EECS 16B Designing Information Devices and Systems II
Spring 2021

Note 7B: System ID

1 Overview

Given a linear differential equation model for a system, we know how we can discretize it and obtain a discrete-time system model. However, this relies on knowing physical parameters to infinite precision and many times, it is not worth investing that much effort in trying to model the system exactly by hand. *System Identification* is about using data collected about the inputs $\vec{u}$ and states $\vec{x}$ to attempt to determine A and B.[1] This process is of great importance, because when we construct real-world mechanical systems, it is practically impossible to determine all their behaviors theoretically - we must do so empirically. In practice, data is also taken continuously to update our models, since in the real world, the model itself can change over time.

Modern system design leverages the ability to learn from data. Fortunately, EECS16A has already given you all the basic tools that you need to do this intelligently, as we see in the following section.

2 System Identification

Let our linear system be:

$$\vec{x}_d[t+1] = A\vec{x}_d[t] + B\vec{u}_d[t] \tag{1}$$

where we observe each of the $\vec{x}_d[t]$. That is, t refers to the current timestep. From here on, it is understood that $\vec{x}_d[t]$ is discrete, so we will use subscripts to denote indices of the discrete-time vector (such as, $\vec{x}_1[t]$ is the value of the first component of $\vec{x}_d[t]$ at time t). We may also refer to $\vec{x}_d[t]$ simply as $\vec{x}[t]$.

Let's express our unknown matrices A and B in terms of scalars a_{ij}, b_{ij}, as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nk} \end{bmatrix} \tag{2}$$

Substituting into our relation for $\vec{x}[t]$, we break down our state, observation, and input vectors into their scalar components as well:

$$\begin{bmatrix} x_1[t+1] \\ x_2[t+1] \\ \\ x_n[t+1] \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1[t] \\ x_2[t] \\ \vdots \\ x_n[t] \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1k} \\ b_{21} & b_{22} & \dots & b_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nk} \end{bmatrix} \begin{bmatrix} u_1[t] \\ u_2[t] \\ \vdots \\ u_k[t] \end{bmatrix} \tag{3}$$

Now comes the interesting part. Recall that our unknowns are in fact all the a_{ij} and b_{ij} scalars, and our knowns are all the components of $\vec{x}$ and $\vec{u}$ (x_1, x_2, u_1, u_2, etc.) When solving linear systems, we know that we

[1] Note that here, we are only considering discrete evolution, so $\vec{x}$, A, B are $\vec{x}_d$, A_d, B_d.

should put our unknowns into a vector. By considering each row of the above matrix separately (using r to index the row), we obtain scalar equations of the form:

$$x_r[t+1] = \begin{bmatrix} a_{r1} & a_{r2} & \dots & a_{rn} \end{bmatrix} \begin{bmatrix} x_1[t] \\ x_2[t] \\ \vdots \\ x_n[t] \end{bmatrix} + \begin{bmatrix} b_{r1} & b_{r2} & \dots & b_{rk} \end{bmatrix} \begin{bmatrix} u_1[t] \\ u_2[t] \\ \vdots \\ u_k[t] \end{bmatrix} \tag{4}$$

for all $1 \leq r \leq n$. Notice that the two terms on the right of the equation are really just inner products, producing a scalar. As the inner product is symmetric for real vectors ($\langle x, y \rangle = \langle y, x \rangle$), we can swap the rows and columns and rewrite the above equation as:

$$x_r[t+1] = \begin{bmatrix} x_1[t] & x_2[t] & \cdots & x_n[t] \end{bmatrix} \begin{bmatrix} a_{r1} \\ a_{r2} \\ \vdots \\ a_{rn} \end{bmatrix} + \begin{bmatrix} u_1[t] & u_2[t] & \cdots & u_k[t] \end{bmatrix} \begin{bmatrix} b_{r1} \\ b_{r2} \\ \vdots \\ b_{rk} \end{bmatrix} \tag{5}$$

again for all $1 \leq r \leq n$.

Combining the two terms on the right, we finally obtain the equation

$$x_r[t+1] = \begin{bmatrix} x_1[t] & x_2[t] & \cdots & x_n[t] & u_1[t] & u_2[t] & \cdots & u_k[t] \end{bmatrix} \begin{bmatrix} a_{r1} \\ a_{r2} \\ \vdots \\ a_{rn} \\ b_{r1} \\ b_{r2} \\ \vdots \\ b_{rk} \end{bmatrix} \tag{6}$$

again for all $1 \leq r \leq n$.

Notice that we have managed to place all the unknowns involving the rth element of the state at time t in a single vector. Since we now have a linear equation with our unknowns in a vector, are we done, since we can do Gaussian elimination to solve for the unknowns?

Unfortunately not. Notice that there are $n+k$ unknowns, but only a single equation. We could try to get more equations by considering all values of r simultaneously (all the rows), but that would also bring in new unknowns, since we'd have to consider the a_{rj} and b_{rj} for *all* r. So is all hope lost?

No! Recall that our equation holds for *all* values of t, since we are assuming that our system's transition equation does not vary over time! Notice that for all timesteps $0 < t \leq T$, where T is the current time, the vector of unknowns remains the same. Thus, by considering all these values of t and stacking them

vertically, we obtain the system

$$\begin{bmatrix} x_r[1] \\ x_r[2] \\ \vdots \\ x_r[T] \end{bmatrix} = \begin{bmatrix} x_1[0] & \cdots & x_n[0] & u_1[0] & \cdots & u_k[0] \\ x_1[1] & \cdots & x_n[1] & u_1[1] & \cdots & u_k[1] \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1[T-1] & \cdots & x_n[T-1] & u_1[T-1] & \cdots & u_k[T-1] \end{bmatrix} \begin{bmatrix} a_{r1} \\ a_{r2} \\ \vdots \\ a_{rn} \\ b_{r1} \\ b_{r2} \\ \vdots \\ b_{rk} \end{bmatrix} \tag{7}$$

which consists of T equations, not just 1! For $T \geq n+k$, we can use least squares to solve for our unknowns, and repeat the process for each value of r in order to fully determine our system. Typically, we create a matrix P of our unknowns across all values of r by stacking them horizontally to obtain

$$P = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{nn} \\ b_{11} & b_{21} & \dots & b_{n1} \\ b_{12} & b_{22} & \dots & b_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{1k} & b_{2k} & \dots & b_{nk} \end{bmatrix} = \begin{bmatrix} A^\top \\ B^\top \end{bmatrix}. \tag{8}$$

We can follow a similar stacking procedure to obtain

$$D = \begin{bmatrix} x_1[0] & \cdots & x_n[0] & u_1[0] & \cdots & u_k[0] \\ x_1[1] & \cdots & x_n[1] & u_1[1] & \cdots & u_k[1] \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1[T-1] & \cdots & x_n[T-1] & u_1[T-1] & \cdots & u_k[T-1] \end{bmatrix} \tag{9}$$

and

$$S = \begin{bmatrix} x_1[1] & x_2[1] & \cdots & x_n[1] \\ x_1[2] & x_2[2] & \cdots & x_n[2] \\ \vdots & \ddots & \ddots & \vdots \\ x_1[T] & x_2[T] & \cdots & x_n[T] \end{bmatrix} \tag{10}$$

Thus, we can combine all our values of r into the single approximate equation

$$DP \approx S$$

and solve this using least squares to obtain

$$P = \left(D^\top D\right)^{-1} D^\top S \tag{11}$$

Why do we use least-squares here? Because we are going to assume that any measurement errors or unmodeled terms are all relatively small. So it makes sense to pick a solution that has a small residual, and least squares gives that to us. (We will see in the next note how we could make such solutions robust to a few points being hit with something much larger.)

Recall that $D^\top D$ is invertible exactly when D has linearly independent columns. We will discuss what happens when the columns made out of the x_r are dependent later - for now, we will simply comment that choosing our inputs randomly will ensure with high probability that the input columns are all linearly independent both of each other and of the earlier x_r columns.

Contributors:

- Neelesh Ramachandran.
- Rahul Arya.
- Anant Sahai.