Analysis of Robust Trajectory Optimization with Ellipsoidal Disturbances

Becky Abramowitz
University of Pennsylvania
Philadelphia, PA
rabram@seas.upenn.edu

Thomas Mulroy
University of Pennsylvania
Philadelphia, PA
mulroyt@seas.upenn.edu

Abstract—**This document analyzes the DIRTREL approach to robust trajectory optimization. DIRTREL (Direct Transcription with Ellipsoidal Disturbances) was proposed by Zachary Manchester and Scott Kuindersma at Harvard University in 2017 and uses a direct transcription approach with an LQR controller and an ellipsoidal model of disturbances. Specifically, this paper looks at the robustness of the DIRTREL method, the time required for algorithm computation, and the trackability of the resulting controller. The paper also discusses the proceedings that led to these findings, the effects of the direct transcription technique on the DIRTREL output, and some suggestions for future work in the domain.**

I. Introduction

Classical trajectory optimization approaches rely on an absolute understanding of system state as well as an accurate depiction of the dynamics and environmental factors that interact with the system. However, most real world systems are inherently non-ideal; erroneous friction coefficients will change system dynamics, unmodeled wiring can alter mass distribution, and wind or air-flows can apply external forces. These factors often cannot be explicitly predicted or accounted for and make it nearly impossible for a real system to behave exactly as planned. This discrepancy underpins the notion of robust control - control systems must be designed to work in cases beyond the ideal. These variations are often predictable and can be approximated within some distribution.

The most common way to handle these discrepancies is a stochastic representation of state, usually with disturbances modeled as a Gaussian Distribution. These variations propagate through time and are usually assumed to increase in a Bayesian manner, such as in an Partially Observable Markov Decision Process. Some previous work exists on the development of trajectories for systems with this noise, but the space is relatively new and far from over-saturated [3].

DIRTREL, a "Robust Trajectory Optimization with Ellipsoidal Disturbances and LQR Feedback" is a recent proposition (2017) from Harvard University that suggests adding disturbances, modeled as ellipsoids, to a direct transcription, or DIRTRAN approach [6]. Direct transcription is a method used in trajectory generation where the control input is defined by a zero-order hold between steps. These steps, known as knot points, also define the state function and must be placed frequently throughout the desired trajectory. A graphic illustrating the direct transcription approach is shown in Fig. 1. Direct transcription is often also accompanied by a first-order Euler integration on the dynamics, which can leave some artifacts on the system state [4].

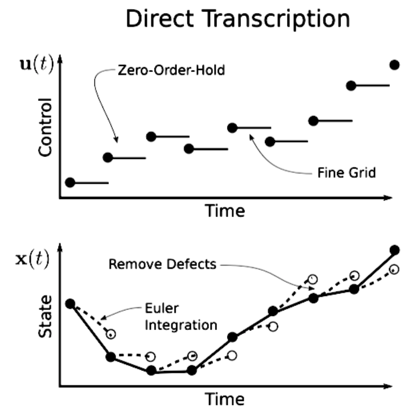

Fig. 1. Direct Transcription [4]

The authors of the DIRTREL paper use ellipsoids to characterize disturbances since they can be easily modeled in a matrix and are synonymous with a bounded Gaussian distribution. The DIRTREL model incorporates these ellipsoids into its LQR-based cost function as well as its constraints over a space of these deviations. These disturbances also propagate over time and can be defined with respect to both magnitude and initial condition. DIRTREL is defined by the nonlinear optimization problem in (1) with a Robust Cost Function as given in Algorithm I. The Robust Cost Function is a take on a shooting method, and traverses the set of knot points four times, performing matrix algebra at each step. The first pass computes the gradients of the f_h function defined in (1), the second pass computes the P matrices for use in the third pass which computes the closed-form solution K to the time-varying LQR problem. The fourth pass updates the model of the ellipsoidal disturbances and determines the effect that these disturbances have on the objective cost. The DIRTREL method thus propagates and incorporates the disturbances in a time-varying manner and is a reasonable candidate for computing robust trajectories that account for unmodeled deviations in state.

$$
\begin{aligned}
&\underset{x_{1:N},u_{1:N-1},h}{\text{minimize}} \; l_W(x_{1:N}, u_{1:N-1}) + g_N(x_N) + \sum_{i=1}^{N-1} g(x_i, u_i) \\
&\text{subject to} \\
&x_{i+1} = f_h(x_i, u_i) = x_i + f(x_i, u_i)h \;\forall i = 1:N-1 \\
&x_i \in X \;\forall i = 1:N \\
&u_i \in U \;\forall i = 1:N-1 \\
&u_i^{\mathcal{W}} = u_i \pm col((K_i E_i K_i^T)^{1/2}) \in U \;\forall i = 1:N-1 \\
&x_i^{\mathcal{W}} = x_i \pm col(E_i^{1/2}) \in X \;\forall i = 1:N \\
&h_{min} \leq h \leq h_{max}
\end{aligned} \tag{1}
$$

Algorithm 1 DIRTREL ROBUST COST FUNCTION

```
Input: $x_{1:N}$ (state at all N knot points), $u_{1:N-1}$ (control at first N-1 knot points), $D$ (magnitude of disturbances squared), $E_1$ (initial disturbance ellipsoids), $Q^l$ (state cost for use in function), $R^l$ (control cost for use in function), $Q_N^l$ (cost-to-go for final state in function), $Q$ (state cost), $R$ (control cost)
Output: $l_w$ (robust cost value)
    Compute Partials
 1: for $i = 1...N-1$ do
 2:     $A_i \leftarrow \partial_{x=x_i} f_h(x,u,w)$
 3:     $B_i \leftarrow \partial_{u=u_i} f_h(x,u,w)$
 4:     $G_i \leftarrow \partial_{w=0} f_h(x,u,w)$
 5: end for
    Perform TVLQR Update
 6: for $i = N...1$ do
 7:     $P_{i-1} = Q + A_i^T P_i A_i -$
        $A_i^T P_i B_i (R + B_i^T P_i B_i)^{-1} (B_i^T P_i A_i)$
 8: end for
 9: for i=1...$N-1$ do
10:     $K_i = (R + B_i^T P_{i+1} B_i)^{-1} (B_i^T P_{i+1} A_i)$
11: end for
    Propagate Disturbances
12: $H-1 \leftarrow 0$
13: for i=1...$N$ do
14:     $l \leftarrow l + Tr((Q^l + K_i^T R^l K_i) E_i)$
15:     $E_{i+1} \leftarrow (A - I - B_i K_i) E_i (A_i - B_i K_i)^T$
        $+ (A_i - B_i K_i) H_i G_i^T$
        $+ G_i H_i^T (A_i - B_i K_i)^T + G_i D G_i^T$
16:     $H_{i+1} = (A_i - B_i K_i) H_i + G_i D$
17: end for
18: $l \leftarrow l + Tr(Q_N^l E_N)$
19: return $l$
```

This paper expands on the work done by Zachary Manchester and Scott Kuindersma in the DIRTREL paper by reproducing results on the cart-pole and pendulum systems as well as including analysis on the algorithm's robustness, speed, and trackability. These analyses are performed by assessing the outputs from DIRTREL on both of the aforementioned systems compared to the outputs from systems with Gaussian noise, a quantified assessment of system run times for both configurations, and a discussion of optimizing and tracking direct transcription based approaches. The work done in this paper was performed in MATLAB using the SNOPT non-linear optimization tool.

II. System Overview

This paper utilizes two dynamical models: a pendulum with unmodeled disturbance in mass, and a cart-pole with unmodeled friction. The dynamics of each of these systems are as follows.

A. Pendulum

The pendulum model used in this simulation is treated as a point mass tethered to a rotating joint; the mass of the pendulum is unknown. As shown in Fig. 2, the system has one degree of freedom in rotation around the joint (θ) and one control input torque (τ) on this joint. The pendulum is governed by (2) where w is the disturbance to the mass.

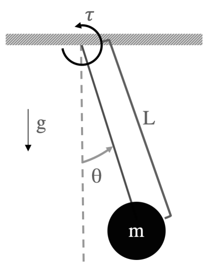

Fig. 2. Pendulum Model Parameters

$$(m+w)L^2\ddot{\theta} + (m+w)gLsin(\theta) = \tau \tag{2}$$

The state of the system is defined to be $x = [\theta, \dot{\theta}]'$ and the control defined as $u = \tau$. The derivative of the state, $\dot{x}$, is defined by (3).

$$
\begin{gathered}
\dot{\theta} = \dot{\theta} \\
\ddot{\theta} = -\frac{g}{L}sin(\theta) + \frac{u}{mL^2}
\end{gathered} \tag{3}
$$

Unless otherwise noted, $Q = Q^l = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix}, Q_N^l = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$, $R = R^l = 0.1$, $m = 1$, $L = 1$, E_1 is zeros, a disturbed system has a value of $D = 0.01$ corresponding to variation of ± 0.1, and all units are SI. The state starts at $\theta = 0, \dot{\theta} = 0$ and ends at $\theta = \pi, \dot{\theta} = 0$.

B. Cart Pole

The cart pole model used in this simulation is said to have an unknown friction value, a reasonable assumption since friction coefficients can be hard to model and a range can encompass both static and dynamic friction values. As shown in Fig. 3, the cart is said to have a mass of m_c, and a point mass at the end of the pole is said to have a mass of m_p. The cart is

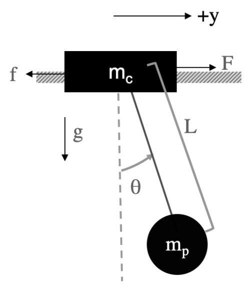

Fig. 3. Cart Pole Model Parameters

governed by (4) where w is the friction value. These equations were derived with reference to [1].

$$\begin{aligned} S = sign(\dot{y}[(m_p+m_c)g+m_p(L\ddot{\theta}sin\theta+L\dot{\theta}^2cos\theta)]) \\ w = S\mu_c[(m_p+m_c)g+m_p(L\ddot{\theta}sin\theta+L\dot{\theta}^2cos\theta)] \\ \tau - w - m_pL(\ddot{\theta}cos\theta-\dot{\theta}^2sin\theta) = (m_c+m_p)\ddot{y} \\ -gsin\theta = L\ddot{\theta}+cos\theta\ddot{y} \end{aligned} \tag{4}$$

The state of the system is defined to be $x = [\theta, \dot{\theta}, y, \dot{y}]'$ and the control defined as $u = F$ applied to the cart as shown in Fig. 3. The derivative of the state, $\dot{x}$ is defined by (5).

$$\begin{aligned} \dot{\theta} = \dot{\theta} \\ \ddot{\theta} = -\frac{(m_p+m_c)gsin\theta + cos\theta(F-w+m_pL\dot{\theta}^2sin\theta)}{L(m_p+m_c)-m_pLcos^2\theta} \\ \dot{y} = \dot{y} \\ \ddot{y} = \frac{F-w-m_pL(\ddot{\theta}cos\theta-\dot{\theta}^2sin\theta)}{m_p+m_c} \end{aligned} \tag{5}$$

The state of the system is defined to be $x = [\theta, \dot{\theta}, y, \dot{y}]'$ and the control defined as $u = \tau$ applied to the cart. Unless otherwise noted, $Q = Q^l = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $Q_N^l = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$, $R = R^l = 1$, $m = 1$, $L = 1$, E_1 is zeros, a system with friction has $D = 1$ corresponding with $f = \pm 1$ and all units are SI. The state starts at $\theta = 0, \dot{\theta} = 0$, $y = 0$, $\dot{y} = 0$ and ends at $\theta = \pi, \dot{\theta} = 0$, $\dot{y} = 0$.

III. Robustness Analysis

In order to test the robustness of this algorithm, Gaussian noise is introduced into the system. The general approach taken is to run the trajectory optimization with the disturbance parametrization matrix D. Then, the outputs of the state and the control inputs are used as the desired trajectory x_d and u_d to run in a closed loop LQR controller (as discussed later). The real, continuous dynamics are used (as opposed to the discrete time, linearized dynamics that are called for in direct transcription in DIRTREL) and are simulated forward in time with ode45 in MATLAB (Note that ode45 is still a numerical solution method. However, it uses a 4th order Runge-Kutta variable time-step method and is therefore a very good approximation for the continuous solution). Noise is also introduced into the simulated dynamics to test the robustness and usefulness of the DIRTREL output. The method by which noise is introduced varies depending on the physical interpretation of the ellipsoidal disturbances introduced in DIRTREL.

A. Pendulum with disturbances in mass

In the case of the pendulum, the noise matrix D is introduced with the physical interpretation of being a unknown pendulum mass. Therefore to test the robustness of the pendulum swing up trajectory, we sampled the mass from a normal distribution centered on the nominal mass used in the trajectory optimization and a standard deviation of $\sigma = 0.3$ as D represents variations with a hard bound of ± 0.1. We then ran the continuous time simulation with the LQR tracking controller with a second LQR controller that stabilizes around the upright position of $\theta = \pi$. The cost matrices for the LQR tracking controller and for the LQR stabilizing controller, denoted Q and R respectively, are as follows.

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \; R = 0.01$$

We ran this simulation in a loop with a different mass each time. The results of this were optimistic in that we had very few failures. Specifically, we ran this 20 times with different masses. There was one failure to reach the upright (as defined by the final position being greater than 0.1 radians from π). The average deviation from the upright was 0.0129. The average max control input for any given sequence was 3730 Nm. Clearly, the control inputs went significantly out of the bounds originally set in DIRTREL. If we were to set a limit on control authority, the success rate would no doubt be much lower considering the amount of control authority used simply to track the nominal trajectory without any deviation from the nominal mass. It seems that the LQR controller compensates for a bad nominal trajectory by significantly increasing the control authority. This happens even when the mass is set to the nominal mass. This can be seen in the plots of the outputs of DIRTREL in Fig. 4, the outputs of an LQR tracking controller with the nominal mass in Fig. 5 and the outputs of the LQR tracking controller with a mass of $m_{nominal}+0.57$ in Fig. 6. Unfortunately, we did not have time to fully implement a direct collocation pendulum swing up model and run it with a mass perturbed by some amount to compare these results to. However we did experiment with some simulations using direct collocation with code modified from homework 5 where the max control asserted for a pendulum with the nominal mass is about 6Nm. This drastic difference demonstrates the shortcomings of the open loop trajectory from DIRTREL. It is also very interesting to note that with the nominal mass this direct collocation implementation of swing up is possible with

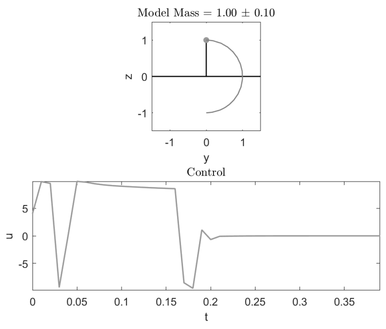

Fig. 4. Robust Pendulum Tracking

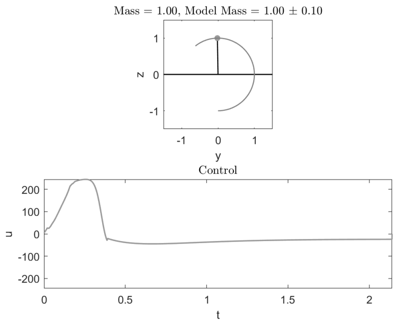

Fig. 5. LQR tracking with nominal mass

just open loop control, while the DIRTREL needed a closed loop LQR tracked to reach the top.

IV. Speed Analysis

The largest challenge encountered in this project is that the optimization problem, as currently implemented in MATLAB using SNOPT, takes nontrivial time. The DIRTREL paper states that DIRTREL algorithm performs two to four times slower than a direct transcription approach without ellipsoidal disturbances. [6]. Table I shows the results from this paper (as averaged over two iterations). These results show DIRTREL computation times on the order of two to four times those of DIRTRAN for lower numbers of knot points and greater latency with more knot points. Table I also conveys the magnitude of the speed of the algorithm; DIRTREL runs on the order of minutes with some cases at higher numbers of knot points taking on the order of hours. It is worth noting that these results were found on a 64-bit computer with an Intel i7 processor and a 4-core CPU at 3GHz. Times on other compute systems will vary, so values are intended to show magnitude and relative sizes.

TABLE I
DIRTREL Computation Time (sec)

Model	**N**	**DIRTRAN**	**DIRTREL**
Pendulum	40	7.0	23.7
Pendulum	50	16.4	66.9
Pendulum	60	25.2	506.8
Cart Pole	40	50.9	343.4
Cart Pole	50	87.3	369.2
Cart Pole	60	217.4	1157.1

The following section discusses the rationale behind this latency by looking at how the time is being spent over varying numbers of iterations and over varying numbers of knot points. There is a following discussion on numerical differentiation in attempting to speed up the solver.

A. Hypothesis and Intuition

From Table I and the previous discussion of DIRTRAN versus DIRTREL run times, it is apparent that a significant amount of computational time is incurred by the ellipsoidal disturbances in the Robust Cost Function. As shown in Equation 1, the E and K matrices produced by the Robust Cost Function also inform the values of $x_i^{\mathcal{W}}$ and $u_i^{\mathcal{W}}$ which are computed in every SNOPT iteration to inform the constraints. The writing of the cost function and the constraints occurs in what SNOPT handles as a single function (*userfun* in SNOPT documentation) which calls the Robust Cost Function above. The userfun also handles the cost function, computation of the dynamics constraints, and the gradients (discussed later). Given that a nontrivial amount of time is spent calculating ellipsoidal disturbances in the Robust Cost Function, it follows that a significant percentage of SNOPT computation time is spent in this userfun. This hypothesis is shown to be true below. (It should be noted that the results below were found with the MATLAB profiling tool, which increases run time to around 130% of the actual value, but we assume that the overhead is distributed evenly and thus normalizing by total time mitigates these effects.)

B. Number of Iterations

To assess the time spent on the DIRTREL problem, it is imperative to begin with a comparison on lower numbers of iterations. Although these systems do not converge, they offer more insight into the calls computed by the SNOPT optimization program. Fig. 7 shows the relative time spent on the cost function and related steps by our DIRTREL implementation. From Fig. 7, it is evident that the majority of computation time in the DIRTREL algorithm is spent on userfun and computing the cost function l_w. This adds

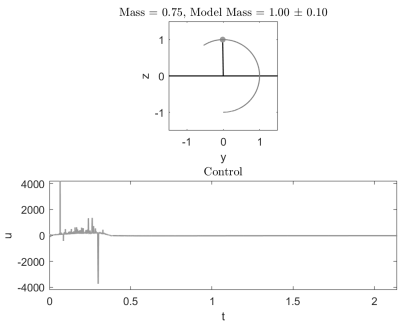

Fig. 6. LQR tracking with $m = m_{nominal}0.25$

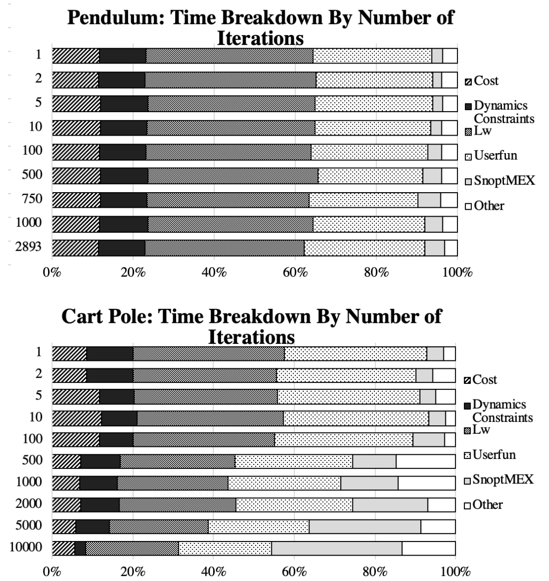

Fig. 7. DIRTREL Time Breakdown with Varying Iterations

latency that otherwise would not be present in a trajectory optimization algorithm. It is also interesting to note that the pendulum has similar time breakdowns from the first iteration through convergence to an optimal solution, while the cart pole system spends an increasing amount of time in the SNOPT MEX function which provides the SNOPT bindings to MATLAB. Due to the nature of the MEX file, it cannot be further analyzed directly. However, increasing time in the MEX function is directly correlated with an increasing number of iterations, implying that the MEX function time is likely spent in computation towards the optimization. Distributing these times by the number of function calls, one can assess the time used by each function call, as shown in Table II.

TABLE II
AVERAGE FUNCTION CALL COMPUTATION TIME (MS)

Model	**Userfun**	l_w	f_h	**Cost**
Pendulum	2.24	2.20	0.010	0.010
Cart Pole	3.12	3.08	0.010	0.011

It is evident from Table II that the time to compute the dynamics constraints (f_h) and the cost is the same between function calls, but the time to compute l_w increases significantly with the complexity of the system, and thus more complex systems and more calls to the l_w function will decrease speed of the DIRTREL algorithm.

With this analysis, it is important to look at the number of calls to each function in the DIRTREL algorithm. As would be expected, experimental results showed that cost function and the function generating dynamics constraints are called an equal number of times, specifically $N-1$ times the number of calls to *userfun*. Likewise, the *userfun* is called the same number of times as the Robust Cost Function, which makes sense since the Robust Cost Function is called once per iteration of *userfun*. More interesting is the relationship between the number of iterations and the number of calls to *userfun* which is shown in Fig. 8. The number of calls

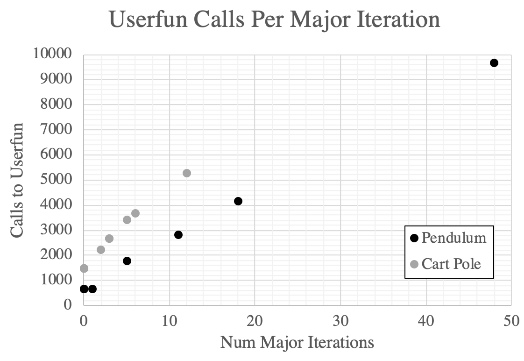

Fig. 8. Calls to Userfun for Pendulum and Cart Pole

is dependent solely on the number of major iterations - not the number of total iterations - and seems to scale linearly with that variation. There is some overhead number of calls with zero major iterations, which are likely derived from the computation that leads to the first iteration as well as the automatic calculation of the gradients.

C. Number of Knot Points

One of the nuances of direct transcription is that the system requires more knot points to reach a desired state than other approaches. For example, given the cost and constraints defined prior, the pendulum needs 13 knot points without ellipsoidal disturbances and 39 knot points with ellipsoidal disturbances to yield a solution. The runtime of the Robust Cost Function is $\mathcal{O}(N)$ and that function is called at least N times per iteration, meaning that the total runtime is polynomial in N. Therefore, the time needed scales dramatically with an increase of knot points. Though the experimental results, as shown in Fig. 9, vary from this ideal as the optimization solver does not have a linear relationship between number of iterations and problem size when reaching convergence, the overall trend can be seen with the data. It takes significantly longer to run both more complex systems and systems with larger numbers of knot points.

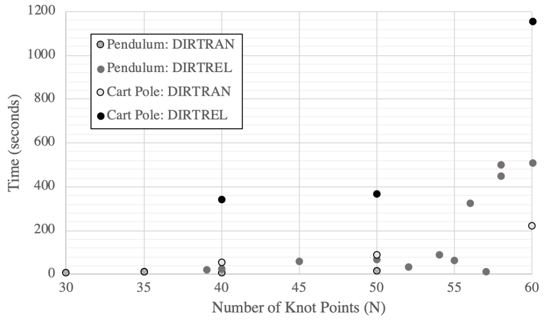

Fig. 9. DIRTREL Run Time with Knot Points

D. Numerical Differentiation

Implementing the gradients into the SNOPT infrastructure is non-trivial, as the robust cost function and therefore a significant number of the constraints are not easily differentiable. Therefore, a cursory gradient implementation provided derivatives only for the easily differentiable dynamics constraints. These derivatives did not yield significant improvement in runtime, likely since there are only $(N-1)n_x$ dynamics constraints (where n_x is the dimension of state) compared to $2n_x^2$ ellipsoidal constraints on state and $2n_u^2$ ellipsoidal constraints on control (where n_u is the dimension of control). The dynamics constraints depend on at most $n_u + 2n_x$ state values while, due to the backwards and forwards passes in the robust cost function, the other constraints could depend on all possible states.

A more careful implementation differentiates the Robust Cost Function and thus the resulting state and control constraints via a line-by-line numerical differentiation. However, with even only a few derivatives, the optimizer to requires a higher convergence threshold to converge to a viable trajectory. It is notable that Manchester and Kuindersma [7] significantly lowered the threshold to 10^{-3} from 10^{-6} [2] for convergence in their implementation. While the results may be marginally faster with this constraint, they are inherently less optimal.

It is possible that further optimization within MATLAB on the cost function could further reduce the time, but cursory experimentation on data structures (cell arrays versus 3D arrays) yielded minimal results. Additional improvements on the run time could be made, as proposed in the paper by running the SNOPT calls in C++.

V. Trajectory Tracking Analysis

The goal of DIRTREL is to produce a feasible trajectory, and the original DIRTREL paper looked at this output directly [6]. However, a feasible trajectory from an optimizer with assumptions does not necessarily imply that the trajectory will work in the real world. This section discusses two trackers: an open loop tracker that follows the control output from DIRTREL and a closed-loop LQR tracker that interpolates state and control values gathered from DIRTREL along with the system's deviations thereof to apply control.

A. Open Loop Tracker

Given that DIRTREL outputs both the states and control inputs at each of the knot points, the first approach to applying DIRTREL is an open loop controller. This system uses the control outputs from DIRTREL to control a system with real dynamics - in the case of the cart pole, this means including the friction force. However, in both systems, the OL system is unable to command the system to the desired result as shown in Fig. 10 and 11. These results were consistent both with and without disturbances added to the system - meaning that a standard direct transcription approach without ellipsoidal disturbances is also not able to be tracked with an open loop controller.

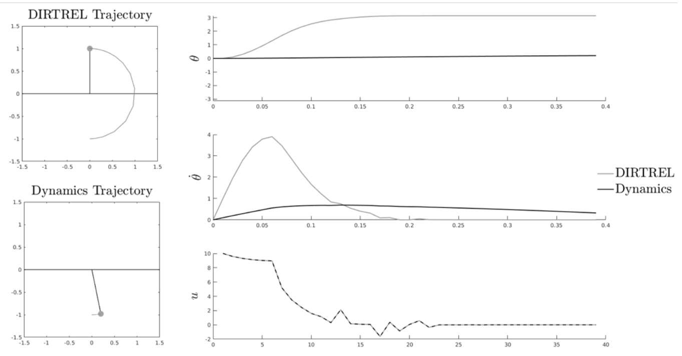

Fig. 10. Pendulum with OL Tracker

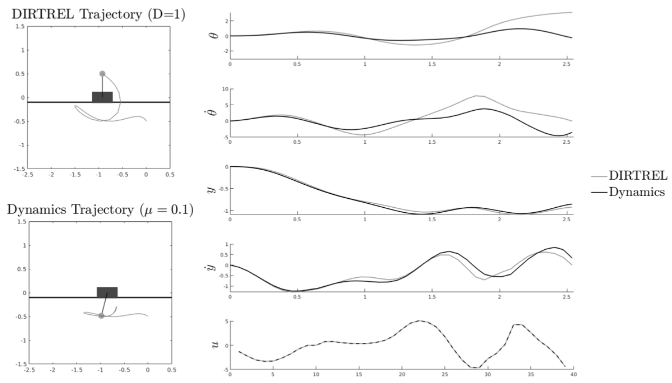

Fig. 11. Cart Pole with OL Tracker

B. Closed Loop LQR Tracker

Since an OL system is unable to get the cart pole or the pendulum to the desired state, a closed loop LQR tracker is the next step. The tracker runs a closed loop LQR controller on the error coordinate between the state of the system and the desired state, as defined by a linear interpolation of the DIRTREL output. The LQR solver is computed within an ODE45 call in MATLAB. There is a secondary LQR controller that handles the final phase of reaching the desired end state. Unlike the OL tracker, this system is able to bring the system to a desired state, as shown in Fig. 12. However, it is notable

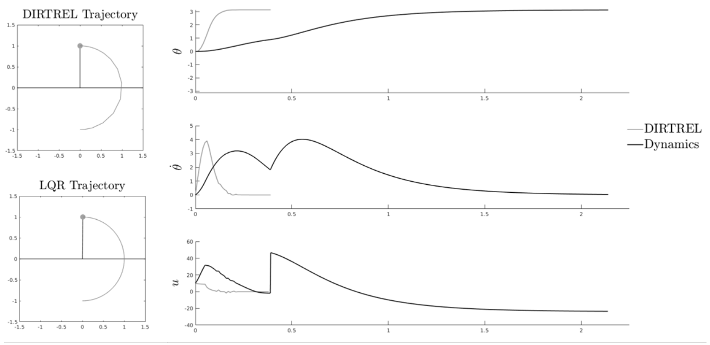

Fig. 12. Pendulum with LQR Tracker

that the trackers do not exactly match the system dynamics and require considerably more control input than the control input designed for by the DIRTREL algorithm. While the control input can be altered somewhat by varying the control cost (R) in the LQR tracker, it is not possible to reach orders of magnitude similar to the DIRTREL controller with this tracker while also driving the pendulum to its final state. This discrepancy implies that the path defined by DIRTREL is not dynamically feasible - it should not require an order of ten to a hundred times more control authority to reach the state if the control path were feasible. It is also notable that this variation is true for systems both with and without the ellipsoidal disturbances, again implying that the issue lies with the underlying direct transcription approach rather than the ellipsoidal wrapper provided by DIRTREL.

VI. Discussion

The following section highlights some higher level thoughts on the project as well as opportunities to further research moving forward.

A. Direct Transcription

Perhaps the biggest takeaway from this analysis are the inherent problems with direct transcription - particularly the first order integration method and the zero order hold on the control inputs.

A zero order hold on control input is implicitly not physically realizable; for a real system, there is some smooth transition between the various control states, and the control input should thus be a continuous function. One ramification of this is that the derivatives, as discussed in the Speed Analysis section of this paper, should be continuous to ensure that they are meaningful, else there are infinite derivatives at various locations. It is probable that this discrepancy between a continuous control function (specified by the finite control derivative values) and the discontinuous control function (resulting from direct transcription) yields an unrealistic model. This could be the reason that a system modeled with gradients requires a higher convergence threshold to get a result from the SNOPT optimizer. These resulting errors are large and thereby reduce the physical realizablility and tracking of the system. This makes any direct transcription based approach far less useful for applications on a real system.

Another ramification of direct transcription is the zero-order hold on the control inputs requires more knot points than a control schema such as direct collocation which varies the control input linearly. The need for an increased number of knot points is further exacerbated as the DIRTREL algorithm uses the time step as a single decision variable that must be the same for all steps. As there are a predefined number of knot points, the time steps cannot be infinitely small, and a more dramatic control input must be held for the entire time step period. This constant step period with constant control makes it harder for the system to converge to an optimal solution thereby requiring extra computation time from the optimization solver.

As mentioned in the prior section, tracking a direct transcription based approach is inherently hard. This is because the direct transcription based method, like other first order Euler methods, is subject to $\mathcal{O}(h)$ error where h is the time step value. For contrast, direct collocation is subject to $\mathcal{O}(h^3)$ error. Since h is less than 1, this cubic error is a significant improvement. The error associated with direct transcription causes general issues for trajectory tracking, especially with underactuated systems [5].

The DIRTREL paper mentions that a midpoint integration scheme can be used in lieu of the first order Euler integration scheme mentioned in the paper. This was implemented by the authors [7] and should provide higher accuracy and reduce run times as it would eliminate some of the problems incurred with the direct transcription approach.

B. Struggles

It would be an injustice to this paper to go without mentioning our many false starts and struggles. From the beginning, we had trouble downloading SNOPT onto our computers and were ultimately unable to get it working on a Mac. Once we had SNOPT installed, we struggled on some initial tests to figure out the SNOPT problem structure and how to define a user function for input to the solver.

Our understanding of, and thus ability to quickly implement, DIRTREL also had its share of problems. After our first few reads, we did not understand that the Robust Cost Function simultaneously computes a trajectory, and we spent considerable time trying to figure out how to pass a nominal trajectory into DIRTREL. We also struggled to implement the time varying LQR, initially trying to solve it with an ODE45 call as we had done in a homework.

Once we had a working implementation of DIRTREL with the pendulum, we got excited about complex dynamics and spent some time working through the equations of motion for a 7-DoF bicycle system. However, the time required to compute our ”intermediate” cart-pole system quickly extinguished this idea.

The continuing time requirements for running the DIRTREL algorithm on the cart pole system continued to be one of our biggest struggles throughout. Given more time (mostly computation time), we would have liked to further flesh out our trackers and robustness analysis on the more complex system.

C. Future Work

Optimal robust trajectory generation is still a relatively new field, and there are many improvements that can be made, both to the DIRTREL algorithm and within the space as a whole. Within DIRTREL, it could be potentially highly beneficial to replace the direct transcription approach with direct collocation, which would help with many of the problems listed above. A higher order integration scheme could be similarly useful.

Given that DIRTREL is designed to handle robustness, it is not far fetched to add a LQG controller to the algorithm to account for missing state feedback or unknowns in the control input. This addition would further increase the robustness of the system.

Given a system with more cores and higher processing power, it would be interesting to run the DIRTREL algorithm on a higher order system with more degrees of freedom and control inputs and to assess the above criterium on such a system.

Lastly, it would be interesting to see the effects of implementing the algorithm in C++, as the paper briefly mentioned, to see if it speeds up the optimization. This could also be accomplished by creating MEX files that would run the slower user function within MATLAB and see what the ramifications on speed would be.

As more algorithms are capable of solving these robust trajectory optimization problems, it would be interesting to compare them with DIRTREL and be able to perform more of a comparative assessment.

For reference, and to be a starting point for any future work, our code is available at the following link: https://github.com/rabramowitz/probabilistic_traj_opt.

Acknowledgment

We would like to thank the MEAM 517 course staff, Dr Michael Posa and Matt Halm, for their continued help and support on this project.

References

[1] Razvan V. Florian. Correct equations for the dynamics of the cart-pole system. 2005.

[2] Philip E. Gill, Walter Murray, and Michael A. Saunders. Users guide for snopt version 7: Software for large-scale nonlinear programming.

[3] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, May 1998.

[4] Matthew P. Kelly. Transcription methods for trajectory optimization: a beginners tutorial. *arXiv:1707.00284 [math.OC]*, 2017.

[5] Scott Kuindersma. Lecture notes optimizing robot motions through contact, July 2016.

[6] Zachary Manchester; Scott Kuindersma. Dirtrel: Robust trajectory optimization with ellipsoidal disturbances and lqr feedback. *2017 Robotics: Science and Systems (RSS)*, 2017.

[7] Zachary Manchester; Scott Kuindersma. Harvard agile robotics lab. https://github.com/HarvardAgileRoboticsLab/drake/blob/dirtrel/drake/matlab/solvers/trajectoryOptimization, 2017.