http://elec3004.com

Frequency Response

ELEC 3004: **Systems**: Signals & Controls
Dr. Surya Singh

Lecture 9

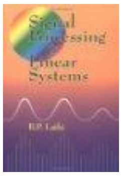
elec3004@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~elec3004/

March 27, 2019

Lecture Schedule:

Week	Date	Lecture Title
1	27-Feb	Introduction
	1-Mar	Systems Overview
2	6-Mar	Systems as Maps & Signals as Vectors
	8-Mar	Systems: Linear Differential Systems
3	13-Mar	Sampling Theory & Data Acquisition
	15-Mar	Aliasing & Antialiasing
4	20-Mar	Discrete Time Analysis & Z-Transform
	22-Mar	Second Order LTID (& Convolution Review)
5	**27-Mar**	**Frequency Response**
	29-Mar	Filter Analysis
6	3-Apr	Digital Filters (IIR) & Filter Analysis
	5-Apr	PS 1: Q & A
7	10-Apr	Digital Filter (FIR) & Digital Windows
	12-Apr	FFT
8	17-Apr	Active Filters & Estimation & Holiday
	19-Apr	**Holiday**
	24-Apr	
	26-Apr	
9	1-May	Introduction to Feedback Control
	3-May	Servoregulation/PID
10	8-May	PID & State-Space
	10-May	State-Space Control
11	15-May	Digital Control Design
	17-May	Stability
12	22-May	State Space Control System Design
	24-May	Shaping the Dynamic Response
13	29-May	System Identification & Information Theory
	31-May	Summary and Course Review

Follow Along Reading:



B. P. Lathi
Signal processing and linear systems
1998
TK5102.9.L38 1998

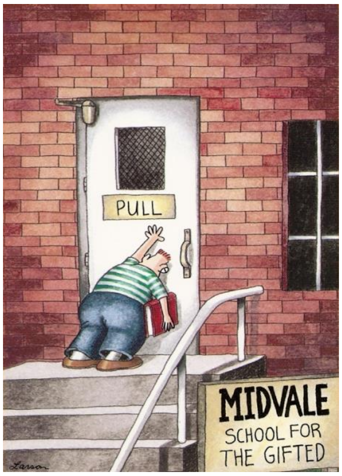
Today

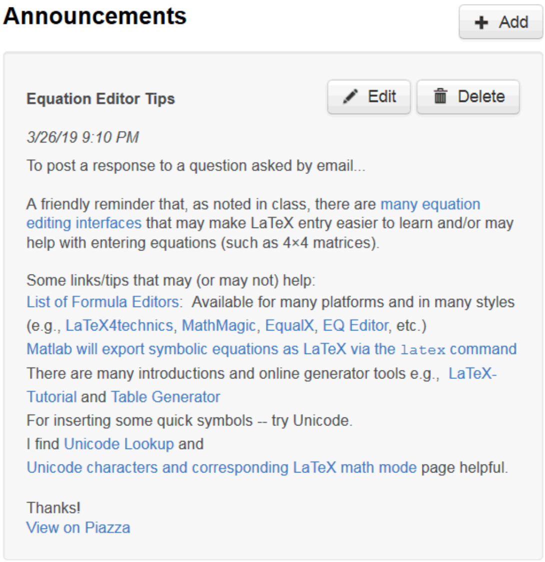
- Review mostly ☺
- Chapter 9 (**Time-Domain Analysis of Discrete-Time Systems**)
  - § 9.4 System Response to External Input
  - § 9.6 System Stability

- Chapter 10 (**Discrete-Time System Analysis Using the *z*-Transform**)
  - § 10.3 Properties of DTFT
  - § 10.5 Discrete-Time Linear System analysis by DTFT
  - § 10.7 Generalization of DTFT to the $\mathcal{Z}$–Transform

Next Time

Announcements





Announcements

Equation Editor Tips

3/26/19 9:10 PM

To post a response to a question asked by email...

A friendly reminder that, as noted in class, there are many equation editing interfaces that may make LaTeX entry easier to learn and/or may help with entering equations (such as 4×4 matrices).

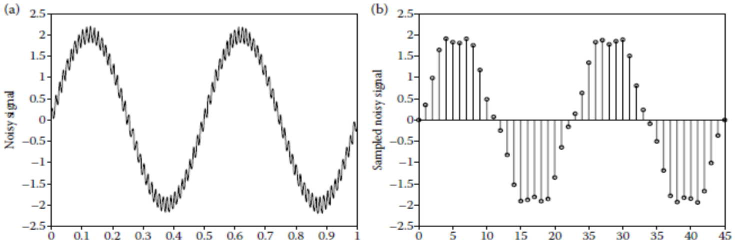
Some links/tips that may (or may not) help:
List of Formula Editors: Available for many platforms and in many styles (e.g., LaTeX4technics, MathMagic, EqualX, EQ Editor, etc.)
Matlab will export symbolic equations as LaTeX via the `latex` command
There are many introductions and online generator tools e.g., LaTeX-Tutorial and Table Generator
For inserting some quick symbols -- try Unicode.
I find Unicode Lookup and
Unicode characters and corresponding LaTeX math mode page helpful.

Thanks!
View on Piazza

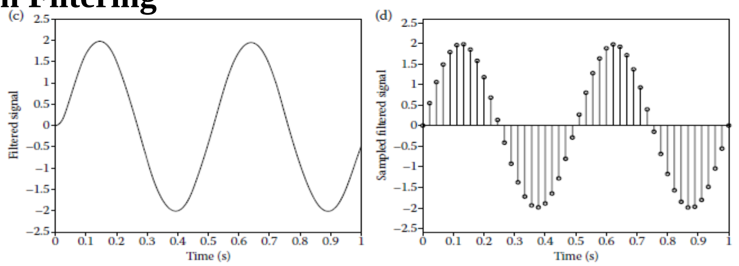
Thinking about Noise

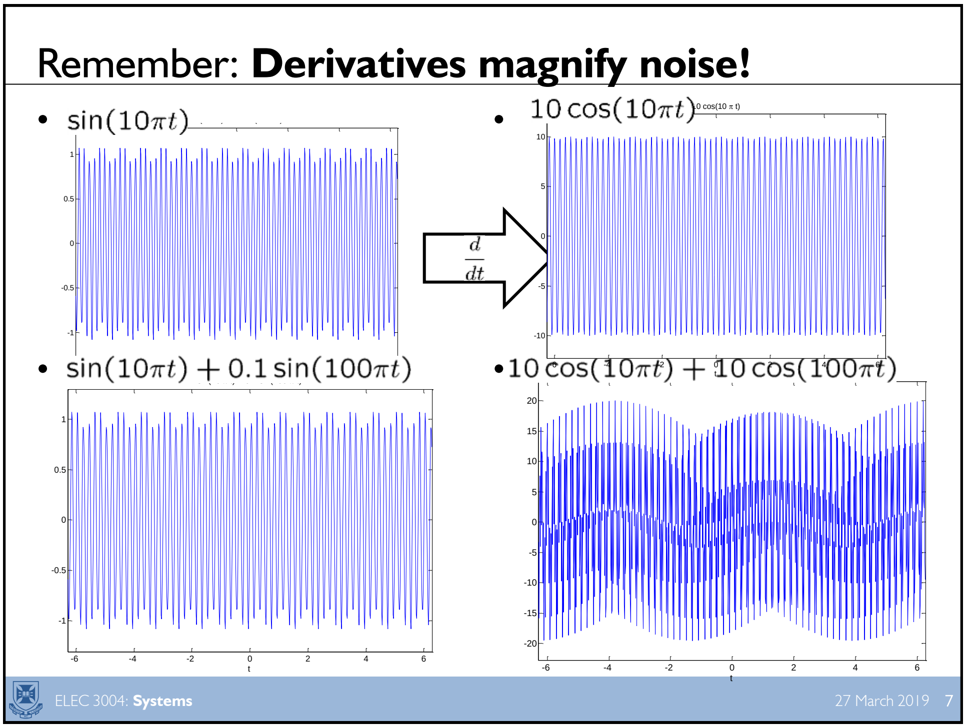
Remember: Effect of Noise…

- Without pre-filtering:



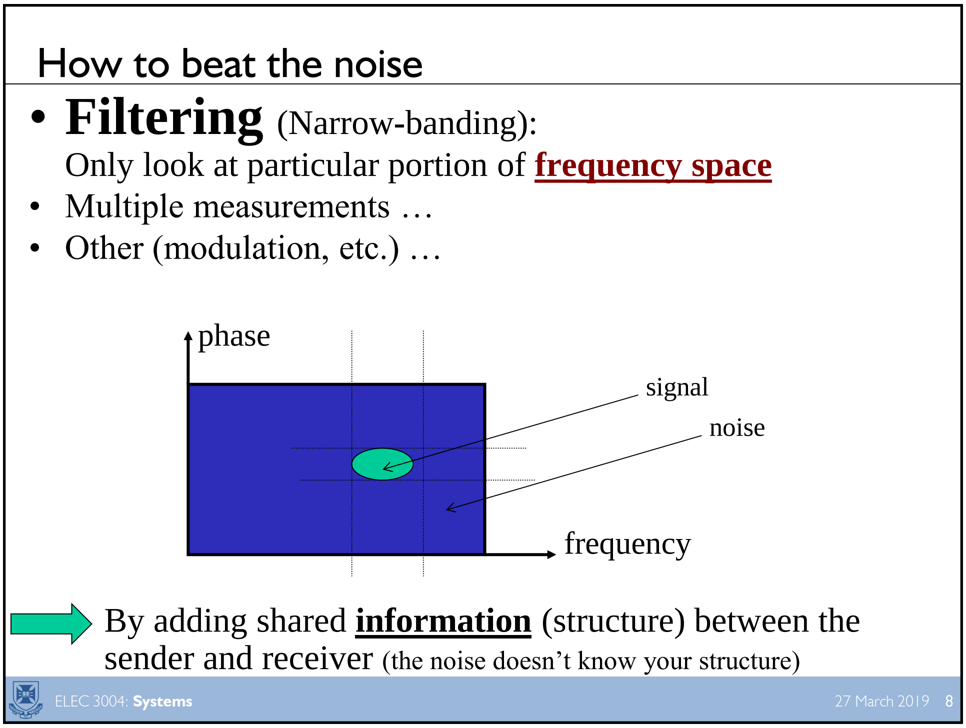
- **With Filtering**





Remember: **Derivatives magnify noise!**

- $\sin(10\pi t)$
- $10\cos(10\pi t)$
- $\sin(10\pi t) + 0.1\sin(100\pi t)$
- $10\cos(10\pi t) + 10\cos(100\pi t)$

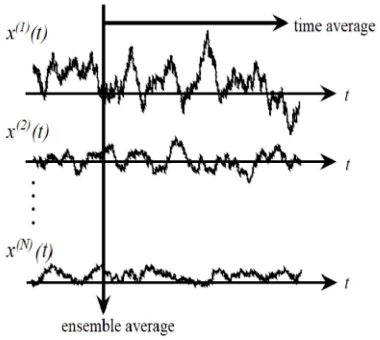


How to beat the noise

- **Filtering** (Narrow-banding):
  Only look at particular portion of **frequency space**
- Multiple measurements …
- Other (modulation, etc.) …

By adding shared **information** (structure) between the sender and receiver (the noise doesn't know your structure)

Treating Uncertainty with Multiple Measurements



1. **Over time:** multiple readings of a quantity over time
    - "stationary" or "ergodic" system
    - Sometimes called "integrating"
2. **Over space:** single measurement (summed) from multiple sensors each distributed in space
3. **Same Measurand**: multiple measurements take of the **same observable quantity** by multiple, related instruments

   e.g., measure position & velocity simultaneously

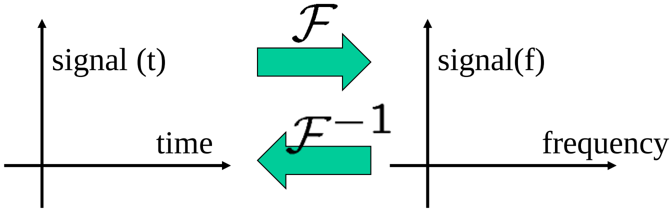
→ Basic "sensor fusion"

.

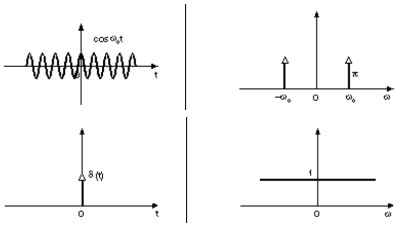
$$\sigma_{\text{final}} = \left[\sigma_1^{-1} + \sigma_2^{-1} + \cdots + \sigma_n^{-1}\right]^{-1}$$

Frequency

- How often the signal repeats
- Can be analyzed through Fourier Transform



- Examples:



Convolution*

(The * tool that let's us $\mathcal{F}$ and $\mathcal{F}^{-1}$ and more!)

Convolution Definition

The convolution of two functions $f_1(t)$ and $f_2(t)$ is defined as:

$$f(t) = \int_{-\infty}^{\infty} f_1(\tau) f_2(t-\tau) d\tau$$

$$= f_1(t) * f_2(t)$$

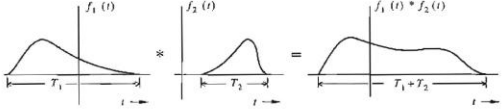
Source: URI ELE436

Convolution Properties

$$f_1(t) * f_2(t) \equiv \int_{-\infty}^{\infty} f_1(\tau) f_2(t-\tau)\, d\tau$$

Properties:

- Commutative: $f_1(t)*f_2(t) = f_2(t)*f_1(t)$
- Distributive: $f_1(t) * [f_2(t) + f_3(t)] = f_1(t) * f_2(t) + f_1(t) * f_3(t)$
- Associative: $f_1(t) * [f_2(t) * f_3(t)] = [f_1(t) * f_2(t)] * f_3(t)$
- Shift:
  if $f_1(t)*f_2(t)=c(t)$, then $f_1(t-\mathbf{T})*f_2(t)= f_1(t)*f_2(t-\mathbf{T})=c(t-\mathbf{T})$
- Identity (Convolution with an Impulse):
  $f(t) * \delta(t) = f(t)$
- Total Width:



Based on Lathi, *SPLS*, Sec 2.4-1

Convolution Properties [II]

- Convolution systems are **linear**:

$$h * (\alpha u_1 + \beta u_2) = \alpha(h * u_1) + \beta(h * u_2)$$

- Convolution systems are **causal**: the output y(t) at time t depends only on past inputs

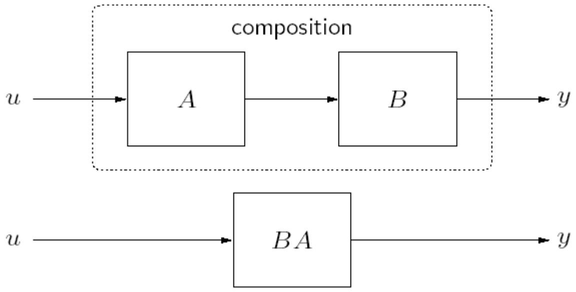
- Convolution systems are **time-invariant**
  (if we shift the signal, the output similarly shifts)

➔ $$\tilde{u}(t) = \begin{cases} 0 & t < T \\ u(t-T) & t \geq 0 \end{cases}$$

$$\tilde{y}(t) = \begin{cases} 0 & t < T \\ y(t-T) & t \geq 0 \end{cases}$$

Convolution Properties [III]

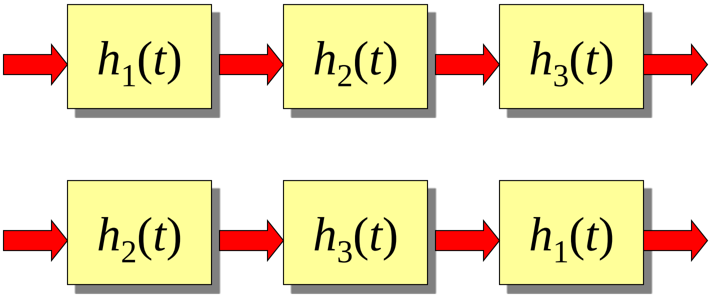
- Composition of convolution systems corresponds to:
  - multiplication of transfer functions
  - convolution of impulse responses



- Thus:
  - We can manipulate block diagrams with transfer functions as if they were simple gains
  - convolution systems commute with each other

Properties of Convolution: Distributive Property

$$[f_1(t) * f_2(t)] * f_3(t) = f_1(t) * [f_2(t) * f_3(t)]$$



- The two systems are identical!

Source: URI ELE436

Properties of Convolution: Commutative Property

$$\boxed{f_1(t) * f_2(t) = f_2(t) * f_1(t)}$$

$$f_1(t)*f_2(t) = \int_{-\infty}^{\infty} f_1(\tau)f_2(t-\tau)d\tau = \int_{\tau=-\infty}^{\tau=\infty} f_1(\tau)f_2(t-\tau)d\tau$$

$$= \int_{t-\tau=-\infty}^{t-\tau=\infty} f_1(t-\tau)f_2[t-(t-\tau)]d(t-\tau)$$

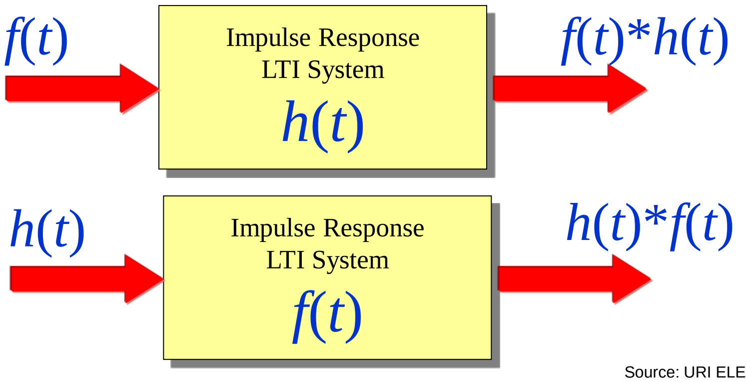
$$= -\int_{\tau=\infty}^{\tau=-\infty} f_1(t-\tau)f_2(\tau)d\tau$$

$$= \int_{-\infty}^{\infty} f_1(t-\tau)f_2(\tau)d\tau = f_2(t) * f_1(t)$$

Source: URI ELE436

Properties of Convolution: LTI System Response

$$\boxed{f_1(t) * f_2(t) = f_2(t) * f_1(t)}$$



$f(t)$

Impulse Response
LTI System
$h(t)$

$f(t)*h(t)$

$h(t)$

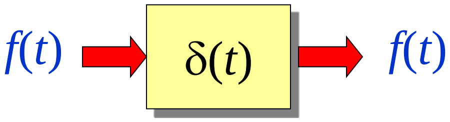
Impulse Response
LTI System
$f(t)$

$h(t)*f(t)$

Source: URI ELE436

Properties of Convolution

$$\boxed{f(t) * \delta(t) = f(t)}$$



$$f(t) * \delta(t) = \int_{-\infty}^{\infty} f(\tau)\delta(t-\tau)d\tau$$

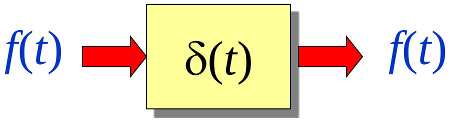
$$= \int_{-\infty}^{\infty} f(t-\tau)\delta(\tau)d\tau$$

$$= f(t)$$

Source: URI ELE436

Properties of Convolution

$$\boxed{f(t) * \delta(t) = f(t)}$$



$$\boxed{f(t) * \delta(t-T) = f(t-T)}$$

$$f(t) * \delta(t-T) = \int_{-\infty}^{\infty} f(\tau)\delta(t-T-\tau)d\tau$$

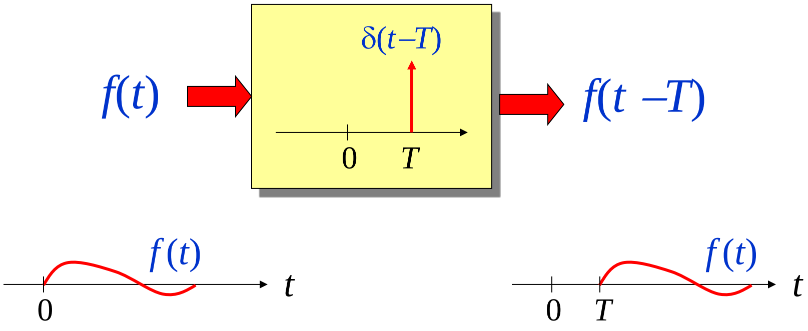
$$= \int_{-\infty}^{\infty} f(t-T-\tau)\delta(\tau)d\tau$$

$$= f(t-T)$$

Source: URI ELE436

Properties of Convolution

$$\boxed{f(t)*\delta(t-T)=f(t-T)}$$



Source: URI ELE436

Properties of Convolution

$$\boxed{f_1(t)*f_2(t)\xleftrightarrow{\mathcal{F}}F_1(j\omega)F_2(j\omega)}$$

$$F[f_1(t)*f_2(t)]=\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}f_1(\tau)f_2(t-\tau)d\tau\right]e^{-j\omega t}dt$$

$$=\int_{-\infty}^{\infty}f_1(\tau)\left[\int_{-\infty}^{\infty}f_2(t-\tau)e^{-j\omega t}dt\right]d\tau$$

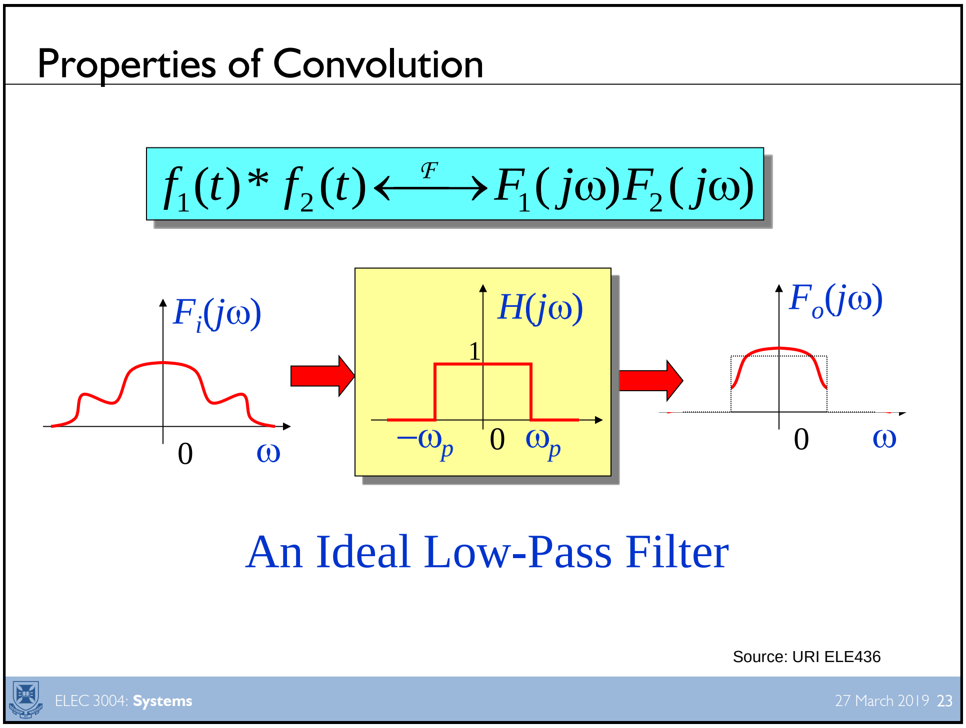
$$=\int_{-\infty}^{\infty}f_1(\tau)F_2(j\omega)e^{-j\omega\tau}d\tau$$

$$=F_2(j\omega)\int_{-\infty}^{\infty}f_1(\tau)e^{-j\omega\tau}d\tau\quad=F_1(j\omega)F_2(j\omega)$$

Time Domain
convolution

Frequency Domain
multiplication

Source: URI ELE436

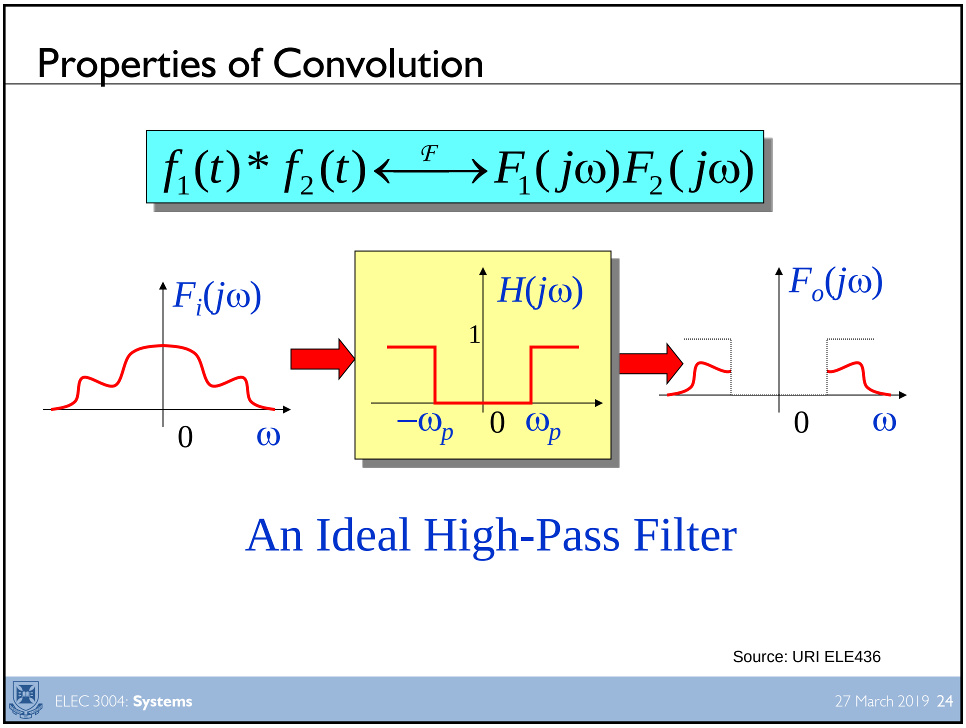


Properties of Convolution

$$f_1(t) * f_2(t) \xleftrightarrow{\mathcal{F}} F_1(j\omega)F_2(j\omega)$$

An Ideal Low-Pass Filter

Source: URI ELE436



Properties of Convolution

$$f_1(t) * f_2(t) \xleftrightarrow{\mathcal{F}} F_1(j\omega)F_2(j\omega)$$

An Ideal High-Pass Filter

Source: URI ELE436

Discrete Convolution

$$y[n] = x[n]*h[n] = \sum_{m=-\infty}^{\infty} x[m]\cdot h[n-m] = \sum_{m=-\infty}^{\infty} h[m]\cdot x[n-m]$$

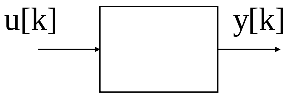
Consider this for the discrete case:

1. Rename the independent variable as **m**. You now have **x**[**m**] and **h**[**m**]. Flip **h**[**m**] over the origin. This is **h**[**-m**]
2. Shift **h**[**-m**] as far left as possible to a point "**n**", where the two signals barely touch. This is **h**[**n-m**]
3. Multiply the two signals and sum over all values of **m**. This is the convolution sum for the specific "**n**" picked above.
4. Shift / move **h**[**-m**] to the right by one sample, and obtain a new **h**[**n-m**]. Multiply and sum over all **m**.
5. 5. Repeat 2~4 until **h**[**n-m**] no longer overlaps with **x**[**m**], i.e., shifted out of the **x**[**m**] zone.

➔ The "*n*" dependency of *y*[*n*] deserves some care:
For each value of "*n*" the convolution sum must be computed ***separately*** over all values of a dummy variable "*m*".

Discrete-Time Systems & Discrete Convolution [1]

Will consider linear time-invariant (<u>LTI</u>) systems



Linear :

input u1[k] -> output y1[k]
input u2[k] -> output y2[k]
hence a.u1[k]+b.u2[k]-> a.y1[k]+b.y2[k]

Time-invariant (shift-invariant)

input u[k] -> output y[k]
hence input u[k-T] -> output y[k-T]

Discrete-Time Systems & Discrete Convolution [2]

Will consider causal systems

iff for all input signals with u[k]=0,k<0 -> output y[k]=0,k<0

Impulse response

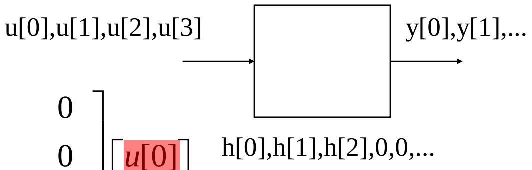
input …,0,0, $\overset{K=0}{1}$,0,0,0,...-> output …,0,0, $\overset{K=0}{h[0]}$,h[1],h[2],h[3],...

General input u[0],u[1],u[2],u[3]   (cfr. linearity & shift-invariance!)

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \end{bmatrix} = \begin{bmatrix} h[0] & 0 & 0 & 0 \\ h[1] & h[0] & 0 & 0 \\ h[2] & h[1] & h[0] & 0 \\ 0 & h[2] & h[1] & h[0] \\ 0 & 0 & h[2] & h[1] \\ 0 & 0 & 0 & h[2] \end{bmatrix} . \begin{bmatrix} u[0] \\ u[1] \\ u[2] \\ u[3] \end{bmatrix}$$

this is called a
`Toeplitz' matrix

Discrete-Time Systems & Discrete Convolution [3]



$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \end{bmatrix} = \begin{bmatrix} h[0] & 0 & 0 & 0 \\ h[1] & h[0] & 0 & 0 \\ h[2] & h[1] & h[0] & 0 \\ 0 & h[2] & h[1] & h[0] \\ 0 & 0 & h[2] & h[1] \\ 0 & 0 & 0 & h[2] \end{bmatrix} . \begin{bmatrix} u[0] \\ u[1] \\ u[2] \\ u[3] \end{bmatrix}$$

$$y[k] = \sum_{\bar{k}} h[k-\bar{k}].u[\bar{k}] \overset{D}{=} h[k]*u[k]$$

= `convolution sum‘

(=more convenient than Toeplitz matrix notation when considering (infinitely) long input and impulse response sequences

Discrete-Time Systems & Discrete Convolution [4]

Z-Transform of system h[k] and signals u[k],y[k]

Definition:

$$H(z)\overset{D}{=}\sum_k h[k].z^{-k} \qquad U(z)\overset{D}{=}\sum_k u[k].z^{-k} \qquad Y(z)\overset{D}{=}\sum_k y[k].z^{-k}$$

Input/output relation:

$$\underbrace{\begin{bmatrix}1 & z^{-1} & z^{-2} & z^{-3} & z^{-4} & z^{-5}\end{bmatrix}.\begin{bmatrix}y[0]\\y[1]\\y[2]\\y[3]\\y[4]\\y[5]\end{bmatrix}}_{Y(z)} = \underbrace{\begin{bmatrix}1 & z^{-1} & z^{-2} & z^{-3} & z^{-4} & z^{-5}\end{bmatrix}.\begin{bmatrix}h[0] & 0 & 0 & 0\\h[1] & h[0] & 0 & 0\\h[2] & h[1] & h[0] & 0\\0 & h[2] & h[1] & h[0]\\0 & 0 & h[2] & h[1]\\0 & 0 & 0 & h[2]\end{bmatrix}}_{H(z).\begin{bmatrix}1 & z-1 & z-2 & z-3\end{bmatrix}}.\begin{bmatrix}u[0]\\u[1]\\u[2]\\u[3]\end{bmatrix}$$

$$\Rightarrow Y(z) = H(z).U(z)$$ H(z) is `transfer function'

Matrix Formulation of Convolution

$$\mathbf{y} = \mathbf{Hx}$$

$$\begin{bmatrix}3\\8\\14\\20\\26\\14\\5\end{bmatrix} = \begin{bmatrix}1&2&3&0&0&0&0&0&0\\0&1&2&3&0&0&0&0&0\\0&0&1&2&3&0&0&0&0\\0&0&0&1&2&3&0&0&0\\0&0&0&0&1&2&3&0&0\\0&0&0&0&0&1&2&3&0\\0&0&0&0&0&0&1&2&3\end{bmatrix}.\begin{bmatrix}0\\0\\1\\2\\3\\4\\5\\0\\0\end{bmatrix}$$

Toeplitz Matrix

Graphical Understanding of Convolution

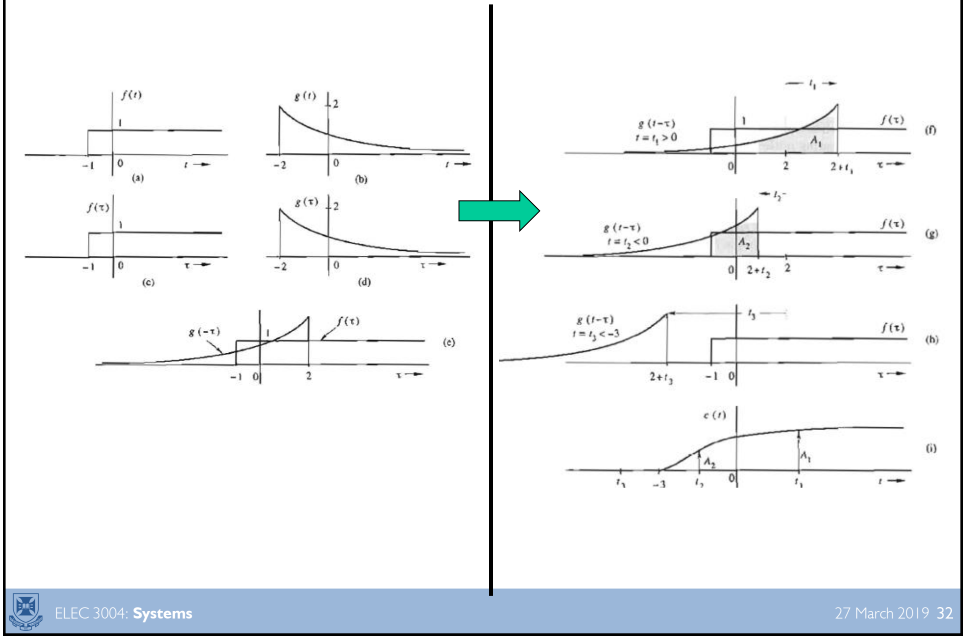
➔ For c(**τ**)= $(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau)\, d\tau$:

1. Keep the function ***f*** **(*τ*)** fixed
2. **Flip** (invert) the function ***g*(*τ*)** about the vertical axis (τ=0)
   = this is ***g*(-*τ*)**
3. **Shift** this frame (***g*(-*τ*)**) along τ (horizontal axis) by **t_0**.
   = this is ***g*(t_0 -*τ*)**

➔ For c(**t_0**):

4. **c(t_0)** = the area under the product of ***f*** **(*τ*) and *g*(t_0 -*τ*)**
5. Repeat this procedure, shifting the frame by different values (positive and negative) to obtain **c(t) for all values of t**.

Graphical Understanding of Convolution (Ex)



Another View

e.g. convolution

x(n) = 1 2 3 4 5
h(n) = 3 2 1

x(k)	0 0 1 2 3 4 5	0 0 1 2 3 4 5	0 0 1 2 3 4 5	
h(n,k)	1 2 3 0 0 0 0	0 1 2 3 0 0 0	0 0 1 2 3 0 0	h(n-k)
y(n,k)	3	2 6	1 4 9	
y(n)	3	8	14	

Notice the gain

Sum over all k

Convolution & Systems

- Convolution system with input u (u(t) = 0, t < 0) and output y:

$$y(t) = \int_0^t h(\tau)u(t-\tau)\,d\tau = \int_0^t h(t-\tau)u(\tau)\,d\tau$$

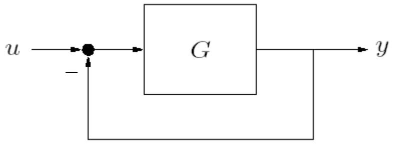
- abbreviated:

$$y = h * u$$

- in the frequency domain:

$$Y(s) = H(s)U(s)$$

Convolution & Feedback



- In the time domain:

$$y(t) = \int_0^t g(t-\tau)(u(\tau) - y(\tau))\, d\tau$$

- In the frequency domain:
  - Y=G(U-Y)

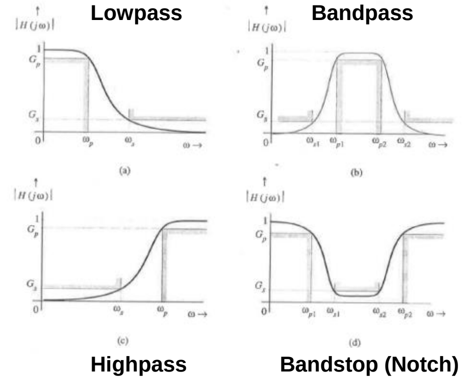
  ➔Y(s) = H(s)U(s)

$$H(s) = \frac{G(s)}{1+G(s)}$$

BREAK! ☺

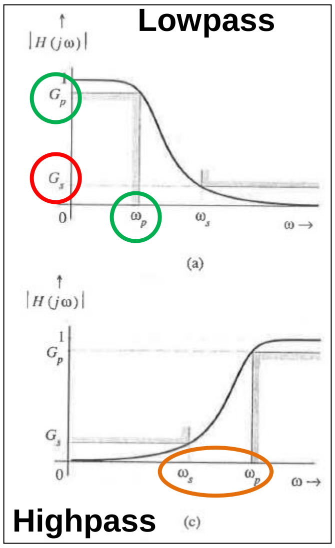
How to Beat the Noise?
Idea 1: (analog) **Filters!**

Filters



- *Frequency-shaping filters*: LTI systems that change the shape of the spectrum
- *Frequency-selective filters:* Systems that pass some frequencies undistorted and attenuate others

Filters



Specified Values:

- Gp = minimum passband gain

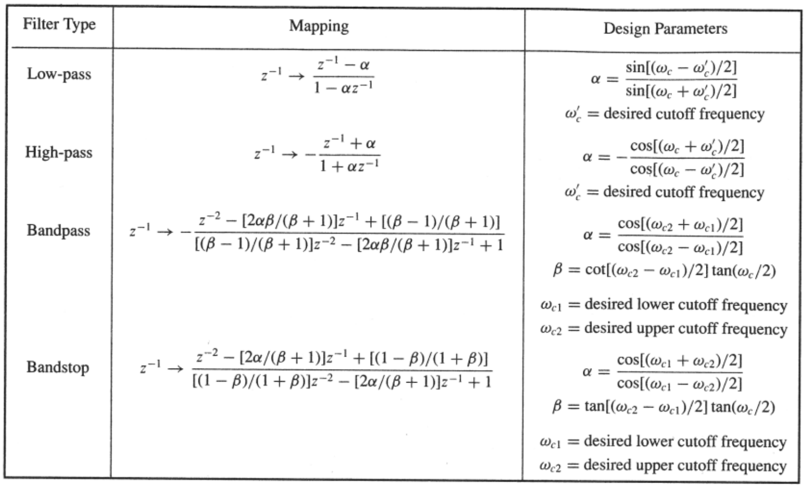
Typically:

$$G_p = \frac{1}{\sqrt{2}} = -3dB$$

- Gs = maximum stopband gain
  - **Low**, not zero (sorry!)
  - For realizable filters, the gain cannot be zero over a finite band (Paley-Wiener condition)
- Transition Band:

  transition from the passband to the stopband ➔ ωp≠ ωs

Filter Design & z-Transform



Filter Type	Mapping	Design Parameters
Low-pass	$z^{-1} \rightarrow \frac{z^{-1}-\alpha}{1-\alpha z^{-1}}$	$\alpha = \frac{\sin[(\omega_c - \omega_c')/2]}{\sin[(\omega_c + \omega_c')/2]}$ ω_c' = desired cutoff frequency
High-pass	$z^{-1} \rightarrow -\frac{z^{-1}+\alpha}{1+\alpha z^{-1}}$	$\alpha = -\frac{\cos[(\omega_c + \omega_c')/2]}{\cos[(\omega_c - \omega_c')/2]}$ ω_c' = desired cutoff frequency
Bandpass	$z^{-1} \rightarrow -\frac{z^{-2} - [2\alpha\beta/(\beta+1)]z^{-1} + [(\beta-1)/(\beta+1)]}{[(\beta-1)/(\beta+1)]z^{-2} - [2\alpha\beta/(\beta+1)]z^{-1} + 1}$	$\alpha = \frac{\cos[(\omega_{c2} + \omega_{c1})/2]}{\cos[(\omega_{c2} - \omega_{c1})/2]}$ $\beta = \cot[(\omega_{c2} - \omega_{c1})/2]\tan(\omega_c/2)$ ω_{c1} = desired lower cutoff frequency ω_{c2} = desired upper cutoff frequency
Bandstop	$z^{-1} \rightarrow \frac{z^{-2} - [2\alpha/(\beta+1)]z^{-1} + [(1-\beta)/(1+\beta)]}{[(1-\beta)/(1+\beta)]z^{-2} - [2\alpha/(\beta+1)]z^{-1} + 1}$	$\alpha = \frac{\cos[(\omega_{c1} + \omega_{c2})/2]}{\cos[(\omega_{c1} - \omega_{c2})/2]}$ $\beta = \tan[(\omega_{c2} - \omega_{c1})/2]\tan(\omega_c/2)$ ω_{c1} = desired lower cutoff frequency ω_{c2} = desired upper cutoff frequency

Butterworth Filters

- Butterworth: Smooth in the pass-band
- The amplitude response $|H(j\omega)|$ of an n^{th} order Butterworth low pass filter is given by:

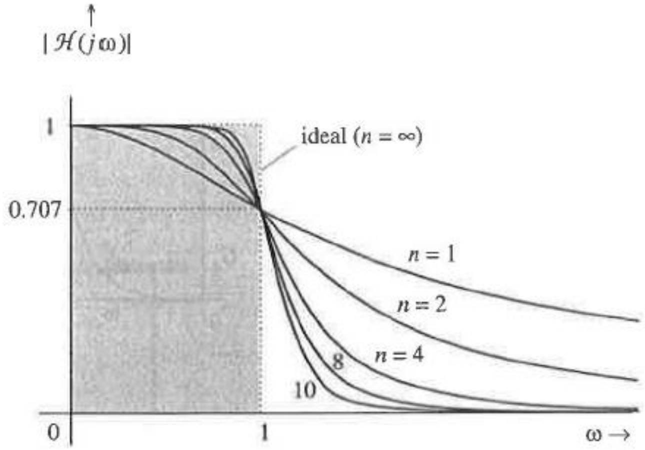
$$|H(j\omega)| = \frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_c}\right)^{2n}}}$$

- The normalized case (ω_c=1)

$$|\mathcal{H}(j\omega)| = \frac{1}{\sqrt{1+\omega^{2n}}} \Rightarrow \mathcal{H}(j\omega)\mathcal{H}(-j\omega) = |\mathcal{H}(j\omega)|^2 = \frac{1}{1+\omega^{2n}}$$

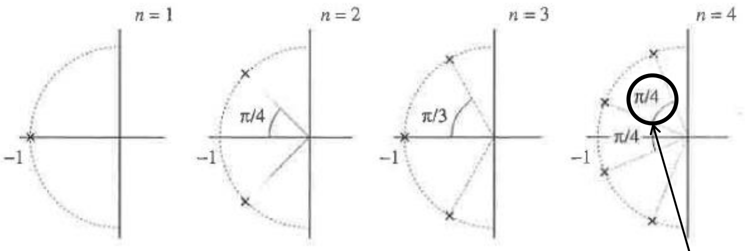
Recall that: $|H(j\omega)|^2 = H(j\omega)\,H(-j\omega)$

Butterworth Filters



Butterworth Filters of Increasing Order: Seeing this Using a Pole-Zero Diagram

- Increasing the order, increases the number of poles:



Angle between poles: $\frac{\pi}{n}$

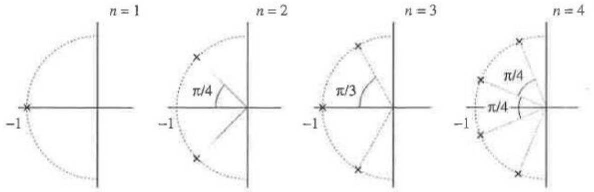
➔Odd orders (n=1,3,5…):

- Have a pole on the Real Axis

➔Even orders (n=2,4,6…):

- Have a pole on the off axis

Butterworth Filters: Pole-Zero Diagram



- Since H(s) is stable and causal, its poles must lie in the LHP
- Poles of -H(s) are those in the RHP
- Poles lie on the unit circle (for a normalized filter)

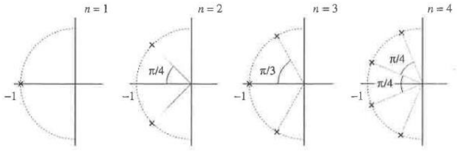
n is the order of the filter

➔$H\left(s\right)=\frac{1}{\left(s-s_1\right)\left(s-s_2\right)\ldots\left(s-s_n\right)}$

Where:

$$\begin{aligned} s_k &= e^{\frac{j\pi}{2n}(2k+n-1)} \\ &= \cos\frac{\pi}{2n}(2k+n-1)+j\sin\frac{\pi}{2n}(2k+n-1) \qquad k=1,\,2,\,3,\,\ldots,\,n \end{aligned}$$

Butterworth Filters: 4th Order Filter Example



- Plugging in for n=4, k=1,…4:

$$\mathcal{H}(s) = \frac{1}{(s+0.3827-j0.9239)(s+0.3827+j0.9239)(s+0.9239-j0.3827)(s+0.9239+j0.3827)}$$
$$= \frac{1}{(s^2+0.7654s+1)(s^2+1.8478s+1)}$$
$$= \frac{1}{s^4+2.6131s^3+3.4142s^2+2.6131s+1}$$

- We can generalize ➔ Butterworth Table

n	a_1	a_2	a_3	a_4	a_5
2	1.41421356				
3	2.00000000	2.00000000			
4	2.61312593	3.41421356	2.61312593		
5	3.23606798	5.23606798	5.23606798	3.23606798	
6	3.86370331	7.46410162	9.14162017	7.46410162	3.86370331

This is for 3dB bandwidth at ω_c=1

Butterworth Filters: Scaling Back (from Normalized)

- Start with Normalized equation & Table
- Replace ω with $\frac{\omega}{\omega_c}$ in the filter equation

- For example:
  for f_c=100Hz → ω_c=200π rad/sec

From the Butterworth table: for n=2, a_1=√2
Thus:

$$H(s) = \frac{1}{\left(\frac{s}{200\pi}\right)^2+\sqrt{2}\left(\frac{s}{200\pi}\right)+1}$$
$$= \frac{1}{s^2+200\pi\sqrt{2}+40{,}000\pi^2}$$

Butterworth: Determination of Filter Order

- Define $\mathbf{G_x}$ as the gain of a lowpass Butterworth filter at $\omega = \omega_x$
- Then:

$$\hat{G}_x = 20\log_{10}|H(j\omega_x)| = -10\log\left[1+\left(\frac{\omega_x}{\omega_c}\right)^{2n}\right]$$

And thus:

$$\hat{G}_p = -10\log\left[1+\left(\frac{\omega_p}{\omega_c}\right)^{2n}\right]$$

$$\hat{G}_s = -10\log\left[1+\left(\frac{\omega_s}{\omega_c}\right)^{2n}\right]$$

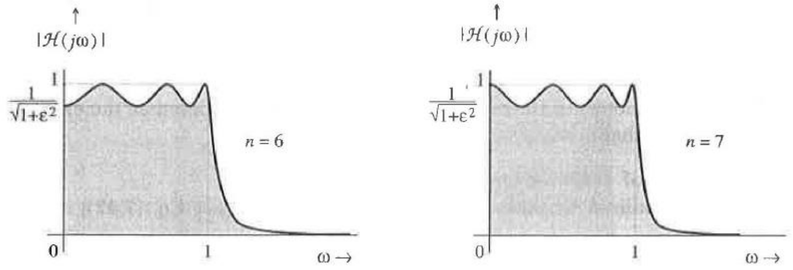
Or alternatively: $\omega_c = \frac{\omega_p}{\left[10^{-\hat{G}_p/10} - 1\right]^{1/2n}}$ & $\omega_c = \frac{\omega_s}{\left[10^{-\hat{G}_s/10} - 1\right]^{1/2n}}$

Solving for n gives:

$$n = \frac{\log\left[\left(10^{-\hat{G}_s/10} - 1\right) / \left(10^{-\hat{G}_p/10} - 1\right)\right]}{2\log(\omega_s/\omega_p)}$$

PS. See Lathi 4.10 (p. 453) for an example in MATLAB

Chebyshev Filters



- **equal-ripple:**
  Because all the ripples in the passband are of equal height
- If we reduce the ripple, the passband behaviour improves, but it does so at the cost of stopband behaviour

Chebyshev Filters

- Chebyshev Filters: Provide tighter transition bands (sharper cutoff) than the same-order Butterworth filter, but this is achieved at the expense of inferior passband behavior (rippling)

➔ For the lowpass (LP) case: at higher frequencies (in the stopband), the Chebyshev filter gain is smaller than the comparable Butterworth filter gain by about **6(*n* - 1) dB**

- The amplitude response of a normalized Chebyshev lowpass filter is:

$$|\mathcal{H}(j\omega)| = \frac{1}{\sqrt{1+\epsilon^2 {C_n}^2(\omega)}}$$

Where Cn(ω), the nth-order Chebyshev polynomial, is given by:

$$C_n(\omega) = \cos\left(n \cos^{-1}\omega\right)$$

$$C_n(\omega) = \cosh\left(n \cosh^{-1}\omega\right)$$

and where C_n is given by:

n	$C_n(\omega)$
0	1
1	ω
2	$2\omega^2 - 1$
3	$4\omega^3 - 3\omega$
4	$8\omega^4 - 8\omega^2 + 1$
5	$16\omega^5 - 20\omega^3 + 5\omega$
6	$32\omega^6 - 48\omega^4 + 18\omega^2 - 1$

Normalized Chebyshev Properties

- It's normalized: The passband is 0<ω<1
- **Amplitude response**: has **ripples** in the passband and is **smooth** (monotonic) in the stopband
- **Number of ripples**: there is a total of ***n*** maxima and minima over the passband 0<ω<1
- $C_n^2(0) = \begin{cases} 0, & n: odd \\ 1, & n: even \end{cases}$ ➔ $|H(0)| = \begin{cases} 1, & n: odd \\ \frac{1}{\sqrt{1+\epsilon^2}}, & n: even \end{cases}$
- ϵ: ripple height → $r = \sqrt{1+\epsilon^2}$
- The Amplitude at ω=1: $\frac{1}{r} = \frac{1}{\sqrt{1+\epsilon^2}}$
- For Chebyshev filters, the ripple ***r*** dB takes the place of G_p

Determination of Filter Order

- The gain is given by: $\hat{G} = -10\log\left[1+\epsilon^2 {C_n}^2(\omega)\right]$

Thus, the gain at ω_s is: $\epsilon^2 {C_n}^2(\omega_s) = 10^{-\hat{G}_s/10} - 1$

- Solving:

$$n = \frac{1}{\cosh^{-1}(\omega_s)} \cosh^{-1}\left[\frac{10^{-\hat{G}_s/10}-1}{10^{\hat{r}/10}-1}\right]^{1/2}$$

- General Case:

$$n = \frac{1}{\cosh^{-1}(\omega_s/\omega_p)} \cosh^{-1}\left[\frac{10^{-\hat{G}_s/10}-1}{10^{\hat{r}/10}-1}\right]^{1/2}$$

Chebyshev Pole Zero Diagram

- Whereas Butterworth poles lie on a semi-circle,
  The poles of an n^{th}-order normalized Chebyshev filter lie on a **semiellipse** of the major and minor semiaxes:

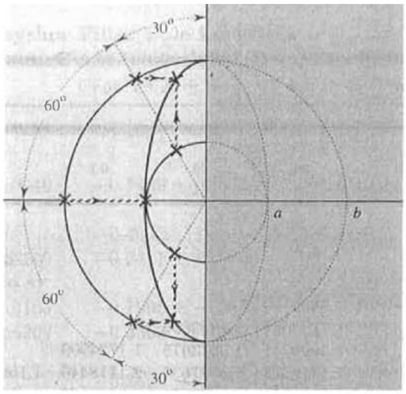
$$a = \sinh\left(\frac{1}{n}\sinh^{-1}\left(\frac{1}{\epsilon}\right)\right) \quad \& \quad b = \cosh\left(\frac{1}{n}\sinh^{-1}\left(\frac{1}{\epsilon}\right)\right)$$

And the poles are at the locations:

$$H(s) = \frac{1}{(s-s_1)(s-s_2)\ldots(s-s_n)}$$

$$s_k = -\sin\left[\frac{(2k-1)\pi}{2n}\right]\sinh x + j\cos\left[\frac{(2k-1)\pi}{2n}\right]\cosh x,\ k = 1,\ldots,n$$

Ex: Chebyshev Pole Zero Diagram for **n=3**



Procedure:

1. Draw two semicircles of radii **a** and **b** (from the previous slide).
2. Draw radial lines along the corresponding Butterworth angles (π/n) and locate the nth-order Butterworth poles (shown by crosses) on the two circles.
3. The location of the kth Chebyshev pole is the intersection of the horizontal projection and the vertical projection from the corresponding kth Butterworth poles on the outer and the inner circle, respectively.

Chebyshev Values / Table

$$\mathcal{H}(s) = \frac{K_n}{C'_n(s)} = \frac{K_n}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0}$$

$$K_n = \begin{cases} a_0 & n \text{ odd} \\ \dfrac{a_0}{\sqrt{1+\epsilon^2}} = \dfrac{a_0}{10^{\hat{r}/20}} & n \text{ even} \end{cases}$$

n	a_0	a_1	a_2	a_3
1	1.9652267			
2	1.1025103	1.0977343		
3	0.4913067	1.2384092	0.9883412	
4	0.2756276	0.7426194	1.4539248	0.9528114

1 db ripple
($\hat{r} = 1$)

Other Filter Types:
Chebyshev Type II = Inverse Chebyshev Filters

- Chebyshev filters passband has ripples and the stopband is smooth.
- **Instead:** this has **passband** have **smooth** response and **ripples** in the stopband.

➔Exhibits maximally flat passband response and equi-ripple stopband

➔ `Cheby2` **in MATLAB**

$$|\mathcal{H}(\omega)|^2 = 1 - |\mathcal{H}_C(1/\omega)|^2 = \frac{\epsilon^2 C_n^2(1/\omega)}{1+\epsilon^2 C_n^2(1/\omega)}$$

Where: H_c is the Chebyshev filter system from before

- Passband behavior, especially for small ω, is **better** than Chebyshev
- **Smallest transition band** of the 3 filters (Butter, Cheby, Cheby2)
- Less time-delay (or phase loss) than that of the **Chebyshev**
- Both needs the **same order *n*** to meet a set of specifications.
- **$$$** (or number of elements):
  Cheby < Inverse Chebyshev < Butterworth (of the same **performance** [not order])

Other Filter Types:
Elliptic Filters (or Cauer) Filters

- Allow **ripple** in **both** the passband and the stopband,
  ➔ we can achieve **tighter** transition band

$$|\mathcal{H}(j\omega)| = \frac{1}{\sqrt{1+\epsilon^2 {R_n}^2(\omega)}}$$

Where: R_n is the nth-order Chebyshev rational function determined from a given ripple spec.
ϵ controls the ripple
$Gp = \frac{1}{\sqrt{1+\varepsilon^2}}$

- Most efficient (η)
  - the **largest ratio** of the passband gain to stopband gain
  - **or** for a given ratio of passband to stopband gain, it requires the **smallest transition band**

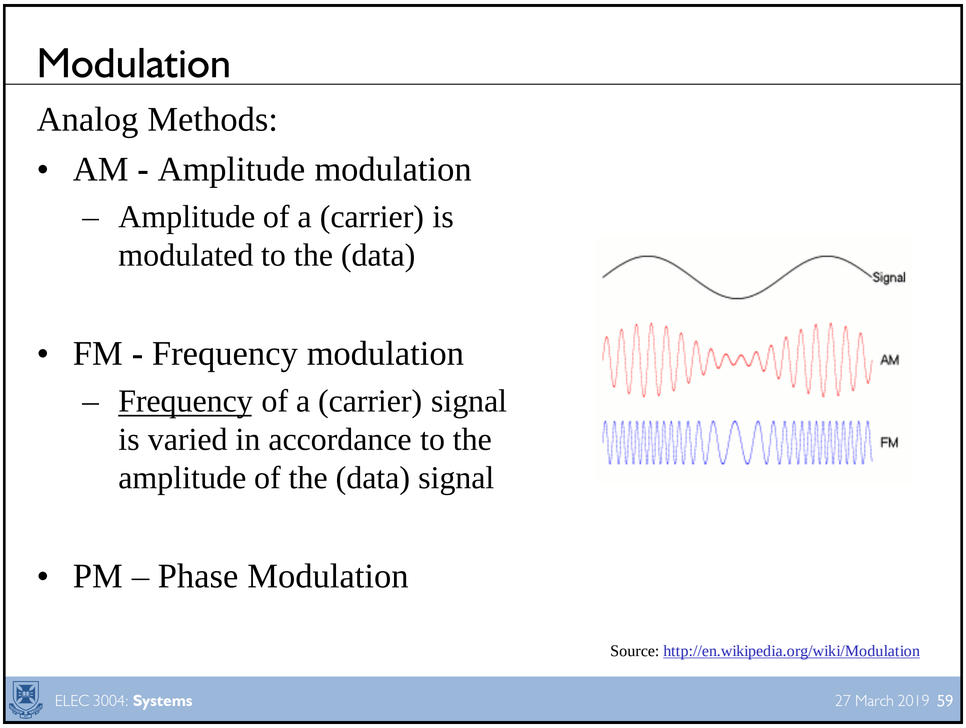
➔ **in MATLAB:** `ellipord` followed by `ellip`

In Summary

Filter Type	Passband Ripple	Stopband Ripple	Transition Band	MATLAB Design Command
Butterworth	No	No	Loose	`butter`
Chebyshev	Yes	No	Tight	`cheby`
Chebyshev Type II (Inverse Chebyshev)	No	Yes	Tight	`cheby2`
Eliptic	Yes	Yes	Tightest	`ellip`

How to Beat the Noise?

Idea 2: **Modulation**

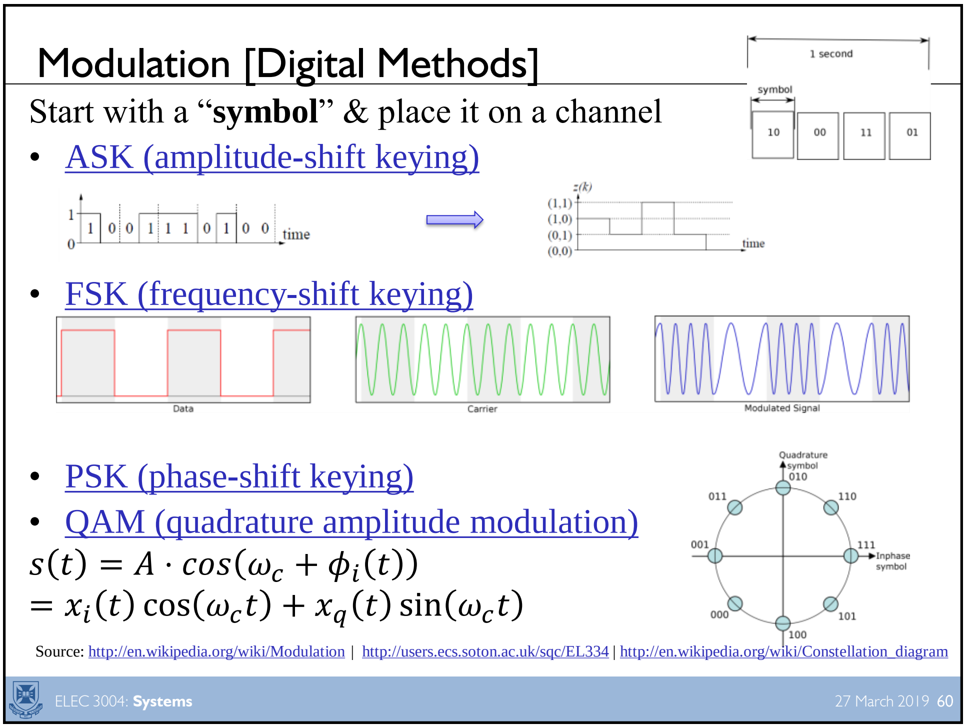


Modulation

Analog Methods:

- AM - Amplitude modulation
  - Amplitude of a (carrier) is modulated to the (data)
- FM - Frequency modulation
  - Frequency of a (carrier) signal is varied in accordance to the amplitude of the (data) signal
- PM – Phase Modulation

Source: http://en.wikipedia.org/wiki/Modulation



Modulation [Digital Methods]

Start with a “**symbol**” & place it on a channel

- ASK (amplitude-shift keying)
- FSK (frequency-shift keying)
- PSK (phase-shift keying)
- QAM (quadrature amplitude modulation)

$s(t) = A \cdot cos(\omega_c + \phi_i(t))$

$= x_i(t)\cos(\omega_c t) + x_q(t)\sin(\omega_c t)$

Source: http://en.wikipedia.org/wiki/Modulation | http://users.ecs.soton.ac.uk/sqc/EL334 | http://en.wikipedia.org/wiki/Constellation_diagram

Modulation [Example – V.32bis Modem]

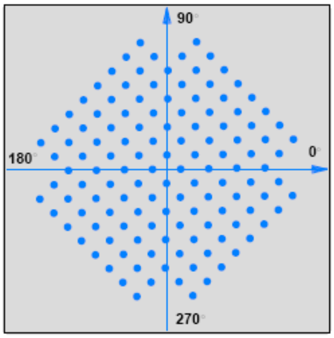


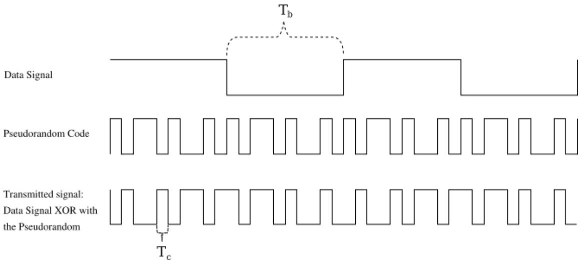
Figure 10.13 Illustration of the QAM constellation for a V.32bis dialup modem.

Source: Computer Networks and Internets, 5e, Douglas E. Comer

---

Multiple Access (Channel Access Method)

- Send multiple signals on 1 to N channel(s)
  - Frequency-division multiple access (FDMA)
  - Time-division multiple access (TDMA)
  - Code division multiple access (CDMA)
  - Space division multiple access (SDMA)
- CDMA:
  - Start with a pseudorandom code (the noise doesn’t know your code)



Source: http://en.wikipedia.org/wiki/Code_division_multiple_access

Next Time…

- **Digital Filters**

- Review:
  - Chapter 10 of Lathi

- A signal has many signals ☺
  [Unless it's bandlimited. Then there is the one ω]

$\mathcal{Z}$ Transform (**Extra** Review^3! – Another Way to L◉ ◉k at it)

Flashback: Euler's approximation (L7, p.26)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \lim_{\delta t \to 0} \frac{x(t+\delta t) - x(t)}{\delta t} \quad \Longrightarrow \quad \frac{\mathrm{d}x}{\mathrm{d}t} \approx \frac{x_{k+1} - x_k}{T}$$

For small enough T, this can be used to approximate a continuous controller by a discrete controller:

1. Laplace transform ⟶ differential equation

   e.g.

$$D(s) = \frac{U(s)}{E(s)} = \frac{K(s+a)}{(s+b)} \quad \Longrightarrow \quad \frac{\mathrm{d}u}{\mathrm{d}t} + b\,u = K\Big(\frac{de}{dt} + a\,e\Big)$$

2. Differential equation ⟶ difference equation

   e.g.

$$\frac{u_{k+1} - u_k}{T} + b\,u_k = K\Big(\frac{e_{k+1} - e_k}{T} + a\,e_k\Big)$$

$$\begin{aligned} \Longrightarrow u_{k+1} &= (1 - bT)u_k + K e_{k+1} + K(aT - 1)e_k \\ &= -a_1 u_k + b_0 e_{k+1} + b_1 e_k \end{aligned}$$

Discrete transfer function

Compare the discrete system time domain model:

$$\begin{aligned} u_k &= -a_1 u_{k-1} - \cdots - a_n u_{k-n} + b_0 e_k + \cdots + b_m e_{k-m} \\ &= -\sum_{i=1}^{n} a_i u_{k-i} + \sum_{j=0}^{m} b_j e_{k-j} \end{aligned}$$

recurrence equation

with the continuous system model:

$$u(t) = -a_1 \frac{\mathrm{d}u}{\mathrm{d}t} - \cdots a_n \frac{\mathrm{d}^n u}{\mathrm{d}t^n} + b_0 e + b_1 \frac{\mathrm{d}e}{\mathrm{d}t} + \cdots + b_m \frac{\mathrm{d}^m e}{\mathrm{d}t^m}$$

differential equation

↓ Laplace transform ↓

$$U(s) = -a_1 s U(s) - \cdots - a_n s^n U(s) + b_0 E(s) + b_1 s E(s) + \cdots + b_m s^m E(s)$$

$$\therefore \frac{U(s)}{E(s)} = D(s) = \frac{b_0 + b_1 s + b_2 s^2 + \cdots + b_m s^m}{1 + a_1 s + a_2 s^2 + \cdots + a_n s^n}$$

transfer function

Can we define a transfer function for the discrete system?

Discrete transfer function [2]

Suppose $u_k = u(kT)$ has transform $U'(s)$ …

…then how can we represent u_{k-1}, u_{k-2}, etc.?

- ⋆ If $x(t) \xrightarrow{L.T.} X(s)$, then $x(t-T) \xrightarrow{L.T.} e^{-sT}X(s)$, so

$$\begin{aligned} u_k &\longrightarrow U'(s) \\ u_{k-1} &\longrightarrow e^{-sT}U'(s) \\ u_{k-2} &\longrightarrow e^{-2sT}U'(s) \end{aligned}$$

etc

- ⋆ Define the discrete frequency domain operator

$$\boxed{z = e^{sT}}$$

then

$$\begin{aligned} u_k &\longrightarrow U(z) \\ u_{k-1} &\longrightarrow z^{-1}U(z) \\ u_{k-2} &\longrightarrow z^{-2}U(z) \end{aligned}$$

etc

Discrete transfer function [3]

Comparison

- ⋆ system representations:

Continuous	Discrete
$u(t) = -a_1\frac{\mathrm{d}u}{\mathrm{d}t} - a_2\frac{\mathrm{d}^2u}{\mathrm{d}t^2} - \cdots + b_0e + b_1\frac{\mathrm{d}e}{\mathrm{d}t} + \cdots$	$u_k = -a_1u_{k-1} - a_2u_{k-2} - \cdots + b_0e_k + b_1e_{k-1} + \cdots$

- ⋆ operators:

Continuous	Discrete
$\frac{\mathrm{d}u}{\mathrm{d}t} \longrightarrow s\,U(s)$	$u_{k-1} \longrightarrow z^{-1}\,U(z)$
differential	delay

Discrete transfer function [4]

Apply the transformation to the linear recurrence equation:

$$\begin{aligned} u_k = & -a_1 u_{k-1} - a_2 u_{k-2} - \cdots - a_n u_{k-n} \\ & + b_0 e_k + b_1 e_{k-1} + \cdots b_n e_{k-n} \end{aligned}$$

$\downarrow$ transform $\downarrow$

$$\begin{aligned} U(z) = & -a_1 z^{-1} U(z) - a_2 z^{-2} U(z) - \cdots a_n z^{-n} U(z) \\ & + b_1 E(z) + b_2 z^{-1} E(z) + \cdots + b_n z^{-n} E(z) \end{aligned}$$

This gives the z domain transfer function:

$$\frac{U(z)}{E(z)} = D(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots b_m z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$$

Rationalize by multiplying top and bottom by z^n

$$\boxed{D(z) = \frac{b_0 z^n + b_1 z^{n-1} + b_2 z^{n-2} + \cdots b_m z^{n-m}}{z^n + a_1 z^{n-1} + a_2 z^{n-2} + \cdots + a_n}}$$

Discrete transfer function [5]

Analysis tools based on s domain transfer functions:

e.g.

Pole & zero locations $\longrightarrow$ damping, natural frequency, settling time & overshoot

Frequency response $\longrightarrow$ gain & phase margins

. . . also apply to z domain transfer functions

Poles and zeros of $D(z)$:

$$D(z) = b_0 \frac{\Pi_{i=1}^{m}(z - z_j)}{\Pi_{i=1}^{n}(z - p_i)} z^{n-m}$$

zeros: z_i

poles: p_j

- z_i & p_i are real or in complex conjugate pairs
- n poles, n zeros, with $n - m$ zeros at $z = 0$
- at least as many poles as zeros

Properties of the the z-transform

- Some useful properties
  - **Delay by n samples**: $\mathcal{Z}\{f(k-n)\} = z^{-n}F(z)$
  - **Linear**: $\mathcal{Z}\{af(k) + bg(k)\} = \mathrm{a}F(z) + bG(z)$
  - **Convolution**: $\mathcal{Z}\{f(k) * g(k)\} = F(z)G(z)$

So, all those block diagram manipulation tools you know and love will work just the same!

The z-Transform

So far we have considered z^{-1} as a delay operator acting on sequences

But to find $E(z)$ from $e(kT)$ we need to define the z-transform:

$$\boxed{\begin{aligned} E(z) &= \mathcal{Z}\{e(kT)\} = \mathcal{Z}\{e_k\} \\ &= \sum_{k=0}^{\infty} e(kT)z^{-k} = \sum_{k=0}^{\infty} e_k z^{-k} \end{aligned}}$$

Note:

- ⋆ Single-sided z-transform — all variables are assumed to be zero for $k < 0$ [Franklin uses a different definition]
- ⋆ Strictly speaking, we should give bounds on $|z|$ for convergence, e.g.

$$r_0 < |z| < R_0$$

where r_0, R_0 depend on $e(kT)$
(these bounds are only needed in order to invert $E(z)$ by integration)

The z-Transform

Example – z-transform of a decaying exponential

Sample $x(t) = Ce^{-at}\mathcal{U}(t)$: ($\mathcal{U}(t)$ = unit step at $t = 0$)

$$x_k = Ce^{-akT}, \quad k \geq 0$$

and take the z-transform:

$$X(z) = \sum_{k=0}^{\infty} x_k z^{-k} = C\sum_{k=0}^{\infty} e^{-akT} z^{-k} = C\sum_{k=0}^{\infty} (e^{-aT} z^{-1})^k$$

this is a geometric series which converges if $|z| > e^{-aT}$:

$$X(z) = \frac{C}{1 - e^{-aT}z^{-1}} = \frac{Cz}{z - e^{-aT}}$$

$$\Downarrow$$

z-transform of exponental = rational polynomial (like Laplace)

The z-Transform

Effect of delay:

$$\mathcal{Z}\{e(kT - T)\} = z^{-1}E(z) \quad \text{where} \quad E(z) = \mathcal{Z}\{e(kT)\}$$

Example – z-transform of a delayed sequence

Take a finite length sequence

$$e_0, e_1, e_2, e_3, e_4, \ldots = 1.5, 1.6, 1.7, 0, 0, \ldots$$

introduce a delay of one sampling interval:

$$f_0, f_1, f_2, f_3, f_4, \ldots = 0, 1.5, 1.6, 1.7, 0, \ldots$$

take z-transforms:

$$E(z) = \sum_{k=0}^{\infty} e_k z^{-k} = 1.5 + 1.6z^{-1} + 1.7z^{-2}$$

$$F(z) = \sum_{k=0}^{\infty} f_k z^{-k} = 1.5z^{-1} + 1.6z^{-2} + 1.7z^{-3}$$

$$= z^{-1}E(z)$$

The z-Transform

Example – z-transform of a delayed exponential

Delay $x(t) = Ce^{-at}\mathcal{U}(t)$ by a time T:

$$y(t) = x(t-T) \quad \Longrightarrow \quad y(t) = Ce^{-a(t-T)}\mathcal{U}(t-T)$$

sample $y(t)$ with sample interval T:

$$y_k = \begin{cases} 0 & k = 0 \\ Ce^{-a(k-1)T} & k = 1, 2, \dots \end{cases}$$

z-transform:

$$Y(z) = \sum_{k=0}^{\infty} y_k\, z^{-k} = \sum_{k=1}^{\infty} Ce^{-a(k-1)T} z^{-k}$$

$$= Cz^{-1}\sum_{j=0}^{\infty}(e^{-aT}z^{-1})^j = \frac{C}{z - e^{-aT}}$$

Comparing $X(z)$ and $Y(z)$:

$$X(z) = \frac{Cz}{z - e^{-aT}} \quad \Longrightarrow \quad Y(z) = z^{-1}X(z)$$

The z-Transform

- In practice, you'll use look-up tables or computer tools (ie. Matlab) to find the z-transform of your functions

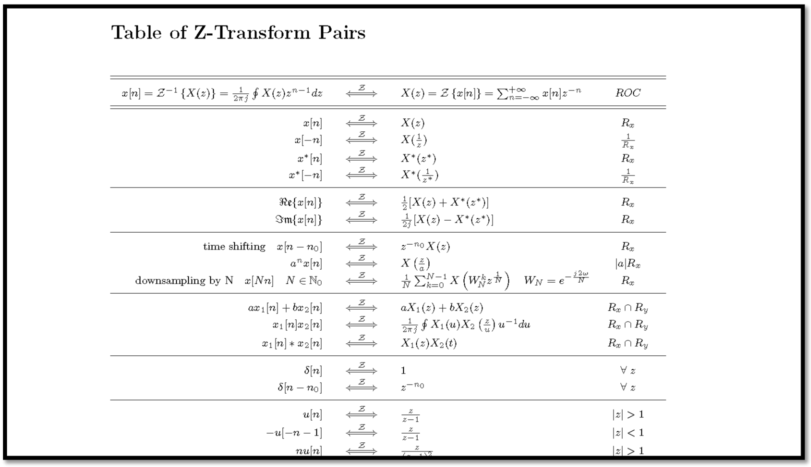
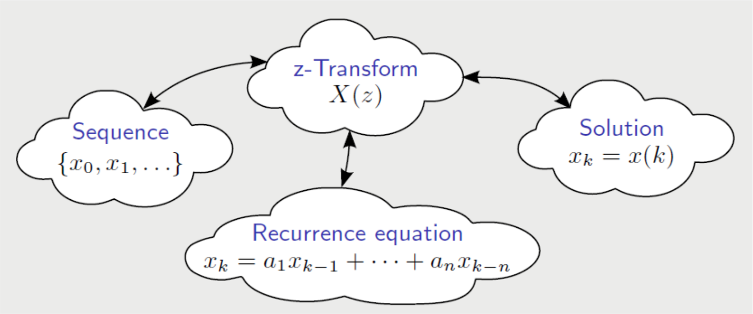


Table of Z-Transform Pairs

$x[n] = \mathcal{Z}^{-1}\{X(z)\} = \frac{1}{2\pi j}\oint X(z)z^{n-1}dz$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{+\infty} x[n]z^{-n}$	ROC
$x[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$X(z)$	R_x
$x[-n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$X(\frac{1}{z})$	$\frac{1}{R_x}$
$x^*[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$X^*(z^*)$	R_x
$x^*[-n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$X^*(\frac{1}{z^*})$	$\frac{1}{R_x}$
$\mathfrak{Re}\{x[n]\}$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$\frac{1}{2}[X(z) + X^*(z^*)]$	R_x
$\mathfrak{Im}\{x[n]\}$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$\frac{1}{2j}[X(z) - X^*(z^*)]$	R_x
time shifting $x[n-n_0]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$z^{-n_0}X(z)$	R_x
$a^n x[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$X\left(\frac{z}{a}\right)$	$\|a\|R_x$
downsampling by N $x[Nn]$ $N \in \mathbb{N}_0$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$\frac{1}{N}\sum_{k=0}^{N-1} X\left(W_N^k z^{\frac{1}{N}}\right) \quad W_N = e^{-\frac{j2\omega}{N}}$	R_x
$ax_1[n] + bx_2[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$aX_1(z) + bX_2(z)$	$R_x \cap R_y$
$x_1[n]x_2[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$\frac{1}{2\pi j}\oint X_1(u)X_2\left(\frac{z}{u}\right)u^{-1}du$	$R_x \cap R_y$
$x_1[n] * x_2[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$X_1(z)X_2(t)$	$R_x \cap R_y$
$\delta[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	1	$\forall\ z$
$\delta[n-n_0]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	z^{-n_0}	$\forall\ z$
$u[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$\frac{z}{z-1}$	$\|z\| > 1$
$-u[-n-1]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$\frac{z}{z-1}$	$\|z\| < 1$
$nu[n]$	$\overset{\mathcal{Z}}{\Longleftrightarrow}$	$\frac{z}{(z-1)^2}$	$\|z\| > 1$

The z-Transform

To summarise:



z-Transform
$X(z)$

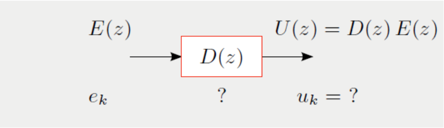
Sequence
$\{x_0, x_1, \ldots\}$

Solution
$x_k = x(k)$

Recurrence equation
$x_k = a_1 x_{k-1} + \cdots + a_n x_{k-n}$

$X(z)$ provides an easy way to convert between sequences, recurrence equations and their closed-form solutions.

Pulse Response



$E(z)$ $\quad U(z) = D(z)\,E(z)$

$D(z)$

e_k $\quad ?$ $\quad u_k = \ ?$

For continuous systems: $D(s) = \mathcal{L}\{d(t)\}$ $\qquad d(t) =$ plant impulse response

What is the equivalent property for the discrete transfer function $D(z)$?

- ⋆ Let $e(kT) =$ discrete unit pulse:

$$e_k = \delta_k = \begin{cases} 1 & k = 0 \\ 0 & k = 1, 2, \ldots \end{cases} \quad \Longleftrightarrow \quad E(z) = \sum_{k=0}^{\infty} e_k z^{-k} = 1$$

- ⋆ then

$$\boxed{U(z) = D(z)E(z) = D(z)}$$

i.e. $\quad D(z) = \mathcal{Z}\{d(kT)\} =$ z-transform of the plant pulse response

Pulse Response [2]

Example – The recurrence equation $u_k = u_{k-1} + \frac{T}{2}(e_k + e_{k-1})$

has transfer function $D(z) = \frac{U(z)}{E(z)} = \frac{T}{2}\frac{(z+1)}{(z-1)}$

Check this by finding the pulse response and taking its z-transform

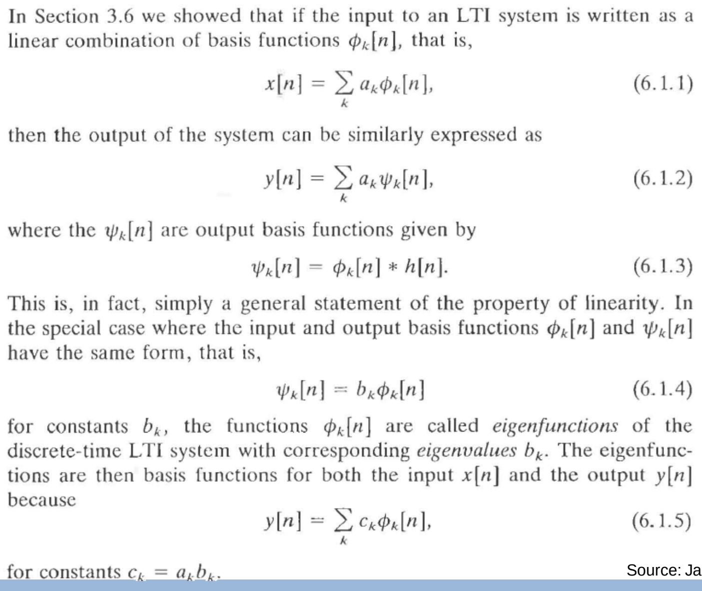
$e_k = \delta_k$ gives

k	u_{k-1}	e_k	e_{k-1}	u_k
0	0	1	0	$T/2$
1	$T/2$	0	1	T
2	T	0	0	T
3	T	0	0	T
$\vdots$				$\vdots$

i.e. $u_k = T/2, T, T, \ldots$

so
$$U(z) = \sum_{k=0}^{\infty} Tz^{-k} - T/2 = \frac{T}{1-z^{-1}} - \frac{T}{2} = \frac{T}{2}\frac{(z+1)}{(z-1)}$$

∴ Eigenfunctions of Discrete-Time LTI Systems



In Section 3.6 we showed that if the input to an LTI system is written as a linear combination of basis functions $\phi_k[n]$, that is,

$$x[n] = \sum_k a_k \phi_k[n], \qquad (6.1.1)$$

then the output of the system can be similarly expressed as

$$y[n] = \sum_k a_k \psi_k[n], \qquad (6.1.2)$$

where the $\psi_k[n]$ are output basis functions given by

$$\psi_k[n] = \phi_k[n] * h[n]. \qquad (6.1.3)$$

This is, in fact, simply a general statement of the property of linearity. In the special case where the input and output basis functions $\phi_k[n]$ and $\psi_k[n]$ have the same form, that is,

$$\psi_k[n] = b_k \phi_k[n] \qquad (6.1.4)$$

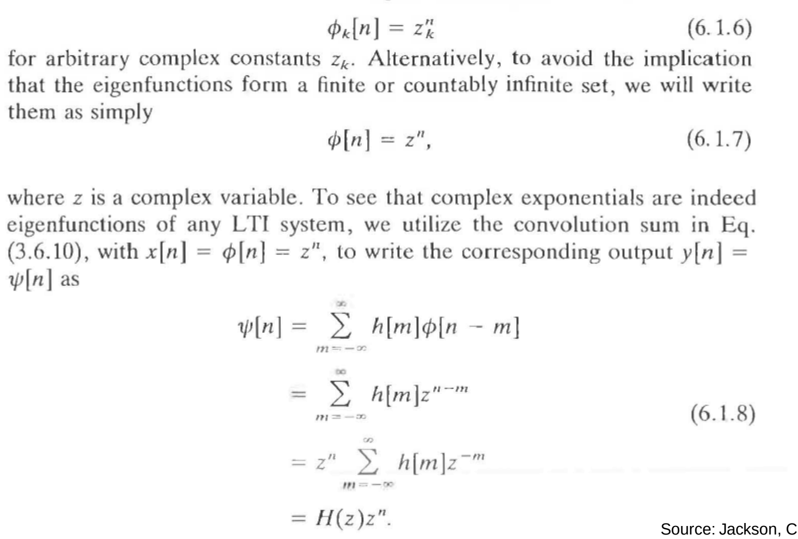
for constants b_k, the functions $\phi_k[n]$ are called *eigenfunctions* of the discrete-time LTI system with corresponding *eigenvalues* b_k. The eigenfunctions are then basis functions for both the input $x[n]$ and the output $y[n]$ because

$$y[n] = \sum_k c_k \phi_k[n], \qquad (6.1.5)$$

for constants $c_k = a_k b_k$.

Source: Jackson, Chap. 6

∴ Eigenfunctions of Discrete-Time LTI Systems



In analogy with the continuous-time case, the eigenfunctions of discrete-time LTI systems are the complex exponentials

$$\phi_k[n] = z_k^n \qquad (6.1.6)$$

for arbitrary complex constants z_k. Alternatively, to avoid the implication that the eigenfunctions form a finite or countably infinite set, we will write them as simply

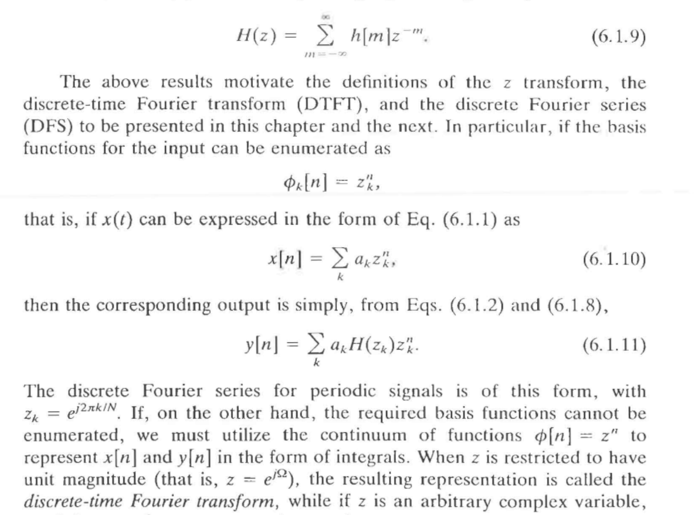
$$\phi[n] = z^n, \qquad (6.1.7)$$

where z is a complex variable. To see that complex exponentials are indeed eigenfunctions of any LTI system, we utilize the convolution sum in Eq. (3.6.10), with $x[n] = \phi[n] = z^n$, to write the corresponding output $y[n] = \psi[n]$ as

$$\begin{aligned} \psi[n] &= \sum_{m=-\infty}^{\infty} h[m]\phi[n-m] \\ &= \sum_{m=-\infty}^{\infty} h[m]z^{n-m} \qquad (6.1.8) \\ &= z^n \sum_{m=-\infty}^{\infty} h[m]z^{-m} \\ &= H(z)z^n. \end{aligned}$$

Source: Jackson, Chap. 6

∴ Eigenfunctions of Discrete-Time LTI Systems



Hence the complex exponential z^n is an eigenfunction of the system for any value of z, and $H(z)$ is the corresponding eigenvalue given by

$$H(z) = \sum_{m=-\infty}^{\infty} h[m]z^{-m}. \qquad (6.1.9)$$

The above results motivate the definitions of the z transform, the discrete-time Fourier transform (DTFT), and the discrete Fourier series (DFS) to be presented in this chapter and the next. In particular, if the basis functions for the input can be enumerated as

$$\phi_k[n] = z_k^n,$$

that is, if $x(t)$ can be expressed in the form of Eq. (6.1.1) as

$$x[n] = \sum_k a_k z_k^n, \qquad (6.1.10)$$

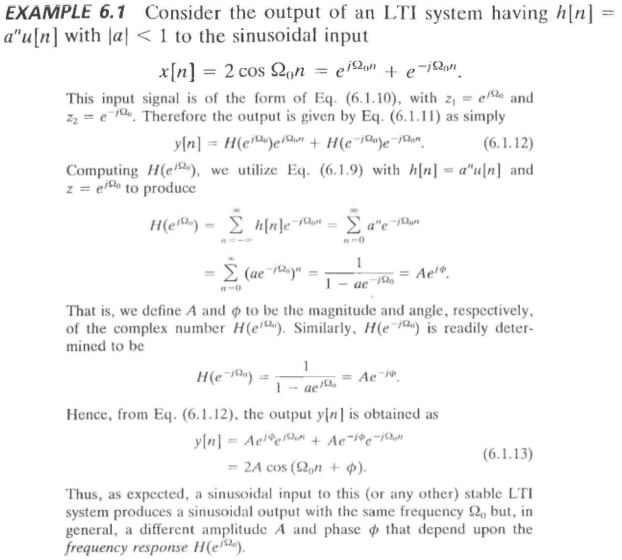
then the corresponding output is simply, from Eqs. (6.1.2) and (6.1.8),

$$y[n] = \sum_k a_k H(z_k) z_k^n. \qquad (6.1.11)$$

The discrete Fourier series for periodic signals is of this form, with $z_k = e^{j2\pi k/N}$. If, on the other hand, the required basis functions cannot be enumerated, we must utilize the continuum of functions $\phi[n] = z^n$ to represent $x[n]$ and $y[n]$ in the form of integrals. When z is restricted to have unit magnitude (that is, $z = e^{j\Omega}$), the resulting representation is called the *discrete-time Fourier transform,* while if z is an arbitrary complex variable, the full *z-transform* representation results.

Source: Jackson, Chap. 6

∴ Eigenfunctions of Discrete-Time LTI Systems



EXAMPLE 6.1 Consider the output of an LTI system having $h[n] = a^n u[n]$ with $|a| < 1$ to the sinusoidal input

$$x[n] = 2\cos\Omega_0 n = e^{j\Omega_0 n} + e^{-j\Omega_0 n}.$$

This input signal is of the form of Eq. (6.1.10), with $z_1 = e^{j\Omega_0}$ and $z_2 = e^{-j\Omega_0}$. Therefore the output is given by Eq. (6.1.11) as simply

$$y[n] = H(e^{j\Omega_0})e^{j\Omega_0 n} + H(e^{-j\Omega_0})e^{-j\Omega_0 n}. \qquad (6.1.12)$$

Computing $H(e^{j\Omega_0})$, we utilize Eq. (6.1.9) with $h[n] = a^n u[n]$ and $z = e^{j\Omega_0}$ to produce

$$\begin{aligned} H(e^{j\Omega_0}) &= \sum_{n=-\infty}^{\infty} h[n]e^{-j\Omega_0 n} = \sum_{n=0}^{\infty} a^n e^{-j\Omega_0 n} \\ &= \sum_{n=0}^{\infty} (ae^{-j\Omega_0})^n = \frac{1}{1 - ae^{-j\Omega_0}} = Ae^{j\phi}. \end{aligned}$$

That is, we define A and ϕ to be the magnitude and angle, respectively, of the complex number $H(e^{j\Omega_0})$. Similarly, $H(e^{-j\Omega_0})$ is readily determined to be

$$H(e^{-j\Omega_0}) = \frac{1}{1 - ae^{j\Omega_0}} = Ae^{-j\phi}.$$

Hence, from Eq. (6.1.12), the output $y[n]$ is obtained as

$$\begin{aligned} y[n] &= Ae^{j\phi}e^{j\Omega_0 n} + Ae^{-j\phi}e^{-j\Omega_0 n} \\ &= 2A\cos(\Omega_0 n + \phi). \end{aligned} \qquad (6.1.13)$$

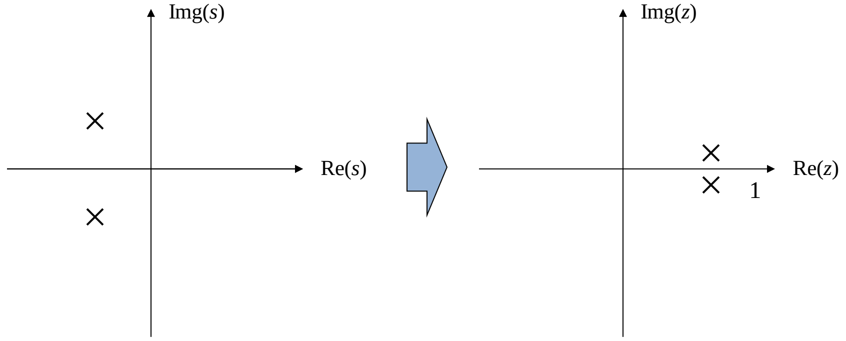
Thus, as expected, a sinusoidal input to this (or any other) stable LTI system produces a sinusoidal output with the same frequency Ω_0 but, in general, a different amplitude A and phase ϕ that depend upon the *frequency response* $H(e^{j\Omega_0})$.

Source: Jackson, Chap. 6

The z-Plane

z-domain poles and zeros can be plotted just like *s*-domain poles and zeros (of the $\mathcal{L}$):

- S-plane:
- $\boldsymbol{z = e^{sT}}$ Plane



  - λ – Plane
  - γ – Plane

The z-Plane & Stability

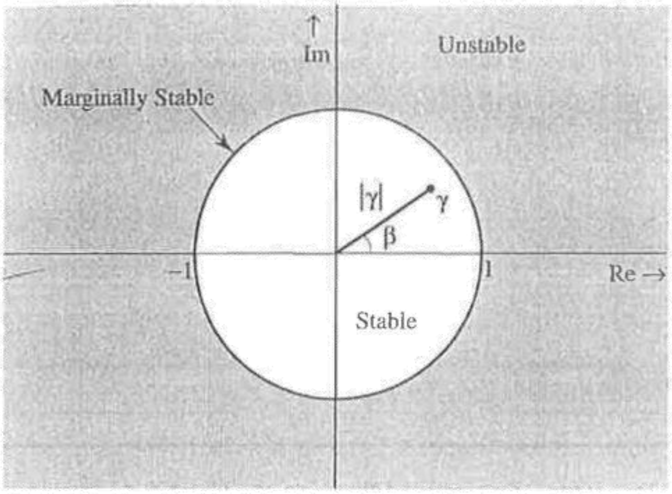
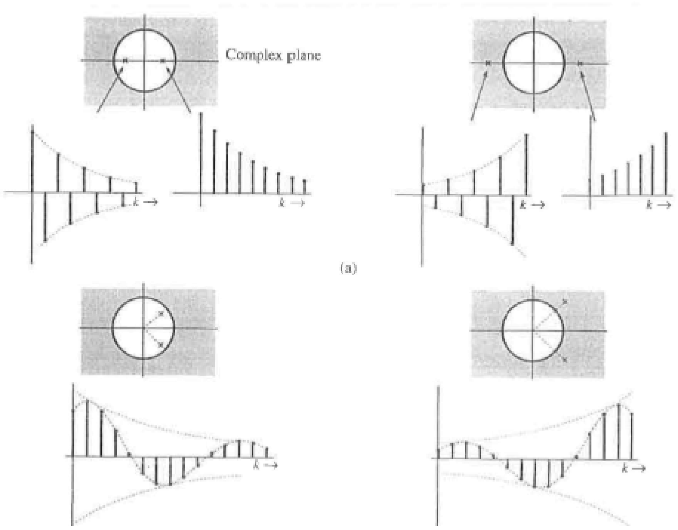


Fig. 9.6 Characteristic roots location and system stability.

The z-Plane & Stability



DT Causality & BIBO Stability

- Causality:

$$h[n] = 0,\ n < 0$$

$$\rightarrow y[n] = \sum_{k=0}^{\infty} h[k]x[n-k] \quad \text{or} \quad \Rightarrow y[n] = \sum_{k=-\infty}^{n} x[k]h[n-k]$$

- Input is Causal if: $x[n] = 0,\ n < 0$

- Then output is Causal:

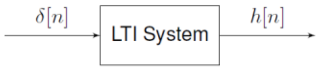
$$y[n] = \sum_{k=0}^{n} h[k]x[n-k] = \sum_{k=0}^{n} x[k]h[n-k]$$

- And, DT LTI is BIBO stable if:

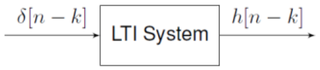
$$\sum_{k=-\infty}^{\infty} |h[k]| < \infty$$

Impulse Response (Graphically)

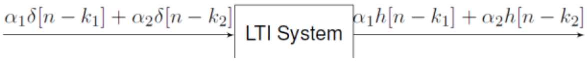
Let's define the *impulse response*, $h[n]$, as the result of applying an LTI system to the unit impulse:

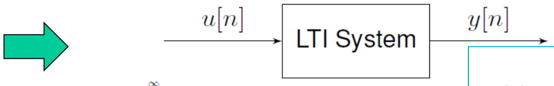


By time invariance, we know



And by linearity, we know





$$u[n] = \sum_{k=-\infty}^{\infty} u[k]\delta[n-k]$$

$$\boxed{y[n] = \sum_{k=-\infty}^{\infty} u[k]h[n-k]}$$

∞ matrix × ∞ vector?

Linear Difference Equations

$$u_k = f(e_0, \ldots, e_k; u_0, \ldots, u_{k-1}).$$

$$u_k = -a_1 u_{k-1} - a_2 u_{k-2} - \cdots - a_n u_{k-n} + b_0 e_k + b_1 e_{k-1} + \cdots + b_m e_{k-m}.$$

$$\begin{aligned} \nabla u_k &= u_k - u_{k-1} && \text{(first difference)}, \\ \nabla^2 u_k &= \nabla u_k - \nabla u_{k-1} && \text{(second difference)}, \\ \nabla^n u_k &= \nabla^{n-1} u_k - \nabla^{n-1} u_{k-1} && \text{(nth difference)}. \end{aligned}$$

$$\begin{aligned} u_k &= u_k, \\ u_{k-1} &= u_k - \nabla u_k, \\ u_{k-2} &= u_k - 2\nabla u_k + \nabla^2 u_k. \end{aligned}$$

$$a_2 \nabla^2 u_k - (a_1 + 2a_2)\nabla u_k + (a_2 + a_1 + 1)u_k = b_0 e_k.$$

Assume a form of the solution

z^k :

- k: “order of difference”
- k: delay

$$Az^k = Az^{k-1} + Az^{k-2}.$$

$$1 = z^{-1} + z^{-2}$$

$$z^2 = z + 1.$$