Consider a mass m on a frictionless table subject to a piece-wise constant force

$$f(t) = f_i \text{ for } t \in [i-1, i) \text{ and } \ i = 1, \ldots, 10.$$

Let $y(t) = (x, v)$ denote the position and velocity of the mass. We'll use the notation $z_i = z(i)$ for $z \in \{x, v, y\}$ since with a piece-wise constant force the problem discretizes.

Suppose $y_0 = (0, 0)$ and we seek a force program f_i which lands the mass in a desired state $y_d = (1, 0)$ at $t = 10$.

Under the assumed initial rest, the velocity $v(t)$ can be written: $v = A_{vf} f$.

Let's find the entries of A_{vf}. Consider the rows a_i of A_{vf}. The entries in the first row a_1 tell us how the forces affect the velocity of the mass at $t = 1$ seconds. Clearly, forces $f_2, f_3, ...$ can have no effect on v_1, since at $t = 1$ they have not yet been applied. Newton's first law for a constant force says $\Delta v = a\Delta t = \frac{f}{m}\Delta t$, and so with $\Delta t = (i+1) - (i) = 1$ we have that $a_1 = (\frac{f_1}{m}, 0, ...)$.

To find a_2, we reason from causality just as before. It's clear that v_2 cannot depend on $f_3, f_4, ...$ The dependency on f_1 and f_2 is very simple: it's additive. In other words, the velocity at time t depends on the running sum of the forces applied so far (each multiplied by the duration of its application). And so we can write:

$$v_i = \frac{1}{m} \Sigma_{j=1}^{i} f_j, \text{ for } \ i = 1, \ldots, 10,$$

or in matrix form:

$$A_{vf} = \frac{1}{m} \begin{pmatrix} f_1 & & & \\ f_1 & f_2 & & \\ \vdots & \vdots & \ddots & \\ f_1 & f_2 & \ldots & f_{10} \end{pmatrix},$$

where the entries not shown are zero.

Similarly, the position can be written as a function of the velocities: $x = A_{xv} v$.

As before, let's think about the first row a_1 of A_{xv}. Since the applied forces are piecewise constant, the velocities are piecewise linear. Under constant acceleration, displacement is simply determined by the average of the velocities at the two endpoints. Under initial rest, we have then

$$x_1 = \frac{1}{2}v_1, \tag{1}$$

and similarly

$$x_2 = x_1 + \frac{1}{2}(v_1 + v_2). \tag{2}$$

With a substitution, we have:

$$x_2 = v_1 + \frac{1}{2}v_2. \tag{3}$$

Carrying out the induction, we see:

$$x_i = \Sigma_{j=0}^{i-1} v_j + \frac{1}{2}v_i. \tag{4}$$

And so the elements of A_{xv} are simply:

$$A_{xv}(i,j) = \begin{cases} 0 & i > j \\ \frac{1}{2} & i = j \\ 1 & i < j \end{cases}$$

Finally form $A_{xf} = A_{xv}A_{vf}$.

Of particular interest to us is the final state $y = (x_{10}, v_{10})$. Letting $a_{xf} = A_{xf}(10, :)$ and $a_{vf} = A_{vf}(10, :)$, and defining:

$$A \in \mathbb{R}^{2\times 10} = \begin{pmatrix} a_{xf} \\ a_{vf} \end{pmatrix}, \tag{5}$$

observe $y = Af$.

As it stands, finding a force program is a feasibility problem. Let's add a regularizer to reign in the solutions. With an l_2 penalty the problem takes the form:

$$\begin{aligned} \underset{f}{\text{minimize}} \quad & \frac{1}{2} f^T f \\ \text{subject to} \quad & Af = y_d \end{aligned} \tag{6}$$

This is nothing but the least-norm problem for an overdetermined system of equations. Let's solve it for practice. We introduce a Lagrangian,

$$\mathcal{L} = f^T f + \lambda^T (Af - y_d) \tag{7}$$

,

and take first order conditions (FOCs):

$$\frac{\partial \mathcal{L}}{\partial f} = f + A^T \lambda = 0 \tag{8a}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = Af - y_d = 0, \tag{8b}$$

Left-multiplying equation (8a) by A we have:

$$Af - AA^T \lambda = 0, \tag{9}$$

and substituting from equation (8b) we have:

$$AA^T \lambda = -y_d. \tag{10}$$

Since A is fat and full-rank, AA^T is invertible, and so equation (10) becomes:

$$\lambda = -(AA^T)^{-1} y_d. \tag{11}$$

Substituting λ back into equation (8a), we obtain the optimal f, which we recognize as the least-norm solution:

$$f = A^T (AA^T)^{-1} y_d. \tag{12}$$

While l_2 gives us traction analytically, other penalties give rise to solutions with very nice interpretations, and in some cases yield a convex formulation. For instance, consider an $\|\cdot\|_1$ penalty on the forces, which in a propulsion application would actually correspond closely to fuel consumption. An $\|\cdot\|_\infty$ would penalize the max thruster.