EN.525.611: Modern Convex Optimization
Assignment 13 (40pts)
Trajectory Optimization of Linearized Cartpole Dynamics

Jonathan Gill

1 Introduction

The goal of trajectory optimization is to figure out what control input should be applied to a system in a *single instantaneous* state in order to achieve some desired final state subject to constraints and a desire to minimize some metric. Contrast this with a *control law*, where the goal is to determine an offline policy to control a dynamical system for *any* configuration that it might be in.

This project uses convex optimization techniques to control the linearized dynamics of an underactuated cartpole system to achieve a user-specified final system state with minimal control effort, which is a flavor of trajectory optimization. Generic trajectory optimization can be written as a nonlinear program (NLP):

$$\begin{aligned} \min_{z_{[0:N]}, u_{[0:N]}} \quad & f(z_{[0:N]}, u_{[0:N]}) \\ \text{s.t.} \quad & z_0 = z_{\text{init}} \\ & z_N = z_{\text{final}} \\ & z_{i+1} = g(z_i, u_i) \qquad \forall i \in [0, N-1] \\ & h_j(u_j) \leq 0 \qquad\qquad \forall j \in [0, P] \end{aligned} \tag{1}$$

Where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is some metric to be minimized, $z_{[0:N]}$ represent $N+1$ states in the time interval $[0, T]$ and $u_{[0:N]}$ represent control inputs at the same time points, $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is an approximation of the continuous time system dynamics, and $h_j(\cdot)$ are inequality constraints on some subset of the control inputs ($P \leq N$).

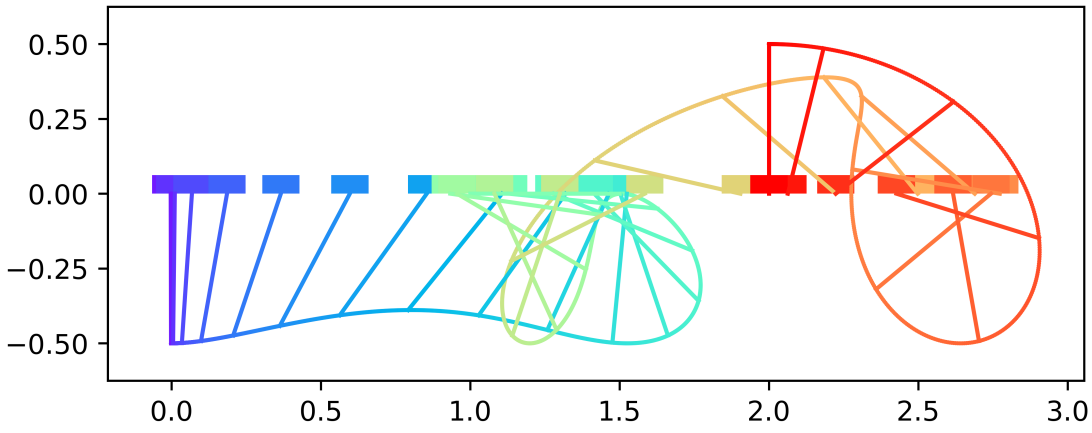

Figure 1: An example of the output of trajectory optimization for a simple cartpole swing-up scenario. Purple is the initial state, red is the final state. Dynamics are linearized. The cart and pendulum arm position are shown as keyframes over the trajectory.

A constraint on the final state, z_N, is not entirely necessary and could be replaced with a penalty term in the objective function. In model predictive control applications with finite horizons, it might be best to replace the final state requirement with either a penalty on the final state or a penalty over many sequential states.

An example of an optimized trajectory is shown in Figure 1, where the initial state is with the cart at $(0, 0)$, pendulum hanging downward, and the final state is at $(2, 0)$ with the pendulum pointing vertically. Only lateral forces are applied to the cart.

2 Approach

An NLP of the form shown in (1) can be written with a quadratic objective function over $z_{[0:N]}$ and $u_{[0:N]}$ using linearized system dynamics and simple box constraints on control inputs. This formulation leads to the quadratic program shown below:

$$\begin{aligned}
\min_{z_{[0:N]}, u_{[0:N]}} \quad & \frac{1}{2}\sum_{i=0}^{N}\left(z_i^T Q_i z_i + p_i^T z_i\right) + \frac{1}{2}\sum_{j=0}^{N} u_j^T R_j u_j \\
\text{s.t.} \quad & z_0 = z_{\text{init}} \\
& z_N = z_{\text{final}} \\
& z_{i+1} = Az_i + Bu_i + r_0 \quad \forall i \in [0, N-1] \\
& u_l \leq u_i \leq u_h \quad \forall i \in [0, N]
\end{aligned} \tag{2}$$

$$\begin{aligned}
z_i &\in \mathbb{R}^n \\
u_i &\in \mathbb{R}^m \\
A &\in \mathbb{R}^{n\times n} \\
B &\in \mathbb{R}^{n\times m} \\
r_0 &\in \mathbb{R}^n \\
Q &\in \mathbb{R}^{n\times n} \\
R &\in \mathbb{R}^{m\times m} \\
N &\in \mathbb{Z}_{++} \\
T &\in \mathbb{R}_{++}
\end{aligned} \tag{3}$$

This formulation is a convex optimization problem, which lends itself perfectly to the analyses and solution methods taught in this course. In general, the approach that this project takes to solving this optimization problem is to:

1. Use a large, finite number of elements from the matrix-exponential solution to the linearized dynamics equation to estimate A, B, and r_0
2. Use the log-barrier Interior Point Method as the outer solver
3. Rewrite (2) as a convex equality-constrained optimization problem by moving the constraints on control inputs into log barrier terms in the objective function
4. Generate an infeasible solution by interpolating from the initial position to the final position with $u_i = 0\ \forall i$
5. Use Infeasible Newton's method to determine if the initial infeasible solution *can* be converted into a feasible solution
6. Use Feasible Newton's method while increasing the log barrier strength terms until an ϵ tolerance is reached

3 Problem Formulation

The following sections detail the way that elements of (2) are calculated in this project.

3.1 Dynamics as Constraints

A general autonomous nonlinear dynamical system can be written as

$$\begin{aligned}
\dot{z} &= f(z, u) \\
f &: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n
\end{aligned} \tag{4}$$

which can be approximated to first-order about an arbitrary z_0 and u_0 as

$$\begin{aligned}
\dot{z} &= p_0 + Fz + Gu \\
&= f(z_0, u_0) + \frac{\partial f(z_0, u_0)}{\partial z}(z - z_0) + \frac{\partial f(z_0, u_0)}{\partial u}(u - u_0) \\
&= f(z_0, u_0) + \nabla_z f(z_0, u_0)^T (z - z_0) + \nabla_u f(z_0, u_0)^T (u - u_0).
\end{aligned} \tag{5}$$

The matrices F and G and the vector p_0 are defined as the following:

$$\begin{aligned}
F &= \frac{\partial f(z_0, u_0)}{\partial z} \\
G &= \frac{\partial f(z_0, u_0)}{\partial u} \\
p_0 &= f(z_0, u_0) - Fz_0 - Gu_0
\end{aligned} \tag{6}$$

The A and B matrices in (2) come from the analytical solution to (5) for piecewise linear control input over a fixed timestep. The general solution to (5) with an unknown control input is given by the following:

$$
\begin{aligned}
z(t) &= e^{F(t-t_0)}z(t_0) + e^{Ft}\int_{t_0}^{t} e^{-F\tau}(Gu(\tau)+p_0)d\tau \\
&= \Phi(t,t_0)z(t_0) + \int_{t_0}^{t}\Phi(\tau,t_0)(Gu(\tau)+p_0)d\tau
\end{aligned} \tag{7}
$$

Where $\Phi(t,t_0)$ is the state transition matrix. When $u(t)$ is a constant u_0 over the time interval $[t_0, t_0+\Delta t]$, the integral in (7) can be written as (assuming F is invertible):

$$
\begin{aligned}
&\int_{t_0}^{t_0+\Delta t}\Phi(\tau,t_0)(Gu(\tau)+p_0)d\tau \\
&= \int_{t_0}^{t_0+\Delta t} e^{F(t_0+\Delta t-\tau)}(Gu_0+p_0)d\tau \\
&= (e^{F\Delta t}-\mathbb{I})F^{-1}(Gu_0+p_0)
\end{aligned} \tag{8}
$$

If F is not invertible, then the integral in (8) can be solved using the series representation of the matrix exponential, shown below.

$$
\begin{aligned}
&\int_{t_0}^{t_0+\Delta t} e^{F(t_0+\Delta t-\tau)}(Gu_0+p_0)d\tau \\
&= \sum_{k=0}^{\infty}\int_{t_0}^{t_0+\Delta t}\frac{(F(t_0+\Delta t-\tau))^k}{k!}d\tau(Gu_0+p_0) \\
&= \left(\sum_{k=1}^{\infty}\frac{F^{k-1}\Delta t^k}{k!}\right)(Gu_0+p_0) \\
&= \left(\Delta t + \frac{F\Delta t^2}{2!} + \frac{F^2\Delta t^3}{3!} + ...\right)(Gu_0+p_0)
\end{aligned} \tag{9}
$$

The matrices A and B and the vector r_0 are short hands for the results from equations (7) and (9):

$$
\begin{aligned}
A &= e^{F\Delta t} = \sum_{k=0}^{\infty}\frac{(F\Delta t)^k}{k!} \\
B &= \sum_{k=1}^{\infty}\frac{F^{k-1}\Delta t^k}{k!}G \\
r_0 &= \sum_{k=1}^{\infty}\frac{F^{k-1}\Delta t^k}{k!}p_0
\end{aligned} \tag{10}
$$

3.2 Interior Point Method

This project places no penalty on system states, only on the control effort exerted to achieve a desired final state. We also weight control effort across all timesteps uniformly. This means that the Q_i matrices and p_i vectors in (2) are zero and $R_j = R\ \forall j \in [0,N]$.

The inequality constraints on control input in (2) are replaced with log-barrier terms in the objective function. The maximum and minimum bounds on the control inputs are converted into the following log barrier terms

$$
\begin{aligned}
\phi_h(u_i) &= -\frac{1}{t}\log(-(u_i-u_h)) \\
\phi_l(u_i) &= -\frac{1}{t}\log(-(u_l-u_i))
\end{aligned} \tag{11}
$$

Where ϕ_h are log barrier terms for the control input upper bound and ϕ_l are log barrier terms on the control input lower bound. Replacing the inequality constraints with ϕ_h and ϕ_l reduces the optimization problem in (2) to:

$$
\begin{aligned}
\min_{z_{[0:N]},u_{[0:N]}} \quad & \frac{1}{2}\sum_{j=0}^{N}u_j^T R u_j + \sum_{i=0}^{N}\phi_h(u_i) + \sum_{i=0}^{N}\phi_l(u_i) \\
\text{s.t.} \quad & z_0 = z_{\text{init}} \\
& z_N = z_{\text{final}} \\
& z_{i+1} = Az_i + Bu_i + r_0 \quad \forall i \in [0, N-1]
\end{aligned} \tag{12}
$$

The Interior Point Method requires that all centering steps must be feasible, however, generating an initial feasible solution to this problem is difficult because of the presence of the final state constraint. Calculating an initial feasible solution amounts to finding a suboptimal trajectory for $z_{[0:N]}$ and $u_{[0:N]}$ that satisfies the linearized system dynamics defined by A, B, and r_0. Fortunately, Infeasible Newton's method exists to handle cases just like this.

The Interior Point Method implementation for this project proceeds as shown in algorithm 1, where superscripts indicate a step number.

Algorithm 1 Interior Point Method

Require: $T > 0$ ▷ Duration of the trajectory)
Require: $N > 0$ ▷ Number of samples along the duration of the trajectory
Require: $z^0_{[0:N]}$, $u^0_{[0:N]}$ ▷ Not necessarily feasible, but with $z_0 = z_{\text{init}}$
Require: $t^0 > 1$ ▷ Log barrier parameter
Require: $\mu \approx 10$ ▷ Log barrier increase term
Require: $\epsilon_{\text{infeas}} \approx 10^{-8}$ ▷ Termination criteria for Infeasible Newton
Require: $\epsilon_{\text{feas}} \approx 10^{-8}$ ▷ Termination criteria for Feasible Newton
Require: $\alpha \in (0, 0.5]$ ▷ Line search comparison parameter
Require: $\beta \in (0, 0.9]$ ▷ Line search step-size reduction parameter

For the sake of clarity, let $Z^i = z^i_{[0:N]}$ and $U^i = u^i_{[0:N]}$ be stacks of states and control inputs, respectively
Z^1, U^1, res = InfeasibleNewton(Z^0, U^0, t^0, A, B, r_0, ϵ_{infeas}, α, β)
if $||\text{res}||_2 > \epsilon_{\text{infeas}}$ **then**
  return Problem is infeasible
end if
while $\frac{m}{t} > \epsilon_{\text{feas}}$ **do** ▷ m is the number of log barrier terms in the objective function
  Z^{k+1}, U^{k+1} = FeasibleNewton(Z^k, U^k, t^k, A, B, r_0, ϵ_{feas}, α, β)
  $t^{k+1} \leftarrow \beta t^k$
end while

3.3 Infeasible Start Newton's Method

Infeasible Newton's method is a primal-dual method for solving equality constrained convex optimization problems of the form show in (13). This method is used in this project only for the first centering step of the Interior Point Method.

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & Ux = w. \end{aligned} \tag{13}$$

Where $x \in \mathbb{R}^n$ is the decision variable, $f : \mathbb{R}^n \to \mathbb{R}$ is convex, $U \in \mathbb{R}^{m \times n}$ is full row rank, and $w \in \mathbb{R}^m$. Solving the nonlinear system of equations that result from the KKT conditions of (13) subsequently solves the optimization problem. The problem's Lagrangian is given by $L = f(x) + \nu^T (Ux - w)$, and the subsequent KKT conditions are shown below.

$$\begin{aligned} \nabla_x L(x, \nu) = \nabla_x f(x^*) + U^T \nu^* = 0 \\ Ux^* = w \end{aligned} \tag{14}$$

Where ν is the unbounded dual variable associated with the equality constraints. The power of Infeasible Newton derives from the fact that an initial infeasible solution converges to a feasible solution as the problem is being optimized. Solutions to these kinds of optimization problems are found by treating the KKT conditions as a root-finding problem and using Newton's method to slowly converge to the roots. Iterates to the primal and dual variables are found by solving the following system of equations (where superscript k indicates an iterate number).

$$\begin{pmatrix} \nabla^2_x f(x^k) & U^T \\ U & 0 \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \nu^k \end{pmatrix} = - \begin{pmatrix} \nabla_x f(x^k) + U^T \nu^k \\ Ux^k - w \end{pmatrix} \tag{15}$$

Where the residual vector, $r(x^k, \nu^k)$, is defined as the RHS of (15),

$$r(x^k, \nu^k) = \begin{pmatrix} \nabla_x f(x^k) + U^T \nu^k \\ Ux^k - w \end{pmatrix} \tag{16}$$

and

$$\begin{aligned} x^{k+1} &= x^k + s\Delta x^k \\ \nu^{k+1} &= \nu^k + s\Delta \nu^k. \end{aligned} \tag{17}$$

Where s is a step-size determined by line search over the norm of the residual (as opposed to the objective function). Infeasible Start Newton's Method works because the *residual* vector is non-increasing over a one-dimensional extrapolation from some given point x^k, ν^k in the direction of $\Delta x^k, \Delta \nu^k$ from (15). It's also worth noting that, because of the infeasible start, the objective function value might sometimes increase over the execution of Infeasible Start Newton's Method. The entire algorithm is shown in Algorithm 2.

Algorithm 2 Infeasible Start Newton's Method

Require: $x^0 \in \text{dom} f, \nu \in \mathbb{R}^m, \alpha \in (0, 0.5], \beta \in (0, 1)$
Require: $\gamma > 0, \gamma \approx 10^{-8}$ ▷ Constraint satisfaction threshold
Require: $\epsilon > 0, \epsilon \approx 10^{-5}$ ▷ Desired residual vector norm threshold
  while $||Ux - w||_2 > \gamma$ and $||r(x^k, \nu^k)||_2 > \epsilon$ **do**
    Calculate Δx^k, $\Delta \nu^k$ from Eq (15)
    $s \leftarrow 1$
    while $x^k + s\Delta x^k \notin \text{dom} f$ or $||r(x^k + s\Delta x^k, \nu^k + s\Delta \nu^k)||_2 > (1 - \alpha s)||r(x^k, \nu^k)||_2$ **do**
      $s \leftarrow \beta s$
    end while
    $x^{k+1} = x^k + s\Delta x^k$
    $\nu^{k+1} = \nu^k + s\Delta \nu^k$
  end while

3.4 Feasible Start Newton's Method

After an initial feasible trajectory is generated via Infeasible Start Newton's Method the Interior Point Method implementation in this project uses Feasible Start Newton's Method to further optimize the solution while increasing the log barrier penalty terms in the objective function.

If the initial solution to the optimization problem in (13) is feasible, then we can solve an approximation of the original problem by letting $x \to x + \Delta x$, approximating $f(x + \Delta x)$ up to a quadratic, and requiring that $U(x + \Delta x) = w$.

$$f(x + \Delta x) \approx f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x) \Delta x \tag{18}$$

$$\begin{aligned} U(x + \Delta x) &= Ux + U\Delta x = w \\ w + U\Delta x &= w \implies U\Delta x = 0 \end{aligned} \tag{19}$$

Replacing the original optimization problem with the approximate values from (18) and (19) leaves

$$\begin{aligned} \min_{\Delta x} \quad & f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x) \Delta x \\ \text{s.t.} \quad & U\Delta x = 0. \end{aligned} \tag{20}$$

with the KKT conditions shown below (where ν is the dual variable associated with the equality constraints).

$$\begin{aligned} \nabla f(x) + \nabla^2 f(x)\Delta x + U^T \nu &= 0 \\ U\Delta x &= 0 \end{aligned} \tag{21}$$

Similarly to Infeasible Start Newton's Method, these are rewritten as a system of linear equations, Δx and $\Delta \nu$ are solved for, and x^k and ν^k are updated using backtracking line search. The system of linear equations is

$$\begin{pmatrix} \nabla^2 f(x^k) & U^T \\ U & 0 \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta \nu^k \end{pmatrix} = - \begin{pmatrix} \nabla f(x^k) \\ 0 \end{pmatrix} \tag{22}$$

and only the primal decision variable is updated, $x^{k+1} = x^k + s\Delta x^k$, where s is a step size that comes from backtracking line search. The algorithm for Feasible Start Newton's Method is shown in 3.

3.5 Cartpole Dynamics

Cartpole dynamics have been derived to death everywhere and anywhere. Figure 2 shows a generic cartpole configuration, where ϕ increases in a counterclockwise direction and the cart is located exactly on the x-axis. The cart has mass m_c, the pendulum arm is massless with length l_p, and a mass m_p is fixed to the end of the pendulum arm. The cart can be subjected to a force along the x-axis within the range $F_l \leq F_c \leq F_h$. The system is conservative, so neither the cart nor the pendulum experience any drag forces.

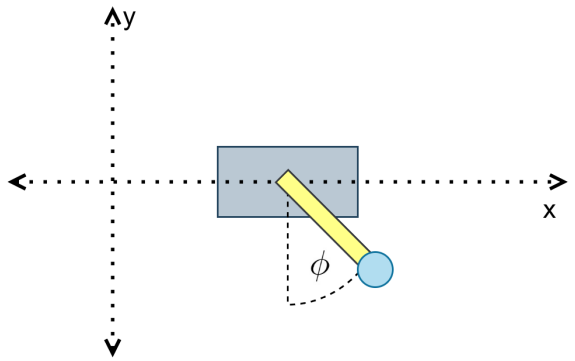

Figure 2: Cartpole system. The cart is constrained to move along the x-axis, and the pendulum makes angle ϕ relative to the cart's body y-axis.

The undamped cartpole's system states are the cart's position and velocity, and the pendulum's angular position and angular velocity. These are arranged into a column vector, $z = [x \quad \phi \quad \dot{x} \quad \dot{\phi}]^T$. The cartpole's nonlinear equations of motion, written in terms of z, are shown below.

$$\dot{z} = \begin{pmatrix} z_3 \\ z_4 \\ \frac{-g \sin z_2 \cos z_2 - l_p u - l_p z_4^2 \sin z_2}{\cos^2 z_2 - l_p \alpha} \\ \frac{\alpha g \sin z_2 + u \cos z_2 + z_4^2 \sin z_2 \cos z_2}{\cos^2 z_2 - l_p \alpha} \end{pmatrix} \tag{23}$$

Where $\alpha = (m_c + m_p)/(m_p l_p)$, g is the constant gravity term 9.8m/s^2, and $F_c = u m_p l_p \in \mathbb{R}$ is a force applied to the cart.

The linearized cartpole dynamics are similarly tedious to derive, so their derivations are omitted and non-trivial entries from the F and G matrices from (6) are shown below in equations (24) - (29). Technically the G matrix is actually a vector, $G \in \mathbb{R}^n$, since $u \in \mathbb{R}$.

For all experiments carried out in this project, the cartpole's dynamics are linearized about the point $z = [0 \quad 0 \quad 0 \quad 0]^T$.

$$F_{3,2} = \frac{g \sin^2(z_2) - g \cos^2(z_2) - l z_4^2 \cos(z_2)}{-\alpha l + \cos^2(z_2)} + \frac{2\left(-g \sin(z_2) \cos(z_2) - l u - l z_4^2 \sin(z_2)\right) \sin(z_2) \cos(z_2)}{\left(-\alpha l + \cos^2(z_2)\right)^2} \tag{24}$$

$$F_{3,4} = -\frac{2 l z_4 \sin(z_2)}{-\alpha l + \cos^2(z_2)} \tag{25}$$

$$\begin{aligned} F_{4,2} = {} & \frac{\alpha g \cos(z_2) - u \sin(z_2) - z_4^2 \sin^2(z_2) + z_4^2 \cos^2(z_2)}{-\alpha l + \cos^2(z_2)} \\ & + \frac{2\left(\alpha g \sin(z_2) + u \cos(z_2) + z_4^2 \sin(z_2) \cos(z_2)\right) \sin(z_2) \cos(z_2)}{\left(-\alpha l + \cos^2(z_2)\right)^2} \end{aligned} \tag{26}$$

$$F_{4,4} = \frac{2 z_4 \sin(z_2) \cos(z_2)}{-\alpha l + \cos^2(z_2)} \tag{27}$$

$$G_3 = -\frac{l}{-\alpha l + \cos^2(z_2)} \tag{28}$$

$$G_4 = \frac{\cos(z_2)}{-\alpha l + \cos^2(z_2)} \tag{29}$$

Algorithm 3 Feasible Start Newton's Method

Require: $x^0 \in \text{dom} f, Ux^0 = w, \nu \in \mathbb{R}^m, \alpha \in (0, 0.5], \beta \in (0, 1)$
Require: $\epsilon > 0, \epsilon \approx 10^{-5}$ ▷ Desired Newton decrement threshold
  while $\lambda^2(x^k) > \epsilon$ **do** ▷ $\lambda^2(x^k) = (\Delta x^k)^T \nabla^2 f(x^k) \Delta x^k$
    Calculate Δx^k, $\Delta \nu^k$ from Eq (15)
    $s \leftarrow 1$
    while $x^k + s\Delta x^k \notin \text{dom} f$ or $f(x^k + s\Delta x^k) > f(x^k) + \alpha s \nabla f(x^k)^T \Delta x^k$ **do**
      $s \leftarrow \beta s$
    end while
    $x^{k+1} = x^k + s\Delta x^k$
    $\nu^{k+1} = \nu^k + s\Delta \nu^k$
  end while

3.6 Nitty gritty details

The control input and state variables are mashed together in one primal decision variable, x. The first $N+1$ entries in x are the scalar control inputs, and the remaining $4 \times (N+1)$ entries contain state variables. The primal decision variable looks like the following:

$$\begin{aligned} x &= \begin{pmatrix} u_0 & u_1 & ... & u_N & z_0 & z_1 & ... & z_N \end{pmatrix}^T \\ &= \begin{pmatrix} u_{[0:N]}^T & z_{[0:N]}^T \end{pmatrix}^T \in \mathbb{R}^{(S+1)(N+1)} \end{aligned} \tag{30}$$

Where S is the size of the state vector for the dynamical system, in this case $S = 4$. A state at some step i occupies indices in the range (assuming zero-based indexing) $[N+1+Si, N+1+Si+S-1]$ in the primal state vector, and a control input at some step i occupies index i.

Since the control inputs and states are smashed into one variable, the matrix R in (12) is the upper-left block of a larger matrix $Q \in \mathbb{R}^{(S+1)(N+1)\times(S+1)(N+1)}$.

This project requires the Jacobian (gradient transpose) and Hessian of the objective function of the objective function from the optimization problem in (12). Since only the control input terms are part of the objective function, the Jacobian is non-zero in the first $(N+1)$ elements, and the Hessian is non-zero in the upper left block of size $(N+1) \times (N+1)$.

The gradient of the objective function with respect to the control inputs (writing $u = u_{[0:N]}$ and $z = z_{[0:N]}$) is

$$\nabla_u f(x) = Ru - \frac{1}{t}\left(\frac{1}{u - u_l} + \frac{1}{u_h - u}\right) \tag{31}$$

Where $\frac{1}{v}$ is taken to mean the element-wise inverse of the vector $v \in \mathbb{R}^n$ and adding or subtracting a scalar from a vector is the same as subtracting the scalar value from each element of the vector.

The Hessian of the objective function with respect to the control inputs is

$$\nabla_u^2 f(x) = Q + \frac{1}{t}\text{diag}\left(\frac{1}{(u - u_l)^2} + \frac{1}{(u_h - u)^2}\right) \tag{32}$$

Where v^2 is taken to mean a vector whose elements are squared individually, or equivalently, the Hadamard product of v with itself. This project uses $Q = \mathbb{I}$, leading the Hessian to be especially sparse. This sparsity is not exploited as part of the implementation, but it certainly could be.

The equality constraints for system dynamics, initial state, and final state are placed in one large, sparse matrix to make it easier to integrate them into the two flavors of Newton's method. The matrix $U \in \mathbb{R}^{S(N+2)\times(S+1)(N+1)}$ is exceptionally sparse, but again, this sparsity is not exploited in this implementation. As an example, the dynamics constraint $z_1 = Az_0 + bu_0 + r_0$ can be rearranged as

$$z_1 - Az_0 - bu_0 = r_0 \tag{33}$$

and which can be written less conveniently using index ranges of the primal decision variable x as

$$x_{[N+1+S:N+1+2S-1]} - Ax_{[N+1:N+1+S-1]} - bx_{[0]} = r_0 \tag{34}$$

Adding dynamics constraints requires inserting A matrices and b vectors into specific portions of the U matrix. The included Jupyter notebook has more details.

4 Experiments

The experiments in this project were haphazardly constructed and mostly designed to verify that implementations of complicated algorithms were working correctly (this project was mostly focused on implementation details).

That said, the core experiment was to verify that a linearized cartpole system, starting in the configuration $z_0 = [0 \quad 0 \quad 0 \quad 0]^T$, could be swung up to the final position $z_f = [* \quad \pi \quad 0 \quad 0]^T$ over a time interval $t \in [0, T]$, with some number of points along the trajectory, N, with control inputs in some user-defined range $u_i \in [u_l, u_h]$, exerting some minimal amount of control effort.

The result of one of these experiments is shown in Figure 1, where the control input is between $[-30, 30]$ over $T = 2$ seconds. The control input for this trajectory is shown in Figure 3.

One particularly interesting experiment is finding the minimum time required to generate a feasible trajectory to a desired final position given some input constraints. For the parameters listed above, it turns out that roughly 1.24 seconds are required to move achieve the desired final state under minimal control effort. Plots of the trajectory and control input for this configuration are included in Figure 4, note that the control inputs exhibit some characteristics of a bang-bang control system.

On the flip side, if the system is given ample time to reach the final state, the control inputs are more subdued over the duration of the trajectory.

This allows the cart to slowly "wiggle" the pendulum in near-vertical configurations until the final time is achieved. It's interesting to note that the control input in 5 exhibits a kind of beat-pattern. This might suggest that one high frequency and one low frequency system mode are being excited and exploited to minimize the total control effort.

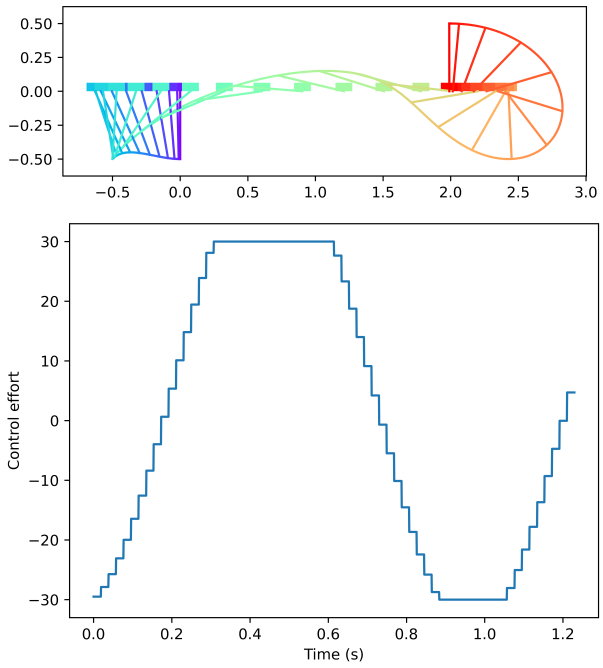

Figure 4: Cartpole trajectory and control input for a cartpole swing-up scenario with $T = 1.24$, $N = 64$, $u \in [-30, 30]$, and final state $z = [2 \quad \pi \quad 0 \quad 0]^T$.

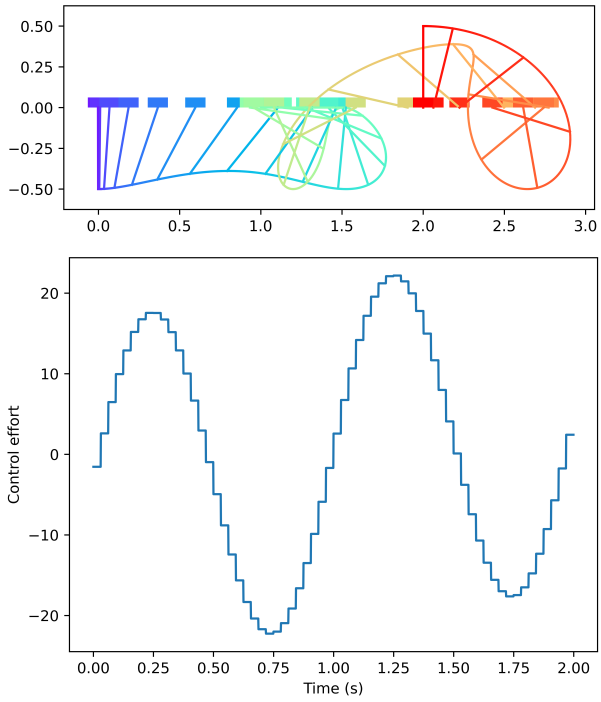

Figure 3: Cartpole trajectory and control input for a cartpole swing-up scenario with $T = 2$, $N = 64$, $u \in [-30, 30]$, and final state $z = [2 \quad \pi \quad 0 \quad 0]^T$.

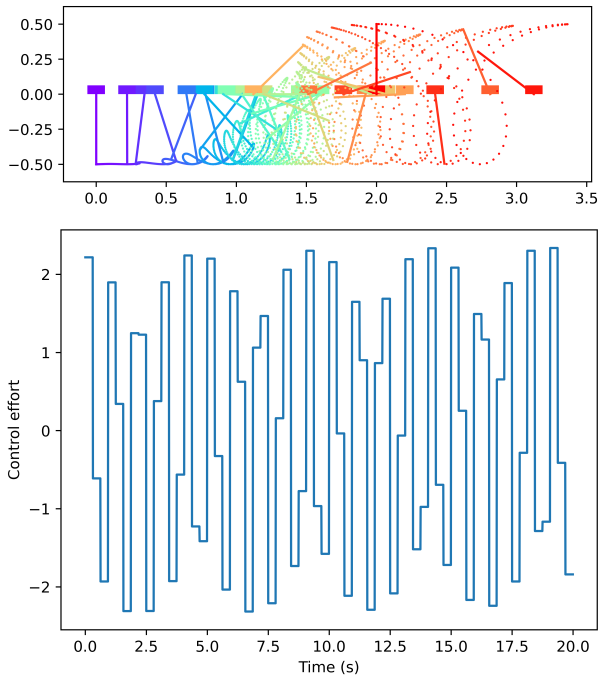

Figure 5: Cartpole trajectory and control input for a cartpole swing-up scenario with $T = 20$, $N = 64$, $u \in [-30, 30]$, and final state $z = [2 \quad \pi \quad 0 \quad 0]^T$.

References

[1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, March 2004.

[2] Matthew Kelly. Trajectory optimization. `https://www.matthewpeterkelly.com/tutorials/trajectoryOptimization/`, 2016.

[3] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, 2006.

[4] Russ Tedrake. Underactuated robotics. `https://underactuated.mit.edu/trajopt.html`, 2024.