POPTIMAL FREEWAY CORNTROL: PREVENTING CARS FROM GOING BANG-BANG

EJ MERCER, NATHAN SCHILL, DALLIN SEYFRIED, BRIGG TRENDLER

Abstract. In this paper, we explore various solvers for an optimal–control–based model for controlling freeway entrance ramps. We create a system designed to control one entrance ramp, and explore how effectively this can be solved. We also abstract this model to allow for more complex transition functions between states, and implement extensions for this model to find the optimal control for freeways involving multiple entrance ramps. Through doing this, we develop a greater understanding of the complexity of traffic modeling.

1. Background & Motivation

As urban sprawl becomes a greater issue in Utah County, we have seen traffic on freeways continue to get worse. In the last few years, Utah has added timed stoplights and meters to the entrance ramps to moderate the number of cars entering the freeway at once. However, most of these traffic systems are likely operated on a timed system or are manually controlled based on current traffic conditions. For our project, we seek to explore how optimal control principles can be used to determine the ideal entry rates for an entrance ramp; implementing these controls will hopefully help ease traffic in congested areas.

1.1. Related Work.

With transportation being such an integral part of a community, there is a large amount of existing optimization research related to these systems. One such example is the article by Sofronova *et al.*, which explores optimal control in networks of roads and solving these systems through genetic algorithms [SBK19]; this is an excellent example of the applications of optimal control, but considerably higher-level than the modeling we will attempt. Another example which is directly related to our project is the research done by Shaw *et al.*, which models the same scenario using queuing theory and threshold controls [Sha72]. We hope to expand upon this research through the use of continuous control.

Date: April 18, 2024.

See `https://github.com/btrendler/v4_project_winter` for code and additional files referenced in this paper.

1.2. **Acknowledgements.** We gratefully acknowledge the utility of the NumPy [HMvdW+20] and SciPy [VGO+20] mathematical tools for Python; these provide an essential foundation for our solvers. We also used Mathematica [Inc] in our exploration.

We also give thanks to Dr. Gregory Macfarlane in the Civil and Construction Engineering Department at Brigham Young University for his feedback & insights regarding how to improve our model.

2. Mathematical Representation

2.1. **Assumptions & Evolution Equations.** To simplify our modeling, we initially use the following assumptions:

- There is only one lane on the freeway, and one lane on the entrance ramp.
- Cars in the entrance ramp accelerate instantly.
- All cars travel at the same speed.
- There is no limit to the size of the queue or of the road segment before the merge region.

For this model, we want to avoid a bang-bang problem so that we can use Pontryagin's Maximum Principle (PMP) and numerical solvers to find the optimal solution. We will also explore linearizing this problem such that we can use the Linear Quadratic Regulator (LQR) algebraic solution to produce a fast real-time solver.

At first, many of our simulations use reasonable but generic 'dummy' values for parameters and initial values. We devoted some effort to determining realistic values, including by estimating road capacity from online maps and visiting Dr. Macfarlane. We regret that it is still likely our choice of parameters would make experts in this field cringe. Even so, our final model is dimensionally homogeneous and sometimes obtains sensible simulations, so we have reason to hope that with proper values our results would hold.

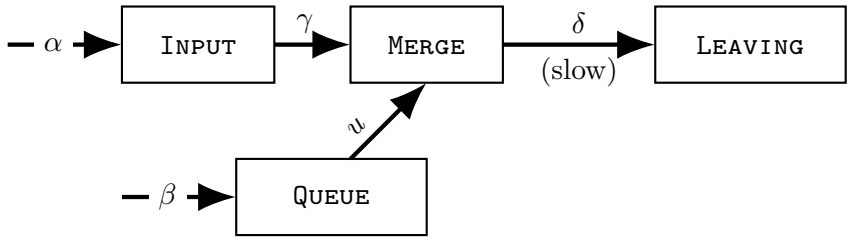
The following sections walk through various sets of equations we used as we iterated on our modeling. For details, see `explore-transitions-between-road-segments.pdf`, and see the corresponding Mathematica notebook (with the `.nb` extension) to experiment with the cells.

2.1.1. *Iteration 1: Baseline Model.* We begin by modeling one entrance ramp. Our first attempt models the number of cars in each of three sections on the main freeway: the 'i**n**put' $n(t)$ before the merge region, the '**m**erge' $m(t)$ region, and the '**l**eaving' $l(t)$ region following the merge region. In addition, we represent the '**q**ueue' or on-ramp by $q(t)$.

We begin with the following system of equations:

$$\begin{aligned}
\dot{n}(t) &= \alpha - \gamma n(t) \\
\dot{m}(t) &= \gamma n(t) - \delta m(t)(1 - m(t)/c) + u(t) \\
\dot{l}(t) &= \delta m(t)(1 - m(t)/c) \\
\dot{q}(t) &= \beta - u(t)
\end{aligned}$$

The variables γ and δ scale the rate at which cars go from n to m and m to l, respectively. The rate at which cars transition from the 'merge' segment to the 'leaving' segment is limited by its carrying capacity, c, through the term $1 - m(t)/c$, slowing the rate at which they leave as capacity is approached. α and β represent the rates at which new cars enter the 'input' and 'queue' segments, respectively. These relationships are expressed by this graph:



2.1.2. *Iteration 2: Preventing Cars from Going Backward.* Having established a baseline model (the results of which will be shared later), we now walk through our next steps at improving our model.

We note that in the definition of $\dot{m}$, the term $-\delta m\,(1 - m/c)$ changes sign if m grows to be higher than c, which would correspond to cars traveling backward from the 'leaving' section to the 'merge' section. Believing this to be a possible cause of instability in our solutions, we propose some variations on this term. One of these variations is to use a function of the form

$$-a\,\sigma(b\,m) + d\,\sigma\big(b\,(m-a)\big) + C$$

where $\sigma(\cdot)$ is the sigmoid function. This achieves a maximal rate of transition at a "carrying capacity," but then flattens out closer to zero as m grows larger, without crossing to the other polarity. See Figure 1 for a comparison. We attempt simulations of this formulation using PMP, though the results (discussed later) are mixed.

2.1.3. *Iteration 3: Dimensional Homogeneity of the Evolution.* Whereas the original evolution equations are dimensionally homogeneous (given the proper units on the constant coefficients), the term proposed above is not, due to the exponential in the sigmoid terms. To combat this, we propose a piecewise variation consisting of the original quadratic term for smaller values of t followed by function with a horizontal asymptote. This is of the form

$$\begin{cases} -m\left(1 - \frac{m}{c}\right) & 0 \le m < m^* \\ -\frac{d}{m-a} - b & m^* \le m. \end{cases}$$

With the proper units on the constants, this is dimensionally homogeneous. We can choose the capacity c and the asymptotic rate b, and then, choosing a switching time m^* somewhat arbitrarily, we can use Mathematica to solve for the other coefficients a and d to make the function C^1. We also

attempt simulating this version with PMP, again with mixed poor results. Again, refer to Figure 1.

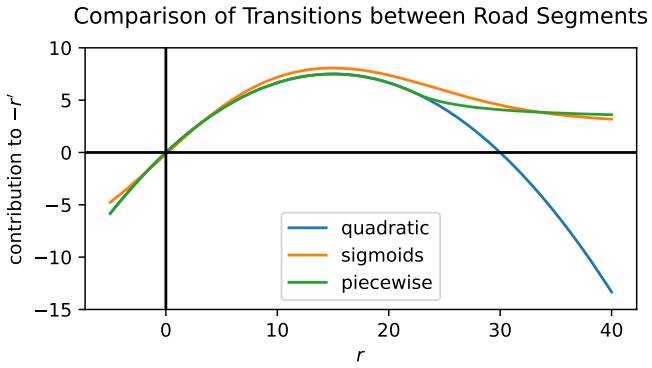


FIGURE 1. The quadratic version changes sign if r becomes too large, while the sum of sigmoids levels out (e.g., due to congestion). The piecewise function also levels out and is dimensionally homogeneous unlike the sum of sigmoids, but it is only C^1.

2.1.4. *Iteration 4: Overhaul Everything.* Our eventual goal is to expand our model to accommodate multiple on-ramps. Obviously, this would require adding equations for multiple queues and multiple merge sections. To simulate differing lengths of road, we also include multiple '**r**oad' sections $r_i(t)$ between merge regions.

After consulting with Dr. Macfarlane, we modify the terms governing the flow between regions. One flaw with the models considered up to this point is that the rate of flow from the input to the merge region depends only on the number of cars in input region, and not at all on the number of cars in the merge region. In reality, we would expect that rate to increase as the number of cars in the input region increases up to a point; then, as the input region or merge region becomes more congested, the rate decreases.

To model this flow between one road segment $r_{i-1}(t)$ to the next segment $r_i(t)$, we write the rate in the form $r_{i-1}\left(1-\frac{r_{i-1}}{c}\right)\left(1-\frac{r_i}{c}\right)$ for some capacity c. Including these "carrying-capacity coefficients" $\left(1-\frac{r(t)}{c}\right)$ on each term in the evolution equations models the influence of each state involved in a transition, and also will prohibit any region from growing above its carrying capacity. This is a departure from the last of our initial assumptions; however, this addresses the issue from section 2.1.2 in which our terms changed sign and the cars flowed backward.

We arrive at a system of the following form:

$$
\begin{aligned}
\dot{n} &= \alpha \left(1 - \frac{n}{c}\right) - \delta_0 \, g(n, r_1) \\
\dot{r}_i &= \delta_{i-1} \, g(r_{i-1}, r_i) - \delta_i \, g(r_i, r_{i+1}), \\
&\qquad i \in \{1, \ldots, s-1, s+1, \ldots, S-1\} \\
\dot{m} &= \delta_{s-1} \, g(r_{s-1}, m) - \delta_s \, g(m, r_{s+1}) + u \\
\dot{l} &= \delta_{S-1} \, g(r_{S-1}, l) \\
\dot{q} &= \beta \left(1 - \frac{q}{c}\right) - u
\end{aligned}
$$

where $r_0 = n$, $r_s = m$, and $r_S = l$, while $g(b, a) = b\left(1 - \frac{b}{c}\right)\left(1 - \frac{a}{c}\right)$ is the transition rate from any '**b**efore' segment to its corresponding '**a**fter' segment. We make the assumption that each road segment is the same length, and thus has the same capacity c, so that a stretch of road of a given length may be modeled using multiple road segments r_i. To extend to multiple on-ramps, we have only to include additional merge and queue regions between road segments.

Finally, in some simulations, the optimal control takes negative values, that is, sending cars back into the queue. This makes sense in our model if the merge region is too full in order to decrease congestion and thus increase throughput, but of course this would be disastrous in a physical setting. To account for this when running our models, after solving for u normally, we clip it to be non-negative when determining the state evolution.

2.2. **Cost Functionals.** As LQR requires requires certain constraints on the cost functional, we use one cost functional for PMP and another for LQR.

For PMP, we use the cost functional

$$J[u] = \int_0^{t_f} \left[Qq(t)^2 - Ll(t)^2 + Uu(t)^2\right] dt$$

where the capital-letter coefficients are constant weights. This functional rewards cars getting to l and penalizes them staying in q. We also include a quadratic cost on u to avoid a bang-bang problem, but we can motivate this choice as it prevents u from becoming unrealistically large—cars can move from the queue to the freeway only so fast, whether or not a meter is present.

For LQR, we use the cost functional

$$J[u] = \int_0^{t_f} \left[x^\top Q x + u^\top R u\right] dt + x(t_f)^\top M x(t_f).$$

We would like to reward getting cars to l and penalize cars in the queue and merge regions. As LQR requires Q and M to be positive semi-definite, we include diagonal terms in Q and M for the m and q components of the state. As with PMP, we also penalize u with R.

As we arrived at the point of generalizing our LQR solver to multiple on-ramps, we realized that we could reward l by replacing it with $-l$ in the state (and negating it throughout the evolution equations) and then 'penalizing' it. We implement this change in our finalized LQR system, otherwise using the same cost functional with various combinations of the diagonal entries in the matrices.

3. Solution

3.1. **Single Entrance Ramps.**

3.1.1. *Pontryagin's Maximum Principle (PMP).* We apply PMP in the usual way with the cost functional described above (section 2.2) and each set of single-ramp evolution equations (section 2.1). We use Mathematica's `NDSolve`, a numeric boundary-value-problem solver. We don't attempt applying PMP to the multi-ramp system because of the difficulties we faced on the single-ramp system (and because LQR solvers are more computationally efficient). We also experiment with variations on the evolution equations and cost functional.

For details of these experiments, see `apply-pmp.pdf`. To experiment with the cells, see the Mathematica notebook `apply-pmp.nb`.

3.1.2. *Finite-Time LQR.* We only attempt the baseline evolution equation in section 2.1.1 using finite-time LQR. To do so, we linearize the system by performing a first-order Taylor series approximation of the transition term between the 'merge' and 'leaving' sections:

$$\begin{aligned} f(m) &= -\delta\, m \left(1 - \frac{m}{c}\right) \\ &\approx -\delta\, m_0 - \delta \frac{m_0^2}{c} - \left(\delta + 2\delta \frac{m_0}{c}\right)(m - m_0). \end{aligned}$$

To allow the incorporation of affine terms into our linear evolution equation, we include α, β, and a new term,

$$\nu = -\delta m_0 - \delta \frac{m_0^2}{c} + \left(\delta + 2\delta \frac{m_0}{c}\right) m_0,$$

into out state, yielding $\mathbf{x} = [\alpha, \beta, \nu, m, l, n, q]^\top$. (We later realized that we could instead carry along a constant 1 in our state and put coefficients like α in the transition matrix computed at each step instead. We do this on our final solver as it greatly reduces the size of the matrix on more complex road networks.)

This produces the state space equation:

$$\begin{bmatrix}\dot{\alpha}\\ \beta\\ \nu\\ m\\ l\\ n\\ q\end{bmatrix} = \begin{bmatrix}0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&-(\delta+2\delta m_0/c)&0&\gamma&0\\ 0&0&-1&(\delta+2\delta m_0/c)&0&0&0\\ 1&0&0&0&0&-\gamma&0\\ 0&1&0&0&0&0&0\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ \nu\\ m\\ l\\ n\\ q\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 0\\ 1\\ 0\\ 0\\ -1\end{bmatrix} u$$

We also break the given interval $[0, t_f]$ down into many sub-intervals, recomputing m_0 for each one, to overcome the error introduced by linearizing the carrying capacity.

Computing the solution for Finite-Time LQR follows from exercise 19.2 in Foundations of Applied Mathematics, Volume 4, Modeling with Dynamics and Control (Volume 4) ([HJW]). We set $X(t)$ and $Y(t)$ to be $n \times n$ matrices satisfying the linear differential equation

$$\begin{pmatrix}\dot{X}\\ \dot{Y}\end{pmatrix} = \mathscr{H}(t)\begin{pmatrix}X(t)\\ Y(t)\end{pmatrix}$$

with the boundary condition $X(t_f) = I, Y(t_f) = -2M$ and where

$$\mathscr{H}(t) = \begin{pmatrix}A(t) & \frac{1}{2}B(t)R^{-1}(t)B^\top(t)\\ 2Q(t) & -A^\top(t)\end{pmatrix}$$

The $P(t)$ matrix can then be solved for via $P(t) = -\frac{1}{2}Y(t)X^{-1}(t)$ which will satisfy the Riccati differential equation:

$$\dot{P} = -P(t)A(t) - A^\top(t)P(t) - Q(t) + P(t)B(t)R^{-1}(t)B^\top(t)P(t)$$

with the endpoint condition $P(t_f) = M$. Since we are given an endpoint condition, we can use `scipy.integrate.solve_ivp` (see [VGO+20]) to solve the differential equation in reverse to treat the endpoint condition as an initial condition. Once we have solved for $P(t)$, we can then solve for the optimal control and co–state vectors using Theorem 19.1.2 from Volume 4 ([HJW]). These then lead to a setup to solve the linear state equation for $\mathbf{x}(t)$ which result in the following equations:

$$\begin{aligned}\tilde{\mathbf{u}}(t) &= -R^{-1}(t)B^\top(t)P(t)\tilde{\mathbf{x}}(t)\\ \mathbf{p}(t) &= -2P(t)\mathbf{x}(t)\\ \dot{\mathbf{x}} &= A(t)\mathbf{x}(t) - B(t)R^{-1}(t)B^\top(t)P(t)\tilde{\mathbf{x}}(t)\end{aligned}$$

Once again, using `scipy.integrate.solve_ivp` solves for $\mathbf{x}(t)$ given initial conditions of the on-ramp and freeway.

3.1.3. *Infinite-Time LQR.* We also explore the single-ramp case as an infinite-horizon problem. The advantage of this is that we can use an algebraic solver, which will allow a much faster algorithmic turnaround. This was a fairly simple extension from the finite-time case, and again we linearize the system and run it over small time intervals.

We also used infinite-time LQR to evaluate a "Time Saved" metric or, in a sense, how many cars our optimal control system was able to safely push through the system versus a system with no control placed on the on-ramp. We compare the different states over time between the two systems as well as examine the number of cars that successfully made it to the leaving state. To define the no-control system, we remove the control u and replace it with the evolution that would "naturally" occur. Specifically, we replace u with $q\left(1-\frac{q}{c}\right)\left(1-\frac{m}{c}\right)$, so that

$$\dot{q} = \beta\left(1-\frac{q}{c}\right) - u$$

becomes

$$\dot{q} = \beta\left(1-\frac{q}{c}\right) - \rho\, q\left(1-\frac{q}{c}\right)\left(1-\frac{m}{c}\right)$$

and corresponding replacements in $\dot{m}$. We use appropriate linearizations as developed in section 2.1.4 to handle a naturally evolving queue and carrying capacities for each part of the road.

When attempting to use the overhauled equations and matrices with `scipy`'s continuous algebraic Riccati equation (ARE) solver, we experienced ill-conditioning problems with the Hamiltonian pencil matrix that the solver constructs using our LQR matrices. This method fails because the eigenvalues of the Hamiltonian pencil matrix are too close to the imaginary axis, causing instability in later decompositions[1]. Due to this, we attempted to switch to the discrete ARE solver offered through `scipy` using a forward Euler discretization of the system solved for in Exercise 23.5 of Volume 4 ([HJW]), but we ran into similar issues with calculating the symplectic pencil matrix that the discrete method uses, since its eigenvalues were well inside the unit circle[2]. In the end, the setup we came to that was the most stable, seeming to provide reasonable results, was using the discrete ARE solver *without* discretizing the matrices. Given more time, we would like to investigate how we can change our LQR setup in an understandable way such that these pencil matrices stabilize.

3.2. **Multiple Entrance Ramps with Infinite-Time LQR.** To solve multiple on-ramps, we developed a Python class to linearize the evolution equations in section 2.1.4 for arbitrary sequences of road and merge regions and then solve that set of equations using infinite-time LQR. This was no

[1] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.solve_continuous_are.html`

[2] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.solve_discrete_are.html`

small feat, and required everyone to collaborate and think of new ways to handle notation, the equations, and the transfer of the equations to code.

Remark. It should be noted that this cumulative brainpower does not differ significantly from the individual brainpower of any one team member due to the rapidly diminishing returns of "collaboration."

The implementation of a simple, single-ramp model through this tool can be explored in the interactive plots found in the `interactive.ipynb` file (or the associated `interactive.py` file, if Jupyter does not cooperate with interactive animations); a non-interactive code sample of how to use multiple ramps can be seen in `multiple_ramps_example.ipynb`.

4. Interpretation

4.1. **Single Entrance Ramps.**

4.1.1. *Pontryagin's Maximum Principle.* For each of the evolution equations described in section 2.1 (except the multi-ramp version), PMP yielded plausible solutions for small end times t_f. However, in all cases, increasing t_f caused nonsensical solutions (such as a negative number of cars) and, for even larger t_f, complete numeric failure. Solutions are also very sensitive to parameters (such as rates) and initial values.

As mentioned in section 3.1.1, we also experiment with variations on the evolution equations and cost functional. In particular, we try including combinations of the control u in the evolution and cost in either linear or quadratic terms, as well as omitting it from the evolution or cost. Most variations resulted in either setting $u = 0$ or a bang-bang problem, and the one variation that yielded a non-zero control resulted in a failed numerical solution.

As before, see `apply-pmp.pdf` for plots. To experiment with the cells, see the Mathematica notebook `apply-pmp.nb`.

4.1.2. *Finite-Time LQR.* We first attempted running our method once over the full time interval $[0, t_f]$, but to avoid the compounding error from the linearization, we found that a more logical and stable approximation of the optimal result was obtained by dividing this interval into many small sub–intervals and computing the LQR solution over each time step. See Figure 2 for a comparison of the two approaches.

4.1.3. *Infinite-Time LQR.* We obtained mixed results when using infinite-time LQR. Solving in multiple steps over time produces a solution which stabilizes quickly, keeping the merge section under its carrying capacity while levelling off the number of cars in each of the influencing areas. This result can be seen in Figure 3. However, this model is very susceptible to parameter changes—for example, in Figure 4, we see that increasing δ changes the optimal solution such that cars are drawn rapidly *out of* the merge section

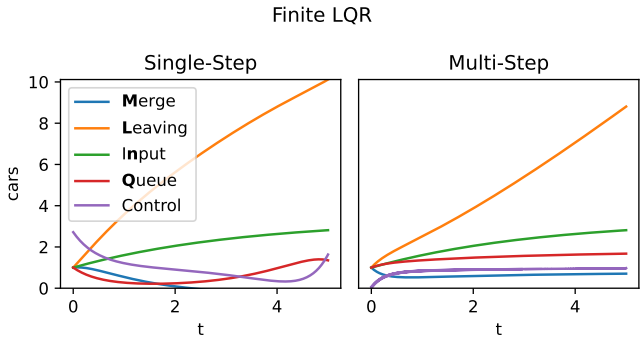


FIGURE 2. Parameters: $\delta, \gamma, m_0, c = 0.5, 0.3, 1.0, 2.0$. In the multi-step case, the solution stabilizes after a few units of time, keeping the merging section under the carrying capacity while levelling off the number of cars in the queue, merge, and before areas. See `finite_time_lqr.ipynb` for a working example.

and back into the queue, and then are allowed to trickle out slowly. This caused us to rethink the basic structure of our model.

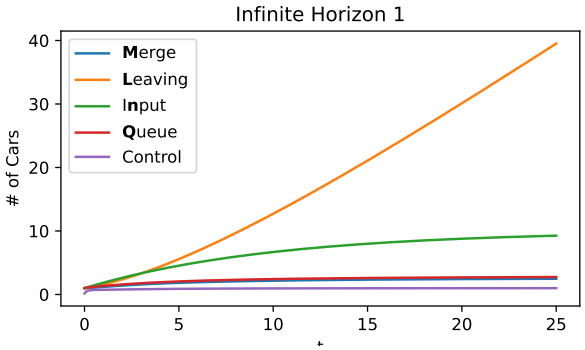


FIGURE 3. Multi-step LQR, with parameters: $\delta, \gamma, m_0, c = 0.1, 0.3, 1.0, 2.0$.

Our results for the "Time-Saved" metric showed that our controlled model performed well against the uncontrolled model. In figure 5 we see that the controlled model on the left was able to send more cars through the to the end while keeping all three sections relatively even as compared to the uncontrolled on the right.

4.2. **Multiple Entrance Ramps.** With much tinkering, we were able to compute a simulation involving multiple on-ramps. Slight perturbations in the choice of initial parameters would cause a `LinAlgError` to occur. We believe the reason for this could be that if any one section has a deficit in

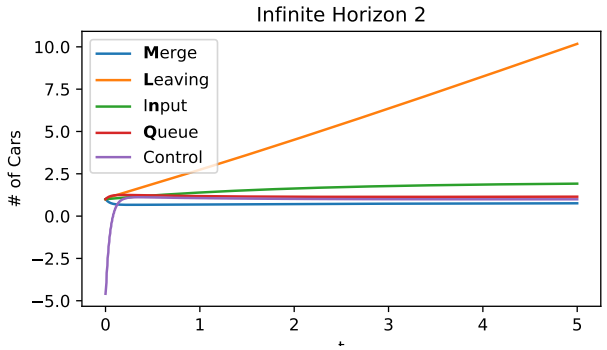


FIGURE 4. Multi-step LQR, with parameters: $\delta, \gamma, m_0, c = 0.5, 0.3, 1.0, 2.0$.

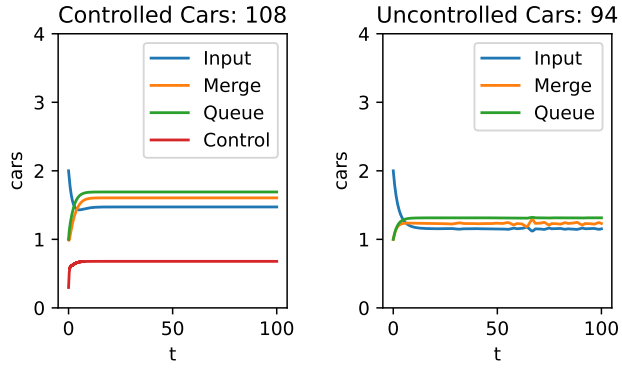


FIGURE 5. "Time Saved" evaluation on a freeway with a carrying capacity of 5 cars per section. See `ext_compl_rd.ipynb` for a working example.

the number of cars for any reason, the A matrix in our LQR computation would no longer be positive semi-definite. See Figure 6 for results.

4.3. **Future Work.** Despite the number of hours we spent exploring models, cost functionals, and solvers, there is a significant amount of work we could still do. For one, there are many variations on the initial values and the cost weights that we could test. In addition, while our multi-ramp solver using the discrete ARE (see section 3.1.3) works with the transition matrices derived for the continuous ARE, it doesn't make sense mathematically. However, if we use either equation with the correct matrices, our solver fails.

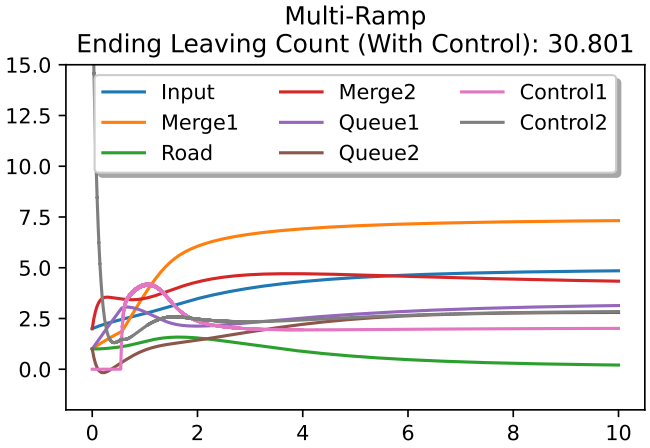


FIGURE 6. Parameters: $\alpha = 5$, $\gamma = 2$, $\delta = 1$, $\beta = 4$, $c = 15$, $n(0) = 2$, $m_1(0) = 1$, $r(0) = 1$, $m_2(0) = 2$, $q_1(0) = 1$, $q_2(0) = 1$. The initial control for the second queue begins at 20, and quickly drops to about 2. Each of the road sections seem to begin to stabilize around $t = 6$.

Future work could determine the cause of this behavior and remedy it to produce more accurate models.

We give Dr. Evans, Dr. Whitehead, and other instructors at BYU teaching ACME classes permission to share our project as an example of a good project in future classes they teach.

References

[HJW] Jeffrey Humpherys, Tyler J. Jarvis, and Jared J. Whitehead. *Foundations of Applied Mathematics, Volume 4, Modeling with Dynamics and Control.*

[HMvdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[Inc] Wolfram Research, Inc. Mathematica, Version 14.0. Champaign, IL, 2024.

[SBK19] E.A. Sofronova, A.A. Belyakov, and D.B. Khamadiyarov. Optimal control for traffic flows in the urban road networks and its solution by variational genetic algorithm. *Procedia Computer Science*, 150:302–308, 2019. Proceedings of the 13th International Symposium "Intelligent Systems 2018" (INTELS'18), 22-24 October, 2018, St. Petersburg, Russia.

[Sha72] L. Shaw. On optimal ramp control of traffic jam queues. *IEEE Transactions on Automatic Control*, 17(5):630–637, 1972.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.