SFWR ENG 2MX3 Summary

Author: Kemal Ahmed
Instructor: Dr. MvM
Date: Winter 2014

Math objects made using MathType; graphs made using Winplot.

CTRL-F (?) to find locations which need to fixed

Bonus: Order of writing symbols, such as exponents: digits-i-variables

Systems

There are 3 ways of representing a system:

- Difference Equation
- State Space Equations (ABCD), and
- Impulse Response

Deltas

2 types of deltas (δ):

Kronecker Delta: discrete time domain, $\delta(n)=\begin{cases}1, & n=0\\ 0, & \text{else}\end{cases}, \int\delta(0)=1$

Dirac Delta: continuous time domain, $\delta(t)=\infty$, $\delta(\text{else})=0$, $\int\delta(0)=1$

Discrete manipulates functions of n.
Continuous manipulates functions of t.

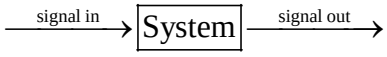
y = output
Causal system: system = 0 before impulse

Only use cosine curves; represent sine curves with a phase shift.

If you have a y(n−2) state, but not a y(n−1) state, include a y(n−1) state anyways.

SISO: single-input, single-output
MIMO system: Multiple Inputs Multiple Outputs.

All the systems we examine are "zero-state". Initial state is zero, s(0) = 0



Discrete Systems

If there's no x in the equation, there must be a mistake because there's no input.

- More precise, more theoretical
- Discrete input signals
- Easier to process because it's digital

Continuous System

- e.g. x(t) = 2cos(t)
- Analog

Difference equation

Definition

The first sum is manipulation of input signals and the second part is manipulation of output signals.

$y(n)=\sum_{k=0}^{N}\alpha_k x(n-k)+\sum_{k-1}^{M}\beta_k y(n-k)$, which is interchangeable with:

$$H(\omega)=\frac{\sum_{k=0}^{\infty}\alpha_k e^{-i\omega k}}{1-\sum_{k=1}^{\infty}\beta_k e^{-i\omega k}}$$

State Space Equations

ABCD

The **state space equations method** is also known as **ABCD Method**. This is because you represent your states by the four matrices, **A**, **B**, **C**, and **D**. There are 2 state space equations:

- Next state equation
- Output

Given *k* states, **next state equation**:

$$\mathbf{S}(n+1)=\begin{bmatrix}S_1(n+1)\\S_2(n+1)\\\vdots\\S_k(n+1)\end{bmatrix}=\mathbf{A}\begin{bmatrix}S_1(n)\\S_2(n)\\\vdots\\S_k(n)\end{bmatrix}+\mathbf{B}x(n)$$

Notice how the left side of the equation is the next state of each state? Expanding the matrices would show you how many of each state and the input each state represents.

The output equation is just re-writing the given equation in terms of the states you determined and your inputs:

$$y(n) = \mathbf{C}\begin{bmatrix} S_1(n+1) \\ S_2(n+1) \\ \vdots \\ S_k(n+1) \end{bmatrix} + \mathbf{D}x(n)$$

The state space equation method use matrices of states to represent any equation in a concise way.

States are any part of the given equation that is not y(n) or x(n). That is because the number of x(n)'s (i.e. inputs) are represented by the **B** matrix. On the other hand, the y(n) is not considered a state, but rather, a collection of states. Thus, if your next state is y(n), expand it by equating it to the initial equation of y(n).

e.g.)

$y(n) = x(n) - y(n-2)$

Recall that x(n) is already going to be represented by the **B** matrix, so it cannot be a state. However, the $y(n-2)$ *can* be represented by a state. Let's call it S_1:

$S_1(n) = y(n-2)$

Now, the next state will be:

$S_1(n+1) = y(n+1-2) = y(n-1)$

Hey! There's no matrix for y(n − 1). There are only matrices for y(n) and x(n)! So we should make another state. Think of it like a temporary variable where the value of the variable is passed down, until S_1.

Ok, so let's call this state S_2:

$S_2(n) = y(n-1)$

Let's look at the next states of both S_1 and S_2:

$S_1(n) = y(n-2)$ $\qquad$ $S_1(n+1) = y(n-1)$

$S_2(n) = y(n-1)$ $\qquad$ $S_2(n+1) = y(n)$

Remember that we're supposed to represent the next states by a combination of the previous states? So let's change those next states. Recall: when you have y(n), you expand it into the states it represents.

$S_1(n+1) = y(n-1) = S_2(n)$

$S_2(n+1) = y(n) = x(n) - S_1(n)$

For the next part, you need to understand matrix expansion. For now, I'll re-write each next-state equation individually before combining it to show what's actually going on:

$$S_1(n+1) = 0S_1(n) + 1S_2(n) + 0x(n)$$
$$= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} S_1(n) \\ S_2(n) \end{bmatrix} + [0]x(n)$$
$$S_2(n+1) = -1S_1(n) + 0S_2(n) + 1x(n)$$
$$= \begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} S_1(n) \\ S_2(n) \end{bmatrix} + [1]x(n)$$

Now put those two together to get:

$$\mathbf{S}(n+1) = \underbrace{\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} S_1(n) \\ S_2(n) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\mathbf{B}} x(n)$$

The output equation is found the same way:

$$y(n) = -1S_1(n) + 0S_2(n) + x(n)$$
$$= \underbrace{\begin{bmatrix} -1 & 0 \end{bmatrix}}_{\mathbf{C}} \begin{bmatrix} S_1(n) \\ S_2(n) \end{bmatrix} + \underbrace{[1]}_{\mathbf{D}} x(n)$$

You can use these matrices for determining impulse response.

Linear Time Invariant (LTI) System: you may flip order of system and delay (?)

Converting to Difference Equation

Euler's equations convert states to the difference equation.

$$e^{i\theta} = \cos\theta + i\sin\theta$$
$$\cos\theta = \tfrac{1}{2}\left(e^{i\theta} + e^{-i\theta}\right)$$
$$\sin\theta = \tfrac{1}{2i}\left(e^{i\theta} - e^{-i\theta}\right)$$

Compound Interest

An application of the state space equation other than signals is for determining compound interest.

n: billing period
α: interest
x: deposits

$$y(n) = y(n-1) + \alpha y(n-1) + x(n)$$
$$= (1+\alpha)y(n-1) + x(n)$$

Similar to an impulse, let's model someone putting money in a bank and leaving it there forever:

$$x(n) = \begin{cases} A, & n = 0 \\ 0, & \text{else} \end{cases}$$

$y(0) = A$

$y(1) = (1+\alpha)A$

$y(2) = (1+\alpha)^2 A$

So you can interpolate: $y(n) = (1+\alpha)^n A$

Impulse Response

Impulses can be modeled by the Kronecker delta function.

FIR

Finite Impulse Response (FIR) system: A system that has an impulse response that has a finite duration (is zero at a finite time). One way of seeing if it is finite is if it has no y's in it,

e.g.) State Space Table

y(n) = x(n) + x(n−2)

How we model systems in this course is using a **state space table**. This table analyzes what the output of the given system is when you are given an impulse. The prof usually leaves the labels blank, but this time, I'll leave them in there. Next time, however, they won't be there:

Time	0	1	2	3	4
Input [x(n)]	1	0	0	0	0
Output [y(n)]	1	0	1	0	0

From this table, you can tell that it's an FIR system, since the output is 0 after t = 2.

IIR

Infinite Impulse Response (IIR) system. A system that has an impulse response that has an infinite duration (continues to respond indefinitely). IIR systems generally have *y* on both sides of the equation

e.g.)

y(n) = x(n) + 0.5y(n−1)

	1	0	0	0	0
	1	½	¼	⅛	1/16

Notice how the system doesn't end? This is why it is called an IIR system. Also, notice how the labels are gone? Get used to it and imagine them there every time you make a table like this.

Impulse Response

Impulse Response is notated as h(*n*). It is the output, y(n), when the input, x(n) is a combination of shifts of the Kronecker delta function.

Remember ABCD representation? Impulse response can be represented by inputting the ABCD matrices:

$$h(n)=\begin{cases}0, & n<0\\ \mathbf{D}, & n=0\\ \mathbf{CA}^{n-1}\mathbf{B}, & n>0\end{cases}$$

Convolution Sum

$$y=h*x$$

$$y(n)=\sum_{k=-\infty}^{\infty}h(n-k)x(k)=\sum_{k=-\infty}^{\infty}h(k)x(n-k)$$

If something appears to have the convolution equation, it will be convolution and thus, you may use its properties.

Impulse response to difference equation

Sometimes your impulse response may be specified. There are 2 ways of converting your impulse response into the difference equation:

- Convolution Sum method
- Table method

e.g.)

We're going to use the same example to demonstrate both methods

$$h(n)=\begin{cases}0, & \text{else}\\ 1, & n=0\\ 2, & n=1\\ 3, & n=2\end{cases}$$

$x(n) = \delta(n) + \delta(n-2)$

Convolution Equation method

$$\boxed{y(n)=\sum_{i=-\infty}^{\infty}h(n-i)x(i)}$$

Let's plug in some values of *n*:

$$y(0)=\sum_{i=0}^{\infty}h(0-i)x(i)$$

However, since h(anything < 0) = 0, we only get output when i = 0:

$$y(0)=h(0)\cdot x(0)$$
$$=1\cdot 1=1$$

$$y(1)=\sum_{i=0}^{\infty}h\underbrace{(1-i)}_{0,1}x(i)$$
$$=h(1-0)\cdot x(0)+h(1-1)x(1)$$
$$=(2\cdot 1)+(1\cdot 0)=2$$
$$y(2)=\sum_{i=0}^{\infty}h(2-i)x(i)$$
$$=h(2-0)x(0)+h(2-1)x(1)+h(2-2)x(2)$$
$$=(3\cdot 1)+(2\cdot 0)+(1\cdot 1)$$
$$=3+0+1=4$$
$$y(3)=\sum_{i=0}^{\infty}h(3-i)x(i)$$
$$=h(3-0)x(0)+h(3-1)x(1)+h(3-2)x(2)+h(3-3)x(3)$$
$$=(0\cdot 1)+(3\cdot 0)+(2\cdot 1)+(1\cdot 0)$$
$$=2$$

We're supposed to stop at this point for some reason.

Table method

x(n) →	1	0	1	0	0	0
h(0)	1	2	3	0	0	0
h(1)	0	0	0	0	0	0
h(2)	0	0	1	2	3	0
y(n)	1	2	4	2	3	0

We're supposed to stop at this point for some reason.

Kemal's Method

Someone else probably thought of this before me, but there's a faster way of directly converting your impulse response to your difference equation:

$$h(n)=\begin{cases}\alpha_0, & n=0\\ \vdots & \\ \alpha_N, & n=N\end{cases}$$
$$y(n)=\sum_{k=0}^{N}\alpha_k x(n-k)$$

Frequency/Time Domain

Upper-case variable: frequency domain
Lower-case variable: time domain

	Discrete	Continuous
Period	$p = \frac{\text{samples}}{\text{cycle}}$ (must be a natural number)	$p = \frac{\text{seconds}}{\text{cycle}}$
Frequency	$f = \frac{1}{p} = \frac{\text{cycles}}{\text{sample}}$	$f = \frac{1}{p} = \frac{\text{cycles}}{\text{second}} = \text{Hz}$
Normalized Frequency	$\omega = 2\pi f = \frac{\text{radians}}{\text{sample}}$	$\omega = 2\pi f = \frac{\text{radians}}{\text{second}}$
Sampling Frequency	$f_s = \frac{\text{samples}}{\text{second}}$	
Pitch	$\text{Pitch} = f_s \cdot f_{\text{discrete}} = \frac{\text{samples}}{\text{second}} \cdot \frac{\text{cycles}}{\text{sample}} = \frac{\text{cycles}}{\text{second}} = \text{Hz}$	

Remember that *upper-case* variables (frequency-domain) are *multiplied*: $Y(\omega) = H(\omega) \times X(\omega)$, and *lower-case* variables (time-domain) are *convoluted*: $y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} x(t-\tau)h(\tau)\,\mathrm{d}\tau$

$$x(n) = k(n + kp)$$
$$= \cos\left(\tfrac{\pi}{4}n\right) \leftarrow p = 8 \qquad (?)$$
$$= \tfrac{\pi}{4}n = k2\pi$$

e.g.

If you wanted to normalize $x = \frac{3\pi}{4}n$ (should be divisible by 2)

$p = 8$

$$\omega = \frac{2\pi}{p} = \frac{\pi}{4}$$

Frequency Response

$$x(n) = e^{i\omega n}$$

$$y(n) = H(\omega)e^{i\omega n}$$

e.g. y(n) + y(n−1) = 2x(n) − 5x(n−2)

$$H(\omega)e^{i\omega n} + H(\omega)e^{i\omega(n-1)} = 2e^{i\omega n} - 5e^{i\omega(n-2)}$$

$$H(\omega)\left(e^{i\omega n} + e^{i\omega n}e^{-i\omega}\right) = 2e^{i\omega n} - 5e^{i\omega n}e^{-2i\omega}$$

$$H(\omega)\left(1 + e^{-i\omega}\right) = 2 - 5e^{-2i\omega}$$

Factor everything out to isolate H(ω):

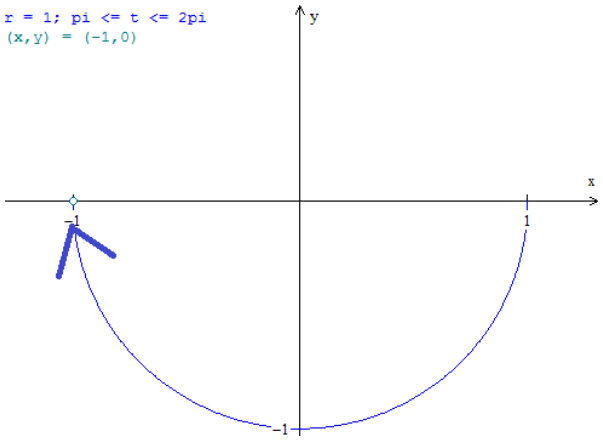
$$H(\omega)=\frac{2-5e^{-2i\omega}}{1+e^{-i\omega}}$$

Formula: $$\boxed{H(\omega)=\frac{\sum_{k=0}^{\infty}\alpha_k e^{-i\omega k}}{1-\sum_{k=1}^{\infty}\beta_k e^{-i\omega k}}}$$

$$H(\omega)=\frac{2-5e^{-2i\omega}}{1+e^{-i\omega}}$$

Table: (the argument will always be negative because you can't look ahead of time)

ω	H(ω)	‖H(ω)‖	$\measuredangle H(\omega)$
0	$-3/2$	3/2	$-\pi$
π/2	$\frac{7+7i}{2}$	$\sqrt{3.5^2+3.5i^2}\approx 5$	$\frac{-7\pi}{4}$
π			



$$\boxed{y(n)=|H(\omega)|[\cos/\sin](\omega n+\measuredangle H(\omega))}$$

$$y(n)=\tfrac{3}{2}\cos(-\pi)=\tfrac{-3}{2}+5\cos\left(\tfrac{\pi}{2}n-\tfrac{7\pi}{4}+\tfrac{\pi}{4}\right)$$

$$y(n)=\tfrac{-3}{2}+\tfrac{7}{\sqrt{2}}\cos\left(\tfrac{\pi}{2}n-\tfrac{3\pi}{2}\right)$$

Filter a signal

$\frac{1}{i}=-i$

Sawtooth signals are periodic, so you have to use the Fourier Series on it.

$$x(t)=\frac{a_0}{2}+\sum_{n=1}^{\infty}a_n\cos\left(\frac{n\pi t}{1}\right)+b_n\sin\left(\frac{n\pi t}{1}\right)$$

State Space

this vs ABCD (?)

S(n+1) = A·s(n) + B·x(n)
y(n) = C·s(n) + D·x(n)

e.g. y(n) = x(n) + 2x(n−2)
let $S_1(n) = x(n-2)$

$S_1(n) = x(n-2)$ $\quad$ $S_1(n+1) = x(n-1) = S_2(n)$

$S_2(n) = x(n-1)$ $\quad$ $S_2(n+1) = x(n)$

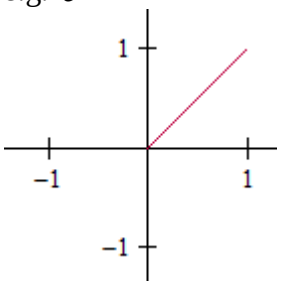
<insert matrices>

Complex Numbers

Draw a graph where the x values represent the real dimension and the y values represent the complex dimension.

Refer to Euler's equations a lot (in the difference equations section).

e.g. $e^{i\frac{\pi}{4}}$



Magnitude = $\sqrt{1^2+1^2} = \sqrt{2}$

Divide by the magnitude on the unit circle. Thus, the point is: $\left(\frac{1}{2}, \frac{i}{2}\right)$

Know your special triangles.

$\measuredangle$ = argument or degree

Now, put it into the form: $x(t) = e^{i\omega t}$

$y(t) = H(\omega)e^{i\omega t}$

$\|H(\omega)\|$ = gain

$\measuredangle H(\omega)$ = phase shift

Impulse Response to Frequency Response

	FS (Fourier Series)	FT (Fourier Transform)

DT (Discrete Time)	$X_k = \frac{1}{p}\sum_{m=0}^{p-1} x(m)e^{-imk\omega_0}$ $x(n) = \sum_{k=0}^{p-1} X_k e^{ik\omega_0 n}$	$X(\omega) = \sum_{m=-\infty}^{\infty} x(m)e^{-i\omega m}$ $x(n) = \frac{1}{2\pi}\int_0^{2\pi} X(\omega)e^{i\omega n}\mathrm{d}\omega$ $\mathrm{DTFT}\left\{e^{i\omega_0 n}\right\} = 2\pi\delta(\omega-\omega_0)$
CT (Continuous Time)	$X_n = \frac{1}{p}\int_0^{p} x(t)e^{-in\omega_0 t}\mathrm{d}t$ $x(t) = \sum_{n=-\infty}^{\infty} X_n e^{in\omega_0 t}$	$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}\mathrm{d}t$ $x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{i\omega t}\mathrm{d}\omega$ Summary: $\mathrm{CTFT}\left(e^{i\omega_0 t}\right) = 2\pi\delta(\omega-\omega_0)$

Forward transform: time → frequency domain (look for negative exponents on e)
Transform^{-1}: frequency → time domain (look for positive exponents on e)

For discrete frequency only: $\omega = \omega_0 \cdot k$
For continuous frequency only: $\omega = \omega_0 \cdot t$

If *h* is a function of time, then CTFT; if not, DTFT. i.e. h(t), do CTFT; h(n), do DTFT
The impulse response, $h(n)$, of a system is the $y(n)$ when $x(n)$ is an impulse: $\delta(n)$.

Step Response

Removal of frequency component at a point means H(point) = 0, so you just pretend the value is a root at that time. i.e. $\tilde{H}(\omega) = \prod\left(e^{-i\omega} - e^{-i\{\text{points}\}}\right)$

DTFT versus DFT

The DFT is actually the most popular transform, because there's an efficient algorithm for it called the **Fast Fourier Transform** (FFT). It's the same as the DTFS, but with different scaling. See the difference:

DTFS	DFT/FFT
$X_k = \frac{1}{p}\sum_{m=0}^{p-1} x(m)e^{-imk\omega_0}$ $x(n) = \sum_{k=0}^{p-1} X_k e^{ik\omega_0 n}$	$X_k = \sum_{m=0}^{p-1} x(m)e^{-imk\omega_0}$ $x(n) = \frac{1}{p}\sum_{k=0}^{p-1} X_k e^{ik\omega_0 n}$

Because the [forward] DFT doesn't divide out the number of points, *p*, using a higher sampling rate will make the DFT larger for the same input. This is why it's actually better to scale the DFT output by 1/p anyways when looking at it afterwards.

Fourier Series

Midterm 1-Question 3:

X_k =

Given . Your coefficient of k should be a multiple of 2π, so make p=4.
calculate your x(n) values before your __ values.

x(n) = 2sin(π/2 n), so you get 0, 2, 0, and −2

X_k's are the fourier coefficients.;jm

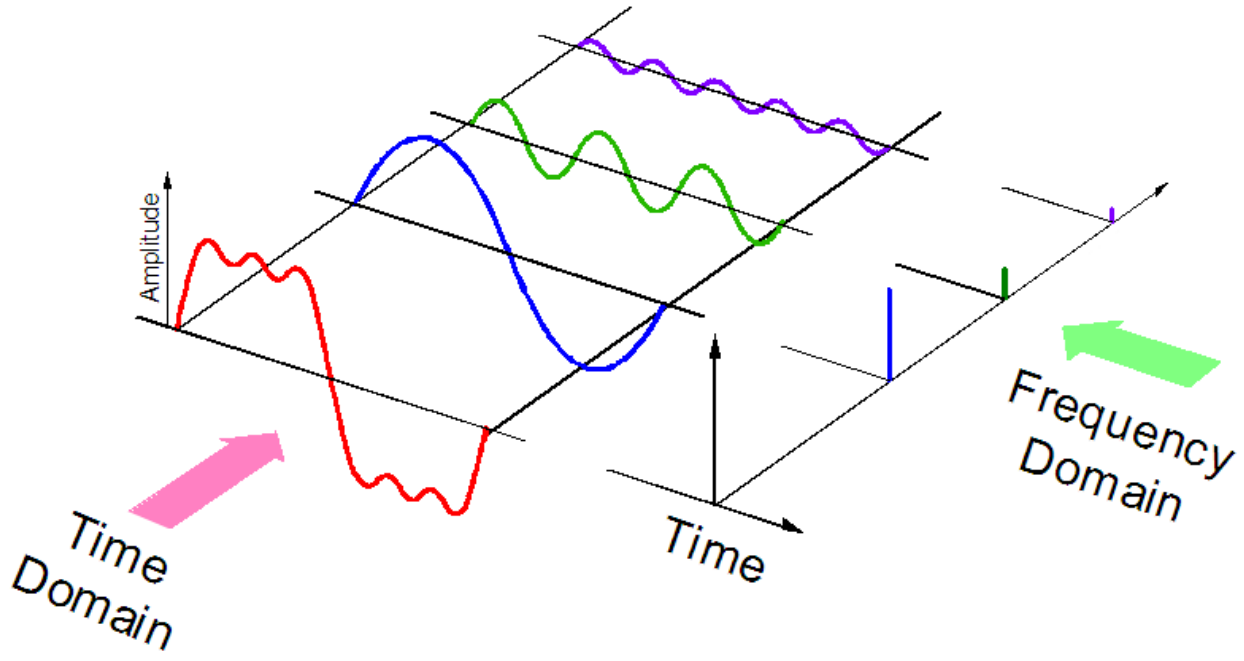
Alternative method:
X(t) = cos(t) + sin(2t) => convert to complex exponentials: , so
k = 1,−1 =>

Question 1, midterm 2
X(n) = 1, 1, 0, 0 →
N = p, so

Fourier Transform
Test 2, Question 3
CTFT:
Recall that integral of a dirac is 1.

Time and Frequency Domain

Gif demonstration of what the frequency domain is.



Z and Laplace Transform

If you aren't given a value for x, assume it's a delta function.

e.g. Practice 3, question 3

$\ddot{y} + 4\dot{y} - 2y = x$

1) Laplace:

$s^2Y(s) + 4sY(s) - 2Y(s) = X(s)$

2) Isolate for Y:

$$Y(s) = \frac{1}{s^2 + 4s - 2} X(s)$$

Poles of the transfer function are the coefficients of X(s) function.

Assume x(t) = δ(t), so X(s) = 1

3)

$$H(s) = \frac{1}{s^2 + 4s - 2} \cdot 1$$

4) Quadratic formula OR factor to find the poles

$$= \frac{1}{\left(s - \left(-2 + \sqrt{6}\right)\right)\left(s - \left(-2 - \sqrt{6}\right)\right)}$$

This region of convergence **doesn't** include the imaginary axis.

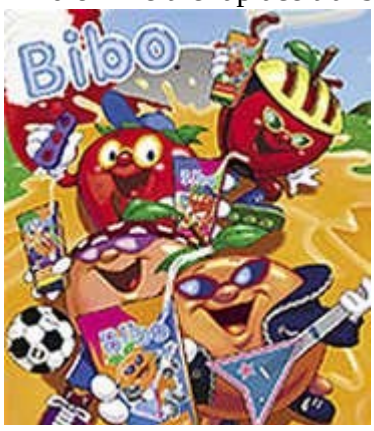
The region of convergence of *H*(*s*) must include the imaginary axis in order for the system to be stable.

H(*s*) is:

1. Called the **transfer function**, and is
2. The Laplace transform of the impulse response, h(t).

Z-transform

Kind of like the laplace transform but for discrete functions.



Block Diagrams

$\oplus$ ALWAYS has 2 inputs, one output. If anything else, ignore. Output = sum of inputs.
When a wire splits up into 2, multiply outputs of the junction by the input.

When a wire encounters a delay or an amplifier or anything without a junction, multiply your current value by the object.

Feedback loop: jump ahead, then look back. Also,

$$H(\omega) = \frac{H_1(\omega)}{1 - H_1(\omega) H_2(\omega)}$$

D = delay = $e^{-i\omega}$
Bonus: D is technically a function, so keep it on the left of things that depend on it, similar to d/dt