Note 10: Discretization, Disturbances, System ID

1 Overview and Recap

Doing control in the real world requires knowing the following concepts:

Key Idea 1 (Discretization, Disturbance, System Identification)

1. *Discretization* is the technique of finding a discrete-time model that approximates a given continuous-time model.
2. *Disturbances* are factors in the underlying system that are not accounted for in our model.
3. *System identification* is the technique of learning estimates for the model parameters from data.

Model 2 (Continuous-Time LTI Differential Equation Model)

The model is of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}_c(t) = A_c\vec{x}_c(t) + B_c\vec{u}_c(t) \qquad t \in \mathbb{R}_+ \tag{1}$$

$$\vec{x}_c(0) = \vec{x}_0 \tag{2}$$

where $\vec{x}_c\colon \mathbb{R}_+ \to \mathbb{R}^n$ is the state of the system as a function of time, $\vec{u}_c\colon \mathbb{R}_+ \to \mathbb{R}^m$ is the control input as a function of time, and $A_c \in \mathbb{R}^{n\times n}$, $B_c \in \mathbb{R}^{n\times m}$ are matrices.

Model 3 (Discrete-Time LTI Difference Equation Model)

The model is of the form

$$\vec{x}_d[i+1] = A_d\vec{x}_d[i] + B_d\vec{u}_d[i] \qquad i \in \mathbb{N} \tag{3}$$

$$\vec{x}_d[0] = \vec{x}_0 \tag{4}$$

where $\vec{x}_d\colon \mathbb{N} \to \mathbb{R}^n$ is the state of the system as a function of timestep, $\vec{u}_d\colon \mathbb{N} \to \mathbb{R}^m$ is the control input as a function of timestep, and $A_d \in \mathbb{R}^{n\times n}$, $B_d \in \mathbb{R}^{n\times m}$ are matrices.

NOTE: We use the c and d subscripts to distinguish continuous and discrete time.

2 Discretization

We want to *approximate* a continuous-time LTI differential equation model by a discrete-time LTI difference equation model.

2.1 Tracing a Continuous-Time System

One way we can start this approximation process is to let our $\vec{x}_d[i]$ be *traces*, e.g., samples of our system state $\vec{x}_c$ at discrete timesteps of length Δt. This means that we will be working with the quantities $\vec{x}_d[0] = \vec{x}_c(0)$, $\vec{x}_d[1] = \vec{x}_c(\Delta t)$, $\vec{x}_d[2] = \vec{x}_c(2\Delta t)$, etc.. In general, we use the traced states

$$\vec{x}_d[i] := \vec{x}_c(i\Delta t) \qquad i \in \mathbb{N}. \tag{5}$$

Similarly, we use the traced inputs

$$\vec{u}_d[i] := \vec{u}_c(i\Delta t) \qquad i \in \mathbb{N}. \tag{6}$$

Now, using these traced states and inputs, and knowing that our system evolves according to Continuous-Time LTI Differential Equation Model, we would like to make a Discrete-Time LTI Difference Equation Model. More formally, knowing that our system obeys the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}_c(t) = A_c\vec{x}_c(t) + B_c\vec{u}_c(t) \qquad t \in \mathbb{R}_+ \tag{7}$$

we would like to find $A_d \in \mathbb{R}^{n\times n}$ and $B_d \in \mathbb{R}^{n\times m}$, independent of the timestep i, the initial condition $\vec{x}_0$, and the inputs $\vec{u}_c$, such that

$$\vec{x}_d[i+1] = A_d\vec{x}_d[i] + B_d\vec{u}_d[i] \qquad i \in \mathbb{N} \tag{8}$$

$$\text{or more explicitly} \qquad \vec{x}_c((i+1)\Delta t) = A_d\vec{x}_c(i\Delta t) + B_d\vec{u}_c(i\Delta t) \qquad i \in \mathbb{N}. \tag{9}$$

2.2 Piecewise Constant Input Assumption

Unfortunately, the desire we have is impossible to achieve, unless we have more information about the control inputs $\vec{u}_c$ between the endpoints $i\Delta t$. The reason is that we don't have any information about how $\vec{u}_c$ behaves in between the traces $\vec{u}_d[i]$ and $\vec{u}_d[i+1]$ – it can be arbitrarily badly behaved within the interval $(i\Delta t, (i+1)\Delta t)$. So, it would be impossible to get coefficients A_d and B_d which multiply $\vec{x}_d[i]$ and $\vec{u}_d[i]$ to get $\vec{x}_d[i+1]$ that work for any input $\vec{u}_c$. To fix this, we will assume that $\vec{u}_c$ is piecewise constant.

Assumption 4 (Piecewise Constant Inputs)
The inputs $\vec{u}_c$ are piecewise constant on intervals of width Δt, that is,

$$\vec{u}_c(t) = \vec{u}_c(i\Delta t) = \vec{u}_d[i] \qquad \text{for all } t \in [i\Delta t, (i+1)\Delta t). \tag{10}$$

This assumption is also referred to as a *zero-order hold* assumption. This is because the input is piecewise a zero-order polynomial in time, i.e., is piecewise constant. At first, this assumption may seem *unreasonable*. For example, if we are studying a circuit and our input is some form of AC current (and hence sinusoidal), then there is *no* time interval on which the input is constant. Nevertheless, the point is that if the input $\vec{u}_c$ is continuous, then the discretized function $\vec{u}_d$ approximates $\vec{u}_c$ for small Δt. In math,

$$\vec{u}_c(t) = \vec{u}_c\left(\Delta t \cdot \frac{t}{\Delta t}\right) \approx \vec{u}_c\left(\Delta t \left\lfloor \frac{t}{\Delta t} \right\rfloor\right) = \vec{u}_d\left[\left\lfloor \frac{t}{\Delta t} \right\rfloor\right] = \vec{u}_d[i]. \tag{11}$$

We should read eq 11 from left to right, to appreciate the connection between the continuous world and the discrete one. $\vec{u}_c(t)$ is a continuous signal, we do the trick of dividing and multipling by the same factor Δt. Recall that we want to retrieve an index i, that can be used for indexing the discrete representation. To do that we notice that if we approximate $\frac{t}{\Delta t}$ to be $\lfloor \frac{t}{\Delta t} \rfloor$ we retrieve that index i.

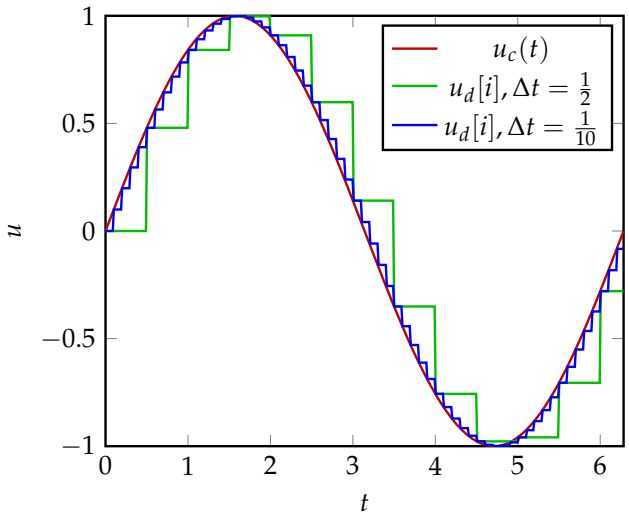

Figure 1: An example of a piecewise constant function where the limit as the time-step Δt goes to 0 approaches a continuous function. The red line, the original signal $u_c(t) = \sin(t)$, is traced almost exactly by the blue line (small Δt) and not nearly as well by the green line (large Δt).

Using our assumption of a piecewise constant input, we can state our model as follows:

Model 5 (Continuous-Time LTI Differential Equation Model with Piecewise Constant Inputs)
The model is of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}_c(t) = A_c\vec{x}_c(t) + B_c\vec{u}_d[i] \qquad t \in [i\Delta t, (i+1)\Delta t) \qquad i \in \mathbb{N} \tag{12}$$

$$\vec{x}_c(0) = \vec{x}_0 \tag{13}$$

where $\vec{x}_c \colon \mathbb{R}_+ \to \mathbb{R}^n$ is the state of the system as a function of time, $\vec{u}_d \colon \mathbb{N} \to \mathbb{R}^m$ is the traced control input as a function of timestep, and $A_c \in \mathbb{R}^{n\times n}, B_c \in \mathbb{R}^{n\times m}$ are matrices.

2.3 Calculating Discretization Coefficients

Theorem 6 (Discretization of Continuous-Time LTI Differential Equation Model with Piecewise Constant Inputs When A_c is Diagonalizable and Invertible)
Suppose in the Continuous-Time LTI Differential Equation Model with Piecewise Constant Inputs that A_c is diagonalizable with diagonalization $A_c = V\Lambda V^{-1}$, and also invertible. The discretization of this model by a Discrete-Time LTI Difference Equation Model has coefficients

$$A_d = \mathrm{e}^{A_c\Delta t} = V\mathrm{e}^{\Lambda\Delta t}V^{-1} \tag{14}$$

$$B_d = V(\mathrm{e}^{\Lambda\Delta t} - I_n)\Lambda^{-1}V^{-1}B_c \tag{15}$$

See optional section for proof.

2.4 Examples

Suppose we have the scalar system

$$\frac{\mathrm{d}}{\mathrm{d}t}x_c(t) = 3x_c(t) + 5u_c(t) \tag{16}$$

where u_c is piecewise constant over intervals of length Δt.

In this case we use theorem 6 with $A_c = 3$ "diagonalized" using $V = 1$ and $\Lambda = 3$, as well as $B_c = 5$, to get the discretization

$$x_d[i+1] = \mathrm{e}^{3\Delta t}x_d[i] + \frac{\mathrm{e}^{3\Delta t} - 1}{3} \cdot 5u_d[i]. \tag{17}$$

Now suppose we have the diagonal system

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}_c(t) = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix} \vec{x}_c(t) + \begin{bmatrix} 5 \\ 1 \end{bmatrix} u_c(t) \tag{18}$$

where u_c is again piecewise constant over intervals of length Δt.

In this case we use theorem 6 with $A_c = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$ "diagonalized" using $V = I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\Lambda = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}$, as well as $B_c = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$, to get the discretization

$$x_d[i+1] = \begin{bmatrix} \mathrm{e}^{3\Delta t} & 0 \\ 0 & \mathrm{e}^{2\Delta t} \end{bmatrix} x_d[i] + \begin{bmatrix} \frac{\mathrm{e}^{3\Delta t}-1}{3} & 0 \\ 0 & \frac{\mathrm{e}^{2\Delta t}-1}{2} \end{bmatrix} \begin{bmatrix} 5 \\ 1 \end{bmatrix} u_d[i]. \tag{19}$$

2.5 (OPTIONAL) Towards Euler Discretization

Since for our applications in control we usually want Δt to be small, it is instructive to see the behavior of the discretization coefficients in the limit $\Delta t \to 0$. The way that we do this is to examine the discretization coefficients, given before, and truncate their series expansions at the terms which are linear in Δt. This is justifiable because when $\Delta t < 1$ is small, $\Delta t^2 \ll \Delta t$, $\Delta t^3 \ll \Delta t^2$, and so on; hence all terms which are second-order or higher in Δt become negligible and may be ignored.

Using the coefficients A_d and B_d, we see

$$A_d = \mathrm{e}^{A_c\Delta t} = I_n + A_c\Delta t + \frac{(A_c\Delta t)^2}{2} + \cdots \approx I_n + A_c\Delta t \tag{20}$$

$$B_d = \mathrm{e}^{A_c\Delta t}\left(\sum_{i=1}^{\infty} \frac{(-A_c)^{i-1}\Delta t^i}{i!}\right) B_c = \left(\sum_{i=0}^{\infty} \frac{(A_c\Delta t)^i}{i!}\right)\left(\sum_{i=1}^{\infty} \frac{(-A_c)^{i-1}\Delta t^i}{i!}\right) B_c \tag{21}$$

$$= (I_n + A_c\Delta t + \cdots)\left(\Delta t - \frac{A_c\Delta t^2}{2} + \cdots\right) B_c \approx B_c\Delta t. \tag{22}$$

This implies that for Δt small enough, an approximate discretization gives a model of the form

$$\vec{x}_d[i+1] \approx (I_n + A_c\Delta t)\vec{x}_d[i] + \Delta t B_c \vec{u}_d[i] \qquad i \in \mathbb{N}. \tag{23}$$

A generalization of this process is called *Euler discretization* and is used to approximately solve nonlinear differential equations.

3 Disturbances and Error

Thus far, our analysis of Continuous-Time LTI Differential Equation Model and Discrete-Time LTI Difference Equation Model has assumed that the original model is a perfect description of the system. In practice, this is not the case. The relevant concept to describe this deviation from the model is *disturbance*, which incorporates random noise from the environment, as well as systemic error, caused by having models which are too simple to describe the very complicated natural system.

Definition 7 (Disturbance)
A *disturbance* is any of the following sources of error in a model:

- random noise from the environment;
- approximation error that comes from having wrong models;
- a combination of the above.

In general a real-world model will have all of these sources of disturbance that we model as *additive disturbances*.

Model 8 (Continuous-Time LTI Differential Equation Model With Additive Disturbance)
The model is of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}_c(t) = A_c\vec{x}_c(t) + B_c\vec{u}_c(t) + \vec{w}_c(t) \qquad t \in \mathbb{R}_+ \tag{24}$$

$$\vec{x}_c(0) = \vec{x}_0 \tag{25}$$

where $\vec{x}_c\colon \mathbb{R}_+ \to \mathbb{R}^n$ is the state of the system as a function of time, $\vec{u}_c\colon \mathbb{R}_+ \to \mathbb{R}^m$ is the control input as a function of time, $\vec{w}_c\colon \mathbb{R}_+ \to \mathbb{R}^n$ is the disturbance in the system as a function of time, and $A_c \in \mathbb{R}^{n\times n}, B_c \in \mathbb{R}^{n\times m}$ are matrices.

Model 9 (Discrete-Time LTI Difference Equation Model With Additive Disturbance)
The model is of the form

$$\vec{x}_d[i+1] = A_d\vec{x}_d[i] + B_d\vec{u}_d[i] + \vec{w}_d[i] \qquad i \in \mathbb{N} \tag{26}$$

$$\vec{x}_d[0] = \vec{x}_0 \tag{27}$$

where $\vec{x}_d\colon \mathbb{N} \to \mathbb{R}^n$ is the state of the system as a function of timestep, $\vec{u}_d\colon \mathbb{N} \to \mathbb{R}^m$ is the control input as a function of timestep, $\vec{w}_d\colon \mathbb{N} \to \mathbb{R}^n$ is the disturbance in the system as a function of timestep, and $A_d \in \mathbb{R}^{n\times n}, B_d \in \mathbb{R}^{n\times m}$ are matrices.

There are generally two ways we deal with additive disturbances in this class:

- We may treat $\vec{w}$ as another input.
  - For the Continuous-Time LTI Differential Equation Model With Additive Disturbance:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}_c(t) = A_c\vec{x}_c(t) + \begin{bmatrix} B_c & I_n \end{bmatrix} \begin{bmatrix} \vec{u}_c(t) \\ \vec{w}_c(t) \end{bmatrix} \tag{28}$$

$$\vec{x}_c(0) = \vec{x}_0 \tag{29}$$

– For the Discrete-Time LTI Difference Equation Model With Additive Disturbance:

$$\vec{x}_d[i+1] = A_d\vec{x}_d[i] + \begin{bmatrix} B_d & I_n \end{bmatrix} \begin{bmatrix} \vec{u}_d[i] \\ \vec{w}_d[i] \end{bmatrix} \qquad i \in \mathbb{N} \tag{30}$$

$$\vec{x}_d[0] = \vec{x}_0 \tag{31}$$

Conceptually, this says that noise is another input to the model, albeit one we cannot control.

- We will thus be able to quantitatively analyze the effects of disturbance using all the techniques we will have developed so far for our familiar disturbance-less systems.
- We may assume the disturbances are negligible. In particular, in many cases it is plausible that disturbances are small, and have a small effect on the system. Thus we may remove the $\vec{w}$ term from our calculations entirely – in effect setting $\vec{w} = \vec{0}_n$ – at the cost of turning the equal signs $=$ in the Continuous-Time LTI Differential Equation Model and Discrete-Time LTI Difference Equation Model into approximations $\approx$.

Key Idea 10

We may develop our theoretical ideas while ignoring disturbance, i.e., only consider models like Continuous-Time LTI Differential Equation Model and Discrete-Time LTI Difference Equation Model. If a problem requires us to consider disturbances, we may fold them into the input, or else approximate them away.

As an example of how disturbance affects a model, consider the following model:

$$x_d[i+1] = 1.1x_d[i] + w_d[i] \tag{32}$$

$$x_d[0] = 1. \tag{33}$$

If we were to see how $x_d[i]$ evolves over time, we might see a plot like in fig. 2.

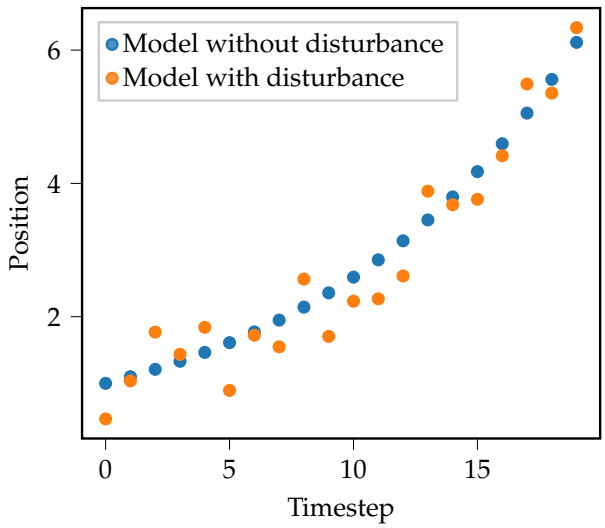

Figure 2: Evolution of a Discrete-Time LTI Difference Equation Model with and without disturbance

4 System Identification

In many cases we don't have a Continuous-Time LTI Differential Equation Model to work with, or at least one where we know the model parameters A_c and B_c with a large degree of certainty. As a result, we would not be able to discretize and get a computational model that we could use – because the discretization coefficients A_d and B_d would be unknown to us.

This is when system identification comes in. Given the assumption that we are using a Discrete-Time LTI Difference Equation Model, but no knowledge of the coefficients A_d and B_d, system identification allows us to learn *estimates* $\widehat{A}_d$ and $\widehat{B}_d$ for our model parameters *from data*, say some data of the form $(\vec{x}_d[i], \vec{u}_d[i], \vec{x}_d[i+1])$ for some integer timesteps $i \in \{i_1, \ldots, i_\ell\}$. Moreover, to get a notion of the "best" estimate for A_d and B_d, we need to define a notion of *how bad* our estimate $(\widehat{A}_d, \widehat{B}_d)$ is, so we can minimize it.

4.1 Least Squares for Error Minimization

Theorem 11 (Least Squares for Vector Estimation)
Suppose $D \in \mathbb{R}^{a\times b}$ has full column rank, and $\vec{s} \in \mathbb{R}^a$. If we define

$$\widehat{p} := (D^\top D)^{-1} D^\top \vec{s} \tag{34}$$

then[a]

$$\widehat{p} \in \operatorname*{argmin}_{\vec{p}\in\mathbb{R}^b} \|D\vec{p} - \vec{s}\|^2. \tag{35}$$

[a] The notation "$\in$ argmin" says that $\widehat{p}$ is one of the minimizers of the squared norm; it does not have to be the only one.

But, we are trying to estimate matrices, not vectors.

Theorem 12 (Least Squares for Matrix Estimation)
Suppose $D \in \mathbb{R}^{a\times b}$ has full column rank, and $S \in \mathbb{R}^{a\times c}$. If we define

$$\widehat{P} := (D^\top D)^{-1} D^\top S \tag{36}$$

then

$$\widehat{P} \in \operatorname*{argmin}_{P\in\mathbb{R}^{b\times c}} \|DP - S\|^2. \tag{37}$$

Notice that the norm we are using is the norm of a matrix and it is defined as follow: Let $A \in \mathbb{R}^{m\times n}$ be a matrix. The *Frobenius norm* of A is

$$\|A\| = \|A\|_F := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |A_{ij}|^2} \tag{38}$$

Proof. Let $P \in \mathbb{R}^{b\times c}$ have columns $\vec{p}_1, \ldots, \vec{p}_c$, $S \in \mathbb{R}^{a\times c}$ have columns $\vec{s}_1, \ldots, \vec{s}_c$, and write

$$\|DP - S\|^2 = \left\| D\begin{bmatrix}\vec{p}_1 & \cdots & \vec{p}_c\end{bmatrix} - \begin{bmatrix}\vec{s}_1 & \cdots & \vec{s}_c\end{bmatrix} \right\|^2 \tag{39}$$

$$= \left\| \begin{bmatrix}D\vec{p}_1 & \cdots & D\vec{p}_c\end{bmatrix} - \begin{bmatrix}\vec{s}_1 & \cdots & \vec{s}_c\end{bmatrix} \right\|^2 \tag{40}$$

$$= \left\| \begin{bmatrix} D\vec{p}_1 - \vec{s}_1 & \cdots & D\vec{p}_c - \vec{s}_c \end{bmatrix} \right\|^2 \tag{41}$$

$$= \sum_{i=1}^{c} \|D\vec{p}_i - \vec{s}_i\|^2. \tag{42}$$

Each of the summands is independent and, by theorem 11 may be minimized by using

$$\vec{\widehat{p}}_i := (D^\top D)^{-1} D^\top \vec{s}_i. \tag{43}$$

Then

$$\widehat{P} := \begin{bmatrix} \widehat{p}_1 & \cdots & \widehat{p}_c \end{bmatrix} \tag{44}$$

$$= \begin{bmatrix} (D^\top D)^{-1} D^\top \vec{s}_1 & \cdots & (D^\top D)^{-1} D^\top \vec{s}_c \end{bmatrix} \tag{45}$$

$$= (D^\top D)^{-1} D^\top \begin{bmatrix} \vec{s}_1 & \cdots & \vec{s}_c \end{bmatrix} \tag{46}$$

$$= (D^\top D)^{-1} D^\top S \tag{47}$$

as desired. □

4.2 Least Squares for System Identification

In order to use the least squares theorems we proved, we need to convert our data into a data matrix D and an output matrix S, and have a parameter matrix P such that $DP \approx S$. Let us start with the simplest case: we have one data point $(\vec{x}_d[i], \vec{u}_d[i], \vec{x}_d[i+1])$. We already know that they are linearly related by

$$\vec{x}_d[i+1] = A_d \vec{x}_d[i] + B_d \vec{u}_d[i]. \tag{48}$$

Writing in matrix-vector form, we have

$$\vec{x}_d[i+1] = \begin{bmatrix} A_d & B_d \end{bmatrix} \begin{bmatrix} \vec{x}_d[i] \\ \vec{u}_d[i] \end{bmatrix}. \tag{49}$$

In order to place our data points in rows that can be vertically stacked, we can take the transpose of both sides of the above equation to obtain:

$$\vec{x}_d[i+1]^\top = \left(\begin{bmatrix} A_d & B_d \end{bmatrix} \begin{bmatrix} \vec{x}_d[i] \\ \vec{u}_d[i] \end{bmatrix} \right)^\top = \begin{bmatrix} \vec{x}_d[i]^\top & \vec{u}_d[i]^\top \end{bmatrix} \begin{bmatrix} A_d^\top \\ B_d^\top \end{bmatrix}. \tag{50}$$

Now we may define:

$$P := \begin{bmatrix} A_d^\top \\ B_d^\top \end{bmatrix}. \tag{51}$$

We can now stack our timesteps $i_1, \ldots, i_\ell$, to form the D and S matrices:

$$D := \begin{bmatrix} \vec{x}_d[i_1]^\top & \vec{u}_d[i_1]^\top \\ \vdots & \vdots \\ \vec{x}_d[i_\ell]^\top & \vec{u}_d[i_\ell]^\top \end{bmatrix} \in \mathbb{R}^{\ell \times (m+n)} \qquad S := \begin{bmatrix} \vec{x}_d[i_1+1]^\top \\ \vdots \\ \vec{x}_d[i_\ell+1]^\top \end{bmatrix} \in \mathbb{R}^{\ell \times n}. \tag{52}$$

Then we have

$$DP \approx S \tag{53}$$

Given a data matrix D and a vector s, Theorem 11 tells us how to choose a vector p that minimizes Dp - s. In this case, we want to choose a matrix P that minimizes DP - S. We can use Theorem 11 for our situation by simply noting that it can be applied repeatedly for each column of P along with the corresponding column of S. In that case, eq 34 tells us that our optimal chose for P is:

$$\widehat{P} := (D^\top D)^{-1} D^\top S \tag{54}$$

The last question to answer is:

When does this D matrix has full column rank, i.e., when is $D^\top D$ invertible? The answer is that for D to be invertible we require $\text{rank}(D) = n + m$, i.e., the number of columns of D. So, we can only use least squares when $\ell \geq n + m$, because otherwise $\text{rank}(D) \leq \ell < n + m$. But even if $\ell \geq n + m$, we may have some rank deficiency, i.e., many samples may somehow be the same. For now, note that if we choose inputs with some randomness or use real data which has random noise, then D will be full column rank with high probability.[1]

Method 13 (System Identification for Discrete-Time LTI Difference Equation Model)

Input: Integers n m, ℓ where $\ell \geq n + m$.

Input: Data $(\vec{x}_d[i], \vec{u}_d[i], \vec{x}_d[i+1])$ for $i \in \{i_1, \ldots, i_\ell\}$.

Output: Least-squares estimates $\widehat{A}_d$ and $\widehat{B}_d$.

1: let $D := \begin{bmatrix} \vec{x}_d[i_1]^\top & \vec{u}_d[i_1]^\top \\ \vdots & \vdots \\ \vec{x}_d[i_\ell]^\top & \vec{u}_d[i_\ell]^\top \end{bmatrix} \in \mathbb{R}^{\ell \times (n+m)}$

2: let $S := \begin{bmatrix} \vec{x}_d[i_1+1]^\top \\ \vdots \\ \vec{x}_d[i_\ell+1]^\top \end{bmatrix} \in \mathbb{R}^{\ell \times n}$

3: compute $\widehat{P} := (D^\top D)^{-1} D^\top S \in \mathbb{R}^{(n+m)\times n}$

4: extract $\begin{bmatrix} \widehat{A}_d^\top \\ \widehat{B}_d^\top \end{bmatrix} := \widehat{P}$

5: **return** $(\widehat{A}_d, \widehat{B}_d)$

4.3 Example

Suppose we are trying want to use System Identification to discover what happens to be the following model:

$$x[t+1] = \frac{1}{3}x[t] + 5u[t]. \tag{55}$$

In this case, $A_d = \frac{1}{3}$ and $B_d = 5$. Let's suppose that we have the following data triplets, in the format $(x_d[i], u_d[i], x_d[i+1])$, taken from different runs of the same system:

$$\left(\frac{3}{5}, \frac{1}{5}, \frac{6}{5}\right) \tag{56}$$

[1] Interested readers who have taken courses in probability will note that, if the inputs are perturbed by a, say, multivariate Gaussian random variable, then D is full column rank with probability 1. This footnote is entirely out of scope of the course and is optional, but provides context as to what guarantees a reasonable error model can get.

$$\left(1, \frac{1}{3}, 2\right) \tag{57}$$

$$\left(\frac{1}{2}, \frac{3}{4}, \frac{47}{12}\right). \tag{58}$$

Then we would form the matrices

$$D := \begin{bmatrix} \frac{3}{5} & \frac{1}{5} \\ 1 & \frac{1}{3} \\ \frac{1}{2} & \frac{3}{4} \end{bmatrix} \qquad S := \begin{bmatrix} \frac{6}{5} \\ 2 \\ \frac{47}{12} \end{bmatrix}. \tag{59}$$

We would compute

$$D^\top D = \begin{bmatrix} \frac{161}{100} & \frac{497}{600} \\ \frac{497}{600} & \frac{2569}{3600} \end{bmatrix} \tag{60}$$

(if you're following along, please do this with a computer). We would see that $D^\top D$ is invertible and so we can use least squares. We would then get

$$(D^\top D)^{-1} D^\top = \begin{bmatrix} \frac{135}{238} & \frac{225}{238} & -\frac{4}{7} \\ -\frac{45}{119} & -\frac{75}{119} & \frac{12}{7} \end{bmatrix}. \tag{61}$$

Finally multiplying by S we would get

$$\widehat{P} := (D^\top D)^{-1} D^\top S = \begin{bmatrix} \frac{1}{3} \\ 5 \end{bmatrix} \tag{62}$$

which is exactly the true parameters P. We didn't have noise in our system, so $\widehat{P} = P$ exactly; otherwise, we would have some error.

5 (OPTIONAL) Validation

We have built up a few techniques for creating new models in this note, but we are still missing one key component – ensuring that the model we created actually works.

The first way we can validate that our model actually works is by using it for what we want to do, and seeing whether it works. For models that are to be used for control, the goal is typically the successful control of the system. If we can successfully control our system by using our model, then the discretization or system identification is good enough for our purposes. This is the equivalent of testing a software system by deploying it in a production environment and verifying that it works.

However, in many learning contexts, we do not have access to the complete system at the time of building the model. To isolate the performance of the model-building component (similar to the so-called "unit testing" in software engineering), the key is to *test the prediction performance of the model using some data we did not use to learn the model in the first place* – the so-called "*validation dataset*", as we use it to validate the performance of our model, as opposed to the "*training dataset*" of data we used to learn the model.

Definition 14 (Training and Validation Data)

- *Training data* is the set of data samples used to construct estimates for the parameters.
- *Validation data* is the set of data samples that are not used in the estimation process, but are instead used to quantify the accuracy of the parameter estimates.

Normally we evaluate our model quantitatively using a so-called "*loss function*".

Definition 15 (Loss Function)
A loss function L takes in a model's input, output, and parameters, and quantifies the accuracy of the estimated output given the input, compared to the true output.

A very typical example is the squared norm loss for Discrete-Time LTI Difference Equation Model.

$$L(\vec{x}_d[i], \vec{u}_d[i], \vec{x}_d[i+1]; \widehat{A}_d, \widehat{B}_d) := \left\| \vec{x}_d[i+1] - (\widehat{A}_d \vec{x}_d[i] + \widehat{B}_d \vec{u}_d[i]) \right\|^2 \tag{63}$$

The end goal is to pick parameters *using the training dataset* that minimize the average "loss" across all data points *in the validation dataset*. For example, in the above case, we want to solve the problem

$$(\widehat{A}, \widehat{B}) := \underset{\widetilde{A}_d, \widetilde{B}_d}{\operatorname{argmin}} \frac{1}{\ell_{\text{val}}} \sum_{k=1}^{\ell_{\text{val}}} L(\vec{x}_d[i_{\text{val};k}], \vec{u}_d[i_{\text{val};k}], \vec{x}_d[i_{\text{val};k}+1]; \widetilde{A}_d, \widetilde{B}_d) \tag{64}$$

but our $\widehat{A}, \widehat{B}$ can only be functions of the training dataset.

One final additional complexity you should keep in mind when using model-building techniques in the real world is that the model itself may change over time. To combat this additional level of difficulty, data is normally taken continuously to update the model via more sophisticated system identification techniques.

A (OPTIONAL) Proofs

Proof of the Discretization of Continuous-Time LTI Differential Equation Model with Piecewise Constant Inputs. The proof is in several steps. We proceed by changes-of-variables and reducing to problems we have already solved. Note that this proof is for the case when A_c is diagonalizable and invertible.

Step 1. We first look at a particular timestep i. Namely, since we are looking for an update equation which writes $\vec{x}_d[i+1]$ in terms of $\vec{x}_d[i]$ and $\vec{u}_d[i]$, we may assume that our differential equation starts at timestep i and goes until timestep $i+1$, and we know everything up to timestep i including $\vec{x}_d[i]$ and $\vec{u}_d[i]$. This gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}_c(t) = A_c\vec{x}_c(t) + B_c\vec{u}_d[i] \qquad t \in [i\Delta t, (i+1)\Delta t) \qquad \text{for this specific } i \tag{65}$$

$$\vec{x}_c(i\Delta t) = \vec{x}_d[i] \qquad \text{for this specific } i. \tag{66}$$

Step 2. Now we solve the differential equation by reducing it to a previous differential equation we have already solved. Recall that in we solved differential equations of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) \tag{67}$$

$$\vec{x}(0) = \vec{x}_0. \tag{68}$$

There are two differences between the differential equations:

- The notation is different – $\vec{x}_c$ instead of $\vec{x}$, $\vec{u}_c$ instead of $\vec{u}$, etc., and the initial condition is $\vec{x}_d[i]$.
- The initial condition is at time $t = i\Delta t$ instead of $t = 0$.
- The differential equation we have to solve is only valid in an interval – the equation we have already solved is valid everywhere.

The second is the only essential difference, since the first is just a relabeling, and the third is solved by declaring that our solution is only valid in the particular interval. To get around the second difference, we will use *change of variables*. More specifically, let s be defined by

$$s := t - i\Delta t \tag{69}$$

so that if $t \in [i\Delta t, (i+1)\Delta t)$ then $s \in [0, \Delta t)$; and let $\vec{\tilde{x}}_c$ be defined pointwise on $[0, \Delta t)$ by

$$\vec{\tilde{x}}_c(s) := \vec{x}_c(s + i\Delta t) = \vec{x}_c(t). \tag{70}$$

Then

$$\vec{\tilde{x}}_c(0) = \vec{x}_c(i\Delta t) = \vec{x}_d[i]. \tag{71}$$

The differential equation in these coordinates becomes

$$\frac{\mathrm{d}}{\mathrm{d}s}\vec{\tilde{x}}_c(s) = \frac{\mathrm{d}}{\mathrm{d}s}\vec{x}_c(t) \tag{72}$$

$$= \underbrace{\frac{\mathrm{d}t}{\mathrm{d}s}}_{=1} \cdot \frac{\mathrm{d}\vec{x}_c(t)}{\mathrm{d}t} \tag{73}$$

$$= A_c\vec{x}_c(t) + B_c\vec{u}_c(t) \tag{74}$$

$$= A_c\vec{\tilde{x}}_c(s) + B_c\vec{u}_d[i] \qquad s \in [0, \Delta t). \tag{75}$$

So, we have to solve the problem

$$\frac{\mathrm{d}}{\mathrm{d}s}\vec{\tilde{x}}_c(s) = A_c\vec{\tilde{x}}_c(s) + B_c\vec{u}_d[i] \qquad s \in [0, \Delta t) \tag{76}$$

$$\vec{\tilde{x}}_c(0) = \vec{x}_d[i]. \tag{77}$$

We know that the solution is given by

$$\vec{\tilde{x}}_c(s) = V\mathrm{e}^{\Lambda s}V^{-1}\vec{x}_d[i] + \int_0^s V\mathrm{e}^{\Lambda(s-\tau)}V^{-1}B_c\vec{u}_d[i]\,\mathrm{d}\tau \qquad s \in [0, \Delta t). \tag{78}$$

Simplifying and using the fact that integrals are linear, we get

$$\vec{\tilde{x}}_c(s) = V\mathrm{e}^{\Lambda s}V^{-1}\vec{x}_d[i] + V\mathrm{e}^{\Lambda s}\left(\int_0^s \mathrm{e}^{-\Lambda\tau}\,\mathrm{d}\tau\right)V^{-1}B_c\vec{u}_d[i] \qquad s \in [0, \Delta t). \tag{79}$$

To compute this integral, we note that Λ is diagonal and invertible, and hence we can write $\Lambda = \mathrm{diag}(\lambda_{11}, \lambda_{22}, \ldots, \lambda_{nn})$ with $\lambda_{ii} \neq 0$.[2] Thus

$$\int_0^s \mathrm{e}^{-\Lambda\tau}\,\mathrm{d}\tau = \int_0^s \begin{bmatrix} \mathrm{e}^{-\lambda_{11}\tau} & & & \\ & \mathrm{e}^{-\lambda_{22}\tau} & & \\ & & \ddots & \\ & & & \mathrm{e}^{-\lambda_{nn}\tau} \end{bmatrix} \mathrm{d}\tau \tag{80}$$

$$= \begin{bmatrix} -\frac{1}{\lambda_{11}}\mathrm{e}^{-\lambda_{11}\tau} & & & \\ & -\frac{1}{\lambda_{22}}\mathrm{e}^{-\lambda_{22}\tau} & & \\ & & \ddots & \\ & & & -\frac{1}{\lambda_{nn}}\mathrm{e}^{-\lambda_{nn}\tau} \end{bmatrix}\Bigg|_{\tau=0}^{\tau=s} \tag{81}$$

$$= \left(-\mathrm{e}^{-\Lambda\tau}\Lambda^{-1}\right)\Big|_{\tau=0}^{\tau=s} \tag{82}$$

$$= \left(I_n - \mathrm{e}^{-\Lambda s}\right)\Lambda^{-1}. \tag{83}$$

Plugging the value of the integral back in, we have

$$\vec{\tilde{x}}_c(s) = V\mathrm{e}^{\Lambda s}V^{-1}\vec{x}_d[i] + V\mathrm{e}^{\Lambda s}\left(I_n - \mathrm{e}^{-\Lambda s}\right)\Lambda^{-1}V^{-1}B_c\vec{u}_d[i] \qquad s \in [0, \Delta t) \tag{84}$$

$$= V\mathrm{e}^{\Lambda s}V^{-1}\vec{x}_d[i] + V\left(\mathrm{e}^{\Lambda s} - I_n\right)\Lambda^{-1}V^{-1}B_c\vec{u}_d[i] \qquad s \in [0, \Delta t). \tag{85}$$

Step 3. Now we solve for $\vec{x}_d[i+1]$. Indeed,

$$\vec{x}_d[i+1] = \vec{x}_c((i+1)\Delta t) \tag{86}$$

$$= \vec{\tilde{x}}_c(\Delta t) \tag{87}$$

$$= V\mathrm{e}^{\Lambda\Delta t}V^{-1}\vec{x}_d[i] + V\left(\mathrm{e}^{\Lambda\Delta t} - I_n\right)\Lambda^{-1}V^{-1}B_c\vec{u}_d[i]. \tag{88}$$

We see that this is a linear equation for $\vec{x}_d[i+1]$ in terms of $\vec{x}_d[i]$ in the form of eq. (3) with coefficients

$$A_d = V\mathrm{e}^{\Lambda\Delta t}V^{-1} \qquad \text{and} \qquad B_d = V(\mathrm{e}^{\Lambda\Delta t} - I_n)\Lambda^{-1}V^{-1}B_c \tag{89}$$

as desired.

[2] This is a side argument, but it may not be clear why we can have $\lambda_{ii} \neq 0$. This is because A_c is invertible, so A_c^{-1} exists, so $\Lambda^{-1} = V^{-1}A_c^{-1}V$ exists and Λ is invertible. Since Λ is diagonal, Λ^{-1} is diagonal, and its entries are $\frac{1}{\lambda_{ii}}$.

Concept Check: We used the assumption that Λ is invertible in Step 2, i.e., eq. (81), when we computed the integral $\int_0^s e^{-\Lambda\tau}\, d\tau$. You can compute this integral even when Λ has some 0 entries, i.e., A has some 0 eigenvalues and is thus not invertible. Do this computation – you won't get a nice closed form, unfortunately. Instead, the i^{th} diagonal entry of the solution to the integral will be calculated differently depending on whether $\lambda_{ii} = 0$ (in which case $e^{\lambda_{ii}\tau} = 1$) or $\lambda_{ii} \neq 0$ (in which case we use the computation above). Carry through the rest of the coefficient calculation again with your more general solution to the integral. What are the discretization coefficients in this case? This calculation allows us to drop the assumption that A_c is invertible, and really we only need that A_c is diagonalizable. □

Contributors:
- Neelesh Ramachandran.
- Rahul Arya.
- Druv Pai.
- Anish Muthali.
- Anant Sahai.
- Chancharik Mitra.
- Matteo Guarrera.

