TODO(austin): Now that the python matches the original problem and solves, confirm the paper matches what got implemented.

osqp!

1 Catapult MPC

We want to phrase our problem as trying to solve for the set of control inputs which get us closest to the destination, but minimizes acceleration. Specifically, we want to minimize acceleration close to the end. We also have a voltage limit.

Our model is

$$\begin{bmatrix} x_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ 0 & a_{11} \end{bmatrix} \begin{bmatrix} x_0 \\ v_0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \begin{bmatrix} u_0 \end{bmatrix} \tag{1}$$

Our acceleration can be measured as:

$$\frac{(\boldsymbol{X_1(1)} - \boldsymbol{X_1(0)})}{\Delta t} \tag{2}$$

This simplifies to:

$$\frac{a_{11}v_0 + b_1u_0 - v_0}{\Delta t} \tag{3}$$

and finally

$$\frac{(a_{11} - 1)v_0 + b_1u_0}{\Delta t} \tag{4}$$

We can also compute our state matrix as a function of inital state and the control inputs.

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \end{bmatrix} X_0 + \begin{bmatrix} B & 0 & 0 & 0 \\ AB & B & 0 & 0 \\ A^2B & AB & B & 0 \\ \vdots & \ddots & & \dots \end{bmatrix} \begin{bmatrix} U_0 \\ U_1 \\ U_2 \\ \vdots \end{bmatrix} \tag{5}$$

2 MPC problem formulation

We want to penalize both final state and intermediate acceleration.

$$C = \sum_{n=0}^{39} \frac{(v(n+1) - v(n))^2}{\Delta t} \pi_n + (X_{40} - X_{final})^T Q_{final} (X_{40} - X_{final}) \tag{6}$$

where π_n is a constant only dependent on n, and designed such that it depends on the distance to the end of the sequence, not the distance from the start.

In order to use OSQP, which has a code generator, we need to get this into the form of

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}X^T P X + q^T X \\ \text{subject to} & l <= AX <= u \end{array}$$

This is the simplest form of a constrained quadratic program that we can solve efficiently. Luckily for us, the problem statement above fits that definition.

3 Manipulating the formulation

We need to separate constant factors from things dependent on U (or X in OSQP parlance) so we can create these matrices easier.

3.1 Terminal cost

Next step is to compute X_{40} using equation 5. We can do this by only computing the final row of the matrix.

$$X_{40} = \begin{bmatrix} A^{39}B & A^{38}B & \dots & B \end{bmatrix} \begin{bmatrix} U_0 \\ \vdots \\ U_{39} \end{bmatrix} + A^{40}X_0 = B_f \boldsymbol{U} + A^{40}X_0 \tag{7}$$

We can substitute equation 7 into equation 1.

$$\begin{aligned} C_f =& \boldsymbol{U}^T B_f^T Q_{final} B_f \boldsymbol{U} \\ &+ 2(A^{40}X_0 - X_{final})^T Q_{final} B_f \boldsymbol{U} \\ &+ (A^{40}X_0 - X_{final})^T Q_{final} (A^{40}X_0 - X_{final}) \end{aligned} \tag{8}$$

3.2 Acceleration costs

We can compute a velocity matrix for all the times by stripping out the positions from equation 5 by using every other row. We can use this to then compute the accelerations for each time slice and then penalize them.

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{40} \end{bmatrix} = M\boldsymbol{U} + \begin{bmatrix} a_{11} \\ a_{11}^2 \\ \vdots \\ a_{11}^{40} \end{bmatrix} v_0 = M\boldsymbol{U} + mv_0 \tag{9}$$

We can then use equation 2 in matrix form to convert a velocity matrix to an acceleration matrix.

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_{40} \end{bmatrix} = \frac{1}{\Delta t} \left(\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{40} \end{bmatrix} - \begin{bmatrix} v_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right) \tag{10}$$

We can pull some of these terms out to make it easier to work with.

$$\boldsymbol{\alpha} = WV + wv_0 \tag{11}$$

Our acceleration cost function is then:

$$C_a = \boldsymbol{\alpha}^T \begin{bmatrix} \pi_1 & & 0 \\ & \pi_2 & \\ 0 & & \ddots \end{bmatrix} \boldsymbol{\alpha} = \boldsymbol{\alpha}^T \Pi \boldsymbol{\alpha} \tag{12}$$

We can substitute everything in to get something as a function of U.

$$C_a = \left(W\left(M\boldsymbol{U} + mv_0\right) + wv_0\right)^T \Pi \left(W\left(M\boldsymbol{U} + mv_0\right) + wv_0\right) \tag{13}$$

And then simplify this down into the expected form.

$$C_a = \left(WM\boldsymbol{U} + (Wm + w)v_0\right)^T \Pi \left(WM\boldsymbol{U} + (Wm + w)v_0\right) \tag{14}$$

$$\begin{aligned} C_a = &\boldsymbol{U}^T M^T W^T \Pi W M \boldsymbol{U} \\ &+ 2v_0 (Wm + w)^T \Pi W M \boldsymbol{U} \\ &+ v_0 (Wm + w)^T \Pi \left(Wm + w\right) v_0 \end{aligned} \tag{15}$$

3.3 Overall cost

We can combine equations 8 and 15 to get our overall cost in the correct form.

$$
\begin{aligned}
C = \boldsymbol{U}^T & \left(M^T W^T \Pi W M + B_f^T Q_{final} B_f \right) \boldsymbol{U} \\
& + \left(2 v_0 (Wm + w)^T \Pi W M - 2 X_{final}^T Q_{final} B_f \right) \boldsymbol{U} \\
& + X_{final}^T Q_{final} X_{final} + v_0 (Wm + w)^T \Pi \left(Wm + w \right) v_0
\end{aligned} \tag{16}
$$

4 Response

For a reasonable request (11 m/s after 90 degrees), we get the following response

This is well within 1% error, and avoid saturation and keeps the acceleration down at the end.