Discrete-time systems & control.

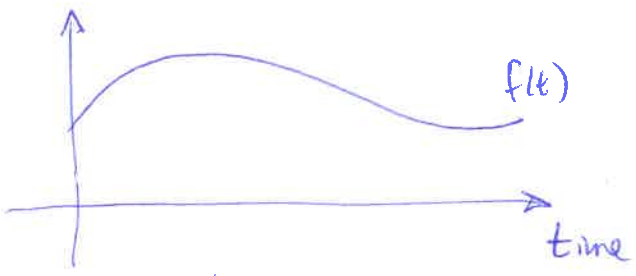
So far in the class, we have considered continuous-time signals such as position $x(t)$, angle $\theta(t)$, and so on.

Systems are what relates one signal to another, for example $m\ddot{x} + b\dot{x} + kx = f$ relates $x(t)$ to $f(t)$.
We looked at the corresponding transfer function $\frac{1}{ms^2 + bs + k}$
and studied the performance (transient, frequency domain) as well as tools for controller design (PID, root locus, Bode).

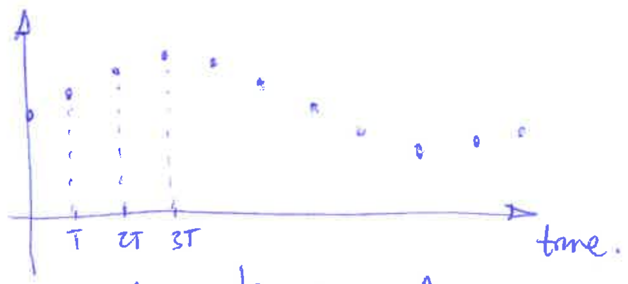
However, many systems and signals are discrete by nature. Instead of time being continuous ($t \in \mathbb{R}$, $t \geq 0$), time is discretized: $t \in \{0, T, 2T, 3T, \ldots\}$. Some examples:

"sampling time".

- stock prices, (daily, monthly, etc).
- digital audio
- quantized sensor readings $\left\{ \begin{array}{l} \text{gyro} \\ \text{temperature} \\ \text{heart rate} \\ \ldots \end{array} \right.$



continuous signal

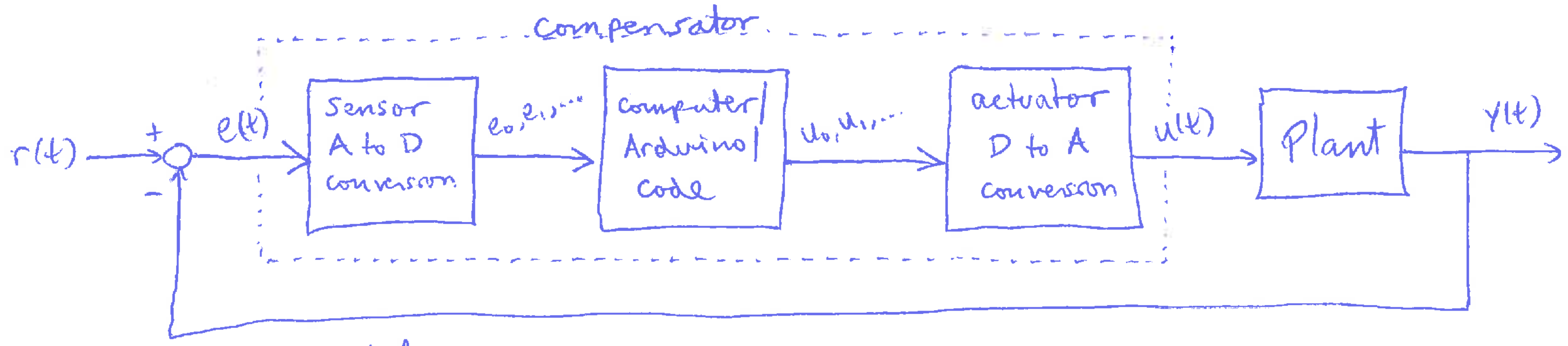


discrete signal

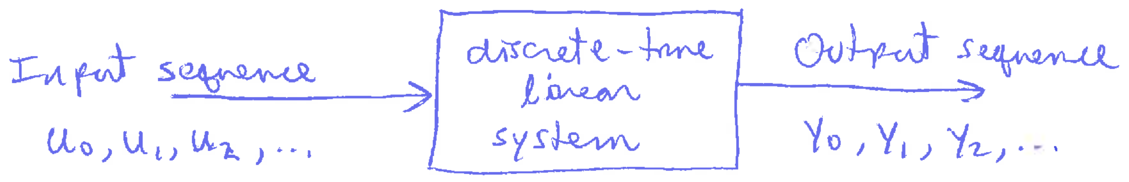
We saw how control systems can be implemented as circuits (e.g. an op amp circuit used to make a PID controller).

But increasingly, we see sensors that provide data/measurements at periodic time intervals and "controllers" implemented in <u>code</u> rather than with circuits. Then, the code outputs a sequence of decisions, which are sent to the ~~actuator~~ to control the system.

Here is what that might look like:



In such a control system, we have a <u>mix</u> of continuous and discrete signals! To understand the discrete part, we need to talk about what a discrete-time linear system looks like. Namely, what does this look like?



In general, at each time step, the <u>output</u> should be some linear combination of <u>past</u> inputs and outputs. (Otherwise, it wouldn't be causal).

Everything we have seen in this class has an analogue for discrete-time systems. Simple example:

<u>ODE</u>: $\dot{y} = ay$ with $y(0) = y_0$ $\longrightarrow$ $y(t) = e^{at} y_0$ for $t \geq 0$.

<u>difference equation</u>: $y_{k+1} = a y_k$ with y_0 given. $\longrightarrow$ $y_k = a^k y_0$ for $k = 0, 1, 2, \ldots$

Instead of <u>exponentials</u>, we get <u>powers</u>.

<u>continuous</u>

(Laplace transform) $\mathcal{L}\{y(t)\} = \int_0^\infty e^{-st} y(t)\, dt$

$$\ddot{y} - 2\dot{y} - 3y = \dot{u} + 5u$$
$$\big\downarrow \mathcal{L}$$
$$(s^2 - 2s - 3)\, Y(s) = (s+5)\, U(s)$$

$$\frac{Y(s)}{U(s)} = \frac{s+5}{s^2 - 2s - 3} = \frac{2}{s-3} - \frac{1}{s+1}$$

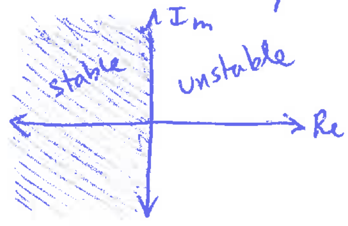
(continuous transfer function)

$$\mathcal{L}^{-1}\left(\frac{Y(s)}{U(s)}\right) = 2e^{3t} - e^{-t}, \quad t \geq 0.$$

(impulse response) $u(t) = \delta(t)$.

Stability if all poles of the transfer function satisfy $Re(s) < 0$

(left-half plane)



<u>discrete</u>

(Z-transform) $Z\{y_k\} = \sum_{k=0}^{\infty} y_k z^{-k}$

$$y_{k+2} - 2y_{k+1} - 3y_k = u_{k+1} + 5u_k$$
$$\big\downarrow Z$$
$$(z^2 - 2z - 3)\, Y(z) = (z+5)\, U(z)$$

$$\frac{Y(z)}{U(z)} = \frac{z+5}{z^2 - 2z - 3} = \frac{2}{z-3} - \frac{1}{z+1}$$

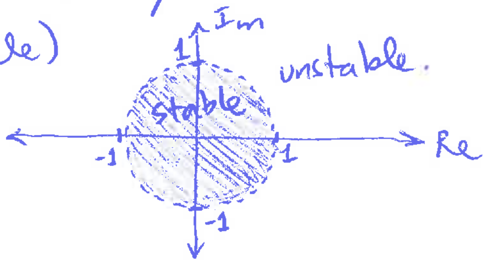
(discrete transfer function)

$$Z^{-1}\left(\frac{Y(z)}{U(z)}\right) = 2 \cdot 3^k - (-1)^k, \quad k = 0, 1, \ldots$$

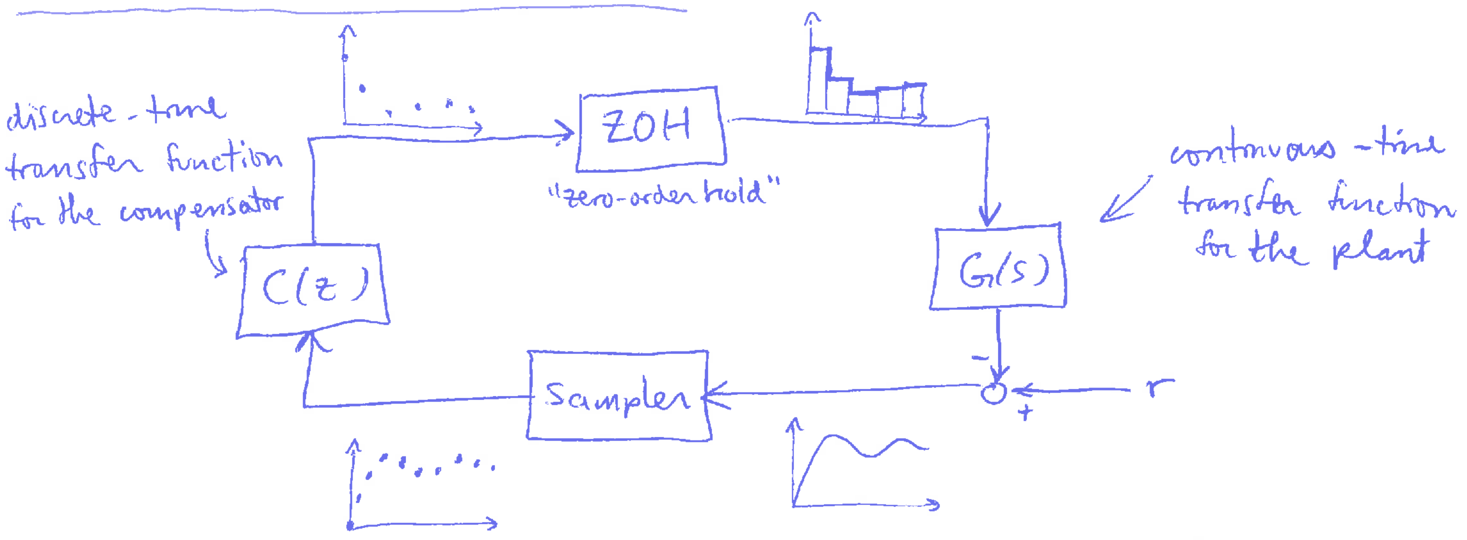
(impulse response), $u_k = \{1, 0, 0, \ldots\}$.

Stability if all poles of the transfer function satisfy $|z| < 1$.

(unit circle)



More abstract representation :


discrete-time transfer function for the compensator
ZOH
"zero-order hold"
continuous-time transfer function for the plant
C(z)
G(s)
Sampler
r
−
+

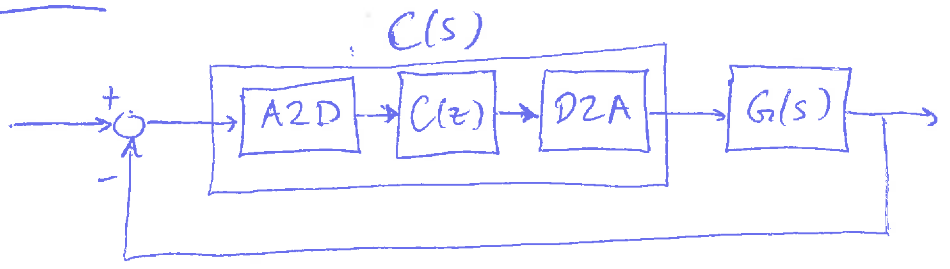
* ideal sampler performs A2D conversion: $f(t) \longrightarrow \{f(0), f(T), f(2T), \dots\}$.

* zero-order hold performs D2A conversion $\{f_0, f_1, f_2, \dots\} \longrightarrow \sum_{k=0}^{\infty} f_k \underbrace{\delta_T(t-kT)\cdot T}_{\text{rectangular pulse.}}$

There are other possibilities, but these are the simplest.

Two options

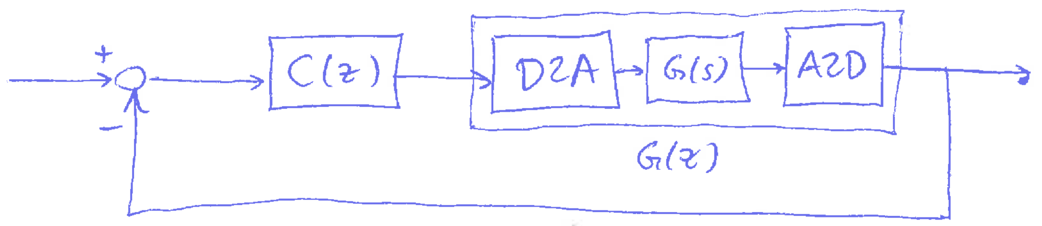
1)



"Emulation"
Design ideal C(s), then find C(z) so that we get a good approximation to C(s) after D to A conversion.

- This method is best when T (sampling time) is small.
- difficult to get good guarantees on performance
- provides insight on performance at all times t, not just at sample points.

2)



"Discrete design"
Derive equivalent discrete-time plant G(z); design C(z) directly.

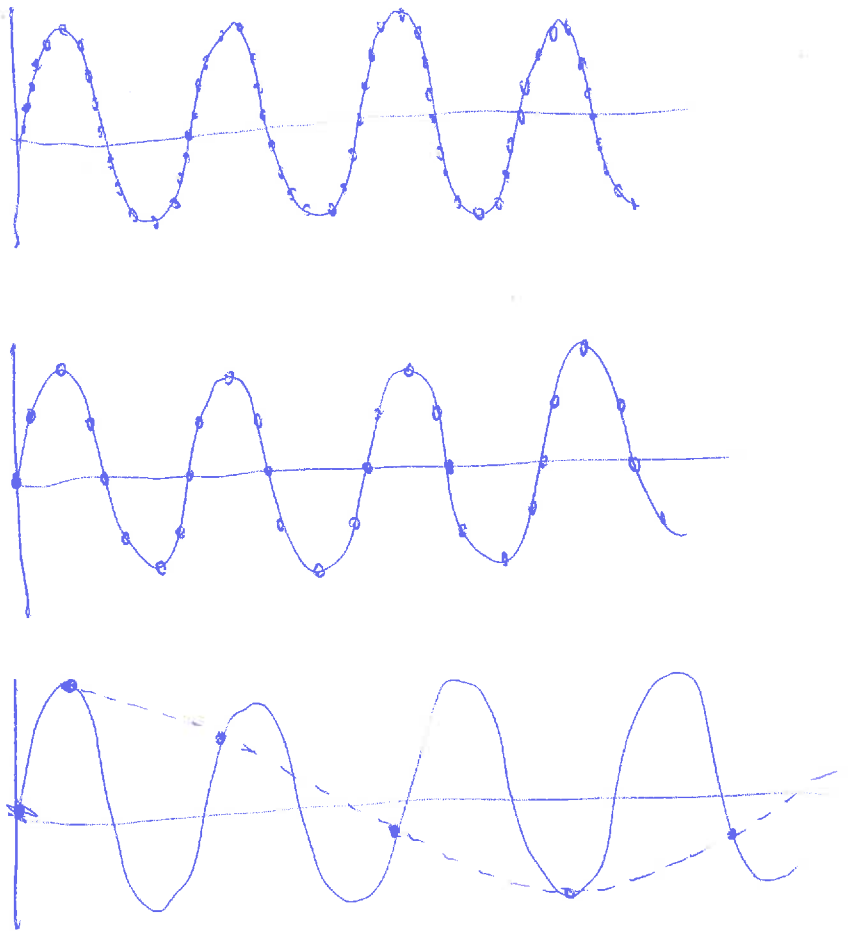
- This method is best when T is larger.
- Can be easier to get robust performance guarantees
- only describes behavior at the sample points $t = 0, T, 2T, \dots$

Emulation approach

It is impossible to pick $C(z)$ such that we perfectly reconstruct $C(s)$.

$$\begin{tikzcd} \arrow[r] & \boxed{A2D} \arrow[r] & \boxed{C(z)} \arrow[r] & \boxed{D2A} \arrow[r] & {} \arrow[r, phantom, "\overset{?}{=}"] & {} \arrow[r] & \boxed{C(s)} \arrow[r] & {} \end{tikzcd}$$

The main limitation is that information is lost during the A2D sampling. Specifically, if frequency is too high, we can't reconstruct a signal. Suppose we try to reconstruct a sampled sinusoid using small, medium, ~~high~~ large sample time:

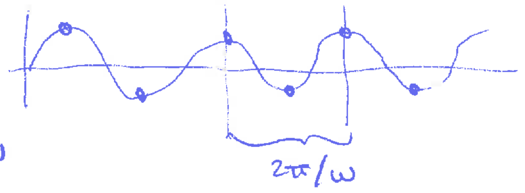


when frequency of sampling is too low, we can't recover original signal.

Instead, we recover low frequencies that are not present in the original signal.

This is called aliasing

The ~~highest~~ lowest sampling frequency we can use is 2 times the frequency of the input, which corresponds to:



We need 2 samples per period, so $T \leq \pi/\omega$

$\uparrow$ sample time

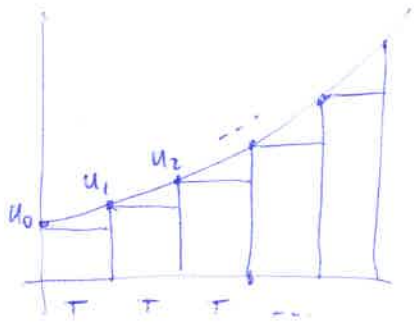
Put another way, if we use sampling time T, the largest frequency we can represent is $\omega < \frac{\pi}{T}$.

The frequency $\frac{\pi}{T}$ is called the Nyquist frequency.

There are many ways to approximate a system.

We can consider a simple integrator: $\dot{y} = u$

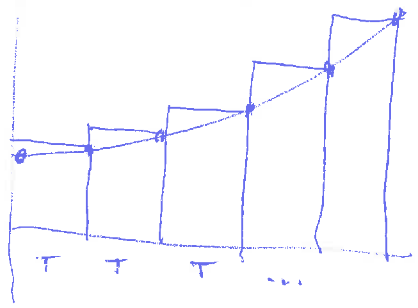
If we write this as $y(t) = \int_0^t u(\tau)\,d\tau$, then we are trying to approximate the integral.



"Forward Euler"

$$y_t \approx (u_0 + u_1 + \cdots + u_{t-1})T$$

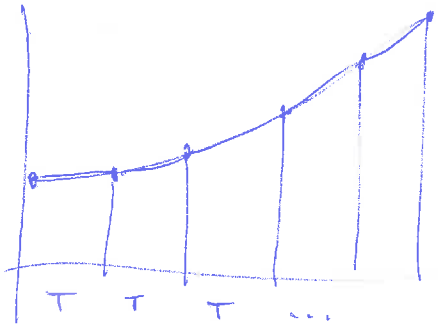
$$\Rightarrow \frac{y_{t+1} - y_t}{T} \approx u_t. \Rightarrow \boxed{\frac{u}{y} = \frac{z-1}{T}}$$



"Backward Euler"

$$y_t \approx (u_1 + u_2 + \cdots + u_t)T$$

$$\Rightarrow \frac{y_t - y_{t-1}}{T} \approx u_t \Rightarrow \boxed{\frac{u}{y} = \frac{z-1}{Tz}}$$



"Trapezoid" / "Tustin"

$$y_t = \left[\left(\frac{u_0+u_1}{2}\right) + \cdots + \left(\frac{u_{t-1}+u_t}{2}\right)\right]T$$

$$\Rightarrow \frac{y_t - y_{t-1}}{T} \approx \frac{u_t + u_{t-1}}{2} \Rightarrow \boxed{\frac{u}{y} = \frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

These are different ways of approximating "s".

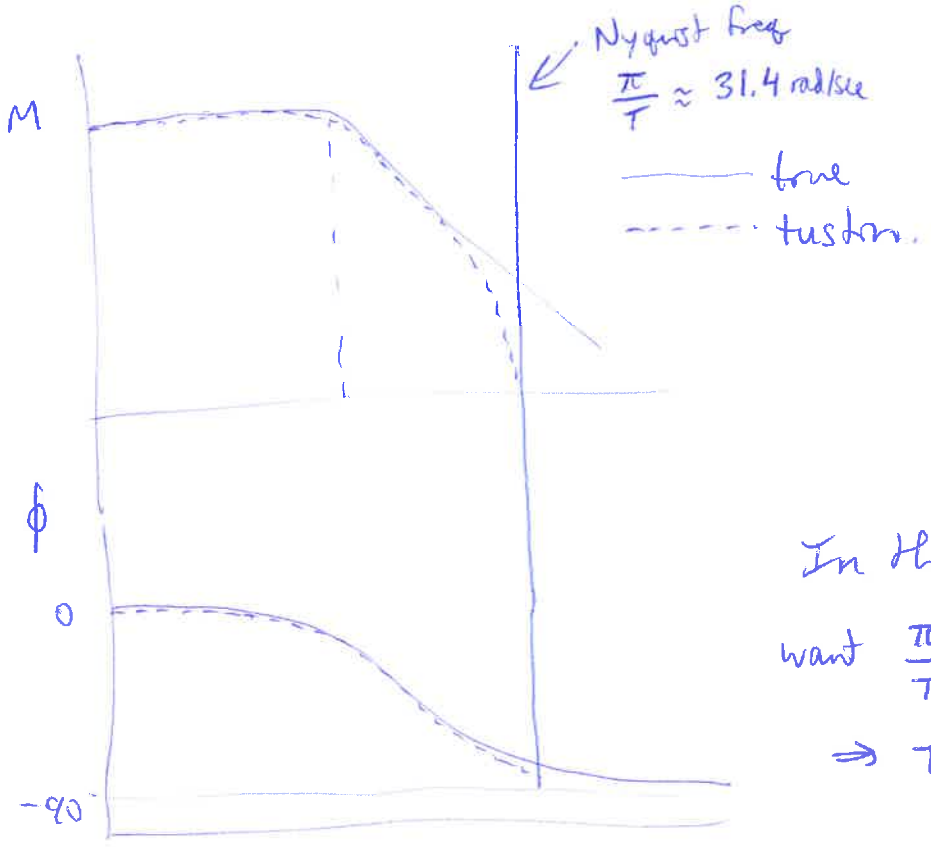
So, for example, to use the Tustin transform, we would do:

$$\underbrace{C(z)}_{\text{discrete TF}} = \underbrace{C(s)}_{\text{continuous TF}}\Big|_{s=\frac{2}{T}\left(\frac{z-1}{z+1}\right)}$$

These methods have varying degrees of success. In general, we can get more accuracy if we use higher-order versions.

<u>Example</u>: Approximate $\frac{1}{s+1}$ using $T = 0.1$ seconds.

Can plot bode for each:



In practice, pick sample rate so that Nyquist freq is at least 5x larger than bandwidth.

In this case, $\omega_b = 1$ rad/sec.

want $\frac{\pi}{T} > 5\omega_b = 5$

$\Rightarrow T < \frac{\pi}{5} \approx 0.628$ seconds

Example: PID.

Say we design $\frac{U(s)}{E(s)} = \underset{\substack{\uparrow \\ P}}{1} + \underset{\substack{\uparrow \\ D}}{0.5\,s} + \underset{\substack{\uparrow \\ I}}{\frac{0.2}{s}}$.

Use Tustin with $T=0.01$.

$$\Rightarrow \frac{U(z)}{E(z)} = \frac{101z^2 - 200z + 99}{z^2 - 1} \longrightarrow \frac{101 - 200z^{-1} + 99z^{-2}}{1 - z^{-2}}$$

$$\Rightarrow u_k = 101e_k - 200e_{k-1} + 99e_{k-2} + u_{k-2}$$

So we need to buffer past errors $e_{k-1}, e_{k-2}, u_{k-2}$. and signals:

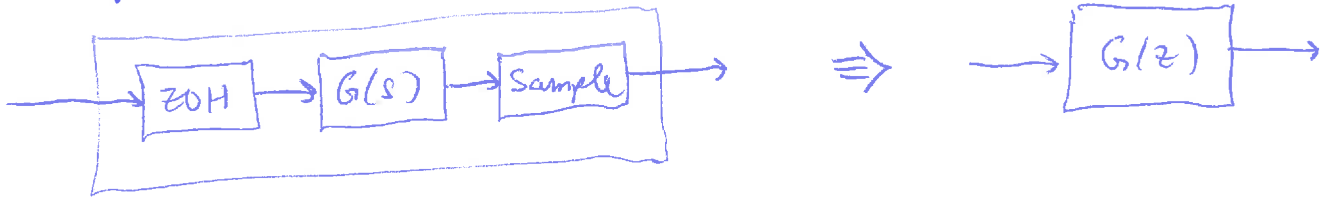
In Matlab, can use the "c2d" to convert continuous to discrete. e.g.

$$\hat{G} = \mathrm{c2d}(G, T, \text{'tustin'})$$

for Tustin discretization.

Discrete design

converting continuous → discrete as in



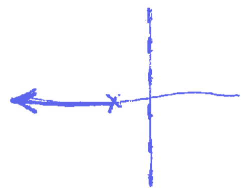
can be computed exactly (exact representation for G(z)).
This is the default option in the c2d command ('zoh' = zero-order hold)

We can do root locus design in discrete time!

- RL rules are exactly the same as in continuous time.
- stability region is now the unit circle instead of imaginary axis

Although RL rules are the same, the intuition we built on the class, e.g. for first and second-order systems, will change!

take $\frac{1}{s+a}$ →

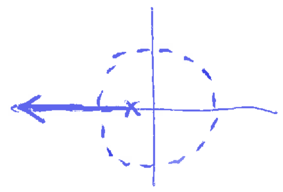


stable closed loop for all $K>0$.

Always stable for K sufficiently large.

but in discrete time...

$\frac{1}{z+a}$ →



stable only for a small range of K values.

unstable for K sufficiently large.