ME C231 Midterm: Model Predictive Control

Your Name and Student ID:

Please answer all questions. Make sure to review the Exam Instructions before completing the exam. Before each answer report the question number you are answering: 3.a.ii means problem 3, question (a), part ii.

Problem:	Max Score	Score
1(a)	5	
1(b)	10	
1(c)	10	
2(a)	5	
3(a)	2	
3(b)	10	
3(c)	5	
4(a)	20	
4(b)	20	
Total	87	

1. **Optimization Basics**

   (a) **Convexity and Polytopes**

   i) Let $f(x) = f_1(x)f_2(x)$ where f_1 and f_2 are convex functions. What can you say about $f(x)$?

   ☐ $f(x)$ is never convex.

   ☐ $f(x)$ can be convex.

   ii) Represent the set $\{x \in \mathbb{R}^2 \mid \|x\|_1 \leq 1\}$ in two different ways: First, using linear inequalities and afterwards as a convex combination of appropriate points.

   (b) **KKT Conditions** Consider the optimization problem

$$\begin{aligned} \min \quad & -\frac{1}{2}(x_1^2 + 0.5x_2^2) \\ \text{s.t.} \quad & x_1 + 2x_2 = 1 \\ & 0 \leq x_1 \leq 1 \\ & 0 \leq x_2 \leq 1 \end{aligned}$$

   i) Report the optimal solution and the associated lagrange multipliers (also called dual variables). (You can use MATLAB to compute them)

ii) Show that KKT conditions hold at the optimal solution.

(c) **Consider the optimization problem**

$$\begin{array}{ll} \min_z & \|3z+2\|_\infty \\ \text{subject to} & -3 \leq z \leq 10 \end{array} \tag{1}$$

where $\|\cdot\|_\infty$ denotes the infinity norm.

i. Transform it into a linear program and write down the LP matrices.

ii. Compute the optimal solution using `linprog`. Print the optimal cost and the optimizer.

2. **Linear Quadratic Regulator**

(a) **Finite-Time LQR** Define the following quadratic cost function over a finite horizon of N steps

$$J_0(x_0, U_0) \triangleq x_N' P x_N + \sum_{k=0}^{N-1} x_k' Q x_k + u_k' R u_k, \tag{2}$$

and consider the finite-time LQR problem

$$\begin{array}{rll} J_0^*(x(0)) = & \min_{U_0} & J_0(x(0), U_0) \\ & \text{subject to} & x_{k+1} = Ax_k + Bu_k,\ k = 0, 1, \ldots, N-1 \\ & & x_0 = x(0). \end{array} \tag{3}$$

In (3) $U_0 = [u_0', \ldots, u_{N-1}']' \in \mathbb{R}^s$, $s \triangleq mN$ is the decision vector containing all future inputs. Assume that the state and input penalty are positive definite $Q = Q' \succ 0$, $R = R' \succ 0$. Set $P = P_\infty$, where P_∞ is the infinite-time optimal costs from LQR, i.e., $[F_\infty, P_\infty] = \texttt{dlqr}(A, B, Q, R)$.

Question: Assume you compute the optimal feedback controller $u_k^* = F_k x_k$ to problem (3) by using Dynamic Programming. What is the link between F_2 (the optimal controller at step 2) and F_∞? Motivate your answer.

3. **Controlling a Rocket Landing: Path Planning with Constrained Finite Time Optimal Control via Batch Approach**
In this question we consider a rocket with a gimbaled thrust. Figure 1 depicts the rocket in a global Cartesian coordinate space where the positive Z direction is pointing down.

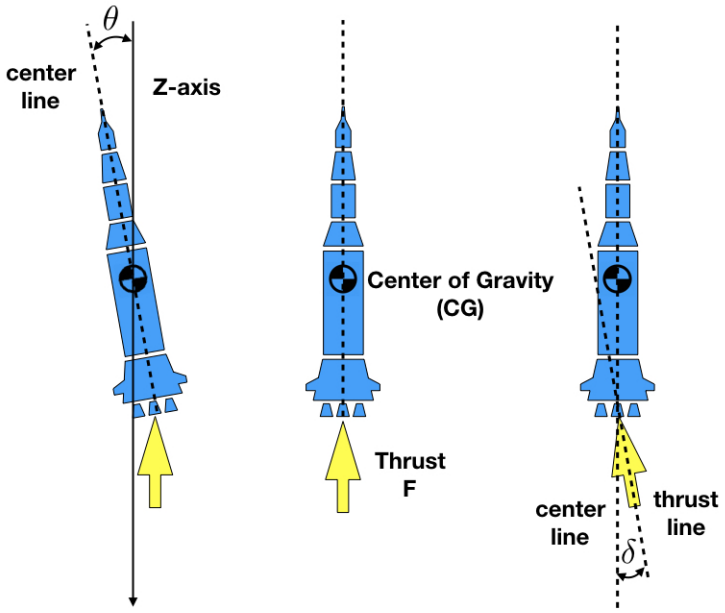

Figure 1: Rocket with gimbaled thrust

The model of the rocket is described by a nonlinear system:

$$\begin{aligned} \dot{\theta}(t) &= \omega(t) \\ J\dot{\omega}(t) &= -\frac{l}{2}F(t)\sin(\delta(t)) \\ \dot{h}(t) &= v(t) \\ m\dot{v}(t) &= mg - F(t)\cos(\delta(t)) \end{aligned} \tag{4}$$

where θ, ω are the rocket angle and angular velocity relative to the z-axis and h, v are the height and velocity at the rocket's center of gravity. The inputs are the thrust force F (positive upwards) and the thrust angle δ. The length of the rocket is l; the mass of the rocket is m.

We simplify the model using the small angle approximation for δ ($\sin\delta = \delta$ and $\cos\delta = 1$), and discretize the model with sampling time $T_s = 0.1$ s to obtain the discrete model:

$$\begin{aligned} \theta(k+1) &= \theta(k) + T_s\omega(k) \\ \omega(k+1) &= \omega(k) - T_s\frac{l}{2J}F(k)\delta(k) \\ h(k+1) &= h(k) + T_s v(k) \\ v(k+1) &= v(k) + T_s(g - \frac{1}{m}F(k)) \end{aligned} \tag{5}$$

Model (5) will be compactly rewritten as

$$z_{k+1} = f(z_k, u_k) \tag{6}$$

where the rocket state at time step k is $z_k = [\theta_k, \omega_k, h_k, v_k]^\top$, and the inputs are $u_k = [F_k, \delta_k]^\top$. We will also use the notation $u_k(1) = F_k$ and $u_k(2) = \delta_k$.

You are asked to use model (6) to formulate and solve a landing problem as the finite-time optimal control problem

$$\begin{aligned}
\min_{z_0,\ldots,z_N,u_0,\ldots,u_{N-1}} \quad & \sum_{k=0}^{N-1} z_k' Q z_k + u_k(2)^2 + (u_k(1)/F_{max})^2 \\
& z_{k+1} = f(z_k, u_k) && \forall k = \{0, \ldots, N-1\} \\
& z_{min} \leq z_k \leq z_{max} && \forall k = \{0, \ldots, N\} \\
& u_{min} \leq u_k \leq u_{max} && \forall k = \{0, \ldots, N-1\} \\
& z_0 = \bar{z}_0 \\
& z_N = \bar{z}_N
\end{aligned} \tag{7}$$

with the following settings:

- The mass of the rocket is $m = 27648$ kg. The rocket length is $l = 70$ m. The moment of inertia of the rocket is $J = \frac{1}{16} m l^2$. The acceleration due to gravity is $g = 9.8$ m/s^2
- The rocket starts from the initial state $\bar{z}_0$. Our goal is to land in 10 seconds at the terminal state $\bar{z}_N = [0; 0; 0; 0]$.
- Horizon $N = 10/T_s = 100$.
- The input constraints u_{min} and u_{max} are defined by: $0 \leq F_k \leq F_{\max} = 1690$ kN, $-5\pi/180 \leq \delta_k \leq 5\pi/180$ (± 5 degrees).
- The state constraints z_{min} and z_{max} are defined by: $-20\pi/180 \leq \theta_k \leq 20\pi/180$, $-100 \leq \omega_k \leq 100$, $-3000 \leq h_k \leq 0$, $-100 \leq v_k \leq 500$.

HERE ARE THE QUESTIONS

(a) Is problem (7) convex or not? Justify your answer.

(b) Read the hints next, use $\bar{z}_0 = [10\pi/180, 0, -1228, 205.2]^\top$, tune the cost matrix Q and submit:

   i. The code solving the rocket landing problem (7).
   ii. Two plots. The first plot should show the time evolution of the states (with four subplots for each state, properly labeled). The second plot should show the time evolution of the inputs F_k and δ_k (with two subplots, please plot angle in degree). The time span of each plot should be $[0, 10]$ seconds.

(c) Try to land from the same initial condition in 5 seconds (solve the same problem with $N = 5/T_s$). Can you find a control sequence which can solve the problem? If not, is there any other control strategy that can land the rocket from this position in 5 seconds? Please motivate your answer.

Important Hints

- All the parameters are defined in the code provided here.

```
% Parameters
m = 27648;
l = 70;
J = 1/16*m*l^2;
g = 9.8;

% Constraints
Fmax = 1690*1000;
dmax = 5*pi/180;
umin = [0; -dmax];
umax = [Fmax; dmax];
tmin = -20*pi/180;
tmax = 20*pi/180;
zmin = [tmin; -100; -3000; -100];
zmax = [tmax; 100; 0; 500];

% Initial conditions
v0 = 205.2;
alt0 = -1228; %negative because the Z-axis is positive pointing downward
t0 = 10*pi/180;
z0 = [t0; 0; alt0; v0];

% Define sampling time
TS = 0.1;

% Define horizon
N = 10/TS;
```

- This is a simple CFTOC problem with terminal constraints to be solved with Yalmip.
- The only challenge is that model (6) is nonlinear because of the product between the two inputs F and δ. This nonlinearity may cause problems when solving the CFTOC problem over a horizon of 100 steps with fmincon or IPOPT. We recommend that you use the following change of variables: $\tilde{u}_k = F_k \delta_k$. After the change of variables, the model $z_{k+1} = f(z_k, \bar{u}_k)$ with $\bar{u_k} = [F_k, \tilde{u}_k]$ becomes linear! (affine, to be precise). You can now use the following model (instead of model (6)) and use a convex solver:

$$\begin{aligned} \theta_{k+1} &= \theta_k + T_s \omega_k \\ \omega_{k+1} &= \omega_k - T_s \frac{l}{2J} \tilde{u}_k \\ h_{k+1} &= h_k + T_s v_k \\ v_{k+1} &= v_k + T_s (g - \frac{1}{m} F_k) \end{aligned} \tag{8}$$

- CAREFUL, now the input constraints need to be changed. You need to compute the new input constraints on $\tilde{u}_k$ from the fact that $\delta_{\min} \leq \delta_k \leq \delta_{\max}$ and $\delta_k = \tilde{u}_k / F_k$.
- You can still use the same form of cost function after the change of variables: $\sum_{k=0}^{N-1} z_k' Q z_k + \tilde{u}_k^2 + F_k / F_{max})^2$. It is a fictitious cost and needs to be tuned anyway.

4. **Controlling a Train: Constrained Finite Time Optimal Control via Dynamic Programming** In this problem, you will be implementing the Dynamic Programming algorithm to solve a CFTOC problem where the objective is to control the speed of a train to travel to a destination in the shortest time while satisfying speed constraints. Before you attempt any coding, make sure to read the **entire** problem statement. Please note that an inefficient implementation may take a long amount of time to compute. We have provided some hints that should guide you in the right direction.

   The train is modeled as a point mass model

$$\begin{aligned} M\dot{v} &= -A - Bv - Cv^2 + F_g(p) + F_R(p) + F_{drive} \\ \dot{p} &= v \end{aligned} \tag{9}$$

   where:

   - v (m/s) is the train velocity, p (m) is its position.
   - M (kg) is the mass of the train.
   - A (N) is a constant term that models the rolling resistance plus the bearing resistance.
   - B (Ns/m) is a constant coefficient.
   - C (Ns^2/m^2) is the aerodynamic coefficient.
   - $F_g(p)$ (N) is the force due to grade resistance, a function of position p, obtained as: $F_g(p) = -M \cdot g \cdot \texttt{slope}(p)$ where $\texttt{slope}(p)$ is the slope of the track, a function of position p.
   - $F_R(p)$ (N) is the force due to curve resistance, a function of position p, obtained as: $F_R(p) = -6M/\texttt{radius}(p)$ where $\texttt{radius}(p)$ (m) is the radius of curvature of the track, a function of position p.
   - F_{drive} (N) is the traction or braking force. This is the input to the system and it is constrained to be $-\mu Mg \leq F_{drive} \leq \mu Mg$.

   M, A, B and C are parameters that depend on the train characteristics, μ is the friction coefficient between the train and the tracks, and g is the gravitational acceleration constant. These parameters are defined as fields in the struct `params` which is included in the `train_data_midterm.mat` file given to you. In addition to these parameters, we have provided you with all required MATLAB functions

   - `slope(p)` returns the slope of the track at position `p`.
   - `radius(p)` returns the radius of curvature of the track at position `p`.
   - `maxspeed(p)` returns the max speed at position `p`.

   Fig. 2 shows the max speed and slope profiles used in these functions.

   To make this problem solvable in a reasonable amount of time, we will use the position of the train p as independent variable. We can then parametrize the velocity of the train v with its position (i.e. v is a function of p). By discretizing the model in space with the sampling interval Δp we obtain:

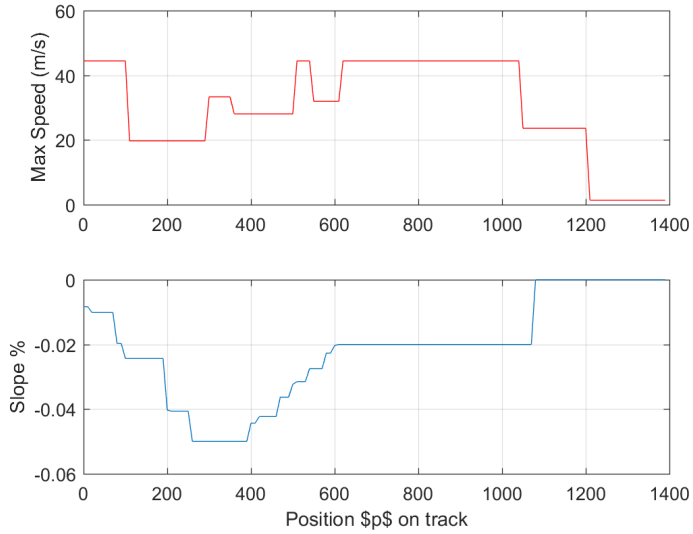

Figure 2: Track slope and max allowed velocity profiles

$$M\left(\frac{v(p_{k+1})-v(p_k)}{\Delta p}v(p_k)\right) = -A - Bv(p_k) - Cv(p_k)^2 + F_g(p_k) + F_R(p_k) + F_{drive}(p_k) \quad (10)$$

where the independent variable is the k-th position p_k. We can compactly write model (10) as

$$v_{k+1} = v_k + \frac{\Delta p}{Mv_k}(-A - Bv_k - Cv_k^2 + F_g(p_k) + F_R(p_k) + u_k) \quad (11)$$

where v_k is the train's speed at position p_k and u_k is the input F_{drive} at position p_k. Now, instead of iterating backwards through time, we can iterate backwards through position when performing DP. We sample the track at an interval of $\Delta p = 10$ m so that $p_k = k\Delta p$ is the position p_k of the train at step k. We want to solve with DP the following problem where the cost minimizes the time it takes to reach the destination:

$$\begin{aligned}
\min_{v_0,\ldots,v_N,u_0,\ldots,u_{N-1}} & \sum_{k=0}^{N}\frac{\Delta p}{v_k} \\
& v_{k+1} = v_k + \frac{\Delta p}{Mv_k}(-A - Bv_k - Cv_k^2 + F_g(p_k) + F_R(p_k) + u_k) \\
& \qquad\qquad \forall k = \{0,\ldots,N\} \quad (12)\\
& 0.1 \le v_k \le \texttt{maxspeed}(p_k) \quad \forall k = \{0,\ldots,N\} \\
& F_{min} \le u_k \le F_{max} \quad \forall k = \{0,\ldots,N-1\} \\
& v_0 = v(0)
\end{aligned}$$

HERE ARE THE QUESTIONS

(a) Implement the Dynamic Programming algorithm (also called recursive approach) for this problem. You may use the starter code provided below. Submit your code.

(b) Simulate system (11)in closed loop with the feedback controller you just computed over the whole length of the track, starting from position p 0 $v_0 = 0.1$. Submit your code. Report the optimal cost. In one plot add three subplots. Subplot 1: plot the closed-loop velocity sequence with the maximum allowable speed `maxspeed`(p) versus position p. Subplot 2: plot the optimal input sequence versus position. Subplot 3: plot slope vs position. My solution looks like the one in Fig. 3.

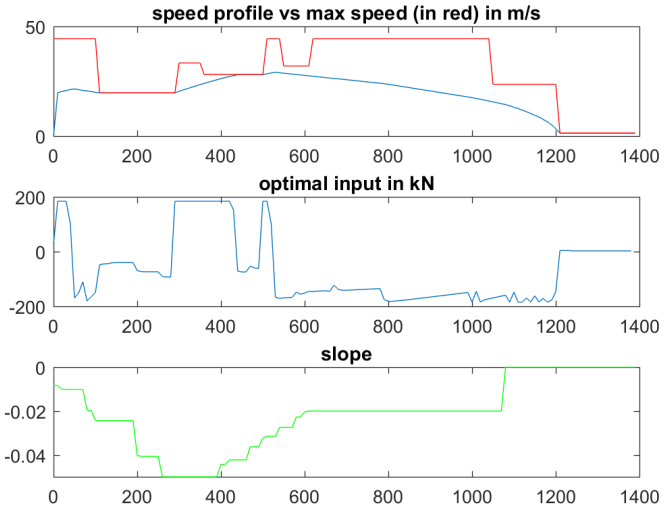

Figure 3: Example solution to the DP problem

Important Hints

- Use the grid and parameters provided in the MATLAB file at the end of these hints.
- At position p_{k-1} for each point in the velocity grid $\bar{v}_{k-1}$ you need to solve the following problem (for simplicity, we denote $v(p_k) = v_k$, `maxspeed`$(p_k) = v_{max,k}$ and $u(p_k) = u_k$)

$$
\begin{aligned}
J^*_{k-1\to N}(v_{k-1}) = \min_{u_{k-1}} \quad & \overbrace{q(v_{k-1}, u_{k-1})}^{\text{stage cost}} + \overbrace{J^*_{k\to N}(v_k)}^{\text{optimal cost-to-go}} \\
\text{subject to} \quad & v_k = v_{k-1} + \tfrac{\Delta p}{M v_{k-1}}(-A - B v_{k-1} - C v_{k-1}^2 \\
& \quad + F_g(p_{k-1}) + F_R(p_{k-1}) + u_{k-1}) \\
& 0.1 \le v_k \le v_{max,k} \\
& v_{k-1} = \bar{v}_{k-1}, \text{where } \bar{v}_{k-1} \text{ is a point on the grid} \\
& v_{k-1} \le v_{max,k-1} \\
& u_{k-1} \text{ is feasible}
\end{aligned}
\tag{13}
$$

It is recommend that you solve problem (13) by using the brute force approach discussed in Homework 4 (look at the MATLAB file DP_parking_brute_force.m)

- Note that the dynamics of this system are simple enough that given two states v_k and v_{k+1}, you are able to calculate the input that corresponds to that transition. This is helpful to prune your input space search. In particular, I suggest you compute the input u_m which brings $\bar{v}_{k-1}$ to the minimum speed 0.1 and the input u_M which brings $\bar{v}_{k-1}$ to the max

speed allowed at position p_k and then take 10 linearly spaced input samples in the range $[u_m, u_M]$. (This is the reason why no input grid is provided in the following starter code. You can try a fixed input grid, but it might take long time to solve this DP problem).

- The code you write should be structurally similar to that in `DP_parking_brute_force.m`. A smart implementation takes a few minutes on an i7 processor, a not-so-smart one can take up to one hour.

```
%% Dynamic Programming - Train Control
%% Author: Francesco Borrelli 2017(c)
clear all
load  train_data_midterm

%% Load parameters
M = param.M;
g = param.g;
A = param.A;
B = param.B;
C = param.C;

%% Define state and input constraints
Fmax0 = param.mumax*M*g;
umin = -Fmax0;
umax = Fmax0;

%% Grid the indipendent variable (same as gridding time)
track_length = profile(end,1); % in meters
dp = 10; % sampling in space
p_sampled = 0:dp:track_length; %sampled train position
N_p = length(p_sampled);

%% Grid the state space
v_sampled = 0.1:0.1:max(maxspeed(p_sampled)); %  grid from 0.1 to the max speed over the track a
N_v = length(v_sampled);
v_idx_set = 1:N_v;

%% This funtion computes the next speed given the current speed, input and position
comp_v_next = @(v,u,p) v+dp/(v*M)*(-A-B*v-C*v^2-M*g*slope(p)-M*6/radius(p)+u);

%% This funtion computes the input to bring speed v to speed v_next
% at position p
comp_u = @(v,v_next,p) M*(v_next-v)/(dp)*v-(-A-B*v-C*v^2-M*g*slope(p)-M*6/radius(p));

%% Define stage cost
Jstage = @(v,u) dp/v;

%% Initialization of Cost-to-Go
for i = v_idx_set
    if v_sampled(i) > maxspeed(p_sampled(N_p))
        J(N_p,i) = inf;
    else
        J(N_p,i) = dp/v_sampled(i);
    end
end
Jtogo = @(v) interpn(v_sampled,J(N_p,:),v,'linear');

%% Perform Dynamic Programming
```

```
tic

for p_indx = N_p-1:-1:1
    fprintf('Solving DP at position = %i (meters) \n',p_sampled(p_indx));
    J(p_indx,:) = inf(1,N_v);
    u(p_indx,:) = nan(1,N_v);
    ...
    ...
    Your code here
    ...
    ...
end

fprintf('Total solution time: %i\n',toc);

%% Simulate the system from the initial condition v(0)=0.1 and p(0)=0 and plot
...
...
Your code here
...
...
```