Note 9: Introduction to Controls

NOTE: Starting from this note, and really for Modules 2 and 3, the material starts to become a little more mathematically sophisticated. As a result, the notes may become more densely packed with information. Here are some tips to read more mathematically dense notes (although this really applies to papers, textbooks, etc. as well).

- Skim through the note first, get an outline of the main ideas and results covered, then go through it again with an eye to detail for proofs. Re-reading for a second time after doing some practice problems to crystallize your understanding can also be helpful, though be aware that this usually isn't a resource you can use for mathematical text you might encounter in "the wild", i.e., your job or in academia.
- When reading carefully, work out some of the intermediate steps of proofs and calculations yourself using pencil and paper (or whiteboard/chalkboard/tablet/etc.). Also, try to come up with small examples and work out what we are saying might be true for those examples.
- Take notes on the notes! It really helps your working memory to have a shortlisted copy of all the important stuff within easy access.
- Last but not least, *don't get discouraged if it takes you a while to understand the notes!* For a mathematical text, reading a handful of pages an hour carefully is a very robust pace, and you should expect to go around this speed, especially when first learning the concepts.

NOTE: For this note we will use the notation $\mathbb{R}_+$ to mean the set of non-negative real numbers, and the notation $\mathbb{N} = \{0, 1, 2, \dots\}$ to mean the set of natural numbers.

1 Controls Overview

Recall that one of the big-picture motivations for the course is to understand what it takes to create artificial systems that interact with the real world. One important component of this is being able to act in the real world to achieve our goals. Interacting with the real world to achieve our objectives is the subject of the field of *control*. Control is one of the foundational disciplines for robotics and artificial intelligence, and is also very critical for many other areas in EECS such as high performance computing and power systems.

We will study control through the framework of so-called "model-based control." A model is a function that attempts to mathematically describe the way a physical system works, in a sense we will make precise shortly. Often-times, people refer to a model and its underlying physical system interchangeably.

Key Idea 1
The main idea of model-based control is that we leverage an explicit model for the physical system we're concerned with in order to plan and execute our actions.

There are two main quantities of interest when discussing a model, usually both represented as vectors.

Definition 2 (State, Control Input)

- The *state* $\vec{x}$ represents the collection of variables we care about.
- The *control input* $\vec{u}$ is the collection of variables that a controller is able to change in order to push the state where we want.

Now we have the vocabulary to precisely define a control model.

Definition 3 (Control Model)
A control model is a rule that

- takes in: (1) an initial condition for the state, and (2) all control inputs over time;
- and produces: a *state trajectory*, i.e., all states achieved over time.

This is a very abstract definition, so let's discuss two main examples of models we will use in this class.

NOTE: Neither example explicitly gives a formula for the state in terms of the initial condition and inputs. This is because, when analyzing a system and setting up a model, it is usually easiest to map our system onto a differential/difference equation, instead of a solution to said equation (which can be very complicated). From there we can solve for the state trajectory using the formulas we will derive in this note.

Model 4 (Discrete-Time LTI Difference Equation Model)
The model is of the form

$$\vec{x}[i+1] = A\vec{x}[i] + B\vec{u}[i] \tag{1}$$
$$\vec{x}[0] = \vec{x}_0 \tag{2}$$

for $\vec{x}\colon \mathbb{N} \to \mathbb{R}^n$ the state vector as a function of timestep, $\vec{u}\colon \mathbb{N} \to \mathbb{R}^m$ the control inputs as a function of timestep, and $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$ matrices.

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

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) \tag{3}$$
$$\vec{x}(0) = \vec{x}_0 \tag{4}$$

for $\vec{x}\colon \mathbb{R}_+ \to \mathbb{R}^n$ the state vector as a function of time, $\vec{u}\colon \mathbb{R}_+ \to \mathbb{R}^m$ the control inputs as a function of time, and $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}$ matrices.

NOTE: We use square brackets in discrete-time (instead of parentheses) to differentiate between discrete-time functions of timestep (i.e., sequences), and continuous-time functions of time.

Right now the names of these models are probably just jargon, and in particular the "LTI" (a.k.a. "linear time-invariant") portion may not make sense. This is absolutely fine; knowing the models is important, but memorizing the names is not, and you certainly aren't expected to divine the definitions of these words without us telling you. In particular, linearity and time-invariance of models is out of scope for the course, though you can read more about it in section 6.

Solving the differential equation or difference equation will tell us the state trajectory as a function of the initial condition and inputs, which is a more explicit form of the model.

It turns out that in many cases that are important to us, we have the ability to explicitly solve for the state trajectory. The remainder of the note will be dedicated to solving for the state trajectory in both the discrete-time and continuous-time cases, as well as some examples of where each model is used.

2 Solving the Discrete-Time LTI Difference Equation Model

We will start with the discrete-time LTI difference equation model. Given an initial condition $\vec{x}_0$ and a sequence of inputs $\vec{u}$, we would like to compute $\vec{x}[i]$ for every i.

Theorem 6 (State Trajectory in Discrete-Time LTI Difference Equation Model)
In the discrete-time LTI difference equation model, we have

$$\vec{x}[i] = A^i \vec{x}_0 + \sum_{k=0}^{i-1} A^{i-k-1} B \vec{u}[k]. \tag{5}$$

Proof. We start with the fact that eq. (1) looks like a recursion, because to compute $\vec{x}[i+1]$ we require only the most recent state $\vec{x}[i]$, as well as the most recent input $\vec{u}[i]$. Motivated by this, we can try to unroll the recursion, and get

$$\begin{aligned}
\vec{x}[i] &= A\vec{x}[i-1] + B\vec{u}[i-1] && (6)\\
&= A(A\vec{x}[i-2] + B\vec{u}[i-2]) + B\vec{u}[i-1] && (7)\\
&= A^2\vec{x}[i-2] + AB\vec{u}[i-2] + B\vec{u}[i-1] && (8)\\
&= A^2(A\vec{x}[i-3] + B\vec{u}[i-3]) + AB\vec{u}[i-2] + B\vec{u}[i-1] && (9)\\
&= A^3\vec{x}[i-3] + A^2B\vec{u}[i-3] + AB\vec{u}[i-2] + B\vec{u}[i-1] && (10)\\
&= \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots && (11)\\
&= A^{i-1}(A\vec{x}[0] + B\vec{u}[0]) + \sum_{k=1}^{i-1} A^{i-k-1} B\vec{u}[k] && (12)\\
&= A^i\vec{x}[0] + \sum_{k=0}^{i-1} A^{i-k-1} B\vec{u}[k] && (13)\\
&= A^i\vec{x}_0 + \sum_{k=0}^{i-1} A^{i-k-1} B\vec{u}[k]. && (14)
\end{aligned}$$

This gives us the state trajectory of the discrete-time LTI difference equation model.

Concept Check: Check that the trajectory given by eq. (5) is correct, in the sense that it obeys the difference eq. (1) and initial condition eq. (2). □

3 Solving the Continuous-Time LTI Differential Equation Model

We now study the Continuous-Time LTI Differential Equation Model. Given an initial condition $\vec{x}_0$ and all inputs $\vec{u}$, we would like to compute $\vec{x}(t)$ for every t.

As you might imagine, it is not possible to use the recursion approach for this model. Instead, we will rely on our knowledge of differential equations that we have already covered (for example in Note 1), and use it to extend the solution to matrices.

3.1 Scalar Model

In the scalar case, the Continuous-Time LTI Differential Equation Model becomes the following:

Model 7 (Scalar Continuous-Time LTI Differential Equation Model)
The model is of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = ax(t) + bu(t) \tag{15}$$
$$x(0) = x_0 \tag{16}$$

for $x\colon \mathbb{R}_+ \to \mathbb{R}$ the state scalar as a function of time, $u\colon \mathbb{R}_+ \to \mathbb{R}$ the control input as a function of time, and $a \in \mathbb{R}, b \in \mathbb{R}$ scalar coefficients.

Theorem 8 (State Trajectory in Scalar Continuous-Time LTI Differential Equation Model)
In the scalar continuous-time LTI differential equation model, we have

$$x(t) = \mathrm{e}^{at}x_0 + \int_0^t \mathrm{e}^{a(t-\tau)} \cdot bu(\tau)\,\mathrm{d}\tau\,. \tag{17}$$

Proof. The first thing to note is that eqs. (15) and (16) looks like a differential equation we already know how to solve! Recall that in Note 1 we showed that the (unique) solution to the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = \lambda x(t) + u(t) \tag{18}$$
$$x(0) = x_0 \tag{19}$$

is given by

$$x(t) = \mathrm{e}^{\lambda t}x_0 + \int_0^t \mathrm{e}^{\lambda(t-\tau)}u(\tau)\,\mathrm{d}\tau\,. \tag{20}$$

But eqs. (15) and (18) look really similar. In fact, if we replace "λ" with "a" and replace "$u(t)$" by "$bu(t)$", they are exactly the same. Thus, the solution to eqs. (15) and (16) is exactly the same as eq. (20) after we make those replacements. So we have

$$x(t) = \mathrm{e}^{at}x_0 + \int_0^t \mathrm{e}^{a(t-\tau)} \cdot bu(\tau)\,\mathrm{d}\tau\,. \tag{21}$$

This gives us the trajectory of the scalar continuous-time LTI differential equation model. □

So, given an input function u and an initial condition x_0, we can solve for the state trajectory x using the formula eq. (17).

3.2 A Diagonal Model

The next step after solving one scalar differential equation is to solve n independent differential equations. This is accomplished whenever A is diagonal. Indeed, suppose A is diagonal, say with entries $a_{11}, a_{22}, \ldots, a_{nn}$ going from top-left to bottom-right (we write this as $A = \mathrm{diag}(a_{11}, a_{22}, \ldots, a_{nn})$). This gives us the model:

Model 9 (Diagonal Continuous-Time LTI Differential Equation Model)
The model is of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) \tag{22}$$
$$\vec{x}(0) = \vec{x}_0 \tag{23}$$

for $\vec{x}\colon \mathbb{R}_+ \to \mathbb{R}^n$ the state scalar as a function of time, $\vec{u}\colon \mathbb{R}_+ \to \mathbb{R}^m$ the control input as a function of time, $B \in \mathbb{R}^{n\times m}$ a matrix, and $A := \operatorname{diag}(a_{11}, a_{22}, \ldots, a_{nn}) \in \mathbb{R}^{n\times n}$ a diagonal matrix.

Theorem 10 (State Trajectory in Diagonal Continuous-Time LTI Differential Equation Model)
In the diagonal continuous-time LTI differential equation model, we have

$$\vec{x}(t) = \mathrm{e}^{At}\vec{x}_0 + \int_0^t \mathrm{e}^{A(t-\tau)}B\vec{u}(\tau)\,\mathrm{d}\tau\,, \tag{24}$$

where we use the notation

$$\mathrm{e}^{At} := \begin{bmatrix} \mathrm{e}^{a_{11}t} & & & \\ & \mathrm{e}^{a_{22}t} & & \\ & & \ddots & \\ & & & \mathrm{e}^{a_{nn}t} \end{bmatrix} \tag{25}$$

for $A = \operatorname{diag}(a_{11}, a_{22}, \ldots, a_{nn})$ a *square diagonal* matrix.

NOTE: e^{At} is a square matrix of the same size as A.

Proof. We can rewrite eq. (22) as

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) \tag{26}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{bmatrix}\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix} + \begin{bmatrix} (B\vec{u}(t))_1 \\ (B\vec{u}(t))_2 \\ \vdots \\ (B\vec{u}(t))_n \end{bmatrix} \tag{27}$$

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t}x_1(t) \\ \frac{\mathrm{d}}{\mathrm{d}t}x_2(t) \\ \vdots \\ \frac{\mathrm{d}}{\mathrm{d}t}x_n(t) \end{bmatrix} = \begin{bmatrix} a_{11}x_1(t) + (B\vec{u}(t))_1 \\ a_{22}x_2(t) + (B\vec{u}(t))_2 \\ \vdots \\ a_{nn}x_n(t) + (B\vec{u}(t))_n \end{bmatrix}. \tag{28}$$

Writing each row of the vector separately as its own equation, we get

$$\frac{\mathrm{d}}{\mathrm{d}t}x_1(t) = a_{11}x_1(t) + (B\vec{u}(t))_1 \tag{29}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}x_2(t) = a_{22}x_2(t) + (B\vec{u}(t))_2 \tag{30}$$

$$\vdots \qquad\qquad \vdots \tag{31}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}x_n(t) = a_{nn}x_n(t) + (B\vec{u}(t))_n \tag{32}$$

We know how to solve each of these equations; along with the initial conditions, which are seen from eq. (23) to be

$$x_1(0) = (\vec{x}_0)_1 \tag{33}$$

$$x_2(0) = (\vec{x}_0)_2 \tag{34}$$

$$\vdots \qquad \vdots \tag{35}$$

$$x_n(0) = (\vec{x}_0)_n \tag{36}$$

Then each equation is of the form that we solved in section 3.1. Hence we can solve each equation to get

$$x_1(t) = e^{a_{11}t}(\vec{x}_0)_1 + \int_0^t e^{a_{11}(t-\tau)}(B\vec{u}(\tau))_1 \, d\tau \tag{37}$$

$$x_2(t) = e^{a_{22}t}(\vec{x}_0)_2 + \int_0^t e^{a_{22}(t-\tau)}(B\vec{u}(\tau))_2 \, d\tau \tag{38}$$

$$\vdots \qquad\qquad \vdots \tag{39}$$

$$x_n(t) = e^{a_{nn}t}(\vec{x}_0)_n + \int_0^t e^{a_{nn}(t-\tau)}(B\vec{u}(\tau))_n \, d\tau \, . \tag{40}$$

Stacking each scalar equation to get a vector solution, we get

$$\vec{x}(t) = \begin{bmatrix} e^{a_{11}t}(\vec{x}_0)_1 + \int_0^t e^{a_{11}(t-\tau)}(B\vec{u}(\tau))_1 \, d\tau \\ e^{a_{22}t}(\vec{x}_0)_2 + \int_0^t e^{a_{22}(t-\tau)}(B\vec{u}(\tau))_2 \, d\tau \\ \vdots \\ e^{a_{nn}t}(\vec{x}_0)_n + \int_0^t e^{a_{nn}(t-\tau)}(B\vec{u}(\tau))_n \, d\tau \end{bmatrix} \tag{41}$$

$$= \begin{bmatrix} e^{a_{11}t}(\vec{x}_0)_1 \\ e^{a_{22}t}(\vec{x}_0)_2 \\ \vdots \\ e^{a_{nn}t}(\vec{x}_0)_n \end{bmatrix} + \int_0^t \begin{bmatrix} e^{a_{11}(t-\tau)}(B\vec{u}(\tau))_1 \\ e^{a_{22}(t-\tau)}(B\vec{u}(\tau))_2 \\ \vdots \\ e^{a_{nn}(t-\tau)}(B\vec{u}(\tau))_n \end{bmatrix} d\tau \, . \tag{42}$$

Now to simplify to the form we want, we use the definition of matrix multiplication to get

$$\begin{bmatrix} e^{a_{11}t}(\vec{x}_0)_1 \\ e^{a_{22}t}(\vec{x}_0)_2 \\ \vdots \\ e^{a_{nn}t}(\vec{x}_0)_n \end{bmatrix} = \begin{bmatrix} e^{a_{11}t} & & & \\ & e^{a_{22}t} & & \\ & & \ddots & \\ & & & e^{a_{nn}t} \end{bmatrix} \begin{bmatrix} (\vec{x}_0)_1 \\ (\vec{x}_0)_2 \\ \vdots \\ (\vec{x}_0)_n \end{bmatrix} = e^{At}\vec{x}_0 \tag{43}$$

and

$$\begin{bmatrix} e^{a_{11}(t-\tau)}(B\vec{u}(\tau))_1 \\ e^{a_{22}(t-\tau)}(B\vec{u}(\tau))_2 \\ \vdots \\ e^{a_{nn}(t-\tau)}(B\vec{u}(\tau))_n \end{bmatrix} = \begin{bmatrix} e^{a_{11}(t-\tau)} & & & \\ & e^{a_{22}(t-\tau)} & & \\ & & \ddots & \\ & & & e^{a_{nn}(t-\tau)} \end{bmatrix} \begin{bmatrix} (B\vec{u}(\tau))_1 \\ (B\vec{u}(\tau))_2 \\ \vdots \\ (B\vec{u}(\tau))_n \end{bmatrix} = e^{A(t-\tau)}B\vec{u}(\tau). \tag{44}$$

Using this to simplify eq. (42), we get

$$e^{At}\vec{x}_0 + \int_0^t e^{A(t-\tau)}B\vec{u}(\tau) \, d\tau \tag{45}$$

as desired. □

Phew! That was a lot of work. Thankfully, this is the mathematically hardest part of the note. Make sure you can replicate each of the steps and justify why they are true.

So, given an input function $\vec{u}$ and an initial condition $\vec{x}_0$, we can solve for the state trajectory $\vec{x}$ using the formula eq. (24).

3.3 A Diagonalizable Model

Now that we have solved the case of n independent differential equations, let us now see what happens if the differential equations are independent in a desirable basis. That is – let us suppose that the A matrix

can be written as $A = V\Lambda V^{-1}$ for some given diagonal matrix Λ and invertible matrix V. Note that this is equivalent to A being diagonalizable, i.e., having n linearly independent eigenvectors; in this case Λ is actually the matrix of eigenvalues and V is the matrix of eigenvectors of A.

This yields the model:

Model 11 (Diagonalizable Continuous-Time LTI Differential Equation Model)
The model is of the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) \tag{46}$$
$$\vec{x}(0) = \vec{x}_0 \tag{47}$$

for $\vec{x}\colon \mathbb{R}_+ \to \mathbb{R}^n$ the state scalar as a function of time, $\vec{u}\colon \mathbb{R}_+ \to \mathbb{R}^m$ the control input as a function of time, $B \in \mathbb{R}^{n\times m}$ a matrix, and $A \in \mathbb{R}^{n\times n}$ a diagonalizable matrix with diagonalization $A = V\Lambda V^{-1}$.

Theorem 12 (State Trajectory in Diagonalizable Continuous-Time LTI Differential Equation Model)
In the diagonalizable continuous-time LTI differential equation model, we have

$$\vec{x}(t) = V\mathrm{e}^{\Lambda t}V^{-1}\vec{x}_0 + \int_0^t V\mathrm{e}^{\Lambda(t-\tau)}V^{-1}B\vec{u}(\tau)\,\mathrm{d}\tau\,, \tag{48}$$

where we again use the notation in eq. (25).

NOTE: Recall that Λ is the diagonal matrix in the diagonalization of $A = V\Lambda V^{-1}$, so the notation $\mathrm{e}^{\Lambda t}$ is valid and represents the diagonal matrix whose entries are the exponentials $\mathrm{e}^{\lambda_{ii}t}$.

Proof. We can rewrite eq. (46) as

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) \tag{49}$$
$$= V\Lambda V^{-1}\vec{x}(t) + B\vec{u}(t). \tag{50}$$

Now we can use the change-of-basis that was introduced when we first solved vector differential equations. Namely, let $\vec{\tilde{x}}(t) := V^{-1}\vec{x}(t)$. Then

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = V\Lambda V^{-1}\vec{x}(t) + B\vec{u}(t) \tag{51}$$
$$V^{-1}\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = V^{-1}V\Lambda V^{-1}\vec{x}(t) + V^{-1}B\vec{u}(t) \tag{52}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\left(V^{-1}\vec{x}(t)\right) = \Lambda V^{-1}\vec{x}(t) + V^{-1}B\vec{u}(t) \tag{53}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{\tilde{x}}(t) = \Lambda\vec{\tilde{x}}(t) + \widetilde{B}\vec{u}(t) \tag{54}$$

if we use the substitution $\widetilde{B} := V^{-1}B$. Thus, after changing the basis for the initial condition eq. (47) by setting $\vec{\tilde{x}}_0 = V^{-1}\vec{x}_0$, we have the vector system

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{\tilde{x}}(t) = \Lambda\vec{\tilde{x}}(t) + \widetilde{B}\vec{u}(t) \tag{55}$$
$$\vec{\tilde{x}}(0) = \vec{\tilde{x}}_0. \tag{56}$$

This is a system of differential equations where the A matrix – in this case Λ – is diagonal, so this is exactly the form of equation we solved in section 3.2! Thus, we can solve eqs. (55) and (56) to get a function $\vec{\tilde{x}}(t)$:

$$\vec{\tilde{x}}(t) = \mathrm{e}^{\Lambda t}\vec{\tilde{x}}_0 + \int_0^t \mathrm{e}^{\Lambda(t-\tau)}\widetilde{B}\vec{u}(\tau)\,\mathrm{d}\tau \tag{57}$$

To finish the proof, it remains to convert back to the usual basis of x coordinates (as opposed to the $\widetilde{x}$ coordinates which the solution is in).

$$\vec{x}(t) = V\vec{\widetilde{x}}(t) \tag{58}$$

$$= Ve^{\Lambda t}\vec{\widetilde{x}}_0 + \int_0^t Ve^{\Lambda(t-\tau)}\widetilde{B}\vec{u}(\tau)\,d\tau \tag{59}$$

$$= Ve^{\Lambda t}V^{-1}\vec{x}_0 + \int_0^t Ve^{\Lambda(t-\tau)}V^{-1}B\vec{u}(\tau)\,d\tau \tag{60}$$

as desired. □

A summary of the above proof is captured by the familiar change of coordinates diagram:

$$\begin{array}{ccc} \frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) & \xrightarrow[\text{Too difficult}]{} & \vec{x}(t) = \ldots \\ \Big\downarrow \text{Change of variables to get a diagonal system} & & \Big\uparrow \text{Undo change of variables} \\ \frac{d}{dt}\vec{\widetilde{x}}(t) = \Lambda\vec{\widetilde{x}}(t) + \widetilde{B}\vec{u}(t) & \xrightarrow[\text{Solve a diagonal system}]{} & \vec{\widetilde{x}}(t) = \ldots \end{array}$$

3.4 Beyond Diagonalizable A

Not all matrices are diagonalizable, so it is necessary to handle the case where we cannot perform the decomposition we used to get eqs. (55) and (56). Unfortunately, there is no neat closed form, such as the one we found in theorem 12.

The primary way to handle this case is to use *upper triangularization* instead of diagonalization. Upper triangularization provides a change of basis to an upper triangular matrix (instead of to a diagonal matrix), i.e., we write $A = VTV^{-1}$ where T is upper triangular. Once in the $\widetilde{x}$ basis, the differential equations we get will not be strictly independent; however, each equation will be solvable without needing any information from the equations above it. So we solve the equations bottom-to-top and back-substitute solutions to lower equations into upper equations. Once we have solved all the equations in the $\widetilde{x}$ basis, we convert back to the x basis.

The process is captured in the modification to the change-of-coordinates diagram:

$$\begin{array}{ccc} \frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + B\vec{u}(t) & \xrightarrow[\text{Too difficult}]{} & \vec{x}(t) = \ldots \\ \Big\downarrow \text{Change of variables to get an upper triangular system} & & \Big\uparrow \text{Undo change of variables} \\ \frac{d}{dt}\vec{\widetilde{x}}(t) = T\vec{\widetilde{x}}(t) + \widetilde{B}\vec{u}(t) & \xrightarrow[\text{Solve an upper triangular system}]{} & \vec{\widetilde{x}}(t) = \ldots \end{array}$$

Upper triangularization is covered later on in this course. We will show that, in fact, every square matrix is upper triangularizable, so this technique is universally applicable.

3.5 (OPTIONAL) Matrix Exponential

Section 3.4 is slightly misleading; in fact, there *is* a theoretical fully general closed form for the state trajectory of the continuous-time LTI differential equation model. However, to find it, we need to expand the notation we introduced in eq. (25). This notation e^{At} is called the *matrix exponential*.

Definition 13 (Matrix Exponential)
Let $X \in \mathbb{R}^{n \times n}$ be a *square* matrix. Then the *matrix exponential* of X is the term

$$e^X = I_n + X + \frac{X^2}{2} + \frac{X^3}{6} + \cdots = \sum_{i=0}^{\infty} \frac{X^i}{i!}. \tag{61}$$

To calculate e^{At}, we can compute eq. (61) with $X = A \cdot t$.

It is insightful to compare eq. (61) with the Maclaurin series (i.e. Taylor series centered at 0) of the usual scalar exponential function:

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!}. \tag{62}$$

This motivates why we call this expansion the matrix exponential. Anyways, here are some properties:

Lemma 14 (Properties of Matrix Exponential)

- If $X = \text{diag}(x_{11}, x_{22}, \ldots, x_{nn})$, then

$$e^X = \begin{bmatrix} e^{x_{11}} & & & \\ & e^{x_{22}} & & \\ & & \ddots & \\ & & & e^{x_{nn}} \end{bmatrix}. \tag{63}$$

  In this sense it coincides with the definition from eq. (25).

- If $X = VYV^{-1}$ is a change of basis of X, then

$$e^X = Ve^Y V^{-1}. \tag{64}$$

- The function $\vec{x}(t) := e^{At}\vec{x}_0$ is the solution to the differential equation

$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) \tag{65}$$

$$\vec{x}(0) = \vec{x}_0. \tag{66}$$

Concept Check: Prove lemma 14.

Concept Check: Show that eqs. (17), (24) and (48) are all versions of eq. (67).

With these facts in hand, we can introduce the general trajectory for model 5, the continuous-time LTI differential equation model.

Theorem 15 (State Trajectory in Continuous-Time LTI Differential Equation Model)
In the continuous-time LTI differential equation model, we have

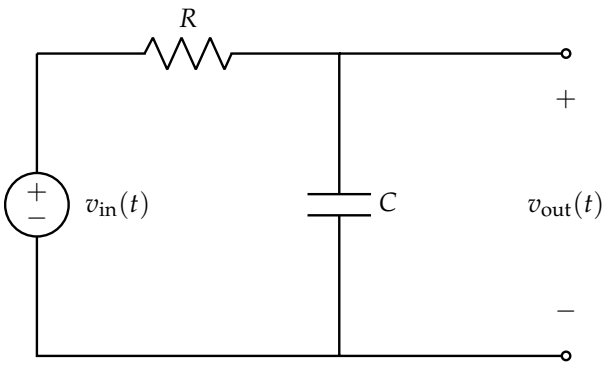
$$\vec{x}(t) = e^{At}\vec{x}_0 + \int_0^t e^{A(t-\tau)}B\vec{u}(\tau)\,d\tau\,. \tag{67}$$

4 Examples

There are many instances where using the Discrete-Time LTI Difference Equation Model and the Continuous-Time LTI Differential Equation Model is practical and we can use them to describe engineering systems. Two such examples follow, one for continuous-time and one for discrete-time.

4.1 Continuous-Time LTI Model Example

Consider the following familiar RC circuit:



Let's say we wanted to model $v_{out}(t)$. We can set up the following differential equation:

$$\frac{d}{dt}v_{out}(t) = \frac{v_{in}(t) - v_{out}(t)}{RC} \tag{68}$$

$$v_{out}(0) = V_0 \tag{69}$$

Let's say we have the ability to change $v_{in}(t)$ via a controller device. This is an explicit instance of a scalar continuous-time LTI differential equation model, which we covered in section 3.1, with the variables $A = -\frac{1}{RC}$, $B = \frac{1}{RC}$ being scalars. We can read off the solution as

$$v_{out}(t) = e^{-\frac{1}{RC}t}V_0 + \frac{1}{RC}\int_0^t e^{-\frac{1}{RC}(t-\tau)}v_{in}(\tau)\,d\tau\,. \tag{70}$$

This is interesting to study as a control model if, for instance, we want to set the trajectory of $v_{out}(t)$ to, for example, be a decaying exponential function with a desired decay rate.

4.2 Discrete-Time LTI Model Example

In the robot car project, we may model the distance traveled d and the velocity v of the left (L) and right (R) wheels at timestep i using the following state model:

$$v_L[i] = d_L[i+1] - d_L[i] = \theta_L u_L[i] - \beta_L \tag{71}$$

$$v_R[i] = d_R[i+1] - d_R[i] = \theta_R u_R[i] - \beta_R \tag{72}$$

where θ_L, θ_R and β_L, β_R are constants given by the physical system that we learn from data, and u_L, u_R are inputs (i.e. power supplied) to the left and right wheels.

From this model, we can compute $u_L[i]$ and $u_R[i]$ such that $v_L[i] = v_R[i] = v_d$ for some desired velocity v_d. In principle, this is how we compute the inputs needed for the wheels to move at the desired velocity.

Note that until now we have not involved any sophisticated control ideas, such as the Discrete-Time LTI Difference Equation Model.

The issue with this computation is that our parameters $\theta_L, \theta_R, \beta_L, \beta_R$ may be imperfectly learned from data. Thus the velocities for the two wheels may disagree, even given the inputs we calculated above. Thus the robot car would go in circles or circular arcs, not in a straight line. In fact, the robot car would go in circles unless the distance traveled by each wheel is the same at each timestep.

Thus, we will define a new variable $\delta[i] = d_L[i] - d_R[i]$, and attempt to find control inputs that send $\delta[i]$ to 0. It turns out that $\delta[i]$ obeys the Discrete-Time LTI Difference Equation Model, but the exact details are left to homework. This system will tell us how to get a car that moves straight even in the face of inacurate θ, β measurement.

5 Final Comments

There are many problems we can tackle once we have fixed a model, even after we know the state trajectory given the inputs and initial conditions. Examples of such problems are:

- Suppose we have a continuous-time model for the system, but we want to implement a digital controller. Since our controller doesn't have infinite memory and infinite computing power, it can only read to/modify the system at discrete timesteps, effectively forcing us to use a discrete-time model. So, given a continuous-time model, we convert it to an approximate discrete-time model and use that to implement the controller.

  The process of coming up with a discrete-time model that approximates a continuous-time model is called *discretization* and is covered in Note 10.

- Suppose we have a discrete-time model (respectively, a continuous-time model) for the system, but we don't know the matrices A and B. We also have data from the system; specifically, we have $(\vec{x}[i], \vec{u}[i], \vec{x}[i+1])$ for lots of values of i (respectively $(\vec{x}(t), \vec{u}(t), \frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t))$ for lots of values of t). Then we can learn the A and B matrices from data and thereby get a discrete-time model that accurately reflects the system.

  The process of learning the model parameters A and B from data is called *system identification* and is covered in Note 10.

- Suppose we have a model for the system, and want to know whether the state vector will grow unboundedly over time, and in particular which inputs will cause the state vector to grow unboundedly over time. We generally do not want this to happen in real systems.

  The process of determining, given a model, under what conditions the state trajectory grows unboundedly is called *stability analysis* and is covered in Note 11.

- Suppose we have a model for the system, and want to know whether the state can ever reach some desired state $\vec{x}_d$. If the state is, for example, the position of the robot car, then this is tantamount to asking whether the car can reach a desired position. This property is desirable in general.

  The process of determining, given a model, if there is a set of inputs that will push the state to a given desired state is called *controllability analysis* and is covered in Note 12.

- Suppose we have a model for the system, and want to find inputs that push the state to a given desired state $\vec{x}_d$ while consuming minimum energy.

  The process of determining the correct inputs to push the state towards a given desired state while consuming minimum energy is called *minimum energy control* and is covered in Note 16.

6 (OPTIONAL) More on LTI

Back when we started the note, we defined some models with the "linear time-invariant" (LTI) property. This property is really two properties which can apply to a model: *linearity* and *time-invariance*. To describe

the definitions of these properties, we will introduce a new notation which makes our description easier.

More particularly, recall that we said a model is a rule which takes in an initial state $\vec{x}_0$ and a control input $\vec{u}$ (a function of time), and returns a state function $\vec{x}$ (again a function of time). We will say in this case that $(\vec{x}_0, \vec{u}) \to \vec{x}$ is an input-output pair of the model.

Now we are ready to give our definitions.

Definition 16 (Linearity)
A model is *linear* if for every $c_1, c_2 \in \mathbb{R}$ and any input-output pairs

$$(\vec{x}_{0;1}, \vec{u}_1) \to \vec{x}_1 \qquad \text{and} \qquad (\vec{x}_{0;2}, \vec{u}_2) \to \vec{x}_2 \tag{73}$$

then the model also has the input-output pair

$$(c_1\vec{x}_{0;1} + c_2\vec{x}_{0;2}, c_1\vec{u}_1 + c_2\vec{u}_2) \to c_1\vec{x}_1 + c_2\vec{x}_2. \tag{74}$$

This is the same definition we use for a linear transformation, or a matrix; it is just abstracted to deal with the inputs and outputs being *functions of time*. Essentially this means that the model's output is linear in its inputs.

Definition 17 (Time-Invariance)
A model is *time-invariant* if for every non-negative time $\tau \geq 0$, and every input-output pair $(\vec{x}_0, \vec{u}) \to \vec{x}$, if we define the notation

$$\vec{\tilde{x}}_0 := \vec{x}(\tau) \qquad \vec{\tilde{x}}(t) := \vec{x}(t+\tau) \qquad \vec{\tilde{u}}(t) := \vec{u}(t+\tau) \tag{75}$$

then the model has the input-output pair $(\vec{\tilde{x}}_0, \vec{\tilde{u}}) \to \vec{\tilde{x}}$. Essentially this means that the model doesn't change its behavior as time goes on.

Note that here we used the continuous-time notation, but this also applies for the discrete-time model. In discrete-time, $\tau \in \mathbb{N}$; in continuous-time, $\tau \in \mathbb{R}_+$.

Theorem 18
The Discrete-Time LTI Difference Equation Model and the Continuous-Time LTI Differential Equation Model are LTI.

Concept Check: Prove theorem 18 by using the solutions to the difference/differential equations (eqs. (5) and (67)) to write the state function explicitly in terms of the initial condition and control input, and showing that it is linear.

Intuitively, the Discrete-Time LTI Difference Equation Model and the Continuous-Time LTI Differential Equation Model are linear because the right-hand side of the equations are linear in the current $\vec{x}$ value and the current $\vec{u}$ value, and they are time-invariant because the coefficients A and B don't depend on time.

For example, an example of a discrete-time linear difference equation model which is *not* time invariant (and in particular called time-varying) is:

$$\vec{x}[i+1] = i^2 \cdot \vec{x}[i] + i \cdot \vec{u}[i] \tag{76}$$
$$\vec{x}[0] = \vec{x}_0. \tag{77}$$

And an example of a continuous-time time-invariant differential equation model which is *not* linear is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = \|\vec{x}(t)\|^2\vec{x}(t) + B\vec{u}(t) \tag{78}$$
$$\vec{x}(0) = \vec{x}_0. \tag{79}$$

This is nonlinear because the first term

$$\|\vec{x}(t)\|^2\vec{x}(t) = \left(\sum_{i=1}^{n} x_i(t)^2\right)\vec{x}(t) \tag{80}$$

is at least quadratic in the $x_i(t)$ (and in particular each coordinate is cubic).

Dealing with time-varying models is out of scope of this class. The one caveat is that we should know qualitatively how to learn parameters that may change over time. The way to do this is to collect more data over time, and use only recent data to learn the current model parameters via system identification.

Dealing with nonlinear models is out of scope for this note, though we will learn in Note 20 how to locally approximate a nonlinear model by a linear model through *linearization*. Once we do this approximation we can use the linear model for designing our inputs for a short time, until the state leaves the region in state space where the approximation is valid, at which point we will have to re-linearize and repeat. We can also do nonlinear system identification using machine learning techniques, discussed in Note 21.

7 (OPTIONAL) More on Model-Based Control

We have discussed linear time-invariant models in discrete-time and continuous-time in the main part of the note, and in section 6 we have discussed ways to produce nonlinear and time-varying models. By mixing and matching these categories, we can generate several types of models we won't be using in this course, but are nonetheless practical and useful to study.

As a summary, here is a diagram of models that are studied in model-based control:

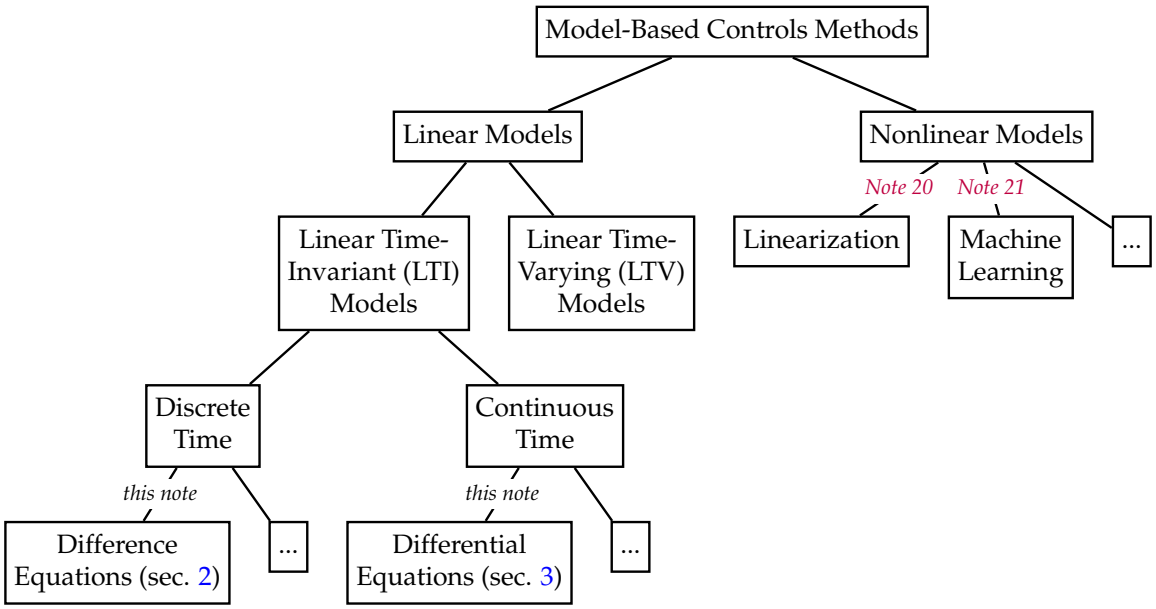


Figure 1: Classification of model-based control techniques. Taken from EE 221A Fall 2020 Lecture Note 1.

Contributors:

- Druv Pai.
- Anish Muthali.