Sample Background

Quill Healey

June 7, 2024

Contents

Document Overview

Notation

1 Function Fitting, Statistical Estimation, and Classification

1.1 Markov Chain Transition Probability Estimation

Problem Overview and Upshot

Notes

- This problem uses what Gilbert Strang would refer to as a *Markov Matrix*. In more traditional texts the transpose this matrix is worked with, and it is called a *stochastic matrix*.

Problem

[BV04] **Exercise 7.5**. *Markov chain estimation.* Consider a Markov chain with n states and a trasition probabilty matrix $P \in \mathbf{R}^{n \times n}$ defined as

$$P_{ij} = \mathbf{Prob}\left(y_{t+1} = i \mid y_t = j\right).$$

The transition probabilities must satisfy $P_{ij} \geq 0$ and $\sum_{i=1}^{n} P_{ij} = 1, j = 1, \ldots, n$. We consider the problem of estimating the transition probabilities, given an observed sample sequence $y_1 = k_1, y_2 = k_2, \ldots, y_N = k_n$.

(a) Show that if there are no other prior constraints on P_{ij}, then the ML estimates are the empirical transition frequencies: $\hat{P}_{ij}$ is the ratio of the number of times the state transitioned from j into i, divided by the number of times it was j, in the observed sample.

(b) Suppose that an equilibrium distribution p of the Markov chain is known, i.e., a vector $q \in \mathbf{R}_+^n$ satisfying $\mathbf{1}^T q = 1$ and $Pq = q$. Show that the problem of computing the ML estimate of P, given the observed sequence and knowledge of q, can be expressed as a convex optimization problem.

My Response

(a). We first clarify

$$p_P(k) = \mathbf{Prob}_{y \sim P}\left(y_1 = k_1, \ldots, y_N = k_n\right)$$

We seek

$$\hat{P} = \underset{P}{\operatorname{argmax}}\, p_P(k).$$

$$\begin{aligned} p_P(k) = {} & \mathbf{Prob}\left(y_N = k_n \mid y_{N-1} = k_{n-1}\right) \mathbf{Prob}\left(y_{N-1} = k_{n-1} \mid y_{N-2} = k_{n-2}\right) \\ & \cdots \mathbf{Prob}(y_2 = k_2 \mid y_1 = y_1) \mathbf{Prob}\left(y_1 = k_1\right). \end{aligned}$$

$$p_P(k) = P_{k_n k_{n-1}} P_{k_{n-1} k_{n-2}} \cdots P_{k_2 k_1}$$

$$p_P(K) = \prod_{i,j=1}^{n} P_{ij}^{c_{ij}}$$

$$l(P) = \log\left(\prod_{i,j=1}^{n} P_{ij}^{c_{ij}}\right) = \sum_{i,j=1}^{n} c_{ij} \log P_{ij}$$

$$\begin{array}{ll} \text{maximize} & \sum_{i,j=1}^{n} c_{ij} \log P_{ij} \\ \text{subject to} & \mathbf{1}^T P = 1. \end{array}$$

$$L(P,\nu) = \sum_{i,j=1}^{n} c_{ij} \log P_{ij} + \sum_{j=1}^{n}\left(\nu_j \sum_{i=1}^{n} P_{ij}\right)$$

$$\frac{\partial L}{\partial P_{ij}}(P,\nu) = \frac{c_{ij}}{P_{ij}} + v_j$$

Setting $\frac{\partial L}{\partial P_{ij}}(P,\nu)$ equal to zero,

$$\frac{c_{ij}}{P_{ij}} = -\nu_j \quad \Longleftrightarrow \quad P_{ij} = \frac{-c_{ij}}{v_j}$$

Using the constraint

$$\mathbf{1}^T P = 1 \Longleftrightarrow \sum_{i=1}^{n} P_{ij}, \quad j = 1,\ldots n$$

plugging in fraction we found above

$$\sum_{i=1}^{n} \frac{-c_{ij}}{v_j} = 1, \quad j = 1,\ldots n \quad \Longleftrightarrow \quad v_j = \sum_{i=1}^{n} -c_{ij}, \quad j = 1,\ldots n.$$

Furthermore we find that

$$\hat{P}_{ij} = \frac{-c_{ij}}{v_j} = \frac{-c_{ij}}{\sum_{i=1}^{n} -c_{ij}} = \frac{c_{ij}}{\sum_{i=1}^{n} c_{ij}},$$

which is of course the empirical transition probability, as desired.

(b) To incorporate additional information about the transition probabilities that we are trying to estimate,

$$\begin{array}{ll} \text{maximize} & \sum_{i,j=1}^{n} c_{ij} \log P_{ij} \\ \text{subject to} & \mathbf{1}^T P = 1 \\ & Pq = q, \end{array}$$

still a convex problem. Could augment our previous Lagrangian with these additional equality constraints.

1.2 Generative Additive Model

Problem Overview and Upshot

Problem

[BV24]**Exercise 6.17**. *Fitting a generalized additive regression model.* A *generalized additive model* has the form

$$f(x) = \alpha + \sum_{j=1}^{n} f_j(x_j)$$

for $x \in \mathbf{R}^n$, where $\alpha \in \mathbf{R}$ is the offset, and $f_j : \mathbf{R} \to \mathbf{R}$, with $f_j(0) = 0$. The functions f_j are called the regressor functions. When each f_j is linear, i.e., has the form $w_j x_j$, the generalized additive model is the same as the standard (linear) regression model. Roughly speaking, a generalized additive model takes into account nonlinearities in each regressor x_j, but not nonlinear interactions among the regressors. To visualize a generalized additive model, it is common to plot each regressor function (when n is not too large). We will restrict the functions f_j to be piecewise-affine, with given knot points $p_1 < \cdots < p_K$. This means that f_j is affine on the intervals $(-\infty, p_1], [p_1, p_2], \ldots, [p_{K-1}, p_K], [p_K, \infty)$, and continuous at $p_1, \ldots, p_K$. Let C denote the total (absolute value of) change in slope across all regressor functions and all knot points. The value C is a measure of nonlinearity of the regressor functions; when $C = 0$, the generalized additive model reduces to a linear regression model. Now suppose we observe samples or data $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(N)}, y^{(N)}\right) \in \mathbf{R}^n \times \mathbf{R}$, and wish to fit a generalized additive model to the data. We choose the offset and the regressor functions to minimize

$$\frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - f(x^{(i)})\right)^2 + \lambda C.$$

where $\lambda > 0$ is a regularization parameter. (The first term is the mean-square error.)

(a) Explain how to solve this problem using convex optimization.

(b) Carry out the method of part (a) using the data in the file `gen_add_reg_data.m`. This file contains the data, given as an $N \times n$ matrix X (whose rows are $\left(x^{(i)}\right)^T$), a column vector y (which give $y^{(i)}$), a vector p that gives the knot points, and the scalar lambda. Give the mean-square error achieved by your generalized additive regression model. Compare the estimated and true regressor functions in a 3×3 array of plots (using the plotting code in the data file as a template), over the range $-10 \leq x_i \leq 10$. The true regressor functions (to be used only for plotting, of course) are given in the cell array f.

My Response

(a)

$$f_j(x) = \sum_{k=0}^{K} \theta_{nk+j} b_k(x)$$

where $b_k(x) = \max\{x - p_k, 0\} + \min\{p_k, 0\}$

Let

$$C = \sum_{j=1}^{n(K+1)} |\theta_j| = \|\theta\|_1$$

where $\theta \in \mathbf{R}^{n(K+1)}$

$$\text{minimize} \quad \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - (\alpha + \sum_{j=1}^{n} f_j(x_j^{(i)}))\right)^2 + \lambda C$$

1.3 Bounding Object Position

Problem Overview and Upshot

Problem

[BV24]**Exercise 8.8**. *Bounding object position from multiple camera views.* $x \in \mathbf{R}^3$, and viewed by a set of m cameras. Our goal is to find a box in $\mathbf{R}^3$,

$$\mathcal{B} = \left\{z \in \mathbf{R}^3 \mid l \preceq z \preceq u\right\},$$

for which we can guarantee $x \in \mathcal{B}$. We want the smallest possible such bounding box. (Although it doesn't matter, we can use volume to judge 'smallest' among boxes.)
Now we describe the cameras. The object at location $x \in \mathbf{R}^3$ creates an image on the image plane of camera i at location

$$v_i = \frac{1}{c_i^T x + d_i}\left(A_i x + b_i\right) \in \mathbf{R}^2.$$

The matrices $A_i \in \mathbf{R}^{2\times 3}$, vectors $b_i \in \mathbf{R}^2$ and $c_i \in \mathbf{R}^3$, and real numbers $d_i \in \mathbf{R}$ are known, and depend on the camera positions and orientations. We assume that $c_i^T x + d_i > 0$. The 3×4 matrix

$$P_i = \left[\begin{array}{cc} A_i & b_i \\ c_i^T & d_i \end{array}\right]$$

is called the camera matrix (for camera i). It is often (but not always) the case the that the first 3 columns of P_i (i.e., A_i stacked above c_i^T) form an orthogonal matrix, in which case the camera is called orthographic. We do not have direct access to the image point v_i; we only know the (square) pixel that it lies in. In other words, the camera gives us a measurement $\hat{v}_i$ (the center of the pixel that the image point lies in); we are guaranteed that

$$\|v_i - \hat{v}_i\|_\infty \leq \rho_i/2,$$

where ρ_i is the pixel width (and height) of camera i. (We know nothing else about v_i; it could be any point in this pixel.) Given the data $A_i, b_i, c_i, d_i, \hat{v}_i, \rho_i$, we are to find the smallest box $\mathcal{B}$ (i.e., find the vectors l and u) that is guaranteed to contain x. In other words, find

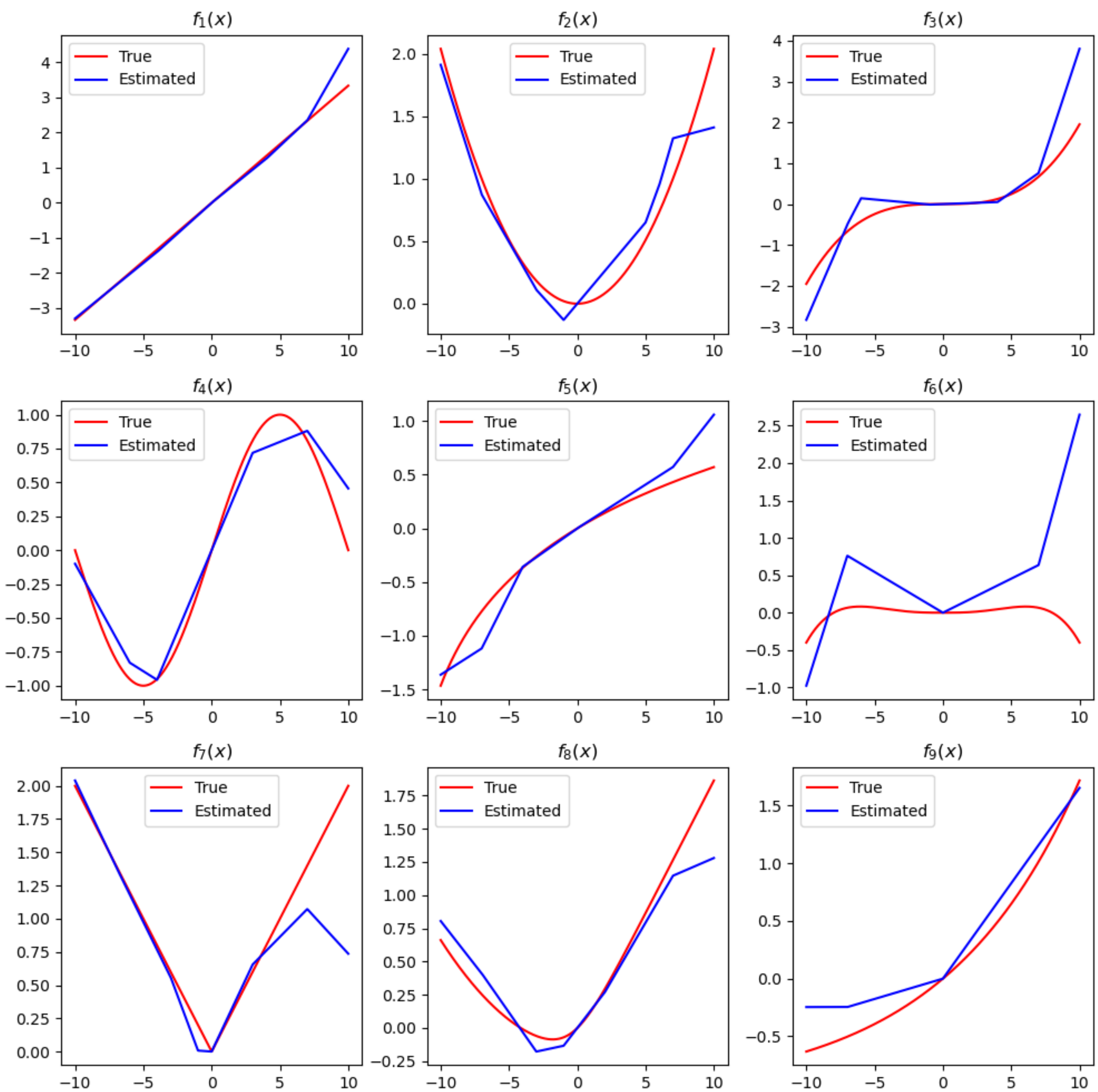

Figure 1: True versus Estimated Regressor Functions.

the smallest box in $\mathbf{R}^3$ that contains all points consistent with the observations from the camera.

(a) Explain how to solve this using convex or quasiconvex optimization. You must explain any transformations you use, any new variables you introduce, etc. If the convexity or quasiconvexity of any function in your formulation isn't obvious, be sure justify it.

(b) Solve the specific problem instance given in the file `camera_data.m`. Be sure that your final numerical answer (i.e., l and u) stands out.

My Response

x is some object in the physical world (modeled as 3-dimensional Euclidean space). We have m cameras which capture images of x. However, because

Let's first formulate the constraints ensuring that our proposed box that x lies in is consistent with the camera measurements; *i.e.*, there isn't a point in the bounding box which contra

$$\|v_i - \hat{v}_i\|_\infty \leq \rho_i/2,, \quad i = 1, \dots, m$$

is expanded as

$$\left\| \frac{1}{c_i^T x + d_i} (A_i x + b_i) - \hat{v}_i \right\|_\infty \leq \rho_i/2, \quad i = 1, \dots, m.$$

The ℓ_∞-norm is requiring that the absolute value of both terms in the $\mathbf{R}^2$ vector

$$\frac{1}{c_i^T x + d_i} (A_i x + b_i) - \hat{v}_i$$

be less than or qual to $\rho_i/2$. Furthermore,

$$-(\rho_i/2)\mathbf{1} \preceq \frac{1}{c_i^T x + d_i} (A_i x + b_i) - \hat{v}_i \preceq (\rho_i/2)\mathbf{1}, \quad i = 1, \dots, m$$

$$(\hat{v}_i - (\rho_i/2)\mathbf{1})(c_i^T x + d_i) \preceq A_i x + b_i \preceq (\hat{v}_i + (\rho_i/2))(c_i^T x + d_i), \quad i = 1, \dots, m$$

1.4 Robust Logistic Regression

Problem Overview and Upshot

Problem

[BV24] **Exercise 6.29**. *Robust Logistic Regresion.* We are given a dataset $x_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$, $i = 1, \dots, n$. We seek a prediction model $y \approx \hat{y} = \text{sign}\,(\theta^1 x)$, where $\theta \in \mathbf{R}^a$ is the model parameter. In logistic regression, θ is chosen as the minimizer of the logistic loss

$$\ell(\theta) = \sum_{i=1}^{n} \log\left(1 + \exp\left(-y_i \theta^T x_i\right)\right)$$

which is a convex function of θ. (We will assume that a minimizer exists.)

We will take into account the idea that the feature vectors x_i are not known precisely. Specifically we imagine that each entry of each feature vector can vary by $\pm\epsilon$, where $\epsilon > 0$ is a given uncertainty level. We define the worst-case logistic loss as

$$\ell^{\text{wc}}(\theta) = \sum_{i=1}^{n} \sup_{\|\delta_i\|_\infty \le \epsilon} \log\left(1 + \exp\left(-y_i \theta^T \left(x_i + \delta_i\right)\right)\right).$$

In words: we perturb each feature vector's entries by up to ϵ in such a way as to make the logistic loss as large as possible. Each term is convex, since it is the supremum of a family of convex functions of θ, and so $\ell^{\text{wc}}(\theta)$ is a convex function of θ.

In *robust logistic regression*, we choose θ to minimize $\ell^{\text{wc}}(\theta)$. (Here too we assume a minimizer exists.)

(a) Explain how to carry out robust logistic regression by solving a single convex optimization problem in disciplined convex programming (DCP) form. Justify any change of variables or introduction of new variables. Explain why solving the problem you propose also solves the robust logistic regression problem.

Hint: $\log(1 + \exp(u))$ is monotonic in u.

(b) Fit a logistic regression model (i.e., minimize $\ell(\theta)$), and also a robust logistic regression model (i.e., minimize $\ell^{\text{wc}}(\theta)$), using the data given in `rob_logistic_reg_data.py`. The x_i s are provided as the rows of a $n \times d$ matrix named x. The y_i s are provided as the entries of a n vector named y. The file also contains a test data set, `x_test`, `y_test`. Give the test error rate (i.e., fraction of test set data points for which $\hat{y} \neq y$) for the logistic regression and robust logistic regression models.

My Response

We will consider a generalization of this problem. Specifically, we'll actually define the worst-case logistic loss as

$$\ell^{\text{wc}}(\theta) = \sum_{i=1}^{n} \sup_{\|\delta_i\| \le \epsilon} \log\left(1 + \exp\left(-y_i \theta^T \left(x_i + \delta_i\right)\right)\right),$$

where $\|\delta_i\|$ is just *some norm* on $\mathbf{R}^d$. We will explore the reasoning behind this generalization after reformulating the model.

The robust logistic regression problem is to solve the unconstrained problem

$$\text{minimize} \quad \ell^{\text{wc}}(\theta) = \sum_{i=1}^{n} \sup_{\|\delta_i\| \le \epsilon} \log\left(1 + \exp\left(-y_i \theta^T \left(x_i + \delta_i\right)\right)\right).$$

However, while this *is* a convex optimization problem, *it is not* DCP compatible. This is because the supremum operator is an analytical expression; *i.e.,* (for our purposes), it cannot be handled by a computer. Furthermore, we need to reformulate the problem so the supremum operator is neither in the objective function nor in any constraint function. To begin reformulating, we use the hint and move the supremum from the objective function to the constraint set. The corresponding problem is,

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n \log\left(1+\exp\left(u_i\right)\right) \\ \text{subject to} & \sup\left\{-y_i\theta^T\left(x_i+\delta_i\right) \mid \|\delta_i\| \le \epsilon\right\} \le u_i, \quad i=1,\ldots,n, \end{array}$$

where now both $\theta \in \mathbf{R}^d$ and $u \in \mathbf{R}^n$ are optimization variables. To see that this problem is equivalent to the original unconstrained problem, consider holding θ fixed and optimizing solely over u. Because the log-sum-exp function is monotonic in its input and the objective function is separable, we achieve the optimum over u by choosing

$$u_i = \sup\left\{-y_i\theta^T\left(x_i+\delta_i\right) \mid \|\delta_i\| \le \epsilon\right\}$$

for $i = 1, \ldots n$.

Now, as previously mentioned, having the supremum operator in the constraint function set does not make the problem DCP compliant. Furthermore, we continue and finish the reformulation effort by first pulling terms out of the supremum:

$$\sup\left\{-y_i\theta^T\left(x_i+\delta_i\right) \mid \|\delta_i\| \le \epsilon\right\} \le u_i, \quad i=1,\ldots,n,$$

is equivalent to

$$-y_i\theta^T x_i + \sup\left\{-y_i\theta^T\delta_i \mid \|\delta_i\| \le \epsilon\right\} \le u_i \quad i=1,\ldots,n,$$

and then scaling the expression within the supremum to recognize that

$$\sup\left\{-y_i\theta^T\delta_i \mid \|\delta_i\| \le \epsilon\right\} = \sup\left\{-\epsilon y_i\theta^T\delta_i \mid \|\delta_i\| \le 1\right\} = \|-\epsilon y_i\theta\|_*,$$

where $\|\cdot\|_*$ is the *dual norm* of whatever norm we choose to define our uncertainty set (again, more on this below). Therefore, we've found that our original unconstrained, non-DCP compatible robust logistic regression optimization problem

$$\text{minimize} \quad \ell^{\text{wc}}(\theta) = \sum_{i=1}^n \sup_{\|\delta_i\|\le\epsilon} \log\left(1+\exp\left(-y_i\theta^T\left(x_i+\delta_i\right)\right)\right)$$

is equivalent to the following constrained, DCP compatible robust logistic regression optimization problem

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n \log\left(1+\exp\left(u_i\right)\right) \\ \text{subject to} & -y_i\theta^T x_i + \|-\epsilon y_i\theta\|_* \le u_i, \quad i=1,\ldots,n. \end{array}$$

2 Design and Control

2.1 Minimum Fuel Control

Problem Overview and Upshot

Problem

[BV04] **Exercise 4.16**. *Minimum fuel optimal control.* Consider the LTI dynamical system with state $x_t \in \mathbf{R}^n$, $t = 0, \ldots, N$, and actuator or input signal $u_t \in \mathbf{R}$, for $t = 0, \ldots, N-1$. The dynamics of the system are governed by the linear recurrence

$$x_{t+1} = Ax_t + bu_t, \quad t = 0, \ldots N-1,$$

where $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^n$ are given. Assume the initial state is $x_0 = 0$. The *minimum fuel optimal control problem* is to choose the inputs $u_0, \ldots u_{N-1}$ so as to minimize the total fuel consumed, which is given by

$$F = \sum_{t=0}^{N-1} f(u_t),$$

subject to the constraint that $x_N = x_{\text{des}}$, where N is the (given) time horizon and $x_{\text{des}} \in \mathbf{R}^n$ is the (given) desired final or target state. The function $f : \mathbf{R} \to \mathbf{R}$ is the *fuel use map* for the actuator, and gives the amount of fuel used as a function of the actuator signal amplitude. In this problem we use

$$f(a) = \begin{cases} |a| & |a| \leq 1 \\ 2|a| - 1 & |a| > 1. \end{cases}$$

Formulate the minimum fuel control problem as an LP.

Response

We want to write this minimum fuel optimal control problem as a LP. However, a LP formulation is rather restrictive, so let's begin by formulating this problem as a convex optimization problem. It is tempting to naively claim that the problem

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{N-1} f(u_t) \\ \text{subject to} & x_{t+1} = Ax_t + bu_t, \qquad t = 0, \ldots, N-1 \\ & x_0 = 0, \quad x_N = x_{\text{des}}. \end{array}$$

is convex. However, we must remember that f is a piecewise function that we are unfamiliar with (*e.g.* it isn't a piecwise linear function). Furthermore, this proprosed formulation is not a conex optimization problem. Consider the graph of the fuel use map in figure 2. As stated in the legend of this figure, and clearly seen when observing the graph, the piecewise function f is equivalent to $\max\{|a|, 2|a| - 1\}$. (Of course one could also provide an algebraic argument if so desired.) Furthermore, the optimal fuel control problem can be formulated as

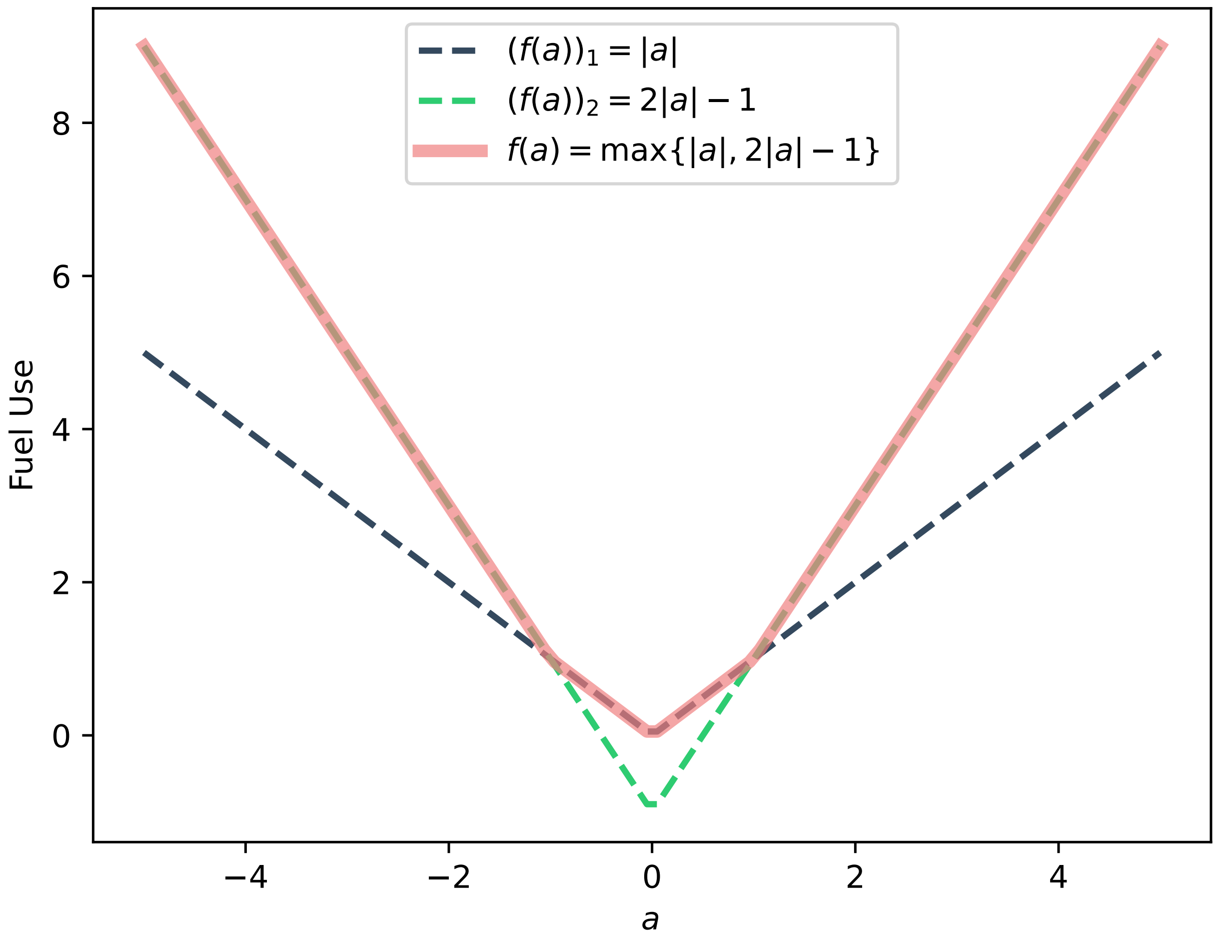

Figure 2: Actuator Fuel Use Map.

$$
\begin{array}{lll}
\text{minimize} & \sum_{t=0}^{N-1} \max\{|u_t|, 2|u_t| - 1\} & \\
\text{subject to} & x_{t+1} = Ax_t + bu_t, & t = 0, \dots, N-1 \\
& x_0 = 0, \quad x_N = x_{\text{des}}, &
\end{array}
$$

which is a valid convex optimization problem. If our goal was simply to find the optimal solution to this fuel problem and we didn't care about formulating the problem as a LP, **we could end our reformulating here.** In fact, the optimal solution $u^* \in \mathbf{R}^{Nm}$, where $Nm = N1 = N$, plotted in figure 3 was obtained by solving this convex problem. Nonetheless, we trudge forward with our LP formulation.

For the time being, to be more concise, let's drop the linear dynamical system constraints

$$x_0 = 0,\ x_N = x_{\text{des}}, \quad \text{and} \quad x_{t+1} = Ax_t + bu_t, \quad t = 0, \dots N-1$$

and just consider the unconstrained problem

$$\text{minimize} \sum_{t=0}^{N-1} \max\{|u_t|, 2|u_t| - 1\}.$$

Our first reformulation uses that an objective function defined as the sum of absolute value/maximum expressions can be rewritten as a sum of auxilary variables, $s \in \mathbf{R}^N$ here, with each summand being less than or equal to an element in the auxilary variable vector. To simplify the problem further we can also remove the max operator from each summand using that if the maximum element in the set being operated on by max is less than or equal to s_t, then so must every other element. These two reformulation techniques yield the problem

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T s \\ \text{subject to} & |u_t| \leq s_t, \qquad\quad t = 1, \dots N \\ & 2\,|u_t| - 1 \leq s_t, \quad t = 1, \dots N. \end{array}$$

To fully linearize the problem, consider $y \in \mathbf{R}^N$ and the problem

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T s \\ \text{subject to} & y \preceq s \\ & 2y - \mathbf{1} \preceq s \\ & -y \preceq u \preceq y, \end{array}$$

which uses this new variable to “pull out” the absolute value from the two other sets of constraints. Now, before returning the LDS constraints to the formulation observe the pattern in the LDS defining recurrence,

$$\begin{aligned} x_1 &= Ax_0 + bu_0 \\ x_2 &= Ax_1 + bu_1 \\ &= A(Ax_0 + bu_0) + bu_1 = A^2 x_0 + Abu_0 + bu_1 \\ x_3 &= A^3 x_0 + A^2 b_0 + Abu_1 + bu_2. \\ &\vdots \end{aligned}$$

Plugging x_0 and x_{des} into the recurrence, the system evolution constraints are introduced back into the optimization problem

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T s \\ \text{subject to} & Hu = x_{\text{des}} \\ & y \preceq s \\ & 2y - \mathbf{1} \preceq s \\ & -y \preceq u \preceq y, \end{array}$$

with the equality constraints $Hu = x_{\text{des}}$, where

$$H = \begin{bmatrix} A^{N-1}b & A^{N-2}b & \cdots & Ab & b \end{bmatrix}.$$

This LP is equivalent to the original minimum fuel optimal control problem, with the corresponding optimal actuator signal plotted in figure 3.

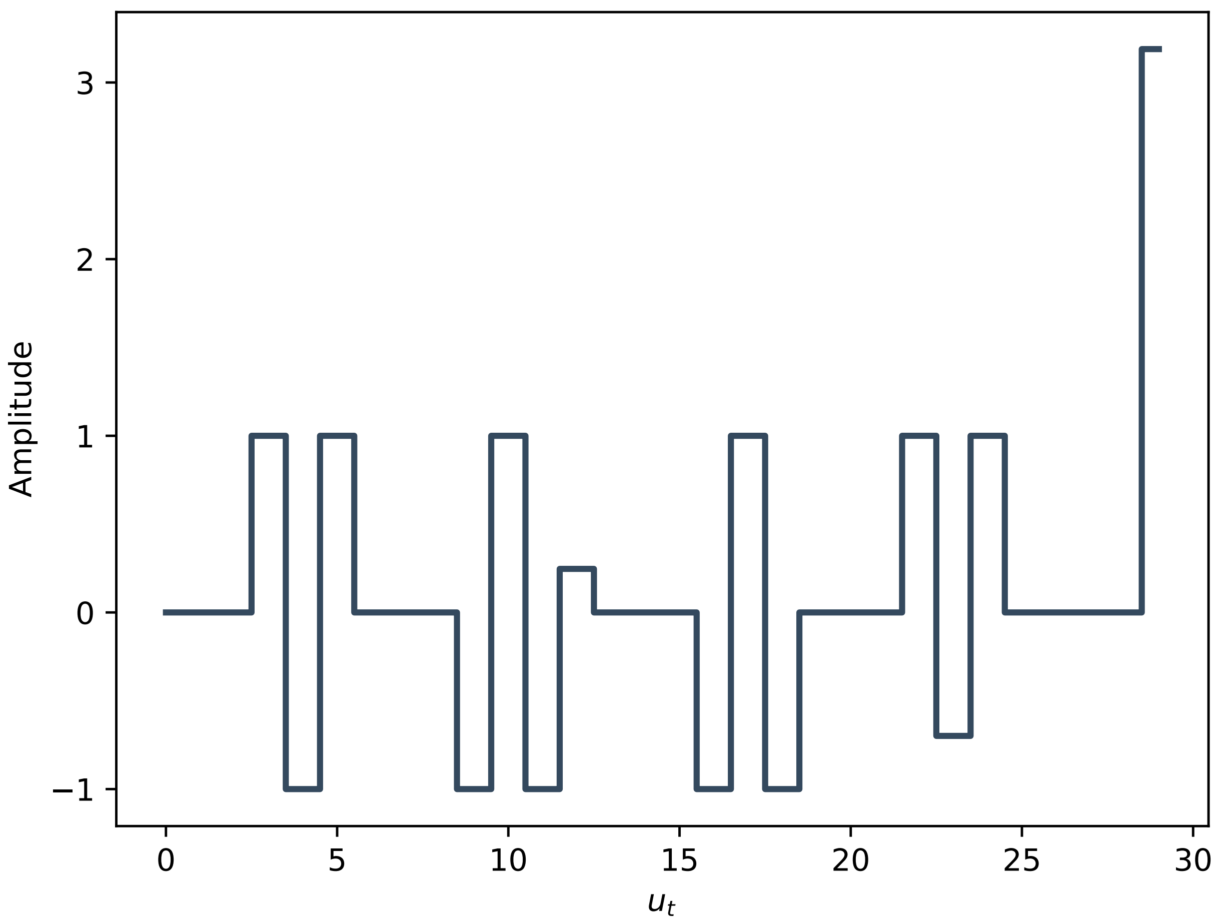

Figure 3: Minimum fuel actuator signal.

2.2 Path Planning with Contingencies

Problem Overview and Upshot

Problem

[BV24] **Exercise 16.9**. *Path Planning with contingencies.* A vehicle path down a (straight, for simplicity) road is specified by a vector $p \in \mathbf{R}^N$, where p_i gives the position perpendicular to the centerline at the point ih meters down the road, where $h > 0$ is a given discretization size. (Throughout this problem, indexes on N-vectors will correspond to positions on the road.) We normalize p so $-1 \leq p_i \leq 1$ gives the road boundaries. (We are modeling the vehicle as a point, by adjusting for its width.) You are given the initial two positions $p_1 = a$ and $p_2 = b$ (which give the initial road position and angle), as well as the final two positions $p_{N-1} = c$ and $p_N = d$. You know there may be an obstruction at position $i = O$. This will require the path to either go around the obstruction on the left, which requires $p_O \geq 0.5$, or on the right, which requires $p_O \leq -0.5$, or possibly the obstruction will clear, and the obstruction does not place any additional constraint on the path. These are the three contingencies in the problem title, which we label as $k = 1, 2, 3$. You will plan three paths for these contingencies, $p^{(i)} \in \mathbf{R}^N$ for $i = 1, 2, 3$. They must each satisfy the given initial and final two road positions and the constraint of staying within the road boundaries. Paths $p^{(1)}$ and $p^{(2)}$ must satisfy the (different) obstacle avoidance constraints given above. Path $p^{(3)}$ does not need to satisfy an avoidance constraint. Now we add a twist: You will not learn which of the three contingencies will occur until the vehicle arrives at position $i = S$, when the sensors will determine which contingency holds. We model this with the information constraints (also called causality constraints or non-anticipatory constraints),

$$p_i^{(1)} = p_i^{(2)} = p_i^{(3)}, \quad i = 1, \ldots, S,$$

which state that before you know which contingency holds, the three paths must be the same. The objective to be minimized is

$$\sum_{k=1}^{3} \sum_{i=2}^{N-1} \left(p_{i-1}^{(k)} - 2p_i^{(k)} + p_{i+1}^{(k)} \right)^2,$$

the sum of the squares of the second differences, which gives smooth paths.

(a) Explain how to solve this problem using convex optimization.

(b) Solve the problem with data given in `path_plan_contingencies_data.*`. The data files include code to plot the results, which you should use to plot (on one plot) the optimal paths. Report the optimal objective value. Give a very brief informal explanation for what you see happening for $i = 1, \ldots, S$. Hint. In Python, use the (default) solver ECOS to avoid warnings about inaccurate solutions.

My Response

$$
\begin{array}{lll}
\text{minimize} & \sum_{k=1}^{3}\sum_{i=2}^{N-1}\left(p_{i-1}^{(k)} - 2p_i^{(k)} + p_{i+1}^{(k)}\right)^2 & \\
\text{subject to} & p_i^{(1)} = p_i^{(2)} = p_i^{(3)}, & i = 1, \dots, S \\
& p_O^{(1)} \geq 1/2, & \\
& p_O^{(2)} \leq -1/2, & \\
& p_1^{(i)} = a, \quad p_2^{(i)} = b, & i = 1, 2, 3 \\
& p_{N-1}^{(i)} = c, \quad p_N^{(i)} = d, & i = 1, 2, 3.
\end{array}
$$

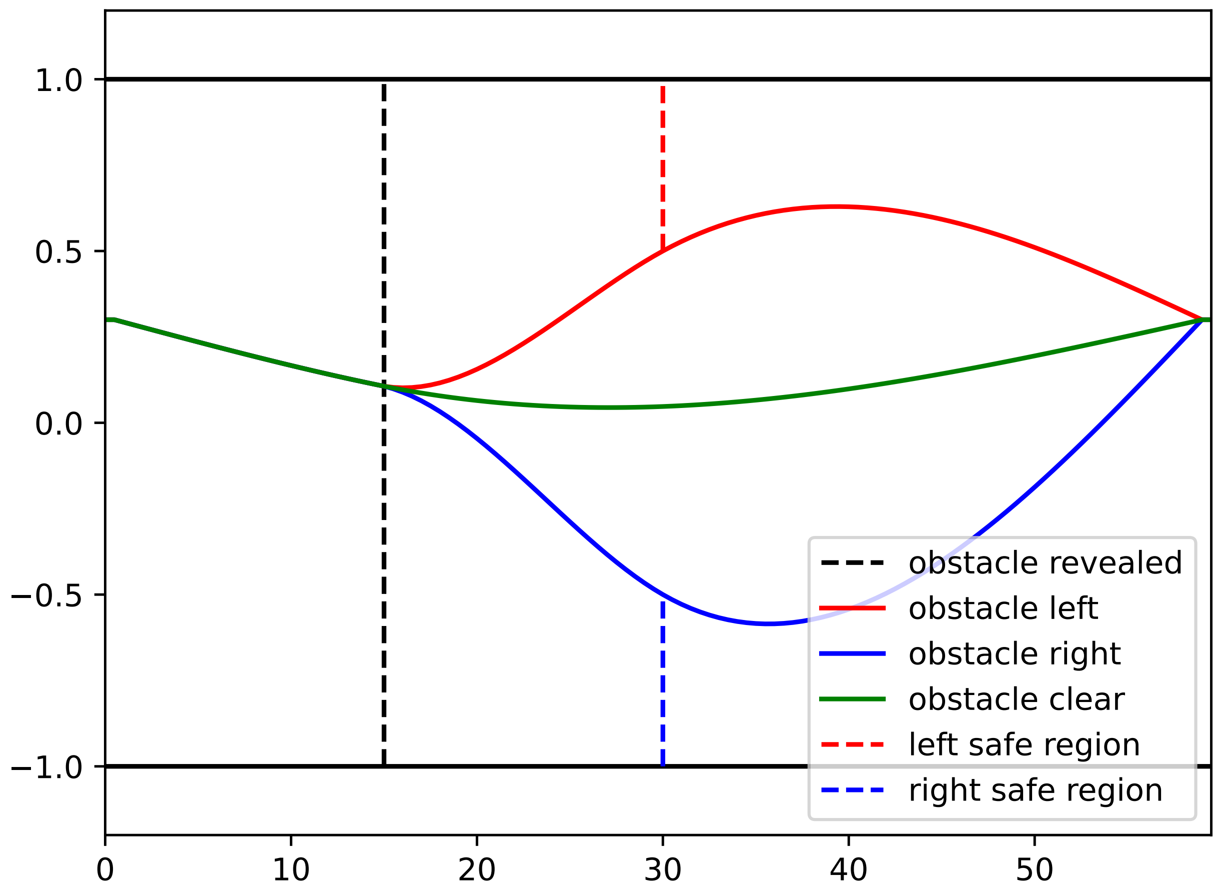

Figure 4: Possible Paths.

2.3 Output Tracking

Problem Overview and Upshot

Problem

[Boy-8] **HW7 Q1.** *MPC for output tracking.* Consider the linear dynamical system

$$x_{t+1} = Ax_t + Bu_t, \quad y_t = Cx_t, \quad t = 0, \ldots, T-1,$$

with state $x_t \in \mathbf{R}^n$, input $u_t \in \mathbf{R}^m$, and output $y_t \in \mathbf{R}^p$. The matrices A and B are known, and $x_0 = 0$. The goal is to choose the input sequence $u_1, \ldots, u_t$ to minimize the output tracking cost

$$J_{\text{output}} = \sum_{t=1}^{T} \left\| y_t - y_t^{\text{des}} \right\|_2^2,$$

subject to $\|u_t\|_\infty \leq U^{\max}$, $t = 0, \ldots, T-1$.
For the remainder of this problem we work with the specific problem instance with associated data

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.5 \\ 1 \end{bmatrix}, \quad C = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix},$$

$T = 100$, and $U^{\max} = 0.1$. The desired output trajectory is given by

$$y_t^{\text{des}} = \begin{cases} 0 & t < 30, \\ 10 & 30 \leq t < 70 \\ 0 & t \geq 70. \end{cases}$$

(a) Find the optimal input u^* and the associated optimal cost J^*.

(b) *Rolling look-ahead.* Now consider the input obtained using an MPC-like method where at time t, we find the values of $u_t, \ldots, u_{t+N-1}$ that solve the following convex optimization problem

$$\begin{array}{ll} \text{minimize} & J_{\text{output}} = \sum_{\tau=t+1}^{t+N} \left\| Cx_\tau - y_\tau^{\text{des}} \right\|_2^2 \\ \text{subject to} & \|u_\tau\|_\infty \leq 0.1, \quad x_{\tau+1} = Ax_\tau + Bu_\tau, \quad \tau = t, \ldots, t+N-1 \\ & x_0 = 0. \end{array}$$

The value N is the amount of *look-ahead*, since it dictates how much of the future of the desired output signal we are allowed to access when we decide on the current input.
Find the input signal for look-ahead values $N = 8$, $N = 10$, and $N = 12$. Compare the cost J_{output} obtained in these three instances to the optimal cost J^*_{output} found in part (a).

My Response

This is simply a *linear* (in the dynamics) *time-invariant quadratic tracking* problem. Instead of doing a theoretical analysis of controllability, etc., we can determine the feasibility of the control problem by formulating and attempting to solve the following convex optimization problem:

$$\begin{array}{ll} \text{minimize} & J_{\text{output}} = \sum_{t=1}^{100} \left\| Cx_t - y_t^{\text{des}} \right\|_2^2 \\ \text{subject to} & \|u_t\|_\infty \leq 0.1, \quad x_{t+1} = Ax_t + Bu_t, \quad t = 0, \ldots, 99 \\ & x_0 = 0, \end{array}$$

(with the provided data for A, B, C, and y^{des}, of course.) Using CVXPY, we obtain the optimal cost $J^*_{\text{outout}} = 112.4157$.
(b) The Python code used to solve this problem can be found in these note's associated examples folder under 364b, so it is omitted here. The cost obtained by applying MPC with

- $N = 8$ is 379.634,
- $N = 10$ is 128.13,
- and $N = 12$ is 123.62.

Figure 5 shows the output trajectories for $N = 8$, $N = 10$, $N = 12$, the optimal output trajectory, and the desired output trajectory.

2.4 UAV Design

Problem Overview and Upshot

Problem

[BV24] **Exercise 18.14**. *Design of an unmanned aerial vehicle.* You are tasked with developing the high-level design for an electric unmanned aerial vehicle (UAV). The goal is to design the least expensive UAV that is able to complete K missions, labeled $k = 1, \ldots, K$. Mission k involves transporting a payload of weight $W_k^{\text{pay}} > 0$ (in kilograms) over a distance $D_k > 0$ (in meters), at a speed $V_k > 0$ (in meters per second). These mission quantities are given. The high-level design consists of choosing the engine weight W^{eng} (in kilograms), the battery weight W^{bat} (in kilograms), and the wing area S (in m^2), within the given limits

$$W_{\min}^{\text{eng}} \leq W^{\text{eng}} \leq W_{\max}^{\text{eng}}, \quad W_{\min}^{\text{bat}} \leq W^{\text{bat}} \leq W_{\max}^{\text{bat}}, \quad S_{\min} \leq S \leq S_{\max}.$$

(The lower limits are all positive.) We refer to the variables W^{eng}, W^{bat}, and S as the design variables. In addition to choosing the design variables, you must choose the power $P_k > 0$ (in watts) that flows from the battery to the engine, and the angle of attack $\alpha_k > 0$ (in degrees) of the UAV during mission k, for $k = 1, \ldots, K$. These must satisfy

$$0 \leq P_k \leq P_{\max}, \quad 0 \leq \alpha_k \leq \alpha_{\max},$$

where $\alpha_{\max}$ is given, and $P_{\max}$ depends on the engine weight as described below. We refer to these $2K$ variables as the mission variables. The engine weight, battery weight, and wing area are the same for all k missions; the power and angle of attack can change with the

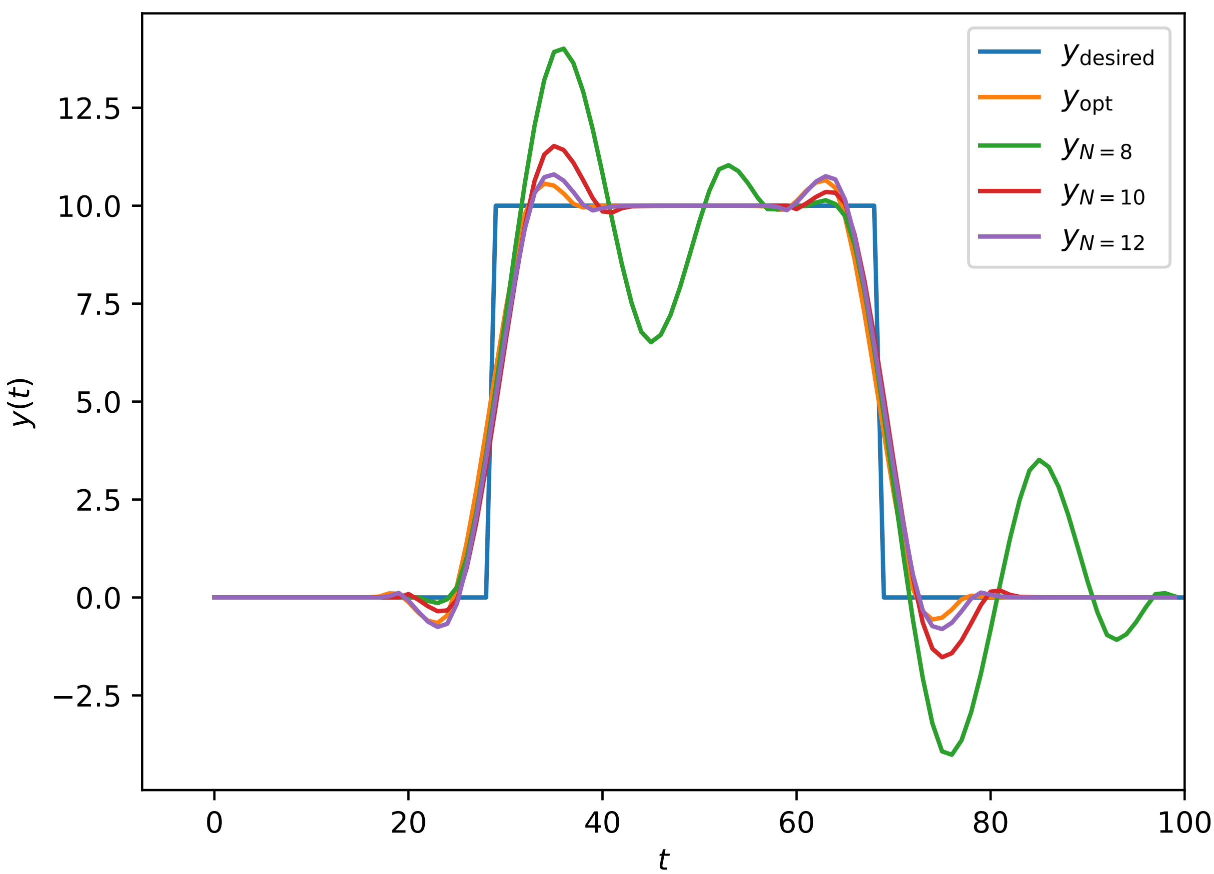

Figure 5: Output Trajectories.

mission. The weight of the wing is W^{wing} (in kilograms) is given by $W^{\text{wing}} = C_W S^{1.2}$, where $C_W > 0$ is given. The total weight of the UAV during mission k, denoted W_k, is the sum of the battery weight, engine weight, wing weight, the payload weight, and a baseline weight W^{base}, which is given. The total weight depends on the mission, via the payload weight, and so is subscripted by k. The lift and drag forces acting on the UAV in mission k are

$$F_k^{\text{lift}} = \frac{1}{2}\rho V_k^2 C_L(\alpha_k) S, \quad F_k^{\text{drag}} = \frac{1}{2}\rho V_k^2 C_D(\alpha_k) S$$

(in newtons), where C_L and C_D are the lift and drag coefficients as functions of the angle of attack α_k, and $\rho > 0$ is the (known) air density (in kilograms per cubic meter). We will use the simple functions

$$C_L(\alpha) = c_L \alpha, \quad C_D(\alpha) = c_{D1} + c_{D0}\alpha^2,$$

where $c_L > 0, c_{D0} > 0$, and $c_{D1} > 0$ are given constants. To maintain steady level flight, the lift must equal the weight, and the drag must equal the thrust from the propeller, denoted T_k (in newtons), i.e.,

$$F_k^{\text{lift}} = W_k, \quad F_k^{\text{drag}} = T_k.$$

The thrust force, power P_k (in watts), and the UAV speed are related via $P_k = T_k V_k$. The engine maximum power is related to its weight by $W^{\text{eng}} = C_P P_{\max}^{0.803}$ where $C_P > 0$ is given. The battery capacity E (in joules) is equal to $C_E W^{\text{bat}}$, where $C_E > 0$ is given. The total energy expended over mission k, with speed V_k, power output P_k, and distance D_k is $P_k D_k / V_k$. This must not exceed the battery capacity E. The overal cost of the UAV is the sum of a design cost and a mission cost. The design cost C_{des}, which is an approximation of the cost of building the UAV, is given by

$$C_{\text{des}} = 100 W^{\text{eng}} + 45 W^{\text{bat}} + 2 W^{\text{wing}}.$$

The mission cost C_{mis} is given by

$$C_{\text{mis}} = \sum_{k=1}^{K} (T_k + 10\alpha_k),$$

which captures our desire that the thrust and angle of attack be small. In summary, $W_{\min}^{\text{eng}}, W_{\max}^{\text{eng}}, W_{\min}^{\text{bat}}, W_{\max}^{\text{bat}}, S_{\min}, S_{\max}, \alpha_{\max}, W_{\text{base}}, C_W, c_L, c_{D0}, c_{D1}, C_P, C_E,$ and ρ are given. Additionally, D_k, V_k, and W_k^{pay} are given for $k = 1, \ldots, K$.

(a) The problem as stated is almost a geometric problem (GP). By relaxing two constraints it becomes a GP, and therefore readily solved. Identify these constraints and give the relaxed versions. Briefly explain why the relaxed constraints will be tight at the solution, which means by solving the GP, you've actually solved the original problem. You do not need to reduce the relaxed problem to a standard form GP, or the equivalent convex problem; it's enough to express it in DGP compatible form.

(b) Solve the relaxed problem you formulate in part (a) with data given in the provided python file. Give the optimal costs C_{des}^* and C_{mis}^*, and the values of all design and mission variables. Check that at your solution the relaxed constraints are tight.

My Response

Recall that an optimization problem of the form

$$\begin{array}{ll}\text{minimize} & f_0(x)\\ \text{subject to} & f_i(x) \le 1, \quad i = 1, \dots, m\\ & h_i(x) = 1, \quad i = 1, \dots, p,\end{array}$$

where $f_0, \dots f_m$ are posynomials and $h_1, \dots h_p$ are monomials is a *geometric program* (GP) in *standard form.* (Note that $\mathcal{D} = \mathbf{R}^n_{++}$; the constraint $x \succ 0$ is implicit.)

Most importantly, note that the equality constraint functions must be monomials. This requirement helps direct our search for invalid constraints, *i.e.*, we should look for constraints, which as formulated "verbally," would require posynomial equalities. The first such constraint(s) is

$$F_k^{\text{lift}} = W_k, \quad k = 1, \dots, K,$$

since

$$\begin{aligned} W_k &= W^{\text{bat}} + W^{\text{eng}} + W^{\text{wing}} + W_k^{\text{pay}} + W^{\text{base}}\\ &= W^{\text{bat}} + W^{\text{eng}} + C_W S^{1.2} + W_k^{\text{pay}} + W^{\text{base}},\end{aligned}$$

is a *posynomial* and F_k^{lift} is a *monomial.* Because posynomials are *closed under division*, the proposed constraint

$$F_k^{\text{lift}} = W_k \Longleftrightarrow \left(F_k^{\text{lift}}\right)^{-1} W_k = 1, \quad k = 1, \dots, K,$$

is a posynomial equality constraint, which to emphasize again, is invalid for GP formulation.

For the same reason, the constraint(s)

$$F_k^{\text{drag}} = T_k, \quad k = 1, \dots, K,$$

is also invalid. However, in this case, it is the force term, F_k^{drag}, which is the posynomial and $T_k = (1/V_k)P_k$ which is the monomial. To formulate a proper GP, we therefore make the following two (or more accurately, $2K$) relaxations

$$W_k = F_k^{\text{lift}}, \quad \text{and} \quad F_k^{\text{drag}} = T_k, \quad k = 1, \dots, K$$

become

$$W_k \le F_k^{\text{lift}}, \quad \text{and} \quad F_k^{\text{drag}} \le T_k, \quad k = 1, \dots, K,$$

or equivalently (and in GP standard form)

$$\left(F_k^{\text{lift}}\right)^{-1} W_k \le 1, \quad \text{and} \quad T_k^{-1} F_k^{\text{drag}} \le 1, \quad k = 1, \dots, K,$$

in the relaxed formulation.

It is now highly tempting to start writing down an optimization problem by just reading off the problem specifications. *This is discouraged.* Even though we have found and proposed relaxations for the two originally invalid GP constraints, it is still *very easy* to formulate a *non-DGP compatible problem.* As an example, a common (and understandable) mistake would be to include the constraints

$$0 \leq P_k, \quad k = 1, \dots, K,$$

in the formulation. However, **including them** would make the formulation **non-DGP compatible**. All constraints, equality or inequality, must have a 1 on the right-hand side of the constraint when reduced to standard form. The constraints $0 \leq P_k$, $k = 1, \dots, K$ cannot be reduced to this form. However, these are valid GP constraints! They are just included *implicitly* since the domain of the problem is $\mathbf{R}^n_{++}$. Including them *explicitly* leads to a non-DGP compatible formulation.

Additionally, unlike in LPs, where additional "linking" constraints are harmless, adding such syntactic constraints here can, again, lead to a non-DGP compatible problem. As an example, consider the specification that the lift force acting on the UAV in mission k is

$$F_k^{\text{drag}} = \frac{1}{2}\rho V_k^2 C_D(\alpha_k) S,$$

which as previously discussed is a *posynomial.* In an LP formulation (at least, as commonly taught in an operations research setting), it is encouraged (and would be totally valid) to declare F_k^{drag} as an optimization variable, include the above equality in the LP **as a constraint**, and then use F_k^{drag} throughout the remainder of the problem in constraints involving drag force. However, this "linking" constraint is **non-DGP compatible** because it would be reduced to a posynomial equality constraint, which as discussed at greath length above, is **not a valid GP constraint**.

Furthermore, we must distinguish between actual *optimization variables* and what I'll call *derived optimization variables.* The former are what they are always defined as: the variables that are to be chosen to minimize the objective function. We will define the latter as a mathematical expression whose value is decided by the optimization variables. To use a Computer Science dialect, the derived optimization variables are functions of references to the optimization variables (where the function can take multiple optimization variables and even other derived optimization variables). Consequently, we can use either the derived variables or the actual variables when formulating our problem. As an example, the constraint

$$\frac{1}{2}\rho V_k^2 C_D(\alpha_k) S \leq (1/V_k) P_k, \quad k = 1, \dots, K$$

is equivalent to

$$F_k^{\text{drag}} \leq T_k, \quad k = 1, \dots, K.$$

However, $F_k^{\text{drag}} = \frac{1}{2}\rho V_k^2 C_D(\alpha_k) S$ and $T_k = (1/V_k) P_k$, $k = 1, \dots, K$, should **not** be included in the problem constraints and F_k^{drag}, T_k, $k = 1 \dots, K$ **are not** optimization variables. They both are merely expressions *encoding* optimization variables. As a final attempt to stress the difference, when building this problem with a declarative language such as CVXPY in Python, the problem should be created as follows (and see Algorithm 1 for the actual code)

1. Declare optimization variables. Ex. `opt_var = cvxpy.Variable(num_entries)` (notice the explicit use of the `Variable` keyword).

2. Create derived optimization variables. Ex. `der_var = 2.5*opt_var` (this becomes a `cvxpy.Expression`).

3. Create the constraints and objective using the optimization variables, the derived optimization variables, or some combination of the two. Ex. `der_var <= 5`, which would be equivalent to adding the constraint `opt_var <= 2`.

4. Solve the problem.

This is a nuanced point, and perhaps is obvious to readers who do not have training in OR formulating LPs, but it is critically important.

We now put everything together; the following is the Relaxed UAV Design Problem.

Optimization Variables

- Design Variables: W^{eng}, W^{bat}, S.
- Mission Variables: P_k, α_k, $\quad k = 1, \ldots, K$.

Derived Variables/CVXPY expressions

- $W^{\text{wing}} = C_W S^{1.2}$,
- $W_k = W^{\text{bat}} + W^{\text{eng}} + W^{\text{wing}} + W_k^{\text{pay}} + W^{\text{base}}$,
- $F_k^{\text{lift}} = \frac{1}{2}\rho V_k^2 C_L\left(\alpha_k\right) S, \quad k = 1, \ldots, K$,
- $F_k^{\text{drag}} = \frac{1}{2}\rho V_k^2 C_D\left(\alpha_k\right) S, \quad k = 1, \ldots, K$,
- $P_{\max} = \left(C_P^{-1} W^{\text{eng}}\right)^{1/0.803}$,
- $T_k = \left(1/V_k\right) P_k, \quad k = 1, \ldots, K$,
- $E = C_E W^{\text{bat}}$,
- $C_{\text{des}} = 100 W^{\text{eng}} + 45 W^{\text{bat}} + 2 W^{\text{wing}}$,
- $C_{\text{des}} = \sum_{k=1}^{K} \left(T_k + 10\alpha_k\right)$.

Relaxed Formulation

$$
\begin{array}{ll}
\text{minimize} & C_{\text{des}} + C_{\text{mis}} \\
\text{subject to} & W_{\min}^{\text{eng}} \leq W^{\text{eng}} \leq W_{\max}^{\text{eng}}, \quad W_{\min}^{\text{bat}} \leq W^{\text{bat}} \leq W_{\max}^{\text{bat}}, \quad S_{\min} \leq S \leq S_{\max} \\
& P_k \leq P_{\max}, \quad \alpha_k \leq \alpha_{\max}, \quad k = 1, \ldots, K \\
& F_k^{\text{lift}} \geq W_k, \quad k = 1, \ldots K \\
& F_k^{\text{drag}} \leq T_k, \quad k = 1, \ldots K \\
& (1/V_k) P_k D_k \leq E, \quad k = 1, \ldots K.
\end{array}
$$

Problem Solution and Analysis.

Firstly, we address why the relaxed constraints, $F_k^{\text{lift}} \geq W_k$ and $F_k^{\text{drag}} \leq T_k$, $k = 1, \ldots, K$,

will be tight at the solution.
The tightness in the constraint $F_k^{\text{drag}} \leq T_k$ at the solution is easy to see. We proceed with a semi-formal argument. Suppose that the drag inequality for mission k holds strictly. That is, $F_k^{\text{drag}} < T_k$. If this is the case, then we can reduce the thrust for mission k by decreasing the power, P_k, that flows from the battery to the engine during that mission as T_k strictly increases and strictly decreases as P_k increases or decreases. We are able to decrease the power because the only lower bound on P_k is the implicit lower bound $P_k \geq 0$, and if it is the case that $P_k \to 0$, then F_k^{drag} must also go to zero since it too is positive. As both terms go to zero, we end up violating the original assumption that F_k^{drag} is strictly less than the thrust. Furthermore, being able to reduce T_k contradicts the assumption that we are at the solution since shrinking T_k lowers the mission cost, and thus the overall cost.
A nearly identical argument can be made for the tightness in the constraint F_k^{lift} with respect to the attack angle optimization variable α_k.

(b) The Python code in Algorithm 1 contains the core Python code required to compute the optimal UAV design (it doesn't include constant instantiations and helper functions). The corresponding optimal values are

- $C_{\text{des}}^* = 4449.324$,
- $C_{\text{mis}}^* = 1556.587$,
- $W^{\text{eng}} = 9.951\,\text{kg}$,
- $W^{\text{bat}} = 74.06\,\text{kg}$,
- $S = 7.999\,\text{m}^2$,
- the power values are

$$\begin{aligned} P_1 =& 40.159\,\text{kW}, \quad P_2 = 12.142\,\text{kW}, \quad P_3 = 20.018\,\text{kW}, \\ & P_4 = 24.998\,\text{kW}, \quad P_5 = 8.080\,\text{kW}, \end{aligned}$$

- and the attack angles are

$$\alpha_1 = 0.1377°, \quad \alpha_2 = 0.3141°, \quad \alpha_3 = 0.2249°, \quad \alpha_4 = 0.2114°, \quad \alpha_5 = 0.3785°.$$

Finally, in table 1 we see that the relaxed constraints are indeed tight.

k	T_k	F_k^{drag}	F_k^{lift}	W_k
1	489.739 N	489.739 N	269.648 N	269.646 N
2	220.763 N	220.763 N	276.647 N	276.646 N
3	307.969 N	307.969 N	276.648 N	276.646 N
4	357.119 N	357.119 N	301.648 N	301.646 N
5	168.332 N	168.332 N	253.946 N	253.946 N

Table 1: Tightness in Relaxed Constraints.

Algorithm 1: UAV Design Python Code

```python
            ### optimization variables ###
            W_eng = cp.Variable(1, pos=True)
            W_bat = cp.Variable(1, pos=True)
            S = cp.Variable(1, pos=True)
            P = cp.Variable(K, pos=True)
            alpha = cp.Variable(K, pos=True)

            ### derived expressions ###
            W_wing = CW*S**(1.2)
            W_k = [W_bat + W_eng + W_wing + W_base + W_pay[k] for k in range(K)]
            P_max = ((1/CP)**(1/0.803)) * (W_eng**(1/0.803))
            F_k_lift = [0.5*rho* V[k]**2 * C_L(alpha[k]) * S for k in range(K)]
            F_k_drag = [0.5*rho * V[k]**2 * C_D(alpha[k])*S for k in range(K)]
            T_k = [P[k] * (1/V[k]) for k in range(K)]
            E = CE*W_bat
            C_des = 100*W_eng + 45*W_bat + 2 * W_wing
            C_mis = cp.sum([T_k[k] + 10*alpha[k] for k in range(K)])
            ### ###

            ### constraints and objective ###
            constraints = [W_eng_min <= W_eng, W_eng <= W_eng_max,
                           W_bat_min <= W_bat, W_bat <= W_bat_max,
                           S_min <= S, S <= S_max]
            constraints += [constr for constr_tuple in [(P[k] <= P_max,
                            alpha[k] <= alpha_max) for k in range(K)]
                            for constr in constr_tuple]
            constraints += [F_k_lift[k] >= W_k[k] for k in range(K)]
            constraints += [F_k_drag[k] <= T_k[k] for k in range(K)]
            constraints += [P[k] * D[k] * (1/V[k]) <= E for k in range(K)]

            obj = cp.Minimize(C_des + C_mis)
            prob = cp.Problem(obj, constraints)
            # print(prob.is_dgp())
            prob.solve(gp=True)
```

2.5 Optimal Spacecraft Landing

Problem Overview and Upshot

Problem

[BV24] **Exercise 16.2**. *Optimal Spacecraft Landing.* We consider the problem of optimizing the thrust profile for a spacecraft to carry out a landing at a target position. The spacecraft dynamics are

$$m\ddot{p} = f - mge_3,$$

where $m > 0$ is the spacecraft mass, $p(t) \in \mathbf{R}^3$ is the spacecraft position, with 0 the target landing position and $p_3(t)$ representing height, $f(t) \in \mathbf{R}^3$ is the thrust force, and $g > 0$ is the gravitational acceleration. (For simplicity we assume that the spacecraft mass is constant. This is not always a good assumption, since the mass decreases with fuel use. We will also ignore any atmospheric friction.) We must have $p\left(T^{\mathrm{td}}\right) = 0$ and $\dot{p}\left(T^{\mathrm{td}}\right) = 0$, where T^{td} is the touchdown time. The spacecraft must remain in a region given by

$$p_3(t) \geq \alpha \left\| (p_1(t), p_2(t)) \right\|_2,$$

where $\alpha > 0$ is a given minimum glide slope. The initial position $p(0)$ and velocity $\dot{p}(0)$ are given.
The thrust force $f(t)$ is obtained from a single rocket engine on the spacecraft, with a given maximum thrust; an attitude control system rotates the spacecraft to achieve any desired direction of thrust. The thrust force is therefore characterized by the constraint $\|f(t)\|_2 \leq F^{\max}$. The fuel use rate is proportional to the thrust force magnitude, so the total fuel use is

$$\int_0^{T^{td}} \gamma \left\| f(t) \right\|_2 dt$$

where $\gamma > 0$ is the fuel consumption coefficient. The thrust force is discretized in time, i.e., it is constant over consecutive time periods of length $h > 0$, with $f(t) = f_k$ for $t \in [(k-1)h, kh)$, for $k = 1, \ldots, K$, where $T^{\mathrm{td}} = Kh$. Therefore we have

$$v_{k+1} = v_k + (h/m)f_k - hge_3, \quad p_{k+1} = p_k + (h/2)\left(v_k + v_{k+1}\right),$$

where p_k denotes $p((k-1)h)$, and v_k denotes $\dot{p}((k-1)h)$. We will work with this discrete-time model. For simplicity, we will impose the glide slope constraint only at the times $t = 0, h, 2h, \ldots, Kh$.

(a) *Minimum fuel descent.* Explain how to find the thrust profile $f_1, \ldots, f_K$ that minimizes fuel consumption, given the touchdown time $T^{\mathrm{td}} = Kh$ and discretization time h.

(b) *Minimum time descent.* Explain how to find the thrust profile that minimizes the touchdown time, i.e., K, with h fixed and given. Your method can involve solving several convex optimization problems.

(c) Carry out the methods described in parts (a) and (b) above on the problem instance with data given in `spacecraft_landing_data.py`. Report the optimal total fuel consumption for part (a), and the minimum touchdown time for part (b). The data files also contain plotting code (commented out) to help you visualize your solution. Use the code to plot the spacecraft trajectory and thrust profiles you obtained for parts (a) and (b).

Remarks. If you'd like to see the ideas of this problem in action, watch these videos:

- Grasshopper Diver, Single Cam
- Grasshopper 24-story Hover
- Falcon 9 Landing at Zone 1
- Falcon 9 First Stage Landing

My Response

(a). Formulating this problem is very straightforward as all control specifications are already DCP compatible, *i.e.*, the specified control problem is convex. After discretizing the integral expression for total fuel use as

$$J_{\text{input}} = \sum_{k=0}^{K-1} \gamma \left\| f_k \right\|_2 ,$$

the minimum fuel descent problem can immediately be formulated as

$$\begin{array}{lll} \text{minimize} & J_{\text{input}} & \\ \text{subject to} & v_{k+1} = v_k(h/m)f_k - hge_3, & k = 0, \dots, K-1 \\ & p_{k+1} = p_k + (h/2)(v_k + v_{k+1}), & k = 0, \dots, K-1 \\ & \alpha \left\| \begin{bmatrix} (p_k)_1 \\ (p_k)_2 \end{bmatrix} \right\|_2 \leq (p_k)_3, & k = 1, \dots, K \\ & \left\| f_k \right\|_2 \leq F^{\max}, & k = 0, \dots, K-1 \\ & v_0 = \dot{p}(0), \quad p_0 = p(0) & \\ & v_K = 0, \quad p_K = 0. & \end{array}$$

(Note that we've adopted a Pythonic zero-based indexing for the formulation, *i.e.*, the first force we apply is at $t = 0$, right when we start controlling the rocket).

(b). Keeping the parameter h fixed, we reduce the time it takes to descend by decreasing K, the number of steps it takes to move from $(p(0), \dot{p}(0)) \to (0, 0)$. The simplest way to find K is with a linear search where we start by solving the feasibility problem,

$$\begin{array}{lll} \text{minimize} & 0 & \\ \text{subject to} & v_{k+1} = v_k(h/m)f_k - hge_3, & k = 0, \dots, K-1 \\ & p_{k+1} = p_k + (h/2)(v_k + v_{k+1}), & k = 0, \dots, K-1 \\ & \alpha \left\| \begin{bmatrix} (p_k)_1 \\ (p_k)_2 \end{bmatrix} \right\|_2 \leq (p_k)_3, & k = 1, \dots, K \\ & \left\| f_k \right\|_2 \leq F^{\max}, & k = 0, \dots, K-1 \\ & v_0 = \dot{p}(0), \quad p_0 = p(0) & \\ & v_K = 0, \quad p_K = 0, & \end{array}$$

with a K which yields a feasible solution, then solving the same feasibility problem again but with $K^{\text{curr}} = K - 1$. We continue to decrement the planning horizon, K^{curr} by one until the problem becomes infeasible. At this point, we know that $K^{\text{curr}} + 1$ and the associated solution vector $f^* = (f_1, \ldots, f_{K^{curr}+1})$ is the minimum number of time steps and associated thrust profile which minimizes the touchdown time.

(c). For part (a), the total fuel consumption is about 193.0. For part (b), we find that $K = 25$ minimizes the touchdown time. The minimum fuel descent trajectory and associated thrust profiles are shown in figure 6 and the minimum touchdown time trajectory and associated thrust profiles are shown in figure 7.

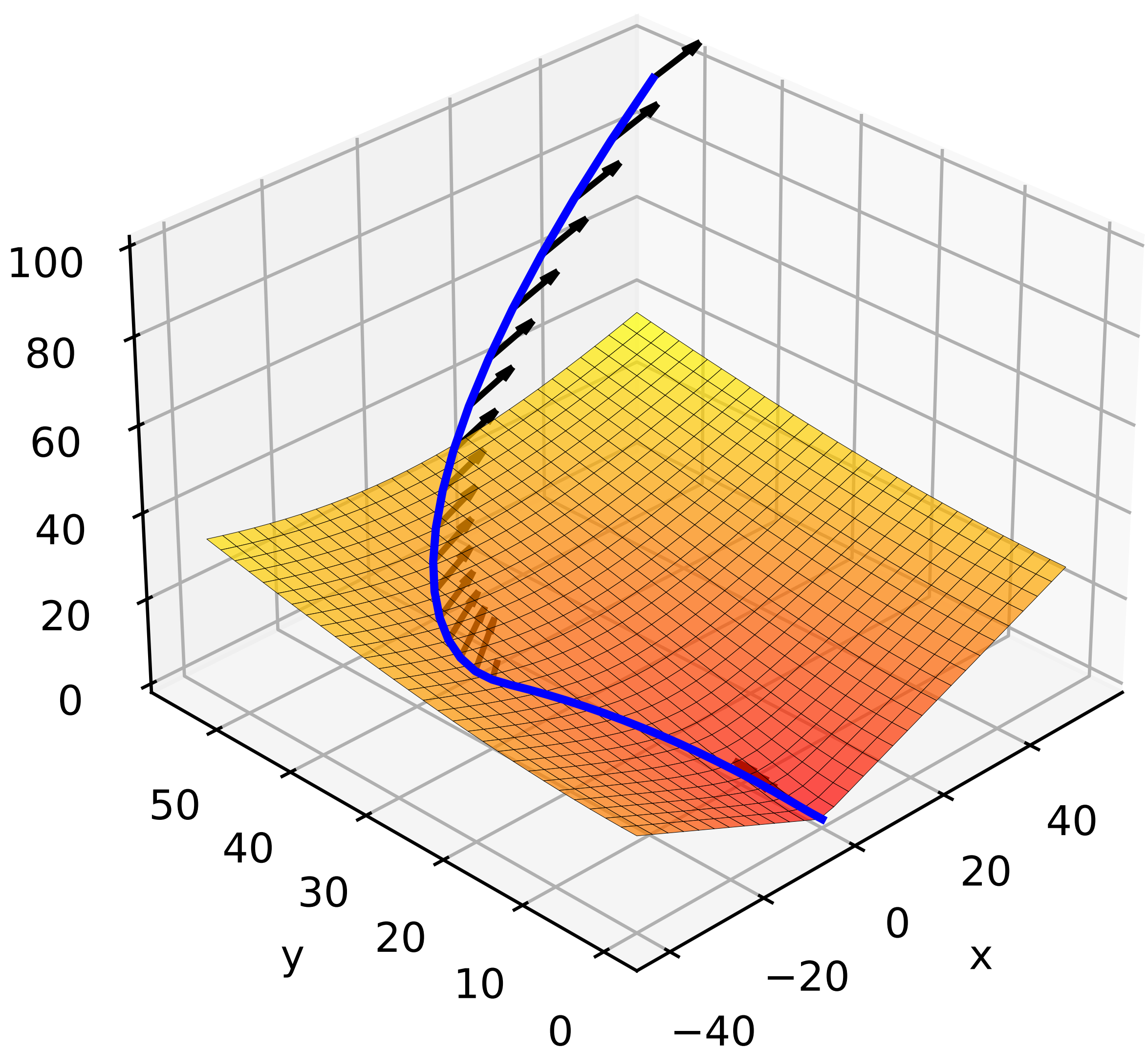

Figure 6: Minimum Fuel Descent Trajectory and Thrust Profiles.

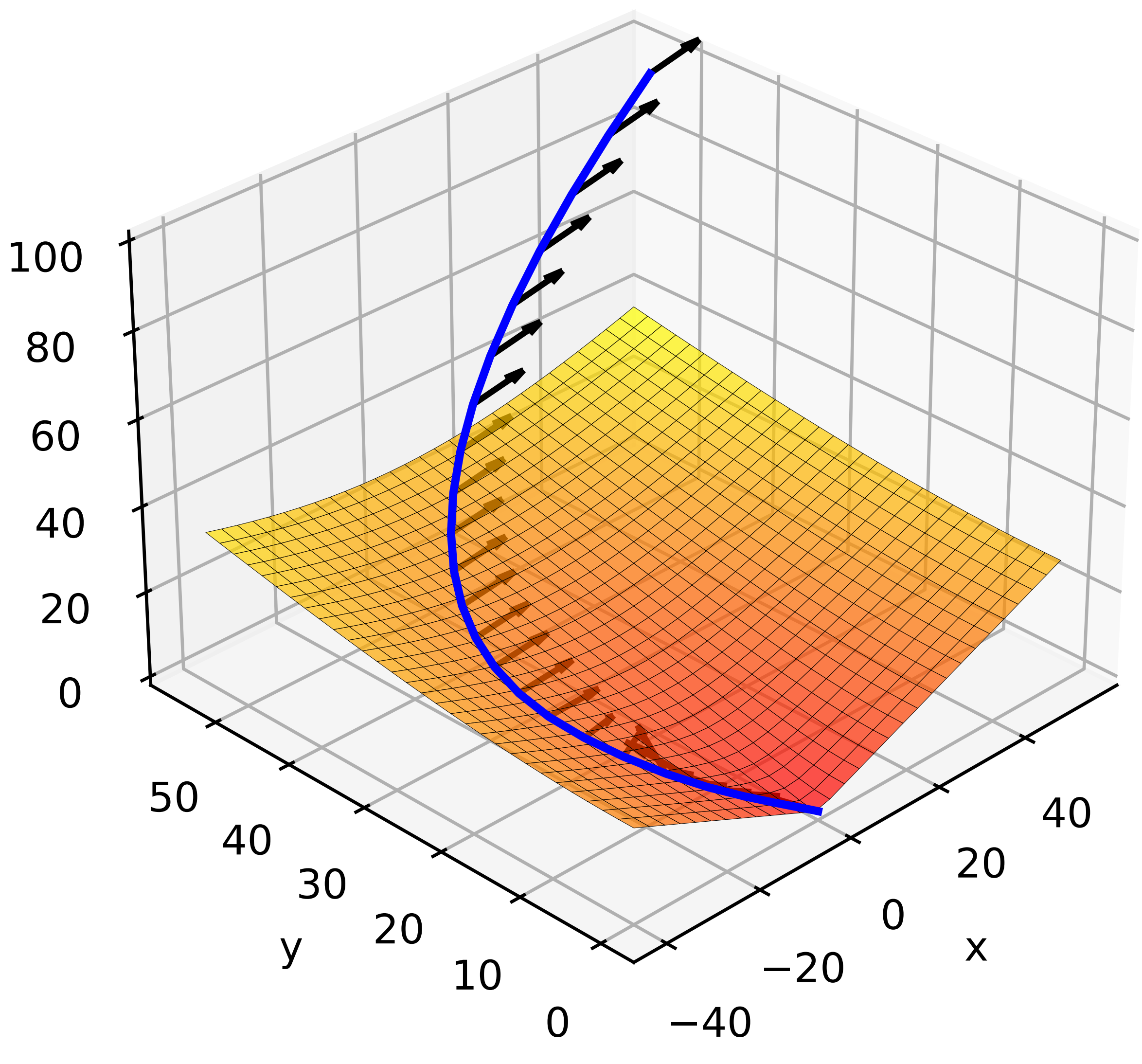

Figure 7: Minimum Touchdown Descent Trajectory and Thrust Profiles.

References

[Boy-8] Stephen Boyd. *EE36b: Convex Optimization II.* 2007-8. URL: https://see.stanford.edu/Course/EE364B.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge, UK: Cambridge University Press, 2004. ISBN: 978-0521833783.

[BV24] Stephen Boyd and Lieven Vandenberghe. *Additional Exercises for Convex Optimization.* 2024. URL: https://github.com/cvxgrp/cvxbook_additional_exercises.