Signals and Systems

Lecture 1: From Continuous Time to Discrete Time

Dr. Guillaume Ducard

Fall 2018

based on materials from: Prof. Dr. Raffaello D'Andrea

Institute for Dynamic Systems and Control

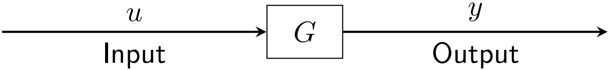
ETH Zurich, Switzerland

Outline

Outline

What is a system?

A *system* G operates on the input signal u to produce the output signal y, an operation denoted as $y = Gu$ and illustrated as:



Note that, depending on G, the output y could depend on past, current, and future values of the input u.

Definitions: (Dynamic) Systems

Dynamic Systems

systems that are not static, i.e., their state evolves w.r.t. time, due to:

- input signals,
- external perturbations,
- or naturally.

For example, a dynamic system is a system which changes:

- its trajectory $\rightarrow$ changes in acceleration, orientation, velocity, position.
- its temperature, pressure, volume, mass, etc.
- its current, voltage, frequency, etc.

Outline

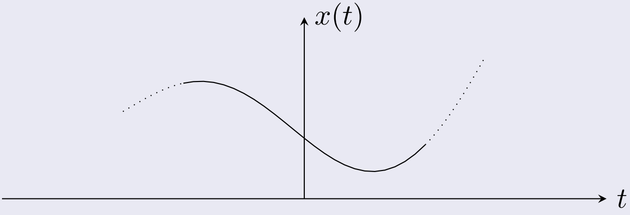
What is a signal?

A *signal* is a function of time that represents a physical quantity such as a force, position, or voltage.

Continuous-time (CT) signals

$x(t)$: t is continuous, $t \in \mathbb{R}$ (the set of real numbers)

$x(t)$ takes on continuous values analog signal

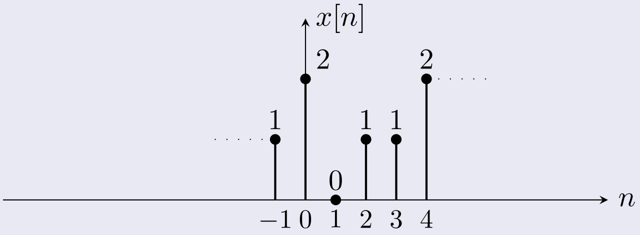


Discrete-time (DT) signals

$x[n]$: n is discrete, an integer,

$n \in \mathbb{Z}$ (the set of integer numbers, $\mathbb{Z} = \{\ldots, -1, 0, 1, \ldots\}$)

$x[n]$ can take continuous or discrete values



In this class, we focus on DT signals:

- CT signals were treated extensively in Control Systems 1 & 2.
- In DT the math is simpler, which allows us to focus on concepts.
  1. Integrals are replaced by sums,
  2. differentiation is replaced by finite differences,
  3. and the Dirac delta function is replaced by the unit impulse.
- Algorithms are implemented in discrete time on computers.

We focus on discrete-time signals with the signal $x[n]$ taking on continuous values:

- Assuming continuous values simplifies the math.
- This is a good approximation: On modern computers the double data type stores numbers using 8 bytes ($= 64$ bits) and hence allows for $2^{64} \approx 10^{20}$ different values.

- $x[n]$ can be a vector, however we mainly deal with scalar signals in this class.
- $x[n]$ can be a complex number:

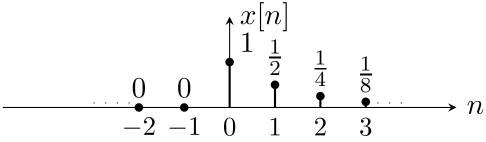
$$x[n] = x_1[n] + jx_2[n], \quad x_1[n], x_2[n] \in \mathbb{R}, \quad j^2 = -1$$

A word on notation

- A sequence is a DT signal. The terms are used interchangeably.
- x denotes a signal. It is a function of time.
- $x[n]$ indicates the value of x at the discrete time n.
- $x(t)$ indicates the value of x at the continuous time t.
- $\{x[n]\}$ refers to the entire sequence. It is the same as x for DT signals.
- Motivated by the above, $\{x(t)\}$ refers to the entire CT signal. It is the same as x for CT signals.

Three signal representations of DT signals

- Graph:



- Rule:

$$x[n] := \begin{cases} \left(\frac{1}{2}\right)^n & n \geq 0 \\ 0 & n < 0 \end{cases}$$

- Sequence:

$$\{x[n]\} = \{\ldots, 0, \underset{\uparrow}{1}, \frac{1}{2}, \frac{1}{4}, \ldots\},$$

where the arrow indicates the value at $n = 0$. If an arrow is absent, the first value of the sequence is assumed to occur at time $n = 0$.

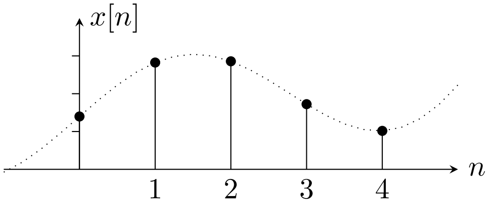
Outline

Sampling a CT signal $\{x(t)\}$ at uniformly-spaced points in time results in the DT sequence $\{x[n]\}$, where

$$x[n] = x(nT_s)$$

for all integers n, and where:

- T_s is the sampling period.
- The variable $f_s = \frac{1}{T_s}$ is the sampling frequency.



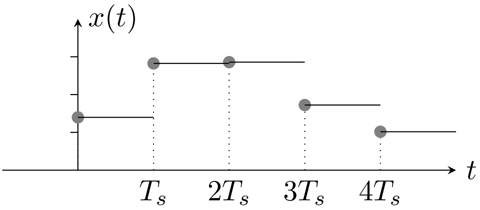
Outline

Zero-order Hold

A CT signal $\{x(t)\}$ can be obtained from a DT signal $\{x[n]\}$ by "holding" the value of the DT signal constant for one sampling period T_s, such that:

$$x(t) = x[n] \qquad nT_s \leq t < (n+1)T_s.$$

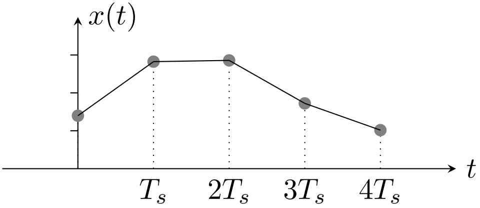
This is known as the **zero-order hold.**



Outline

First-order Hold

Alternative methods exist to obtain a CT signal from a DT signal, such as the *first-order hold*:



However, in this class we will exclusively work with *zero-order hold*, which we will simply call hold.

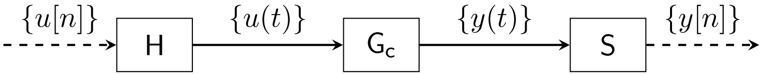
Outline

Interface between CT and DT systems

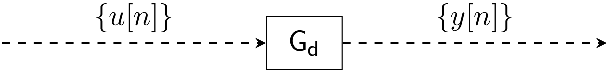
To interact with a CT (eg. "real-world") system $\mathsf{G_c}$ from a DT environment (eg. a computer), we need to:

- convert a DT signal into a CT signal, (ex: Digital to Analog Converters (DAC)
- and vice versa. (ex: Analog to Digital Converters (ADC))

This can be captured formally with the hold and sample operators as follows:



By defining the DT system $\mathsf{G_d} := \mathsf{S G_c H}$, the previous figure can be replaced by:



In the following, we consider the case where $\mathsf{G_c}$ is a CT linear time-invariant (LTI) system with a state-space description

$$\begin{aligned} \dot{q}(t) &= A_{\mathsf{c}}q(t) + B_{\mathsf{c}}u(t) \\ y(t) &= C_{\mathsf{c}}q(t) + D_{\mathsf{c}}u(t). \end{aligned} \tag{1}$$

The solution to the differential equation is

$$q(t) = e^{A_{\mathsf{c}}t}q(0) + \int_0^t e^{A_{\mathsf{c}}(t-\tau)}B_{\mathsf{c}}u(\tau)d\tau.$$

Demonstration recalled in class on blackboard.

The goal is to find

a description for the DT LTI system

$$\mathsf{G_d} = \mathsf{SG_cH} \quad .$$

The discrete-time equivalent of Equation 1 is of the form:

$$\begin{aligned} q[n+1] &= A_\mathsf{d} q[n] + B_\mathsf{d} u[n] \\ y[n] &= C_\mathsf{d} q[n] + D_\mathsf{d} u[n] \end{aligned}$$

which is the description of the DT system $\mathsf{G_d} = \mathsf{SG_cH}$ that we are looking for.

Outline

The discrete-time equivalent of Equation 1 is of the form

$$\begin{aligned} q[n+1] &= A_\mathsf{d} q[n] + B_\mathsf{d} u[n] \\ y[n] &= C_\mathsf{d} q[n] + D_\mathsf{d} u[n] \end{aligned}$$

which is the description of the DT system $\mathsf{G_d} = \mathsf{SG_cH}$ looked for. It can be shown, starting from the solution of the continuous state-space equation

$$q(t) = e^{A_\mathsf{c} t} q(0) + \int_0^t e^{A_\mathsf{c}(t-\tau)} B_\mathsf{c} u(\tau) d\tau.$$

that the corresponding discrete matrices are obtained as:

$$A_\mathsf{d} = e^{A_c T_s} \text{ and } \quad B_\mathsf{d} = \left(\int_0^{T_s} e^{A_c \tau} d\tau \right) B_c$$

with the sampling period T_s.

Demonstration is done on the board during the class. **Remarks**:

- This is the *exact* solution to the differential equation, there are no discretization errors.
- While it is exact, information is still lost by the discretization: the inter-sample behavior.

Outline

In this section, we will show how the discrete matrices A_d and B_d can be obtained through a slightly different approach. It consists of :

1. simplifying the state equations in (1), by taking into account that the input is constant during a sampling period (zero-order hold assumption).
2. rewriting the system as a no-input (or homogeneous) system of the type

$$\dot{r}(t) = Mr(t). \tag{2}$$

Such a system has the straightforward solution

$$r(t) = e^{Mt} r(0), \tag{3}$$

where e^{Mt} is the *matrix exponential*:

$$e^{Mt} := I + Mt + \frac{(Mt)^2}{2} + \cdots + \frac{(Mt)^k}{k!} + \cdots = \sum_{k=0}^{\infty} \frac{(Mt)^k}{k!}.$$

Remarks:

- When M is a scalar (matrix of dimension 1×1), we recover the standard definition of the exponential.
- It can be shown that the power series defining the matrix exponential converges for all real (or complex) square matrices.

Proof that the solution (3) satisfies the differential equation (2)

Let $r(t) = e^{Mt}r(0)$. It follows that $\dot{r}(t) = \frac{de^{Mt}}{dt}r(0)$.
For convergent power series, the derivative can be evaluated as:

$$\begin{aligned}
\frac{de^{Mt}}{dt} &= \frac{d}{dt}\sum_{k=0}^{\infty}\frac{(Mt)^k}{k!} \\
&= \sum_{k=0}^{\infty}\frac{kM^k t^{k-1}}{k!} = \sum_{k=1}^{\infty}\frac{kM^k t^{k-1}}{k!} \\
&= \sum_{k=1}^{\infty}\frac{M^k t^{k-1}}{(k-1)!} = M\sum_{k=1}^{\infty}\frac{M^{k-1}t^{k-1}}{(k-1)!} \\
&= M\sum_{l=0}^{\infty}\frac{M^l t^l}{l!} \text{ with the substitution } l := k-1 \\
&= Me^{Mt}.
\end{aligned}$$

$$\therefore \quad \dot{r}(t) = Me^{Mt}r(0) = Mr(t).$$

Example : Double integrator with zero input

"Standard solution approach"

Consider the system described by the differential equation

$$\ddot{y}(t) = 0$$

with initial conditions $\dot{y}(0) = v_0, \ y(0) = y_0$.

We can immediately write the solution:

$$\begin{aligned} \dot{y}(t) &= v_0, \\ y(t) &= y_0 + v_0 t. \end{aligned}$$

Example : Double integrator with zero input

The solution can also be derived using the matrix exponential: let

$$r(t) = \begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix} := \begin{bmatrix} y(t) \\ \dot{y}(t) \end{bmatrix}$$

$$r(0) = \begin{bmatrix} y_0 \\ v_0 \end{bmatrix}.$$

Then

$$\begin{bmatrix} \dot{r}_1(t) \\ \dot{r}_2(t) \end{bmatrix} = \begin{bmatrix} r_2(t) \\ 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{=M} \begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix}.$$

Note that the matrix M is *nilpotent* because $M^2 = 0$, and therefore

$$e^{Mt} = I + Mt$$

$$\begin{bmatrix} r_1(t) \\ r_2(t) \end{bmatrix} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ v_0 \end{bmatrix}.$$

which is identical solution as the one found before, i.e.

$$\dot{y}(t) = v_0, \quad y(t) = y_0 + v_0 t. \tag{4}$$

Relation to the Euler forward method

Matrix exponential approximation

When t is very small, the matrix exponential can be approximated by

$$e^{Mt} \approx I + Mt$$

and therefore $r(t) = e^{Mt}r(0) \approx (I + Mt)r(0) \quad \approx r(0) + tMr(0)$.

Euler forward method

The Euler forward method approximates a function's derivative as

$$\dot{r}(0) \approx \frac{r(t) - r(0)}{t}.$$

For small t, and with $\dot{r}(t) = Mr(t)$ we have $\frac{r(t)-r(0)}{t} \approx Mr(0)$, which yields $r(t) \approx r(0) + tMr(0)$.

$\Rightarrow$ The Euler forward method gives the same result as a first-order approximation to the matrix exponential.

Outline

The original problem was, for the first time step

$$\begin{aligned} \dot{q}(t) &= A_c q(t) + B_c u(t), \quad q(0) = q[0], \\ u(t) &= u[0] \quad 0 \leq t < T_s \end{aligned}$$

where

1. u is constant over the sampling interval due to the zero-order hold device
2. and we set the initial state $q(0)$ to its discrete-time counter-part $q[0]$.

We can rewrite these equations as the zero-input system (2)

$$\begin{bmatrix} \dot{q}(t) \\ \dot{u}(t) \end{bmatrix} = \underbrace{\begin{bmatrix} A_c & B_c \\ 0 & 0 \end{bmatrix}}_{=:\, M} \begin{bmatrix} q(t) \\ u(t) \end{bmatrix}$$

by including the input in the state of this adapted system.

Therefore, the solution at time T_s^-, just before the sampling time, is

$$\begin{bmatrix} q(T_s^-) \\ u(T_s^-) \end{bmatrix} = e^{MT_s} \begin{bmatrix} q(0) \\ u(0) \end{bmatrix}$$

with

$$F = e^{MT_s} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}.$$

It is easy to show that

$$F_{21} = 0 \text{ and } F_{22} = I,$$

since

$$M^k = \begin{bmatrix} * & * \\ 0 & 0 \end{bmatrix} \text{ for any integer } k \geq 1.$$

This result is expected, since u is constant over the sampling interval and thus $u(T_s^-) = u(0)$.

$$A_{\mathsf{d}} = F_{11}, \quad B_{\mathsf{d}} = F_{12}$$
$$C_{\mathsf{d}} = C_{\mathsf{c}}, \quad D_{\mathsf{d}} = D_{\mathsf{c}}.$$

Then

$$\begin{aligned} q[1] &= A_{\mathsf{d}} q[0] + B_{\mathsf{d}} u[0] \\ y[0] &= C_{\mathsf{d}} q[0] + D_{\mathsf{d}} u[0]. \end{aligned}$$

We can do this for *any* time period, since the CT system is time invariant, and thus obtain:

$$\begin{aligned} q[n+1] &= A_{\mathsf{d}} q[n] + B_{\mathsf{d}} u[n] \\ y[n] &= C_{\mathsf{d}} q[n] + D_{\mathsf{d}} u[n] \end{aligned}$$

which is the description of the DT system $\mathsf{G_d} = \mathsf{SG_cH}$ that we were looking for.

- This is the *exact* solution to the differential equation, there are no discretization errors.
- While it is exact, information is still lost by the discretization: The inter-sample behavior.

Mass-spring-damper system

Consider a mass-spring-damper system with the differential equation

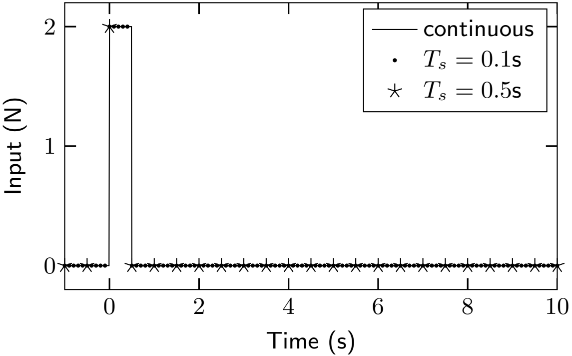
$$\ddot{p}(t) + \dot{p}(t) + p(t) = f(t)$$

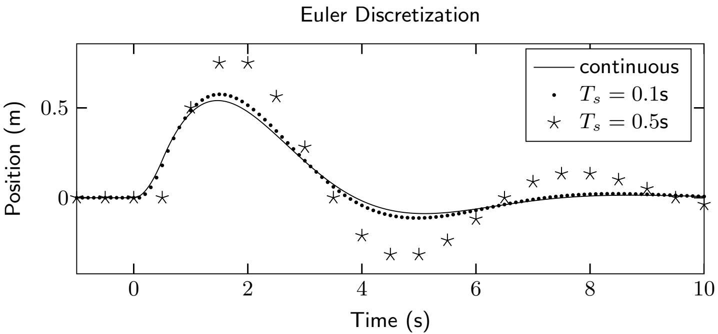
and compare:

1. the Euler discretization,
2. to the exact discretization,

with sampling times of 0.1s and 0.5s.

The input to the continuous and discretized models is a pulse with a width of 0.5 seconds, identical to the longest sampling time used.





When discretizing using the Euler discretization, the output strongly depends on the discretization time, and differs from the continuous-time output even for small sampling times (remember that the Euler discretization is identical to a first-order approximation of the matrix exponential – the errors seen here stem from this approximation)

The exact discretization has no discretization errors. The discrete-time output coincides with the continuous-time output at the sampling times: