Robust Trajectory Optimization by Iteratively Learning Uncertainties

Teerthaa Parakh, Meenal Parakh

***Abstract*—In this project, we learn about robust trajectory optimization and implement a part of the paper by Luo and Hauser [1]. The approach we implement is a trajectory optimization, in which given a geometric path with uncertain model parameters, we use robust optimization and iterative parameter learning to find a dynamically feasible trajectory. The robust trajectory optimization is essentially a time scaling optimization problem that takes in confidence intervals for the uncertain parameters along with the constraints in a given environment, and formulates it as a convex problem. The optimization problem minimizes the time duration and returns a dynamically feasible trajectory which we execute in the environment. Based on the feedback from the executed trajectory, the iterative learning approach updates its estimate on the confidence intervals for the parameters, and then re-optimize the trajectory. In this project, we assume that the only uncertainty in the model parameters comes from uncertain friction coefficients between pair of surfaces. We test our algorithm for the 'waiter' task in Drake[2] in which a robotic arm Kuka IIWA tries to move an object along with it on a specified path.**

I. Introduction

.

Optimizing trajectories in environments such as robotic arm, rugged terrain and so on, is challenging because of the high uncertainties in environment parameters that may affect the optimization. The optimal solution for the trajectory in such scenarios is often the edge case/extreme case which is just on the verge of satisfying constraints. Even a small error in a parameter estimate may leave the trajectory dynamically infeasible.

In our project, we especially consider the case when the only source of uncertainty are the friction coefficients. For some environment, in order to always remain within bounds set by friction coefficients one can try to be as conservative as possible while moving on a given path. But this may be very inefficient and slow. So, we need algorithms that can optimally find a path while taking into account the possible uncertainties in friction coefficients.

We implement the approach given in the paper by Lou and Hauser [1] in which they estimate uncertain parameters through iterative learning, and plan a trajectory based on the estimates. The iterative learning is a series of optimizing for the trajectory and executing them to gather information about the uncertain parameters. This series continues until the trajectories estimated converge or the execution is near optimal to what obtained from the optimization. While the algorithm proposed in the paper is general and applies also to execution errors, which are quite common, we only consider the case of uncertainty in friction coefficients. Also, the paper discusses an efficient way to solve the optimization problem by separating out a linear program from the rest of the convex optimization problem and solving these two parts hierarchically. The linear program stems from approximating the friction cone constraints as multiple linear constraints on the friction force. While we do formulate the friction cone constraint as linear constraints, we do not currently have an hierarchical structure for the optimization problem.

Some assumptions in the algorithm are: (a) the position and velocity error in the trajectory are small and only the second order uncertainties affect the planned trajectory (for the project, we ignore this as well), and (b) no external observations are available, otherwise we could have included that in our iterative learning approach (currently the only feedback that one obtains from the environment is whether a planned trajectory failed or succeeded).

II. Problem Formulation

In this problem, we assume that a twice-differential path is given, that is $q(s) : s \in [0,1] \mapsto \mathbb{R}^n$ is given for for a robot with n DOFs. Our goal is to find time parameterization $s(t) : t \in [0, t_f] \mapsto [0,1]$ and the final time t_f such that path is traversed in minimum time while satisfying a set of constraints. We considered modeling error in coefficient of friction and random noise in joint angle acceleration.

A. Robot Motion Equations

1) ***Objective Function****:* The objective function is the total time required to complete the path and we want to minimize the total time. There are N collocation points, and $\Delta s = \frac{1}{N}$, so the objective function is shown in eq(1).

$$f_{obj}(\dot{S}) = \sum_{i=0}^{N} \frac{2\Delta s}{\dot{s}_i + \dot{s}_{i+1}} \tag{1}$$

2) ***Dynamics Equations****:* For robot, the dynamics equation is

$$M(q)\ddot{q} + C(q, \dot{q} + G(q) = \tau + \sum_i J_i(q)^T f_i \tag{2}$$

Here, $M(q)$ is the mass matrix, $C(q, \dot{q}$ is the centrifugal and Coriolis forces and $G(q)$ is gravity vector, f_i is the contact force at point p_i and there are four contact points between box and slab, $J_i(q)$ is the jacobian of the ith contact point in world frame with respect to joint angles of robot or equivalently it

is the jacobian of ith contact point translational velocity in world frame with respect to joint velocities of robot.
For box the dynamics equations are as follows:

$$M_{obj}\alpha_L = \sum_i f_i + M_{obj}g \tag{3}$$

$$I_{obj}\alpha_A = \sum_i (p_i - c)f_i \tag{4}$$

Here, α_L is the linear acceleration, α_A is the angular acceleration, c is the center of mass of the box, M_{obj} is the mass matrix and I_{obj} is the inertia matrix, such that

$$M_{obj} = m_{obj}I_{3\times3} \qquad I_{obj} = \frac{m_{obj}l_{obj}^2}{6}I_{3\times3} \tag{5}$$

Since the box is placed on the slab which is attached to the robot, so the linear and angular acceleration of the box should also satisfy the below equations.

$$\begin{pmatrix}\alpha_L \\ \alpha_A\end{pmatrix} = J(q)\ddot{q} + \dot{q}^T H(q)\dot{q} \tag{6}$$

Here, $J(q)$ is the jacobian of spatial velocity of the center of mass of box in robot's frame with respect to joint velocities and $H(q)$ is the hessian of spatial acceleration bias of the center of mass of box in robot's frame with respect to joint velocities.

*3) **Constraints**:*

$$\dot{q}_{min} \leq \dot{q}_{exec} \leq \dot{q}_{max}$$

$$\ddot{q}_{min} \leq \ddot{q}_{exec} \leq \ddot{q}_{max}$$

Here $\dot{q}_{exec}$ and $\ddot{q}_{exec}$ are the joint velocities and joint accelerations obtained after execution.

$$\tau_{min} \leq \tau \leq \tau_{max} \tag{7}$$

$$\dot{q}(0) = 0 \qquad \dot{q}(1) = 0 \tag{8}$$

The external disturbances, friction between joints etc. are not considered in the dynamic equations, so due to these, planned joint velocities and accelerations will be different from what we obtain from execution. That is

$$\dot{q}_{exec} - \dot{q}_{plan} \in [\dot{q}_{low}, \dot{q}_{upp}]$$

$$\ddot{q}_{exec} - \ddot{q}_{plan} \in [\ddot{q}_{low}, \ddot{q}_{upp}]$$

We assume that error between planned joint velocities and velocities obtained after execution is negligible. We model the error between planned joint acceleration and acceleration obtained after execution as a normal distribution with U as mean and Δ as standard deviation. That is:

$$(\ddot{q}_{exec} - \ddot{q}_{plan}) \in \mathcal{N}(U, \Delta)$$

Now, we assume that in the eq(9) and eq(10), U and Δ are known and we pick K depending upon how conservative we want our solution to be. The future work will be to estimate U and Δ from the executed trajectories feedback.

$$\ddot{q}_{low} = U - K\Delta \tag{9}$$

$$\ddot{q}_{high} = U + K\Delta \tag{10}$$

Since it is a time-scaling problem, therefore,

$$\dot{q}_{plan} = \frac{dq}{ds}\dot{s} \tag{11}$$

We assumed that $\dot{q}_{low}$ and $\dot{q}_{upp}$ are negligible, therefore

$$\dot{q}_{exec} = \dot{q}_{plan}$$

Hence,

$$\dot{q}_{min} \leq \dot{q}_{plan} \leq \dot{q}_{max} \tag{12}$$

Similarly for joint acceleration,

$$\ddot{q}_{plan} = \frac{d^2q}{ds^2}\dot{s}^2 + \frac{dq}{ds}\ddot{s} \tag{13}$$

And,

$$\ddot{q}_{exec} = \frac{d^2q}{ds^2}\dot{s}^2 + \frac{dq}{ds}\ddot{s} + [\ddot{q}_{low}, \ddot{q}_{upp}]$$

So on planned trajectory we put following constraint,

$$\ddot{q}_{plan} \in [\ddot{q}_{min} + |\ddot{q}_{low}|,\ \ddot{q}_{max} - |\ddot{q}_{upp}|] \tag{14}$$

There are four contact points and at each contact point p_i, we inner approximate a non-linear friction cone by the following linear equations

$$\begin{aligned} |f_i|_t &< \frac{\mu}{\sqrt{2}}|f_i|_n \\ |f_i|_b &< \frac{\mu}{\sqrt{2}}|f_i|_n \end{aligned} \tag{15}$$

Here $\mu = \frac{2*\mu_1\mu_2}{\mu_1+\mu_2}$, μ_1 is the COF of slab and μ_2 is COF of box.

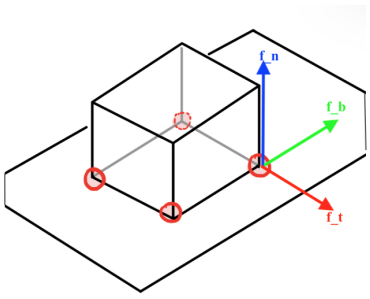

Fig. 1. $(f_i)_t, (f_i)_b, (f_i)_n$ directions and the four contact points

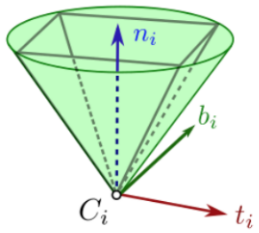

Fig. 2. inner approximation of friction cone

Equation (1) - (15) are the equations for optimization problem. Equation (2) - (15) should be satisfied at all the N collocation points. We want to minimize equation (1) so the parameters that we want to find are $\dot{S}$ and $f_{i,j}$'s where $i \in [1,4]$ and $j \in [1,N]$. So there are total number of $(N + 3 \times 4 \times N)$ parameters.

*4) **Initial Guess**:* Since,

$$\dot{q}_{min} \leq \dot{q} = \frac{dq}{ds}\dot{s} \leq \dot{q}_{max}$$

Therefore following initial guess for $\dot{S}$ is given to the solver,

$$\dot{s}^0 = \min\left(\min_i\left(\left|\frac{q^i_{max}}{\frac{dq}{ds}^i_{max}}\right|\right), \min_i\left(\left|\frac{q^i_{min}}{\frac{dq}{ds}^i_{min}}\right|\right)\right) \quad (16)$$

B. Iterative Learning Algorithm

Iterative Learning improves the confidence intervals for uncertain parameters, in our case the friction coefficients. As discussed in Section I, it is important to find a good estimate for such parameter values, as an optimistic estimate would cause trajectory failure while an underestimate may make the the trajectory inefficient and not optimal.

One of the methods that the paper [1] proposes is using Binary Search on the uncertain parameter, the pseudo-code for which is given in Figure 3.

Algorithm 1 Bisection Search for COF

```
μ_low ← 0, μ_upp ← μ_init
while μ_upp − μ_low > δ do
    μ ← (μ_low + μ_upp)/2
    traj ← RobustOptimize(μ)
    if Execute(traj)= Success then
        traj' ←Speedup(traj, 1 + ϵ )
        if Execute(traj') ≠ Success then return μ_low
        else
            μ_low ← μ
    else
        μ_upp ← μ
return μ_low
```

Fig. 3. Iterative Learning using Bisection Search. (Taken from [1])

The bisection search starts with the upper and lower bounds on the coefficient of friction (COF) μ and optimize the problem formulated above with μ. On executing the trajectory obtained from the optimization, if we find that the object has fallen from the robot's tray, or has shifted (violated the no sliding constraint), then we set the new upper bound μ_{upp} equal to μ. On the other hand, if the trajectory succeeds, μ_{low} is set to μ and we try to execute a "speed-up" version of the trajectory, that is increase the speed by $1 + \epsilon$. If the seed-up trajectory fails, it implies that we are within the ϵ boundary of the optimal trajectory and we can terminate. We can also terminate if the uncertainty has reduced to below threshold. Note that, a low value of ϵ implies that we are more conservative in the speed up, and would require larger number of iterations.

III. Environment and Implementation Details

We used Drake for simulation. The environment has a Kuka arm, attached to this at the top is a slab and a box is kept at the center of slab. The environment set-up is shown in 4

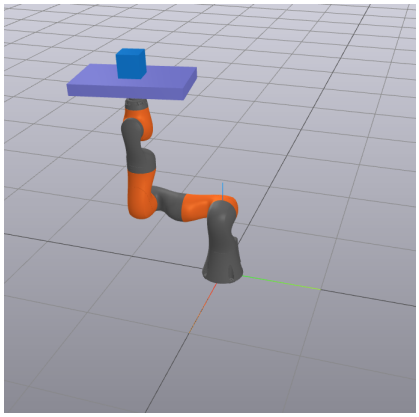

Fig. 4. Environment

We set the following limits:

$$\dot{q}_{min} = -5rad/s \qquad \dot{q}_{max} = 5rad/s$$

$$\ddot{q}_{min} = -5rad/s^2 \qquad \ddot{q}_{max} = 5rad/s^2$$

We assumed $U = 0$ and $\Delta = 0.5$ and $K = 1$, therefore

$$\ddot{q}_{low} = -0.5rad/s^2 \qquad \ddot{q}_{max} = 0.5rad/s^2$$

So,

$$-4.5 \leq \ddot{q}_{plan} \leq 4.5$$

$$M_{slab} = 50gm \qquad m_{obj} = 50gm \qquad l_{obj} = 10cm$$

IV. Results

A. Known Friction Coefficients

We first assumed that the friction coefficient is known and we fixed the number of collocation point to 25. We want the robot to rotate the slab by π rad, while making sure that the box does not slip. The trajectory obtained after optimization is interpolated for 100 nodes. We tested the solution for different values of friction coefficient.

CASE-I: μ = 1
In this setup to move in a circular path, only the first joint KUKA arm is rotating. Therefore planned trajectory of first joint is plotted.

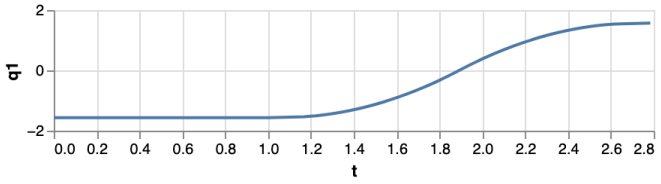

Fig. 5. first joint position vs time for μ = 1

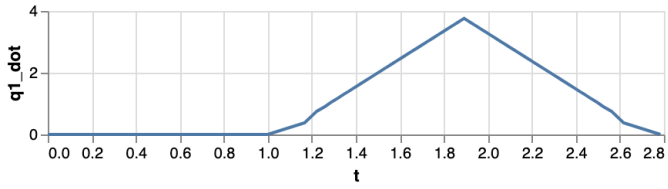

Fig. 6. first joint velocity ($\dot{q}_{plan}$)vs time μ = 1

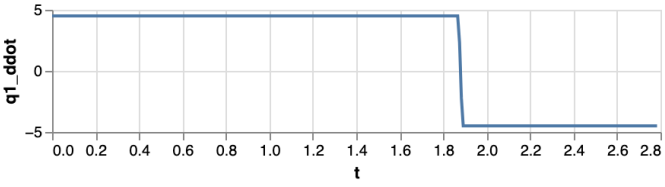

Fig. 7. first joint acceleration ($\ddot{q}_{plan}$)vs time μ = 1

For simulation to get ready, an initial buffer of 1s is given, after 1s the trajectory is executed. The time taken to complete the path is 1.67s. As we can the robot joint moved with maximum acceleration and friction coefficient is high enough to provide necessary force to box so that it does not fall.

CASE-II: μ = 0.5
The time taken to complete the path is 1.72 which is more than CASE-I time. The plots for the planned trajectory are:

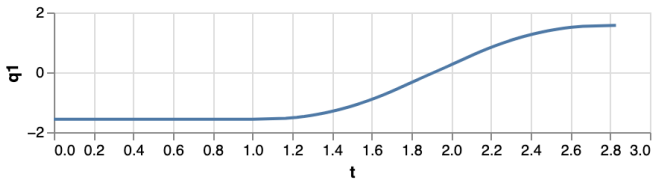

Fig. 8. first joint position vs time μ = 0.5

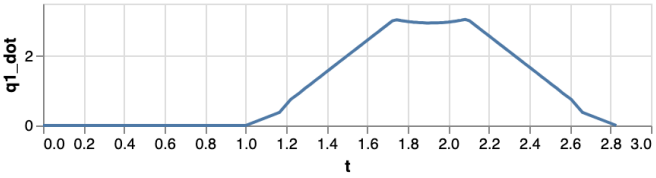

Fig. 9. first joint velocity ($\dot{q}_{plan}$)vs time μ = 0.5

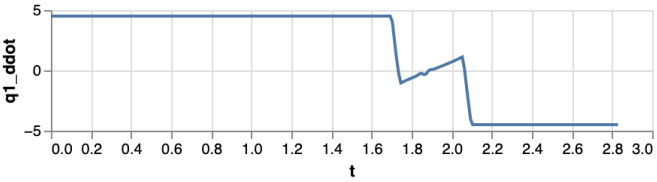

Fig. 10. first joint acceleration ($\ddot{q}_{plan}$)vs time μ = 0.5

If the slab moves with high velocity, then friction force is not enough to support the box because μ is low as compared to CASE-I, so slab moves slowly and from the fig(9) we can see that it is truncated from top.

CASE-III: μ = 0.2
The time required is 2.19s, which is higher than CASE-I&II.

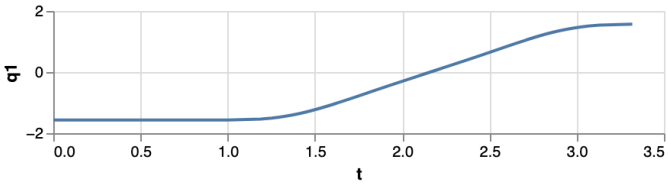

Fig. 11. first joint position vs time μ = 0.2

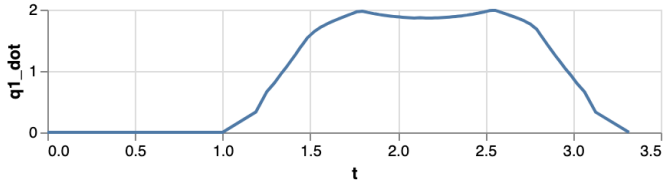

Fig. 12. first joint velocity ($\dot{q}_{plan}$)vs time μ = 0.2

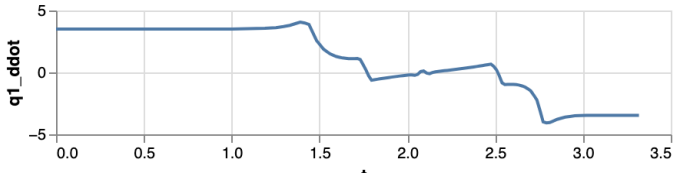

Fig. 13. first joint acceleration ($\ddot{q}_{plan}$)vs time μ = 0.2

Since the number of parameters to find and constraints to satisfy are directly proportional to number of collocation points, the following plot is obtained. For N = 100, time taken is 40s, therefore for bigger problems, this method will be inefficient. To tackle this issue [1] presented a hierarchical optimization method, which divides the problem into N sub-problems and solve them independently. In each subproblem there are $3m$ parameters that are to be found where m is the number of contact points and since these are independent sub-problems, they can be solved in parallel, thus making this method more efficient.

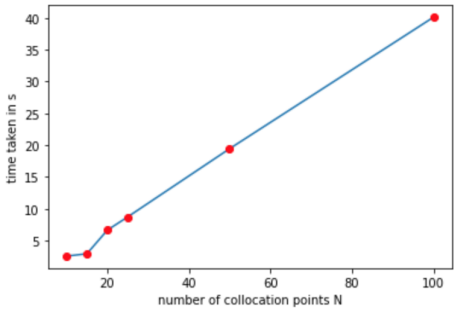

Fig. 14. Number of collocation points vs time taken (in s)

B. Iterative Algorithm for estimating friction coefficient

In this part we estimated friction coefficient using binary search method. We took the actual μ to be 0.1. The speed-up factor is (1.05), it is the factor by which $\dot{s}$ gets increased, hence $\ddot{s}$ is increased by a factor of 1.05^2. Initial μ_{low} = 0.01 and μ_{high} = 1. The threshold is 0.01, that is the algorithm will stop when $(\mu_{high} - \mu_{low})$ is less than 0.01.

Iter	μ_{avg}	Time(s) for trajectory completion	Success	Attempted Speedup	μ_{low}	μ_{upp}	$\mu_{upp} - \mu_{low}$
1	0.525	1.656	FALSE		0.05	1	0.95
2	0.288	1.923	FALSE		0.05	0.525	0.475
3	0.169	2.448	FALSE		0.05	0.288	0.238
4	0.109	3.006	FALSE		0.05	0.169	0.119
5	0.08	3.509	TRUE	TRUE	0.05	0.109	0.059
6	0.095	3.227	TRUE	TRUE	0.08	0.109	0.029
7	0.102	3.11	FALSE		0.095	0.109	0.014
Loop Exited					0.095	0.102	0.007

Fig. 15. Coefficient of Friction Binary search

Using binary search, the coefficient of friction estimated is 0.0945, which is lower than the actual value. This can be due to simulation setup and assumptions made like the box at the start of simulation is always at the center of slab and with these assumptions the jacobians are calculated. Also we inner-approximated friction cone to get linear constraints. Also note that the estimated friction coefficient is a conservative estimate.

V. Conclusion and Future Work

In this project the objective was to make a robot traverse a given path in minimum time such that the box does not slip. This problem was formulated as time-scaling convex optimization problem which was solved and simulated in Drake. To estimate coefficient of friction, an iterative learning method based on binary search is implemented. which takes feedback from the executed trajectory. The relative error between estimated and true COF was 5.5%. A future work would be to compare the planned trajectory with the trajectory obtained from Drake after simulation. Also in this formulation, it was assumed that the distribution of error between planned and executed joint acceleration is normal and for that the mean and standard deviation are already known, so another future work is to implement Iterative Learning with Robust Optimization and Binary Search [1] method.

VI. Acknowledgement

I would like to thank Meenal Parakh for helping me in the Drake setup.

References

[1] Luo, J., Hauser, K. (2017, March). Robust trajectory optimization under frictional contact with iterative learning. Autonomous Robots, 1447–1461. https://doi.org/10.1007/s10514-017-9629-x

[2] Drake https://drake.mit.edu/doxygen_cxx/modules.html