EECS 16B Designing Information Devices and Systems II
Fall 2019

Note: System ID

1 Overview

We know how given a linear differential equation model for a system 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. This process is of great importance, as when constructing real-world mechanical systems, it is practically impossible to determine all their behaviors theoretically - we must do so empirically. And in practice, data is also taken continuously to update our models since in the real world, things can change over time.

Modern system design leverages the ability to learn from data. Fortunately, 16A has already given you all the basic tools that you need to do this intelligently.

2 System Identification

Let our linear system be

$$\vec{x}(i+1) = A\vec{x}(i) + B\vec{u}(i)$$

where we observe each of the $\vec{x}(i)$.

Let's express our unknown matrices A and B in terms of scalars, as follows:

$$A = \begin{bmatrix} a[1][1] & a[1][2] & \dots & a[1][n] \\ a[2][1] & a[2][2] & \dots & a[2][n] \\ \vdots & \vdots & \ddots & \vdots \\ a[n][1] & a[n][2] & \dots & a[n][n] \end{bmatrix}$$

$$B = \begin{bmatrix} b[1][1] & b[1][2] & \dots & b[1][k] \\ b[2][1] & b[2][2] & \dots & b[2][k] \\ \vdots & \vdots & \ddots & \vdots \\ b[n][1] & b[n][2] & \dots & b[n][k] \end{bmatrix}.$$

Substituting into our relation for $\vec{x}$, we obtain

$$\begin{bmatrix} x[1](i+1) \\ x[2](i+1) \\ \\ x[n](i+1) \end{bmatrix} = \begin{bmatrix} a[1][1] & a[1][2] & \dots & a[1][n] \\ a[2][1] & a[2][2] & \dots & a[2][n] \\ \vdots & \vdots & \ddots & \vdots \\ a[n][1] & a[n][2] & \dots & a[n][n] \end{bmatrix} \begin{bmatrix} x[1](i) \\ x[2](i) \\ \vdots \\ x[n](i) \end{bmatrix} + \begin{bmatrix} b[1][1] & b[1][2] & \dots & b[1][k] \\ b[2][1] & b[2][2] & \dots & b[2][k] \\ \vdots & \vdots & \ddots & \vdots \\ b[n][1] & b[n][2] & \dots & b[n][k] \end{bmatrix} \begin{bmatrix} u[1](i) \\ u[2](i) \\ \vdots \\ u[k](i) \end{bmatrix},$$

breaking down our state, observation, and input vectors into their scalar components as well.

Now comes the interesting part. Recall that our unknowns are in fact all the $a[i][j]$ and $b[i][j]$, and our knowns are all the $\vec{x}[i]$ and $\vec{u}[i]$. When solving linear systems, we know that we should put our unknowns into a vector. By considering each row of the above matrix separately, we obtain scalar equations of the form

$$x[j](i+1) = \begin{bmatrix} a[j][1] & a[j][2] & \dots & a[j][n] \end{bmatrix} \begin{bmatrix} x[1](i) \\ x[2](i) \\ \vdots \\ x[n](i) \end{bmatrix} + \begin{bmatrix} b[j][1] & b[j][2] & \dots & b[j][k] \end{bmatrix} \begin{bmatrix} u[1](i) \\ u[2](i) \\ \vdots \\ u[k](i) \end{bmatrix}$$

for all $1 \leq j \leq n$. 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, we may rewrite the above equation as

$$x[j](i+1) = \begin{bmatrix} x[1](i) & x[2](i) & \cdots & x[n](i) \end{bmatrix} \begin{bmatrix} a[j][1] \\ a[j][2] \\ \vdots \\ a[j][n] \end{bmatrix} + \begin{bmatrix} u[1](i) & u[2](i) & \cdots & u[k](i) \end{bmatrix} \begin{bmatrix} b[j][1] \\ b[j][2] \\ \vdots \\ b[j][k] \end{bmatrix}$$

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

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

$$x[j](i+1) = \begin{bmatrix} x[1](i) & x[2](i) & \cdots & x[n](i) & u[1](i) & u[2](i) & \cdots & u[k](i) \end{bmatrix} \begin{bmatrix} a[j][1] \\ a[j][2] \\ \vdots \\ a[j][n] \\ b[j][1] \\ b[j][2] \\ \vdots \\ b[j][k] \end{bmatrix}$$

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

Notice that we have managed to place all the unknowns involving the jth element of the state 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 j simultaneously, but that would also bring in new unknowns, since we'd have to consider the $a[j][i]$ and $b[j][i]$ for *all* j. So is all hope lost?

No! Recall that our equation holds for *all* values of i, since we are assuming that our system's transition equation does not vary over time! Notice that for all timesteps $0 < i \leq t$, the vector of unknowns remains

the same. Thus, by considering all these values of i and stacking them vertically, we obtain the system

$$\begin{bmatrix} x[j](1) \\ x[j](2) \\ \vdots \\ x[j](t) \end{bmatrix} = \begin{bmatrix} x[1](0) & \cdots & x[n](0) & u[1](0) & \cdots & u[k](0) \\ x[1](1) & \cdots & x[n](1) & u[1](1) & \cdots & u[k](1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \\ x[1](t-1) & \cdots & x[n](t-1) & u[1](t-1) & \cdots & u[k](t-1) \end{bmatrix} \begin{bmatrix} a[j][1] \\ a[j][2] \\ \vdots \\ a[j][n] \\ b[j][1] \\ b[j][2] \\ \vdots \\ b[j][k] \end{bmatrix}$$

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

$$P = \begin{bmatrix} a[1][1] & a[2][1] & \dots & a[n][1] \\ a[1][2] & a[2][2] & \dots & a[n][2] \\ \vdots & \vdots & \ddots & \vdots \\ a[1][n] & a[2][n] & \dots & a[n][n] \\ b[1][1] & b[2][1] & \dots & b[n][1] \\ b[1][2] & b[2][2] & \dots & b[n][2] \\ \vdots & \vdots & \ddots & \vdots \\ b[1][k] & b[2][k] & \dots & b[n][k] \end{bmatrix} = \begin{bmatrix} A^T \\ B^T \end{bmatrix}.$$

We can follow a similar stacking procedure to obtain

$$D = \begin{bmatrix} x[1](0) & \cdots & x[n](0) & u[1](0) & \cdots & u[k](0) \\ x[1](1) & \cdots & x[n](1) & u[1](1) & \cdots & u[k](1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x[1](t-1) & \cdots & x[n](t-1) & u[1](t-1) & \cdots & u[k](t-1) \end{bmatrix}$$

and

$$S = \begin{bmatrix} x[1](1) & x[2](1) & \cdots & x[n](1) \\ x[1](2) & x[2](2) & \cdots & x[n](2) \\ \vdots & \ddots & \vdots & \\ x[1](t) & x[2](t) & \cdots & x[n](t) \end{bmatrix}.$$

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

$$DP \approx S$$

and solve this using least squares to obtain

$$P = (D^T D)^{-1} D^T S.$$

Why do we use least-squares here? Because we are going to assume that any measurement errors or unmod-

eled 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 $D^T D$ is invertible exactly when D has linearly independent columns. We will discuss what happens when the columns made out of the $x[j]$ 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 $x[j]$ columns.

Contributors:

- Rahul Arya.
- Anant Sahai.

