Systems, Filters

- a linear time-invariant system, H, can be described in a number of ways, whether it is discrete- or continuous-time:

great for analysis of existing systems, getting a transfer function or estimating a solution

1) INPUT-OUTPUT modelling: (no state variables)

a) the differential or difference-equation description, e.g.,

→ CT: $\ddot{y} + \alpha_1 \dot{y} + \alpha_0 y = \beta_2 \ddot{u} + \beta_1 \dot{u} + \beta_0 u$, $\forall t \geq 0$, $t \in \mathbb{R}$

- initial conditions:

$$y(0) = y_0 \in \mathbb{R}$$
$$\dot{y}(0) = y_{01} \in \mathbb{R}$$
$$u(0) = u_0 \in \mathbb{R},$$
$$\dot{u}(0) = u_{01} \in \mathbb{R}.$$

→ DT: $y[n-2] + \alpha_1 y[n-1] + \alpha_0 y[n] = \beta_2 u[n-2] + \beta_1 u[n-1] + \beta_0 u[n]$, $\forall n \geq 0$, $n \in \mathbb{Z}$

- initial conditions:

$$y[k] = 0 \quad \forall k < 0$$
$$u[k] = 0 \quad \forall k < 0$$
$$y[0] = y_0 \in \mathbb{R}$$
$$u[0] = u_0 \in \mathbb{R}$$

great for checking BIBO stability

b) the transfer function (assumes all ICs are "at rest" or zero):

→ CT: take LTs of both sides —

$$s^2 \hat{y} + \alpha_1 s \hat{y} + \alpha_0 \hat{y} = \beta_2 s^2 \hat{u} + \beta_1 s \hat{u} + \beta_0 \hat{u}$$

$$\sim (s^2 + \alpha_1 s + \alpha_0) \hat{y} = (\beta_2 s^2 + \beta_1 s + \beta_0) \hat{u}$$

$$\sim \hat{y} = \underbrace{\left(\frac{\beta_2 s^2 + \beta_1 s + \beta_0}{s^2 + \alpha_1 s + \alpha_0} \right)}_{\text{the transfer function, } H(s)} \hat{u}$$

→ DT: – take ZTs of both sides –

$$z^{-2}\hat{y} + \alpha_1 z^{-1}\hat{y} + \alpha_0 \hat{y} = \beta_2 z^{-2}\hat{u} + \beta_1 z^{-1}\hat{u} + \beta_0 \hat{u}$$

$$\sim (z^{-2} + \alpha_1 z^{-1} + \alpha_0)\hat{y} = (\beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0)\hat{u}$$

$$\sim \quad \hat{y} = \underbrace{\left(\frac{\beta_2 z^{-2} + \beta_1 z^{-1} + \beta_0}{z^{-2} + \alpha_1 z^{-1} + \alpha_0}\right)}_{H(z) \text{ is the transfer function}} \hat{u}$$

c) the <u>convolution</u> description:

→ CT: $$y(t) = y_0 + \int_0^t h(t-\tau)\, u(\tau)\, d\tau$$

→ DT: $$y(n) = y_0 + \sum_{k=0}^{n} h(n-k)\, u(k)$$

$h(\cdot)$ in both cases is the system impulse response, often obtained using an inverse Laplace or Z transform.

2) <u>internal or state modelling</u>:

great for realizing or building a filter

→ CT: $$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$
, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $u \in \mathbb{R}^p$,
(usually, in this course, $m = p = 1$, the SISO case)

IC: $x(0) = x_0 \in \mathbb{R}^n$

e.g.

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ -2 & -3 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \quad , \quad x_0 = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

$$y = \begin{bmatrix} 4 & -7 \end{bmatrix} x + [0]\, u$$

→ DT: $x(n+1) = Ax(n) + Bu(n)$

$y(n) = Cx(n) + Du(n)$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, $u \in \mathbb{R}^p$,

IC: $x(0) = x_0 \in \mathbb{R}^n$

eg.

$$\left.\begin{aligned} x(n+1) &= (0.1)x(n) + (0.25)u(n) \\ y(n) &= 3x(n) + 2u(n), \\ x(0) &= -7. \end{aligned}\right]$$

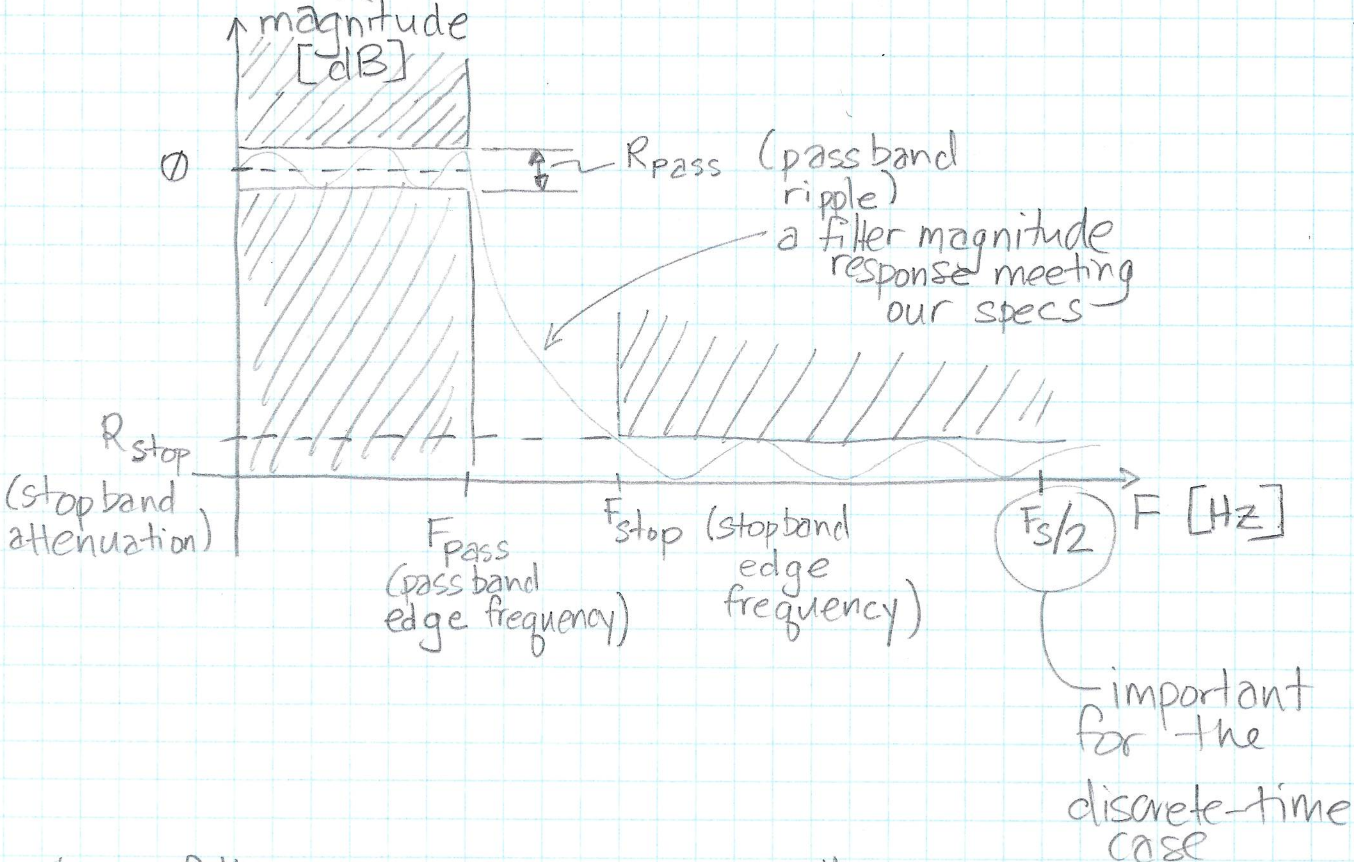
$n=1$ in this example. ($m=p=1$ for SISO case)

→ each of these model types has its uses

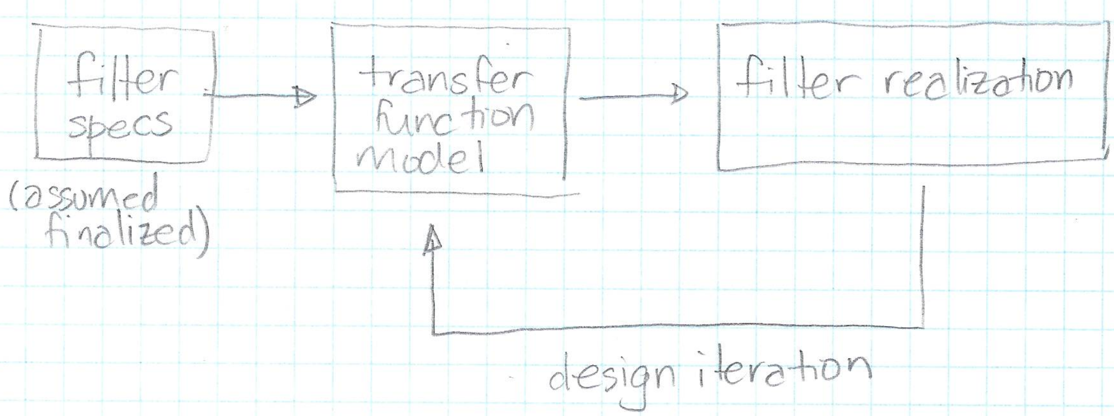
→ Matlab offers tools to switch easily amongst these various descriptions.

FIR & IIR filters

- we often start w/ a filter design in the frequency domain; that is, a filter is created to meet certain frequency-domain specifications, which, in the lowpass case, look like



- The filter design process usually goes as follows:



Getting the Filter TF

- in embedded systems, we must design filters in both continuous- and discrete-time.
- most often, CT (analog) filters are used for anti-aliasing (before the sampling process), equalization, amplification and noise removal.
- DT filters are often implemented in software, and are used for a wide-range of effects and signal conditioning
- In Matlab, there are a number of filter "families" available, and TFs can be be generated in CT or DT by each of the filter family "functions".

Filter Family	Characteristics	CT	DT
Butterworth	"maximally flat" passbands, monotonic response, slower roll-off	[num, den] = butter(N,Wn,ftype,'s');	[num,den] = butter(N,fn,ftype);
Chebyshev Type I and Type II	Type I: passband ripple, flat stopband Type II: stopband ripple, flat passband Both have faster roll-off than Butterworth	[num, den] = cheby1(N,Rp,Wn,ftype,'s'); [num, den] = cheby2(N,Rs,Wn,ftype,'s');	[num, den] = cheby1(N,Rp,fn,ftype); [num, den] = cheby2(N,Rs,fn,ftype);
Elliptical	passband & stopband ripple for fastest roll-off	[num, den] = ellip(N,Rp,Rs,Wn,ftype,'s');	[num,den] = ellip(N,Rp,Rs,fn,ftype);