and $\ddot{x_p}, \ddot{y_p}$ are the same as the control $\mathbf{u}$.

B. *Acceleration from Friction*

To calculate the total frictional froce of the table acting on the slider, the exact method involves integrating individual friction force of every differential area of the contact surface (Figure 3). Let the velocity at a differential area be

$$v(\vec{r}) = [\dot{x}_s, \dot{y}_s] + \vec{r}_\perp \cdot (|\vec{r}|\theta),$$

where $\vec{r}$ is the relative coordinate from the center of mass of the slider, and $\vec{r}_\perp$ is a unit vector at position $\vec{r}$ and pointing in counter-clockwise direction about the center of mass.

$$\int_A \mu_t p(\vec{r}) \frac{-v(\vec{r})}{|v(\vec{r})|} dA,$$

where μ_t is the coulomb kinetic friction parameter, and $p(\cdot)$ denotes the pressure distribution.

The pressure distribution here is approximated at discrete points [4], so it becomes a function composed of delta functions at these support points. Then I can write the force as a sum instead of an integral. The translational force can be written as

$$F_f = \sum_{m=1}^{|M|} \mu_t f_{N_m} \frac{-v(\vec{r_m})}{|v(\vec{r_m})|}$$

The torque caused by friction can be expressed as

$$\tau_f = \sum_{m=1}^{|M|} \left| \vec{r}_m \times \frac{-v(\vec{r_m})}{|v(\vec{r_m})|} \right|$$

By Newton's second law of motion, the total frictional force will cause an translational acceleration of

$$[\ddot{x}_f, \ddot{y}_f]^T = \frac{F_f}{m}$$

and angular acceleration of

$$\ddot{\theta}_f = \frac{\tau_f}{I}$$

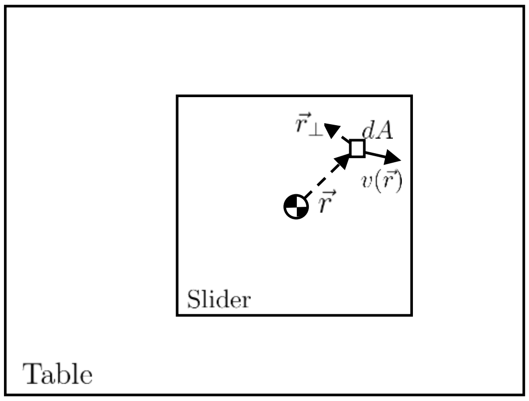


Fig. 3: Illustration of differential area of a slider on table

C. Acceleration from pushing

The pushing force F_p causes an translational acceleration of $[\ddot{x}, \ddot{y}]_F = F_p/m$ and rotational acceleration of $\ddot{\theta}_F = |\vec{r}_p \times F|/I$, where $r_p = [x_p, y_p] - [x_s, y_s]$ is the pusher's position relative to the slider's center of mass. In sum, the total acceleration of slider is $\ddot{\mathbf{q}}_s = \ddot{\mathbf{q}}_f + \ddot{\mathbf{q}}_F$

IV. Direct Trajectory Optimization

A. Direct transcription

In this method, we have decision vector,

$$\mathbf{z} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N, \mathbf{u}_0, \mathbf{u}_1, ...\mathbf{u}_{N-1}]^T$$

and want to minimize an objective function

$$\mathbf{x}'^T_{s,N} Q \mathbf{x}'_{s,N} + \sum_{i=1}^{N-1} \left(\mathbf{x}'^T_{s,n} Q \mathbf{x}'_{s,n} + \mathbf{u}_n^T R \mathbf{u}_n \right) dt,$$

where $\mathbf{x}'_{s,n} = \mathbf{x}_{s,n} - \mathbf{x}_{s,goal}$. Dynamic constraint is by Euler integration:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + dt \cdot \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n).$$

Thus, the optimization tries to minimize the inconsistency with dynamics.

B. Direct collocation

In direct collocation, a piecewise polynomial function is used to represent both a state trajectory $\mathbf{x}(\cdot)$ and an input trajectory $\mathbf{u}(\cdot)$. $\mathbf{u}(t)$ is represented with first-order hold model, and $\mathbf{x}(t)$ is approximated as a hermite cubic spline, which is defined as piecewise cubic polynomials. Let $\mathbf{x}_s(t)$ be a cubic polynomial defined in the interval of t_n to t_{n+1}. The constraints imposed on $\mathbf{x}_s$ are

$$\mathbf{x}_s(t_n) = \mathbf{x}_n,$$

$$\mathbf{x}_s(t_{n+1}) = \mathbf{x}_{n+1},$$

$$\dot{\mathbf{x}}_s(t_n) = f(\mathbf{x}_n, \mathbf{u}_n),$$

$$\dot{\mathbf{x}}_s(t_{n+1}) = f(\mathbf{x}_{n+1}, \mathbf{u}_{n+1}).$$

As derived in Homework 5, with $t_n \le t \le t_{n+1}, h = t_{n+1} - t_n, f_n = f(\mathbf{x}_n, \mathbf{u}_n)$, to let a cubic polynomial

$$\text{poly}(h) = ah^3 + bh^2 + ch + d$$

satisfies the above constraints, we have

$$a = \frac{2}{h^3}(\mathbf{x}_n - \mathbf{x}_{n+1}) + \frac{1}{h^2}(\mathbf{f}_n + \mathbf{f}_{n+1})$$

$$b = \frac{\mathbf{f}_{n+1} - \mathbf{f}_n}{2h}$$

$$c = \mathbf{f}_n$$

$$d = \mathbf{x}_n$$

Let the collocation point between t_n and t_{n+1} be:

$$\mathbf{x}_n^c = \text{poly}_n(h/2) = \frac{\mathbf{x}_n + \mathbf{x}_{n+1}}{2} + h\frac{\mathbf{f}_n - \mathbf{f}_{n+1}}{8}$$

The derivative at collocation points can be expressed using the decision variables:

$$\dot{\mathbf{x}}_n^c = -\frac{3}{2h}(\mathbf{x}_n - \mathbf{x}_{n+1}) - \frac{\mathbf{f}_n + \mathbf{f}_{n+1}}{4}$$

Then, we can impose the constraint so that the derivative matches system dynamics.

$$\dot{\mathbf{x}}_n^c[n] - \mathbf{f}(\mathbf{x}_n^c, \mathbf{u}_n^c) = 0,$$

where $\mathbf{u}_n^c = (\mathbf{u}_n + \mathbf{u}_{n+1})/2$.

C. Complementarity constraints

Let $\phi_k(\mathbf{x})$ denotes the closest distance from the pusher to k'th side of the slider, and $F_{p,k}$ be the constraint force. In order to let the solver schedule time of contacts, I specify complementarity constraints for all n as follows:

$$F_{p,k,\perp}[n] \ge 0$$

$$\phi_k(\mathbf{x}[n]) \ge 0$$

$$F_{p,k,\perp}[n] \cdot \phi_k(\mathbf{x}[n]) = 0,$$

where $F_{p,k,\perp}$ is the push force perpendicular to the contacting side. This constraint set imposes that only when contact happens at k'th side – $\phi_k(\mathbf{x}) = 0$–, can the pusher exert force on that side of the slider.

D. Non-slipping pushing

To constrain the pusher do not slide on the contact point, I restrict the lateral push force to be inside the friction cone caused by the normal pushing force:

$$\mu |F_{p,\perp}| \ge |F_{p,\|}|,$$

and the relative velocity of the two object should be zero if the pusher is exerting forces: $\psi(\mathbf{q}, \dot{\mathbf{q}}) F_\perp = 0$

E. Other constraints

In realistic scenerio, our motor of the robot arm do have actuator limits. Also, if we do not impose the limit, the solution space can be huge, and solver may spend time searching in space which is undesirable. Thus, we have a input limit constraint:

$$|\mathbf{u}| \leq u_{\max}$$

To constrain the trajectory to agree with the specified start and goal state, we have:

$$\mathbf{x}_0 = \mathbf{x}_{start}; |\mathbf{x}_{s,N+1} - \mathbf{x}_{s,goal}| \leq \text{tol}$$

For the goal constraint, we added a tolerance to help the solver find a solution faster. Otherwise, it is very hard to end at a specific pose and velocity.

V. Experiment

As mentioned in the introduction, for learning purpose, I tried implementing the system with different combinations. Among those implementations, I found the version implemented using SNOPT but without drakesystem gives the most reliable optimization result. In other implementations, the failures usually happens because of the nonlinear complementarity constraints are not satisfied, or collocation constraints are not satisfied, which needs further investigation. In the experiment $N = 10$, and the optimization reach to convergence in about 180 seconds, Figure 4 shows a planned push trajectory planned. At t_0, the pusher does not touch the slider so the push force is computed as 0. From t_1 through t_9, the pusher pushes on a side toward the goal $(0.1, 0.1, 0)$. After t_9 the pusher leaves the slider and let the friction stop the slider at the goal pose.

VI. Discussion

A. Gradient of Frictional constraint

During the implemntation, I found that $\mathbf{f}(x, u)$ is not partially differentiable with respect to $\dot{q}$ at $\dot{q} = 0$, which is required when calculating the gradient. Moreover, numerical difference would result in very huge value near the non-differentiable point. Here we provide a solution by rewriting the equation. Let v denotes $\dot{q}$, and a denotes $\ddot{q}$. Originally we have an equation of the form:

$$\frac{[v_x, v_y]}{\sqrt{v_x^2 + v_y^2}} = \alpha \circ [a_x, a_y],$$

where $\alpha = [\alpha_x, \alpha_y]$ is a constant vector. We can fix it by adding two variables v'_x, v'_y. Rewrite the equation as:

$$[v_x, v_y] = [v'_x, v'_y] \cdot \sqrt{v_x^2 + v_y^2}$$

and

$$[v_x, v_y] = \alpha \circ [a_x, a_y],$$

where $\circ$ is Hadamard product. Then the equations are clearly differentiable at $\dot{q} = 0$.

B. Constraint relaxation

In order to help SNOPT find a solution satisfying complementarity constraints, I introduce some slack variable to relax these constraint. That is, the equality constraints are fixed into a bounding box constraint from 0 to a small value ϵ. ϵ is decreased everytime SNOPT find a new solution based on the last optimal solution. This can improve the rate of convergence greatly and avoid having poor solutions.

VII. Techniques and Concepts Learned

Here is a list of techniques/concepts learned during the project.

A. Ordinary Differential Equation (ODE)

– Initial value problem: given $\dot{x} = f(x)$ and $x(t = 0) = x_0$, find $x(t)$.
– Runge-Kutta: Numerical method to integrate an ODE through time.
– RK4: A widely used configuration of Runge-Kutta method

$$y_{n+1} = y_n + h/6(k_1 + 2k_2 + 2k_3 + k_4) + O(h^5),$$

where

$$\begin{gathered} t_{n+1} = t_n + h \\ k_1 = f(t_n, y_n), \\ k_2 = f(t_n + h/2, y_n 1/2 k_1 h), \\ k_3 = f(t_n + h/2, y_n 1/2 k_2 h), \\ k_4 = f(t_n + h, y_n + k_3 h). \end{gathered}$$

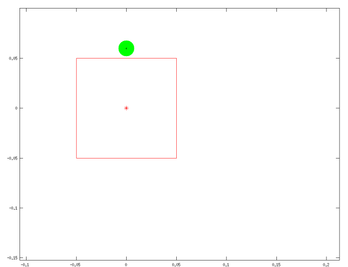
– Stiff equation: A differential equation with bad numerical stability when calculate it, unless with extremely small step.
– Numerical stability: Calculations that can be proven not to magnify approximation errors are stable.
– Forward Euler method: $x_{k+1} = x_k + hf(x_k)$
– Backward Euler method: $x_{k+1} = x_k + hf(x_{k+1})$

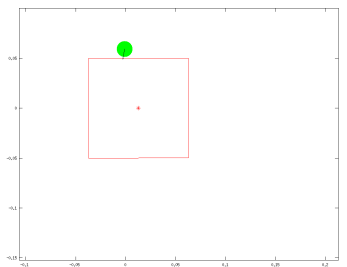
B. Numerical Method

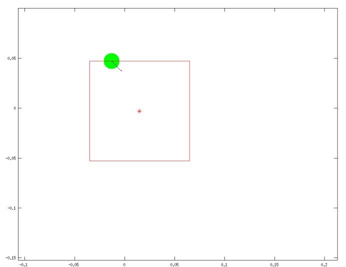
– Newton-Raphson method: find the root of $f(x)$ by iteratively computing $x_{n+1} = x_n - f(x_n)/f'(x_n)$ till convergence.

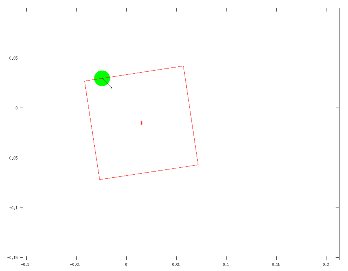
VIII. Conclusion and Future Work

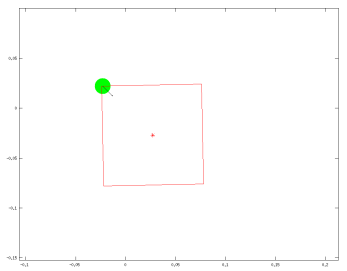
In this project, I have modeled a pusher-slider system, and applied direct trajectory optimization to plan a push trajectory to push the slider from a start to an end pose. Complementarity constraints are used to let the pusher schedule the contactin time by itself. However, to satisfy this sort of constraints is hard for optimization solvers. Thus, I employed a constraint relaxation scheme to help solver converge to a valid solution. I also provide a fix to the frictional acceleration, which is non-differentiable w.r.t the velocity in the original formulation. In the future, a multi-slider can be studied, so that the dynamics of friction can help to move sliders in parallel. However, multi-sliders would also require special care in collision between objects. On the other hand, due to the uncertainty in pushing motion in real world, and errors in approximating the trajectory, using a time variant LQR to follow the planned trajectory is needed in real application.

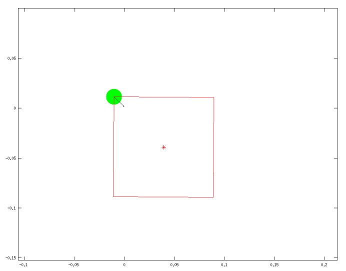

(a) t_0

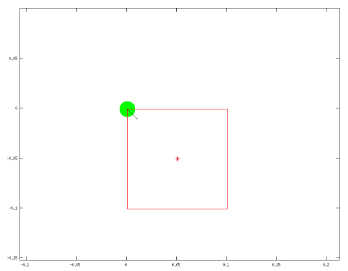

(b) t_1

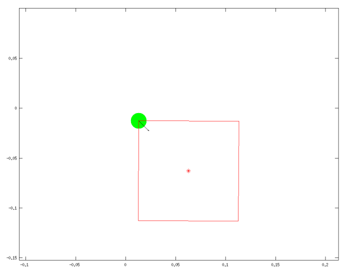

(c) t_2

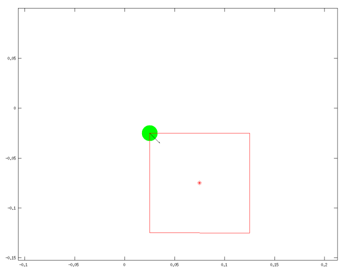

(d) t_3

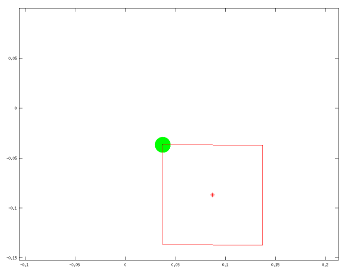

(e) t_4


(f) t_5


(g) t_6


(h) t_7


(i) t_8


(j) t_9

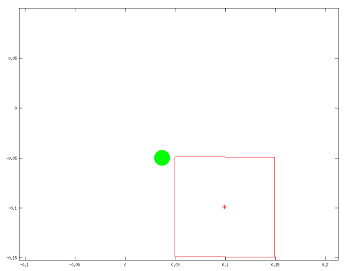

(k) t_{10}

Fig. 4: A push plan found using direct transcription. The green circles denote the pusher, which is volumeless in model, but enlarged here for visualization. The black arrows denote the push force. The red squares denote the slider's pose

ACKNOWLEDGMENTS

I thank Hongkai Dai for his insights in solving the nondifferentiable problem in finding gradient of frictional constraint, and Michael Posa for his guidance in contact-implicit trajectory planning.

REFERENCES

[1] Suresh Goyal. *Planar sliding of a rigid body with dry friction: limit surfaces and dynamics of motion*. PhD thesis, Cornell University, 1989.

[2] Charles R Hargraves and Stephen W Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[3] Kevin M Lynch and Matthew T Mason. Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15(6):533–556, 1996.

[4] Matthew T Mason. Mechanics and planning of manipulator pushing operations. *The International Journal of Robotics Research*, 5(3):53–71, 1986.

[5] MA Peshkin and Arthur C Sanderson. The motion of a pushed, sliding object. part 1. sliding friction. Technical report, DTIC Document, 1985.

[6] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.