EECS 16B Designing Information Systems and Devices II UC Berkeley Spring 2022

Discussion 10A

The following notes are useful for this discussion: Note 16

1. Minimum Energy Control

In this question, we build up an understanding for how to get the minimum energy control signal to go from one state to another. What do we mean by the "energy" of the control? In this context, we will use the squared norm of the input vector $\|u\|^2 = u_1^2 + \cdots + u_n^2$ as the model for the cost to applying a set of controls to our system.

Why do we use this definition? This formula for the cost model is closely connected to many physical scenarios relating to energy. Consider a few examples below:

- $E_{\text{capacitor}} = \frac{1}{2}CV^2$
- $E_{\text{spring}} = \frac{1}{2}kx^2$
- $E_{\text{kinetic}} = \frac{1}{2}mv^2$

And so we find that the definition we use is a natural one.

(a) Consider the scalar system:

$$x[i+1] = ax[i] + bu[i] \tag{1}$$

where $x[0] = 0$ is the initial condition and $u[i]$ is the control input we get to apply based on the current state. Consider if we want to reach a certain state, at a certain time, namely $x[\ell]$. **Write a matrix equation for how a choice of values of $u[i]$ for $i \in \{0, 1, \ldots, \ell - 1\}$ will determine the output at time ℓ.**

(HINT: Write out all the inputs as a column vector $\begin{bmatrix} u[\ell-1] & u[\ell-2] & \cdots & u[1] & u[0] \end{bmatrix}^\top$ and figure out the combination of a and b that gives you the state at time ℓ.

This involves some amount of recursion unrolling, which you have seen in discussion before; the same logic can be carried over to write the state at some time ℓ in terms of all the previous inputs.)

Solution: From the hints, we can derive inspiration for this problem from the structure of our discretization equation, many discussions ago. We write out the state for a few timesteps, to get a feel for the pattern.

$$\begin{aligned}
x[0] &= 0 && (2)\\
x[1] &= ax[0] + bu[0] && (3)\\
&= bu[0] && (4)\\
x[2] &= ax[1] + bu[1] && (5)\\
&= abu[0] + bu[1] && (6)\\
x[3] &= a^2bu[0] + abu[1] + bu[2] && (7)
\end{aligned}$$

At this point, we can see the pattern; each time we proceed a timestep, the existing terms get multiplied by a as they travel through the system, and the most recent input is multiplied by b. The general case, then, is:

$$x[\ell] = a^{\ell-1}bu[0] + a^{\ell-2}bu[1] + \cdots + abu[\ell-2] + bu[\ell-1] \tag{8}$$

Composing this in matrix-vector form:

$$x[\ell] = \begin{bmatrix} b & ab & \cdots & a^{\ell-2}b & a^{\ell-1}b \end{bmatrix} \begin{bmatrix} u[\ell-1] \\ u[\ell-2] \\ \vdots \\ u[1] \\ u[0] \end{bmatrix} \tag{9}$$

(b) Consider the scalar system:

$$x[i+1] = 1.0x[i] + 0.7u[i] \tag{10}$$

where $x[0] = 0$ is the initial condition and $u[i]$ is the control input we get to apply based on the current state. Suppose if we want to reach a certain state at a certain time, namely $x[\ell] = 14$ at $\ell = 10$. **With our dynamics ($a = 1$), make an argument for what the best way is to get to a specific state $x[\ell] = 14$, when $\ell = 10$.** When we say *best way* to control a system, we want the squared sum of the inputs to be minimized (see top of worksheet for motivation behind this definition). Mathematically:

$$\underset{u[i]}{\operatorname{argmin}} \sum_{i=0}^{\ell-1} u[i]^2 \tag{11}$$

(HINT: Think about making a symmetry-based argument. Also, consider the following variant to this problem. Suppose we only have 2 inputs, and had to reach the same final state $x[\ell] = 14$. Would you choose to apply inputs $u[0] = 10, u[1] = 10$ or $u[0] = 9, u[1] = 11$, which combination would you choose and why?)

Solution: Starting from the previous part, and noticing that $a = 1, b = 0.7$ in this specific problem, we have

$$x[\ell] = 0.7u[\ell-1] + 0.7u[\ell-2] + \cdots + 0.7u[1] + 0.7u[0]. \tag{12}$$

Then, we can plug in what we know about $x[\ell]$ in terms of our ultimate goal to get

$$14 = 0.7u[\ell-1] + 0.7u[\ell-2] + \cdots + 0.7u[1] + 0.7u[0]. \tag{13}$$

First, let's divide our original equation by 0.7 to normalize the input scalar coefficients.

$$20 = u[\ell-1] + u[\ell-2] + \cdots + u[1] + u[0]. \tag{14}$$

Now, let's begin to think about whether this optimization problem has any symmetry, as the hint guides us to do. There are 2 ways to go about this; the first is to consider the structure of the equation above. From the perspective of the operations involved (simple addition of all the inputs), none of the inputs has any "preference" or difference from the other inputs. This means that there is no reason to make any one input larger than any other, and ultimately, we can conclude that the optimal input sequence is to make every input the same value here.

The other approach to conclude the same result (that the inputs should all be the same) is to consider the case in the hint. If we had only 2 inputs to apply to reach the state $x[\ell] = 14$, we can choose among many options; a couple important cases are $u[0] = 10, u[1] = 10$ or $u[0] = 9, u[1] = 11$ (these options both sum to 20). The first case, based on our cost being the squared sum of inputs, has a cost of $10^2 + 10^2 = 200$. The second case has a cost of $9^2 + 11^2 = 202$. The first case has lower cost; in fact, we can show using single-variable calculus (as in a later subpart) that this distribution of input strengths has the minimum cost out of *all* input combinations.

In line with this reasoning, if we now set $u[k] = \overline{u}$ (making them all the same while meeting the constraint that we reach the desired state), for all $k < \ell$, then we see that:

$$20 = \ell \cdot \overline{u}, \implies \overline{u} = \frac{20}{\ell} = 2. \tag{15}$$

This gives us a solution of $u[k] = 2$, at every timestep k before ℓ. This can be interpreted as pushing our state forward by two each timestep.

(c) The previous system was simpler to think about because the full effect of our inputs carried over from one time timestep to another, leading to linear accumulation of the inputs in the overall state. Now, consider a variant of this scalar system:

$$x[i+1] = 0.5x[i] + 0.7u[i] \tag{16}$$

where $x[0] = 0$ is the initial condition and $u[i]$ is the control input we get to apply based on the current state. Consider if we want to reach a certain state at a certain time, again $x[\ell] = 14$, when $\ell = 10$. **Suppose that we can only apply an input at a single time-step to reach the desired state, and it can only be at the very end ($u[9]$) or at the very start ($u[0]$). Based on the scalar model in eq. (16), which option would minimize the squared sum of the single input we apply? Why?**

(HINT: Remember, we still need to reach the same end state $x[10] = 14$.)

Solution: The key realization to make in this subpart is that the new scalar model takes only half of the current timestep's state in constructing the next timestep's state. What does this mean? If we apply some input $u[0] = 1$ (at the very start of the state's evolution), then the effect of this input on the final state, k timesteps later, is $0.7 \cdot 0.5^k$ (the 0.7 comes from the first time the input enters the system, and the remainder of the 0.5 factors come from how the input is now part of the state, and evolves in accordance with a.

However, if we apply this input at the very end, then the same kind of decay is not experienced; the input term $u[9]$ gets multiplied by 0.7 and directly is added to the current state $x[9]$ to arrive at $x[10]$.

Intuitively, this means that given a choice to place inputs closer to the end or closer to the start, we should try and place them closer to the end to make sure they don't decay too much as a result of the accumulation of 0.5 multiplications.

(d) In the previous subpart, we discovered that for the model in eq. (16) it is better to apply inputs at the end rather than at the start. Now, let's suppose that instead of applying our entire input in only a single timestep, we actually distributed it across 2 timesteps (the last input $u[9]$, and the second to last input $u[8]$). Note that if we only apply an input starting $u[8]$, and $x[0] = 0$, then the state remains zero until $x[8]$.

Given that we have two inputs to move our state from $x[8] = 0$ to $x[10] = 14$, what are the optimal values of $u[8], u[9]$ to minimize the squared sum of these 2 inputs? Use a calculus-based approach.

Solution: First, we formulate the goal of the problem: we need to reach $x[10] = 14$ from $x[8] = 0$, using 2 inputs $u[8], u[9]$. In equations, we find that when we unroll the recursion in the style of the first subpart (but only for 2 timesteps), we arrive at the following equation:

$$\begin{aligned} x[10] &= 0.5x[9] + 0.7u[9] && (17)\\ 14 &= 0.5(0.5x[8] + 0.7u[8]) + 0.7u[9] && (18)\\ 14 &= 0.25x[8] + 0.35u[8] + 0.7u[9] && (19)\\ 14 &= 0.35u[8] + 0.7u[9] && (20) \end{aligned}$$

We are trying to solve this equation subject to the minimization of the sum of input norms; that is, we want to minimize $u[8]^2 + u[9]^2$.

We can solve this using calculus; from the first equation, we can say that $u[8] = \frac{14-0.7u[9]}{0.35} = 40 - 2u[9]$. The second equation then becomes a single-variable equation we want to minimize:

$$\begin{aligned} u[8]^2 + u[9]^2 &= (40 - 2u[9])^2 + u[9]^2 && (21)\\ &= 1600 - 160u[9] + 4u[9]^2 + u[9]^2 && (22)\\ f_u(u[9]) &= 1600 - 160u[9] + 5u[9]^2 && (23) \end{aligned}$$

To minimize this final expression (call it f_u), we differentiate with respect to $u[9]$, set the result equal to zero to find extrema (stationary points), and back-substitute to find $u[8]$ from $u[9]$.

Notice that the shape of f_u is an upward-facing parabola; this will have a *minimum* in $u[9]$, as expected.

$$\frac{\mathrm{d}f_u(u[9])}{\mathrm{d}u[9]} = -160 + 10u[9] = 0 \tag{24}$$

$$\implies u[9] = 16 \tag{25}$$

From the equation that $u[8] = 40 - 2u[9]$, we see that $u[8] = 8$. Therefore, the optimal combination of inputs here is $u[8] = 8, u[9] = 16$, and we can confirm from the original constraint equation that this does indeed satisfy the requirement in our system (that we reach $x[10] = 14$).

The single-input cost (a single input $u[9] = 20$ at the very end) is $20^2 = 400$, whereas this optimal we found has cost $8^2 + 16^2 = 320$.

(e) With the previous subparts having established useful results and given practical insight, we are well-equipped to solve the most general form of the problem. As a reminder, we are given the scalar system:

$$x[i+1] = 0.5x[i] + 0.7u[i] \tag{26}$$

where $x[0] = 0$ is the initial condition and $u[i]$ is the control input we get to apply based on the current state. Consider if we want to reach a certain state at a certain time, $x[\ell] = 14$, when $\ell = 10$. **Explain in words the trend of the control input that will be used to solve this problem. It will help to consider the intution built in part 1.d; more recent inputs need to be stronger, and the ratio between consecutive inputs is related to the value of a (here, 0.5).**

Solve for the exact inputs that minimize the energy (squared sum) of the inputs.

Solution: Similar to the previous part, we begin by writing $x[\ell]$ in terms of the input sequence $u[0] \cdots u[\ell - 1]$:

$$x[\ell] = \begin{bmatrix} 0.7 & 0.7 \cdot 0.5 & \cdots & 0.7 \cdot 0.5^{\ell-2} & 0.7 \cdot 0.5^{\ell-1} \end{bmatrix} \begin{bmatrix} u[\ell-1] \\ u[\ell-2] \\ \vdots \\ u[1] \\ u[0] \end{bmatrix} = \langle \vec{v}, \vec{u} \rangle . \tag{27}$$

In the previous parts, we built intuition on what kind of input sequence will lead to the desired reuslt under the minimum norm constraint. We also solved rigorously in the case of 2 inputs, demonstrating that the more recent input should be twice as strong as the other input (when $a = 0.5$). The key result is that each input must become stronger and stronger. From a fact of mathematical analysis called the Cauchy-Schwarz inequality (the proof of which relies on vector projections, but is omitted here to save time during section), we know that, to minimize $\|\vec{u}\|^2$, the input $\vec{u}$ and the controllability matrix C are linearly dependent. In the scalar case, this means that the minimum norm solution $\vec{u}$ will be one that is linearly dependent (aligned and proportional) to $\vec{v}$.

Thus:

$$\vec{u} = \alpha \vec{v} \tag{28}$$

$$\implies x[\ell] = 14 = \langle \vec{v}, \vec{u} \rangle \tag{29}$$

$$= \alpha \langle \vec{v}, \vec{v} \rangle \tag{30}$$

$$= \alpha \|\vec{v}\|^2 \tag{31}$$

$$14 \approx \alpha \cdot 0.808^2 \tag{32}$$

$$\implies \alpha \approx 21.42 \tag{33}$$

$$\implies \vec{u} = 21.42 \cdot \vec{v}. \tag{34}$$

We know exactly what $\vec{v}$ is from the definition of the system, so this gives us a final solution of $u[i] = 21.42 \cdot 0.7 \cdot 0.5^{(\ell-1-i)} = 15 \cdot 0.5^{(\ell-1-i)} = 15 \cdot 0.5^{(9-k)}, \forall k \in \{0 ... \ell - 1\}$.

The main idea to notice here is that the effect of earlier inputs carries through for longer because of the accumulation, and the effect of these inputs on later states *diminishes over time*. Specifically, we want to put our strongest "push" input closest to the end timestep (applied on the 9th timestep) because this input does not get multipled by a (halved) in its effect on the final system. This is the same insight developed in a previous part.

By this same logic, we want our first input $u[0]$ to be somewhat weak; we know that by the time the 10th timestep comes around, the effect of $u[0]$ is 512 times weaker because of the repeated multiplications by 0.5. Putting a ton of energy in this control doesn't make sense for that reason.

We similarly know that applying an extreme input at the very end (like $u[9] = 20$) is not optimal; *the cost of applying larger inputs increases as a quadratic function of the input magnitude.* This is by definition, because we have defined our "optimal" to be in terms of minimum *energy* (proportional to the square of the value). We could have very reasonably defined a different metric, but given what we have, we know that doubling the input actually multiplies the cost by a factor of 4. So, there's a balance to be struck between waiting to apply big inputs at the end, and applying too large of inputs early on which don't have much ultimate impact.

(f) Now, consider the following linear discrete time system

$$\vec{x}[i+1] = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \vec{x}[i] + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[i] \tag{35}$$

Set up the system of equations to calculate the state at time $\ell = 20$.

i First, write out the matrices in symbolic form:
Solution:

$$\vec{x}[\ell] = \begin{bmatrix} \vec{b} & A\vec{b} & \cdots & A^{\ell-2}\vec{b} & A^{\ell-1}\vec{b} \end{bmatrix} \begin{bmatrix} u[\ell-1] \\ u[\ell-2] \\ \vdots \\ u[1] \\ u[0] \end{bmatrix} \tag{36}$$

ii In the previous equation, substitute in the first matrix (the one with powers of A) with numbers:
Solution:

$$\vec{x}[\ell] = \begin{bmatrix} 0 & 1 & 0 & 1 & \cdots & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & \cdots & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u[\ell-1] \\ u[\ell-2] \\ \vdots \\ u[1] \\ u[0] \end{bmatrix} \tag{37}$$

(g) Now, suppose that we want to reach the state $x[20] = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ (that is, we reach this state at $\ell = 20$.) **Using the results of the previous subpart, what form does the minimum norm solution take in this problem (vector-case) to reach $x[20] = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$?**

Solution: If we observe the structure of the matrix-vector multiplication above, we arrive at a very interesting observation that also makes our calculations convenient. Let's consider just the first row of the left matrix; this tells us the following:

$$x_1[\ell] = 0 \cdot u[\ell-1] + 1 \cdot u[\ell-2] + \cdots + 0 \cdot u[1] + 1 \cdot u[0] \tag{38}$$

$$= u[\ell-2] + u[\ell-4] + \cdots + u[2] + u[0] \tag{39}$$

And similarly from the second row, we have the following:

$$x_2[\ell] = 1 \cdot u[\ell-1] + 0 \cdot u[\ell-2] + \cdots + 1 \cdot u[1] + 0 \cdot u[0] \tag{40}$$
$$= u[\ell-1] + u[\ell-3] + \cdots + u[3] + u[1] \tag{41}$$

Notice what this means; we have two parallel scalar problems, of the exact same form as part **1.**b! What result did we find there? Applying the result of that problem (that we want to apply the same input at all timesteps), we can solve for the optimal input sequences.

For the first component of the state, we see that we add 10 inputs ($u[0], u[2], \cdots, u[18]$) to reach α; each input should therefore be $\frac{\alpha}{10}$. Similarly, the second set of 10 inputs $u[1], u[3], \cdots, u[19]$ must reach β, which means they are $\frac{\beta}{10}$. Combining everything, our input sequence looks as follows:

$$\vec{u} = \frac{1}{10}\begin{bmatrix}\beta \\ \alpha \\ \vdots \\ \beta \\ \alpha\end{bmatrix} \tag{42}$$

(h) **(PRACTICE) Repeat part 1.g with a time horizon of $\ell = 21$.**

Solution: What has changed from the previous part? Now, we apply one more input, for a total of 21 inputs. Naturally, this means that the shared ratio of $\frac{1}{10}$ that we had in the previous problem will no longer hold.

If we write out the same equations as above, we find that we end up applying 10 inputs to x_1, but 11 inputs to x_2. What does this impact in the solution? Our input in each timestep was weighted uniformly to make sure that we reached α and β respectively; now that we have one more input applied to the component being driven to β, the scalar in front of those inputs becomes $\frac{1}{11}$. It stays $\frac{1}{10}$ for the first component, because there are still only 10 total inputs that impact the α component. Finally, we have:

$$\vec{u} = \begin{bmatrix}\frac{1}{11}\beta \\ \frac{1}{10}\alpha \\ \frac{1}{11}\beta \\ \vdots \\ \frac{1}{10}\alpha \\ \frac{1}{11}\beta\end{bmatrix} \tag{43}$$

Notice that the leftmost column in the matrix will always be $\vec{b}$, and therefore always corresponds to an input applied to β. The rightmost column either has an even or odd power of A multiplying $\vec{b}$, where even multiplications yield $\begin{bmatrix}0 \\ 1\end{bmatrix}$ and odd multiplications yield $\begin{bmatrix}1 \\ 0\end{bmatrix}$.

Contributors:

- Anant Sahai.
- Nathan Lambert.
- Kareem Ahmad.
- Neelesh Ramachandran.