Digital Control: Fundamentals

ENGI 7825: Control Systems II

Andrew Vardy

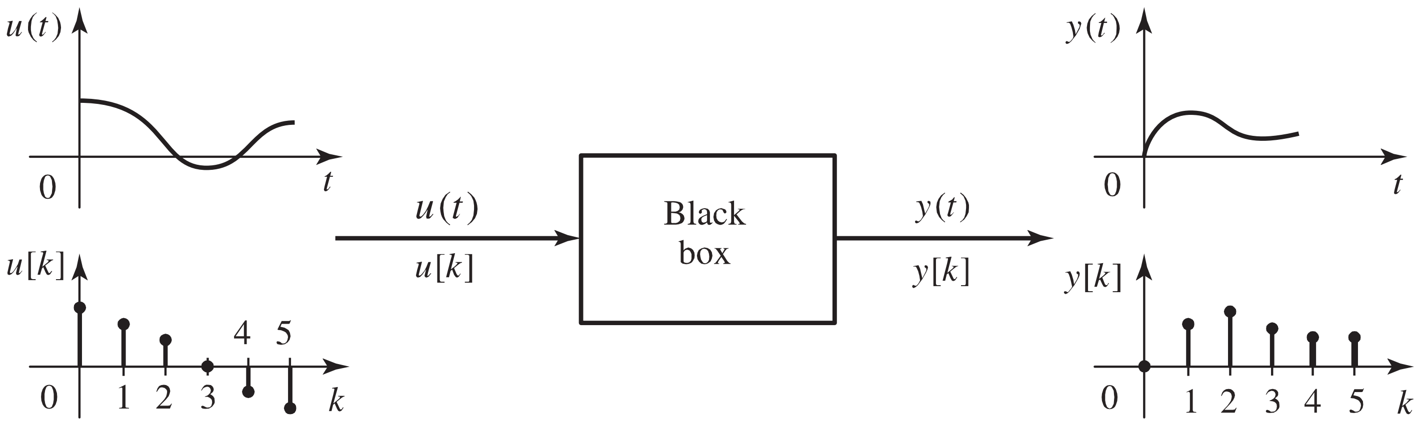
Introduction

- So far we have considered only continuous-time (CT) systems. However, computers are very often incorporated into modern control systems. Computers are discrete-time (DT) components.
- Computers have the following advantageous characteristics for control systems:
  - They continuously grow both faster and cheaper; Microcontrollers (complete computer on one IC) often cost < $5
  - Fully customizable through software
  - Operation is static and largely invariant to environmental conditions
- We can contrast digital computers with analog electronic components (e.g. resistors, inductors, capacitors, op-amps)
  - Not easily customizable once installed and may deviate from specs
  - Affected by variations in temperature
  - Produce analog (i.e. CT) signals which are quite susceptible to noise

Notation

- The material to come on Digital Control comes from different sources:
  - "Feedback Control of Dynamic Systems", 6th Edition, by Franklin, Powell, and Emami-Naeini (Sections 8.1, 8.2)
  - "Feedback Control Systems", 5th Edition, by Phillips and Parr (Sections 11.6, 11.7, 11.8, 12.9, 13.4, 14.2)
  - "Control Systems Engineering", 5th Edition, by Nise (z-transform tables)
  - "Linear System Theory and Design", 4th Edition by Chen (Section 4.2)
- The notation will consequently vary:
  - Quantities with unqualified references to time are continuous-time: e.g. $u(t)$
  - Quantities which refer to integer multiples of the sampling period, T, are discrete samples: e.g. $u(kT)$
    - Often the T is dropped, leaving an index k: $u(k)$
    - Sometimes square brackets are used for discrete-time signals: $u[k]$

Discrete-Time Systems



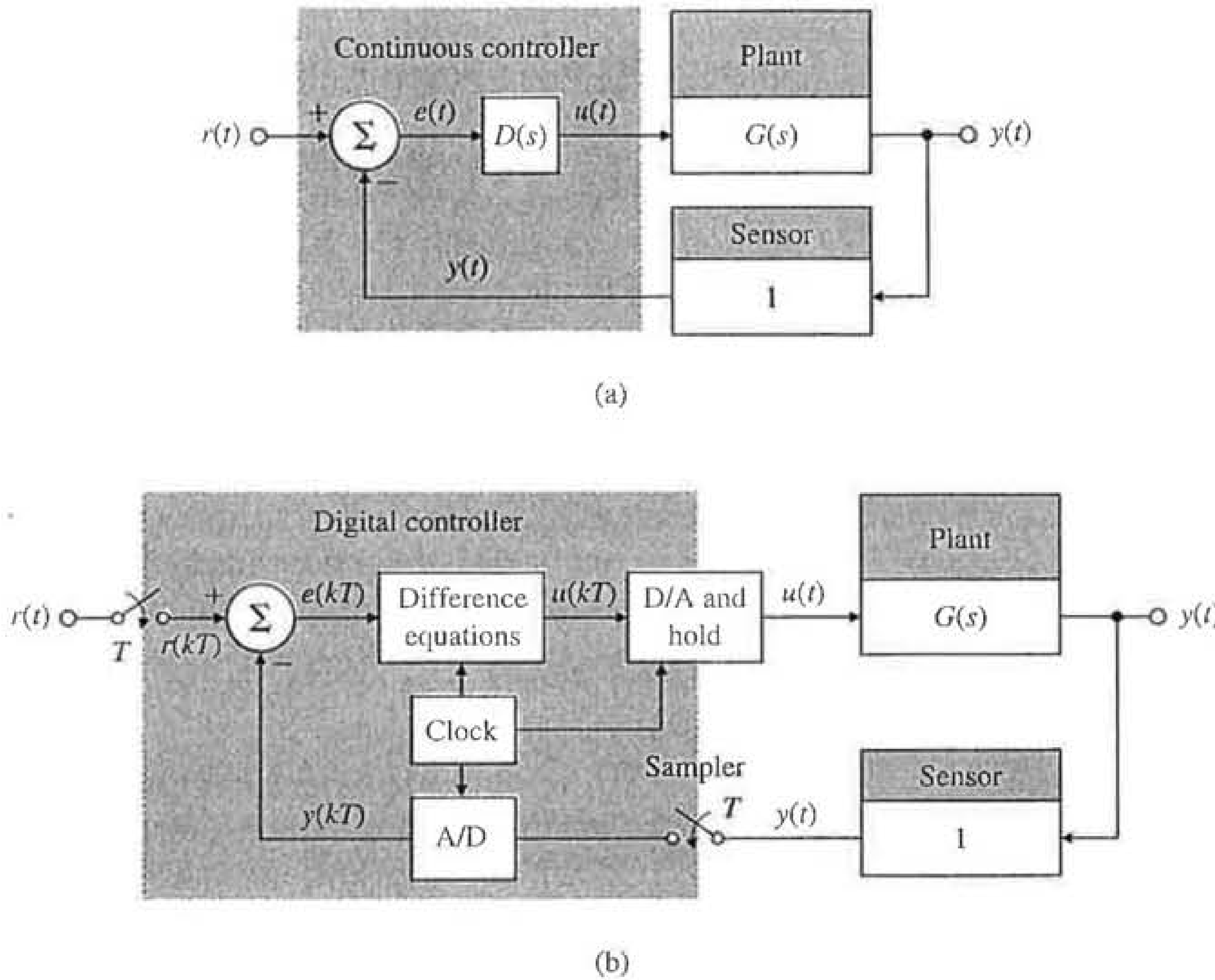
Discrete-time signals are sequences of numbers (e.g. u[k] and y[k]) above. They may be sampled from CT signals (as above) or they may be produced by inherently discrete processes.

Sampled Data Systems

- A system with both CT and DT components is often referred to as a **sampled data system**
- In practice, most control systems are sampled data systems
- An analog-to-digital (A/D) converter <u>samples</u> a physical variable and translates it into a digital number
  - We usually assume this occurs at a fixed sampling period, T
- In most sampled data systems, a digital controller replaces a continuous controller as shown...

Figure 8.1

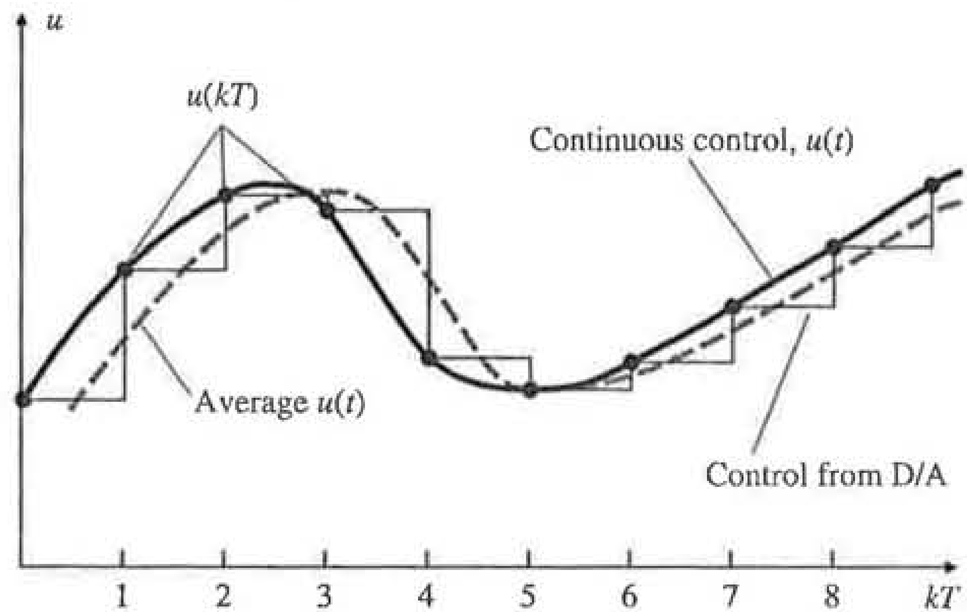
Block diagrams for a basic control system: (a) continuous system; (b) with a digital computer



- If the input is CT then r(t) must be sampled as shown. However, the input will often be DT in which case no sampling is required on the input.

- The D/A converter converts the controller output to an analog signal which is held until the next period of duration T. This is known as zero-order hold (ZOH). ZOH incurs an average delay of T/2.

Figure 8.2
The delay due to the hold operation



- This delay is one reason we generally wish to increase the frequency of the digital controller's operation.

Fundamental Differences

- CT systems are governed by differential equations
- For DT systems the notion of a derivative is not so well defined; DT systems are governed by **difference equations**
- The following is a general 2^{nd} order difference equation:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_0 u(k) + b_1 u(k-1) + b_2 u(k-2)$$

- The “new value” at y(k) is obtained from the past values of y(k-1) and y(k-2) as well as current and past values of u.

Motivation for the Z-Transform

- Why did we ever start using the Laplace Transform (LT) for control systems?
  - The LT simplifies the treatment of derivatives, allowing differential equations to be easily solved and allowing transfer functions to be found
- Recall the definition of the LT and the crucial identity that establishes the LT of a derivative under zero initial conditions:

$$\mathcal{L}\{f(t)\} = F(s) = \int_0^{\infty} f(t)e^{-st}\, dt$$

Definition of LT

$$\mathcal{L}\{\dot{f}(t)\} = sF(s)$$

Derivative property of LT

Z-Transform

- The z-transform is the counterpart of the Laplace transform for DT systems. It is defined as follows:

$$\mathcal{Z}\{f(k)\} = F(z) = \sum_{k=0}^{\infty} f(k) z^{-k}$$

- Note that k is an index referring to discrete sample times. Like s, z is a complex variable.
- Analogous to the derivative property of the LT we have:

$$\mathcal{Z}\{f(k-1)\} = z^{-1} F(z)$$

- This allows us to turn difference equations into transfer functions in the same way as the derivative property in the Laplace domain.

The Transfer Function of a DT System

- We can find the transfer function of a DT system by using the z-transform. Consider again the general second-order difference equation:

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) + b_0 u(k) + b_1 u(k-1) + b_2 u(k-2)$$

- Now apply $\mathcal{Z}\{f(k-1)\} = z^{-1}F(z)$

$$Y(z) = (-a_1 z^{-1} - a_2 z^{-2})Y(z) + (b_0 + b_1 z^{-1} + b_2 z^{-2})U(z)$$

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

- Similar to LT we seldom use the raw definition of the z-transform but make use of tables instead...

TABLE 13.1 Partial table of z- and s-transforms

	$f(t)$	$F(s)$	$F(z)$	$f(kT)$
1.	$u(t)$	$\frac{1}{s}$	$\frac{z}{z-1}$	$u(KT)$
2.	t	$\frac{1}{s^2}$	$\frac{Tz}{(z-1)^2}$	kT
3.	t^n	$\frac{n!}{s^{n+1}}$	$\lim_{a\to 0}(-1)^n \frac{d^n}{da^n}\left[\frac{z}{z-e^{-aT}}\right]$	$(kT)^n$
4.	e^{-at}	$\frac{1}{s+a}$	$\frac{z}{z-e^{-aT}}$	e^{-akT}
5.	$t^n e^{-at}$	$\frac{n!}{(s+a)^{n+1}}$	$(-1)^n \frac{d^n}{da^n}\left[\frac{z}{z-e^{-aT}}\right]$	$(kT)^n e^{-akT}$
6.	$\sin \omega t$	$\frac{\omega}{s^2+\omega^2}$	$\frac{z \sin \omega T}{z^2 - 2z\cos \omega T + 1}$	$\sin \omega kT$
7.	$\cos \omega t$	$\frac{s}{s^2+\omega^2}$	$\frac{z(z-\cos \omega T)}{z^2 - 2z\cos \omega T + 1}$	$\cos \omega kT$
8.	$e^{-at}\sin \omega t$	$\frac{\omega}{(s+a)^2+\omega^2}$	$\frac{ze^{-aT}\sin \omega T}{z^2 - 2ze^{-aT}\cos \omega T + e^{-2aT}}$	$e^{-akT}\sin \omega kT$
9.	$e^{-at}\cos \omega t$	$\frac{s+a}{(s+a)^2+\omega^2}$	$\frac{z^2 - ze^{-aT}\cos \omega T}{z^2 - 2ze^{-aT}\cos \omega T + e^{-2aT}}$	$e^{-akT}\cos \omega kT$

TABLE 13.2 *z*-transform theorems

	Theorem	Name
1.	$z\{af(t)\} = aF(z)$	Linearity theorem
2.	$z\{f_1(t) + f_2(t)\} = F_1(z) + F_2(z)$	Linearity theorem
3.	$z\{e^{-aT}f(t)\} = F(e^{aT}z)$	Complex differentiation
4	$z\{f(t-nT)\} = z^{-n}F(z)$	Real translation
5.	$z\{tf(t)\} = -Tz\dfrac{dF(z)}{dz}$	Complex differentiation
6.	$f(0) = \lim_{z\to\infty} F(z)$	Initial value theorem
7.	$f(\infty) = \lim_{z\to 1}(1-z^{-1})F(z)$	Final value theorem

Note: kT may be substituted for t in the table.

This is perhaps the most important theorem for our purposes and will more typically be written like this:

$$Z\{f(k-n)\} = z^{-n}F(z)$$

Note that this property is defined for positive n. If n is negative (i.e. positive time shift) then we have the following:

$$Z\{f(k+n)\} = z^{n}\left\{F(z) - \sum_{i=0}^{n-1} f(i)z^{-i}\right\}$$

We will soon need to apply this theorem for n = 1:

$$Z\{f(k+1)\} = z(F(z) - f(0))$$

Inverse z-Transform

In a somewhat similar fashion to the ILT, we can obtain the inverse z-Transform using partial fraction expansion. In the s-domain the terms corresponding to exponentials were of the form,

$$\frac{1}{s+a} \Longleftrightarrow e^{-at}$$

In the z-domain we have,

$$\frac{z}{z-e^{-aT}} \Longleftrightarrow e^{-akT}$$

Therefore we will try to reduce z-domain expressions into the following form,

$$F(z) = \frac{Az}{z-z_1} + \frac{Bz}{z-z_2} + \cdots$$

We require the z in the numerator. This can be achieved by expanding $F(z)/z$ instead of $F(z)$, then multiplying by z...

e.g. Find the sampled time function corresponding to $F(z)$.

$$F(z) = \frac{0.5z}{(z-0.5)(z-0.7)}$$

We expand $F(z)/z$,

$$\frac{F(z)}{z} = \frac{0.5}{(z-0.5)(z-0.7)} = \frac{A}{z-0.5} + \frac{B}{z-0.7} = \frac{-2.5}{z-0.5} + \frac{2.5}{z-0.7}$$

Therefore,

$$F(z) = \frac{-2.5z}{z-0.5} + \frac{2.5z}{z-0.7}$$

Apply the inverse z-Transform on each term,

$$f(kT) = -2.5(0.5)^k + 2.5(0.7)^k$$

Notice that do not get $f(t)$. We only get back its sampled values.

Discrete-Time State Space

- The discrete-time (DT) state space representation is of the following form:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$$

- This is quite similar to the continuous-time (CT) representation except that we have a shift in time as opposed to a derivative
- In CT we needed to handle higher-order derivatives which was achieved by defining a stack of state variables
- In DT we can apply the same idea to handle larger time shifts.

- Consider the following common form of difference equation:

$$y(k+n) + a_1 y(k+n-1) + a_2 y(k+n-2) + \ldots + a_{n-1} y(k+1) + a_n y(k) \quad = \quad bu(k)$$

- We can assign the "base" variable y(k) as our first state variable and create further state variables to represent all subsequent time shifts:

$$\begin{aligned}
x_1(k) &= y(k) \\
x_1(k+1) &= x_2(k) \\
x_2(k+1) &= x_3(k) \\
\vdots \quad & \quad \vdots \\
x_{n-1}(k+1) &= x_n(k) \\
x_n(k+1) &= -a_1 x_n(k) - a_2 x_{n-1}(k) - \ldots - a_n x_1(k) + bu(k)
\end{aligned}$$

$$
\begin{aligned}
x_1(k) &= y(k) \\
x_1(k+1) &= x_2(k) \\
x_2(k+1) &= x_3(k) \\
\vdots \quad &\quad \vdots \\
x_{n-1}(k+1) &= x_n(k) \\
x_n(k+1) &= -a_1 x_n(k) - a_2 x_{n-1}(k) - \ldots - a_n x_1(k) + bu(k)
\end{aligned}
$$

- The final state space representation easily follows:

$$
\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \vdots \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ -a_n & -a_{n-1} & -a_{n-2} & & -a_1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ b \end{bmatrix} u(k)
$$

$$
y(k) = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix}
$$

- Notice that this is in Controller Canonical Form (CCF)

Solution of the DT State Space Equations

- Consider just the DT state space difference equation:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$

- We will assume that x(0) and the input u(k) are known. We can solve for k = 0, 1, 2, ...

$$\mathbf{x}(1) = \mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u}(0)$$

$$\begin{aligned}\mathbf{x}(2) &= \mathbf{A}\mathbf{x}(1) + \mathbf{B}\mathbf{u}(1) = \mathbf{A}[\mathbf{A}\mathbf{x}(0) + \mathbf{B}\mathbf{u}(0)] + \mathbf{B}\mathbf{u}(1)\\ &= \mathbf{A}^2\mathbf{x}(0) + \mathbf{A}\mathbf{B}\mathbf{u}(0) + \mathbf{B}\mathbf{u}(1)\end{aligned}$$

$$\begin{aligned}\mathbf{x}(3) &= \mathbf{A}\mathbf{x}(2) + \mathbf{B}\mathbf{u}(2) = \mathbf{A}[\mathbf{A}^2\mathbf{x}(0) + \mathbf{A}\mathbf{B}\mathbf{u}(0) + \mathbf{B}\mathbf{u}(1)] + \mathbf{B}\mathbf{u}(2)\\ &= \mathbf{A}^3\mathbf{x}(0) + \mathbf{A}^2\mathbf{B}\mathbf{u}(0) + \mathbf{A}\mathbf{B}\mathbf{u}(1) + \mathbf{B}\mathbf{u}(2)\\ &\vdots\end{aligned}$$

$$\mathbf{x}(n) = \mathbf{A}^n\mathbf{x}(0) + \mathbf{A}^{n-1}\mathbf{B}\mathbf{u}(0) + \mathbf{A}^{n-2}\mathbf{B}\mathbf{u}(1) + \cdots + \mathbf{A}\mathbf{B}\mathbf{u}(n-2) + \mathbf{B}\mathbf{u}(n-1)$$

$$\mathbf{x}(n) = \mathbf{A}^n\mathbf{x}(0) + \sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\mathbf{u}(k)$$

$$\mathbf{x}(n) = \mathbf{A}^n\mathbf{x}(0) + \sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\mathbf{u}(k)$$

- In the CT solution to the state-space equation we found something similar, only more exotic because it involved the matrix exponential. Our approach then was to try using Laplace to obtain an easier-to-compute solution. Here we do the same, except we use the z-transform instead.
- First we restate the state-space difference equation:

$$\mathbf{x}(k + 1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$$

- We rewrite this as individual row equations:

$$\begin{aligned} x_1(k+1) &= a_{11}x_1(k) + \cdots + a_{1n}x_n(k) + b_{11}u_1(k) + \cdots + b_{1r}u_r(k) \\ &\vdots \\ x_n(k+1) &= a_{n1}x_1(k) + \cdots + a_{nn}x_n(k) + b_{n1}u_1(k) + \cdots + b_{nr}u_r(k) \end{aligned}$$

- Now take the z-transform…

$$x_1(k+1) = a_{11}x_1(k) + \cdots + a_{1n}x_n(k) + b_{11}u_1(k) + \cdots + b_{1r}u_r(k)$$

$$\vdots$$

$$x_n(k+1) = a_{n1}x_1(k) + \cdots + a_{nn}x_n(k) + b_{n1}u_1(k) + \cdots + b_{nr}u_r(k)$$

↓ z-transform

$$z[X_1(z) - x_1(0)] = a_{11}X_1(z) + \cdots + a_{1n}X_n(z) + b_{11}U_1(z) + \cdots + b_{1r}U_r(z)$$

$$\vdots$$

$$z[X_n(z) - x_n(0)] = a_{n1}X_1(z) + \cdots + a_{nn}X_n(z) + b_{n1}U_1(z) + \cdots + b_{nr}U_r(z)$$

Express as a vector-matrix equation and solve for **X**(z):

$$z[\mathbf{X}(z) - x(0)] = \mathbf{A}\mathbf{X}(z) + \mathbf{B}\mathbf{U}(z)$$

$$[z\mathbf{I} - \mathbf{A}]\mathbf{X}(z) = z\mathbf{x}(0) + \mathbf{B}\mathbf{U}(z)$$

$$\mathbf{X}(z) = z[z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{x}(0) + [z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B}\mathbf{U}(z)$$

$$\mathbf{x}(n) = \mathbf{A}^n\mathbf{x}(0) + \sum_{k=0}^{n-1}\mathbf{A}^{n-1-k}\mathbf{B}\mathbf{u}(k)$$

$$\mathbf{X}(z) = z[z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{x}(0) + [z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B}\mathbf{U}(z)$$

We can equate the time-domain solution (above left) and the z-transform solution (above right) and denote $\mathbf{\Phi}$(k) = A^k. Note that $\mathbf{\Phi}$(k) is the state transition matrix for our discrete-time solution, <u>but it is not the matrix exponential</u>. Now we have an alternative way of computing $\mathbf{\Phi}$(k):

$$\Phi(k) = \mathfrak{z}^{-1}(z[z\mathbf{I} - \mathbf{A}]^{-1}) = \mathbf{A}^k$$

$$\mathbf{x}(n) = \Phi(n)\mathbf{x}(0) + \sum_{k=0}^{n-1}\Phi(n - 1 - k)\mathbf{B}\mathbf{u}(k)$$

Notice how closely this followed the derivation of the state-space solution in CT

Example

- Consider the DT system with the following transfer function:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{z}{z^2 - 3z + 2} = \frac{z}{(z-1)(z-2)}$$

- We can determine the difference equation for this system and then obtain the state-space equation

COVERED ON BOARD

$$\mathbf{x}(k+1) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}(k)$$

$$\mathbf{x}(k+1) = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix}\mathbf{x}(k) + \begin{bmatrix} 0 \\ 1 \end{bmatrix}u(k)$$

$$y(k) = [0 \quad 1]\mathbf{x}(k)$$

$$\Phi(k) = \mathcal{Z}^{-1}(z[z\mathbf{I} - \mathbf{A}]^{-1}) = \mathbf{A}^k$$

- Now that we have the state-space representation, we can solve for Y(z) and then y(k):

$$[z\mathbf{I} - \mathbf{A}] = \begin{bmatrix} z & -1 \\ 2 & z-3 \end{bmatrix} \qquad |z\mathbf{I} - \mathbf{A}| = z^2 - 3z + 2$$

$$[z\mathbf{I} - \mathbf{A}]^{-1} = \frac{1}{z^2 - 3z + 2}\begin{bmatrix} z-3 & 1 \\ -2 & z \end{bmatrix}$$

- Recall our general solution: $\mathbf{X}(z) = z[z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{x}(0) + [z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B}\mathbf{U}(z)$
  In this example we have x(0) = 0 (after all we started with a transfer function which implicitly assumes x(0) = 0).

$$\begin{aligned} \mathbf{X}(z) &= [z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B}U(z) \\ &= \frac{1}{z^2 - 3z + 2}\begin{bmatrix} z-3 & 1 \\ -2 & z \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}U(z) \\ &= \frac{1}{z^2 - 3z + 2}\begin{bmatrix} 1 \\ z \end{bmatrix}U(z) \end{aligned}$$

$$
\begin{aligned}
\mathbf{X}(z) &= [z\mathbf{I} - \mathbf{A}]^{-1}\mathbf{B}U(z) \\
&= \frac{1}{z^2 - 3z + 2}\begin{bmatrix} z-3 & 1 \\ -2 & z \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix}U(z) \\
&= \frac{1}{z^2 - 3z + 2}\begin{bmatrix} 1 \\ z \end{bmatrix}U(z)
\end{aligned}
$$

- Lets assume we have a step input. The z-transform of the step function is z/(z-1). We fill this into the above solution and into Y(z) = C X(z):

$$
Y(z) = \mathbf{C}\mathbf{X}(z) = [0 \quad 1]\begin{bmatrix} \dfrac{z}{(z-1)(z^2-3z+2)} \\ \dfrac{z^2}{(z-1)(z^2-3z+2)} \end{bmatrix}
$$

- Applying partial fraction expansion and then the inverse z-transform:

$$
Y(z) = \frac{z^2}{(z-1)^2(z-2)} = \frac{-z}{(z-1)^2} + \frac{-2z}{z-1} + \frac{2z}{z-2}
$$

$$
y(k) = -k - 2 + 2(2)^k
$$

- The output is the sequence 0, 1, 4, 11, 26,...

Solution for Sampled Data Systems

- The solution just presented works for purely DT systems, but what if the plant is CT? There are a couple of possibilities (we take the **bolded** path):
  - Design controller in CT then translate to DT
    - Using approximate mappings from s-domain to z-domain (e.g. Tustin's method, MPZ)
  - **Translate CT plant model to DT, then design controller in DT**
    - Use approximate mappings from from s- to z- (as above)
    - Or...
    - **Use the exact mapping for state-space representations we are about to discuss**

- *[We are following the solution presented in section 4.2 of "Linear System Theory and Design" by Chen]*
- Assume we begin with a purely CT state-space representation and its solution, developed earlier this term:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$

$$\mathbf{x}(t) = e^{\mathbf{A}t}\mathbf{x}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}\mathbf{u}(\tau)\,d\tau$$

- The input u(t) will be produced by a computer and will be held constant throughout each sample period:

$$\mathbf{u}(t) = \mathbf{u}(kT) =: \mathbf{u}[k] \qquad \text{for } kT \le t < (k+1)T$$

Zero-Order Hold

- We evaluate our CT solution at discrete time steps t = kT and t = (k+1) T

$$\mathbf{x}[k] := \mathbf{x}(kT) = e^{\mathbf{A}kT}\mathbf{x}(0) + \int_0^{kT} e^{\mathbf{A}(kT-\tau)}\mathbf{B}\mathbf{u}(\tau)\,d\tau$$

$$\mathbf{x}[k+1] := \mathbf{x}((k+1)T) = e^{\mathbf{A}(k+1)T}\mathbf{x}(0) + \int_0^{(k+1)T} e^{\mathbf{A}((k+1)T-\tau)}\mathbf{B}\mathbf{u}(\tau)\,d\tau$$

$$\mathbf{x}[k] := \mathbf{x}(kT) = e^{\mathbf{A}kT}\mathbf{x}(0) + \int_0^{kT} e^{\mathbf{A}(kT-\tau)}\mathbf{B}\mathbf{u}(\tau)\,d\tau$$

$$\mathbf{x}[k+1] := \mathbf{x}((k+1)T) = e^{\mathbf{A}(k+1)T}\mathbf{x}(0) + \int_0^{(k+1)T} e^{\mathbf{A}((k+1)T-\tau)}\mathbf{B}\mathbf{u}(\tau)\,d\tau$$

- We can re-write the second equation as follows and then recognize that it contains the first:

$$\mathbf{x}[k+1] = e^{\mathbf{A}T}\left[e^{\mathbf{A}kT}\mathbf{x}(0) + \int_0^{kT} e^{\mathbf{A}(kT-\tau)}\mathbf{B}\mathbf{u}(\tau)d\tau\right] + \int_{kT}^{(k+1)T} e^{\mathbf{A}(kT+T-\tau)}\mathbf{B}\mathbf{u}(\tau)\,d\tau$$

$$\mathbf{x}[k+1] = e^{\mathbf{A}T}\mathbf{x}[k] + \left(\int_0^{T} e^{\mathbf{A}\alpha}\,d\alpha\right)\mathbf{B}\mathbf{u}[k]$$

- where $\alpha = kT + T - \tau$. We have also substituted in our DT u[k] which is constant within the integrated interval and can therefore be factored out.

$$\mathbf{x}[k+1] = e^{\mathbf{A}T}\mathbf{x}[k] + \left(\int_0^T e^{\mathbf{A}\alpha}\,d\alpha\right)\mathbf{B}\mathbf{u}[k]$$

- This is now a purely DT representation. We can establish a correspondence between the given CT system (A, B, C, D) and its DT equivalent (A_d, B_d, C_d, D_d):

$$\mathbf{x}[k+1] = \mathbf{A}_d\mathbf{x}[k] + \mathbf{B}_d\mathbf{u}[k]$$
$$\mathbf{y}[k] = \mathbf{C}_d\mathbf{x}[k] + \mathbf{D}_d\mathbf{u}[k]$$

$$\mathbf{A}_d = e^{\mathbf{A}T} \qquad \mathbf{B}_d = \left(\int_0^T e^{\mathbf{A}\tau}\,d\tau\right)\mathbf{B} \qquad \mathbf{C}_d = \mathbf{C} \qquad \mathbf{D}_d = \mathbf{D}$$

- The only practical issue is in computing B_d. A few short manipulations (see Chen for details) lead to the following:

$$\mathbf{B}_d = \mathbf{A}^{-1}(\mathbf{A}_d - \mathbf{I})\mathbf{B} \qquad \text{(if } \mathbf{A} \text{ is nonsingular)}$$

Example

- Assume we start with the following transfer function (appropriate form for a servomotor):

$$G(s) = \frac{10}{s^2 + s}$$

- We can obtain the CT state space model quite directly

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0 \\ 10 \end{bmatrix} u(t)$$

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}(t)$$

- Simply apply our derived DT equivalents. We'll say that T = 0.1 s

$$\mathbf{A}_d = e^{\mathbf{A}T} \qquad \mathbf{B}_d = \mathbf{A}^{-1}(\mathbf{A}_d - \mathbf{I})\mathbf{B} \qquad \mathbf{C}_d = \mathbf{C} \qquad \mathbf{D}_d = \mathbf{D}$$

- Unfortunately, A is singular, so this doesn't work! Phillips and Parr present another method which does work, but we'll just use Matlab.

Conversion from CT to DT using Matlab

- The function c2d converts from CT to DT state-space representations. In fact you have been using this all along, since Matlab is inherently DT (it runs on a computer).

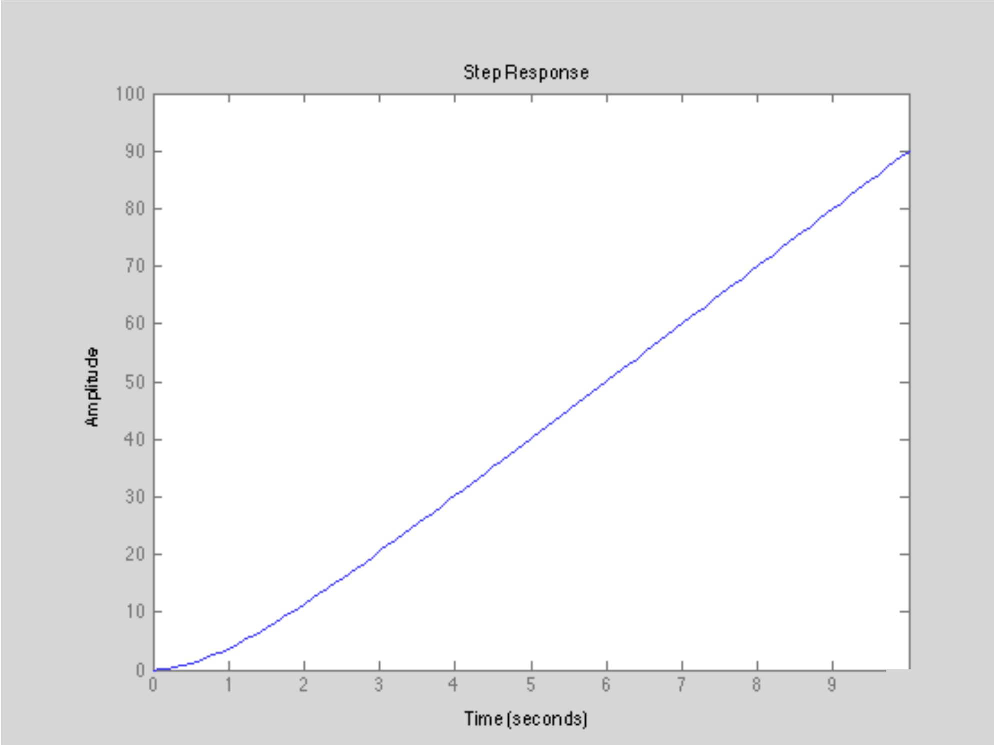
```
c2d  Converts continuous-time dynamic system to discrete time.

   SYSD = c2d(SYSC,TS,METHOD) computes a discrete-time model SYSD with
   sampling time TS that approximates the continuous-time model SYSC.
   The string METHOD selects the discretization method among the following:
      'zoh'       Zero-order hold on the inputs
      'foh'       Linear interpolation of inputs
      'impulse'   Impulse-invariant discretization
      'tustin'    Bilinear (Tustin) approximation.
      'matched'   Matched pole-zero method (for SISO systems only).
   The default is 'zoh' when METHOD is omitted. The sampling time TS should
   be specified in the time units of SYSC (see "TimeUnit" property).
```

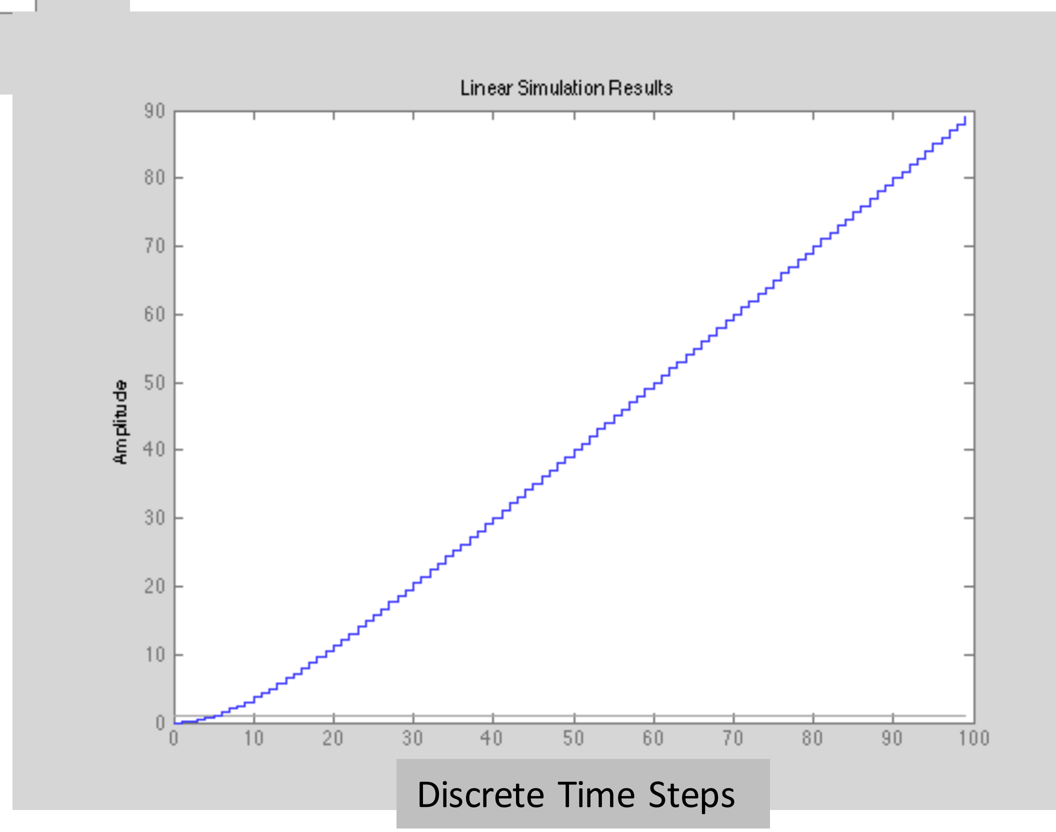
- For the example we execute: [Ad, Bd] = c2d(A, B, 0.1)

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & 0.0952 \\ 0 & 0.905 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} 0.0484 \\ 0.952 \end{bmatrix} m(k)$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}(k)$$



This is the step response of the original system. Note that the system is a DC motor so as we continue to apply a step input it is quite reasonable for the output (motor shaft angle) to increase continually.



This is the step response of the converted DT system. Note that T = 0.1 s, so the responses do match.