LINEAR TIME-INVARIANT SYSTEMS AND THEIR FREQUENCY RESPONSE

Professor Andrew E. Yagle, EECS 206 Instructor, Fall 2005

Dept. of EECS, The University of Michigan, Ann Arbor, MI 48109-2122

I. Abstract

The purpose of this document is to introduce EECS 206 students to linear time-invariant (LTI) systems and their frequency response. It also presents examples of designing a digital speedometer (i.e., differentiator) and a digital low-pass filter.

A. Table of contents by sections:

II. LTI Systems and Other System Properties

So just what is a Linear Time-Invariant (LTI) system, and why should you care?

Systems are used to perform signal processing. The effect of a system on the spectrum of a signal can be analyzed easily if and only if the system is LTI, as we will see in these notes.

The ability to *design* a filter to perform a specific signal processing task (e.g., filter out noise, differentiate, equalize frequency distortion) requires an ability to *analyze* what a filter will do. We will show how to analyze the effect of a given LTI system on the spectrum of a signal. Then we will design LTI systems for low-pass filtering and differentiation. With a few exceptions (e.g., median filtering), most filters are LTI systems.

A. Systems

A *system* is a device that accepts an input signal $x[n]$, processes it somehow, and spits out an output signal $y[n]$. So a system is how a signal gets processed (hence, "signal processing"). Some examples of systems:

$$y[n] = 3x[n] + 1; \quad y[n] = x[n-2]^2; \quad y[n] - 2y[n-1] + 3y[n-2] = 4x[n] + 5x[n-1]$$

The last equation makes these important points:

- The output $y[n]$ at time n can depend on the present input $x[n]$, past inputs $x[n-1], x[n-2]\ldots$ and future inputs $x[n+1], x[n+2]\ldots$, and on known functions of n;
- The output $y[n]$ can also depend on past values of *itself*: $y[n-1], y[n-2]\ldots$
- The system equation need *not* be an explicit formula $y[n] = F(x[n], x[n-1]\ldots, y[n], y[n-1]\ldots)$, although it should be theoretically possible to write such a formula.

A system is often designated using the diagram $x[n] \rightarrow \boxed{\text{SYSTEM}} \rightarrow y[n]$. Note that this does *not* mean that $y[n_o]$ depends *only* on $x[n_o]$. The *complete output* $\{y[n]\}$ depends on the *complete input* $\{x[n]\}$.

A system can be a mathematical model of a real-world device, derived using equations of biology, economics, or some other discipline. Some examples of such models:

- Biology: Population models in which $x[n]$ is some environmental factor and $y[n]$ is the population of some species in generation n;
- Economics: Stock market models in which $x[n]$ is the federal reserve interest rate and $y[n]$ is the daily Dow Jones close at the end of day n.

Note these are all discrete-time systems. In continuous time, systems can be used to model almost any physical phenomenon (e.g., circuits in EECS 215 or 314).

However, in EECS 206 we will *design* discrete-time systems to accomplish a specific task, e.g., filter a *sampled* continuous-time noisy signal. How are we able to do this?

B. Linear Systems

A *linear* system has the property that its response to the sum of two inputs is the sum of the responses to each input separately:

$$x_1[n] \rightarrow \boxed{\text{LIN}} \rightarrow y_1[n] \quad \text{and} \quad x_2[n] \rightarrow \boxed{\text{LIN}} \rightarrow y_2[n] \text{ implies } (x_1[n] + x_2[n]) \rightarrow \boxed{\text{LIN}} \rightarrow (y_1[n] + y_2[n])$$

This property is called **superposition**. By induction, this immediately extends to any number of inputs.

Superposition also implies that scaling the input $x[n]$ by a *constant* a scales the output by a as well:

$$x[n] \rightarrow \boxed{\text{LINEAR}} \rightarrow y[n] \quad \text{implies} \quad ax[n] \rightarrow \boxed{\text{LINEAR}} \rightarrow ay[n]$$

This property is called **scaling**. Although it is listed as a separate property, it follows from superposition. To see this, let's use the notation $x[n] \rightarrow y[n]$ as a shorthand to designate that input signal $\{x[n]\}$ results in output signal $\{y[n]\}$. Let $a = \frac{m}{n}$, where m and n are *integers*, be any rational number. Then we have:

1. Let $\frac{x[n]}{n} \rightarrow z[n]$ where output $z[n]$ is something we wish to determine.
2. Use superposition on n identical inputs $\frac{x[n]}{n}$: Then $n\frac{x[n]}{n} = x[n] \rightarrow y[n] = nz[n]$.
3. So $z[n] = \frac{y[n]}{n}$ and $\frac{x[n]}{n} \rightarrow \frac{y[n]}{n}$,
4. Use superposition on m identical inputs $\frac{x[n]}{n}$: Then $m\frac{x[n]}{n} = ax[n] \rightarrow m\frac{y[n]}{n} = ay[n]$.

So the superposition property implies the scaling property for any rational number a. To extend this result to any real a actually requires some high-level mathematical analysis, but since any real number is arbitrarily close to a rational number, we can fudge the result in EECS 206.

How can you tell whether a system is linear? You have been exposed to enough linear operators in Math 115 and 116 that your intuition will work, with one exception noted below (affine systems). But if your intuition isn't working, the following rule often works:

$$\boxed{\text{If doubling the input doubles the output, then often the system is linear}}$$

This is not always true: the system $y[n] = \frac{y[n]^2}{x[n]} + x[n]$ has this property but is not linear. But this rule usually works. Note the rule is just the scaling property for $a = 2$.

To implement this test, you can use either of two methods:

- Replace $\{x[n]\}$ with $\{2x[n]\}$ and $\{y[n]\}$ with $\{2y[n]\}$, and see if you can reduce to the original equation;
- **Or:** Double the entire equation, and see if you can pull out $\{2x[n]\}$ and $\{2y[n]\}$ everywhere.
- Here $\{x[n]\}$ means $x[n], x[n-1], x[n-2] \ldots x[n+1], x[n+2] \ldots$

Example: $y[n] = 3x[n]$ is linear since $(2y[n]) = 3(2x[n])$ reduces to $y[n] = 3x[n]$.

Example: $y[n] = 3x[n]$ is linear since doubling it yields $2y[n] = 6x[n] \rightarrow (2y[n]) = 3(2x[n])$.

Example: $y[n] = x[n]^2$ is not linear since $(2y[n]) = (2x[n])^2$ does not reduce to $y[n] = x[n]^2$.

Of course, here it is easy to see that doubling the input quadruples the output.

Example: $y[n] - 2y[n-1] + ny[n-2] = 3x[n] + 4x[n-1]$ is linear since doubling it yields $2y[n] - 4y[n-1] + 2ny[n-2] = 6x[n] + 8x[n-1]$ which becomes $(2y[n]) - 2(2y[n-1]) + n(2y[n-2]) = 3(2x[n]) + 4(2x[n-1])$.

Note that you don't need an explicit formula for $y[n]$ in terms of $\{x[n]\}$ here. In fact, you can easily show that superposition holds for this equation:

$$y_1[n] - 2y_1[n-1] + ny_1[n-2] = 3x_1[n] + 4x_1[n-1] \text{ and } y_2[n] - 2y_2[n-1] + ny_2[n-2] = 3x_2[n] + 4x_2[n-1]$$

implies $(y_1[n]+y_2[n]) - 2(y_1[n-1]+y_2[n-1]) + n(y_1[n-2]+y_2[n-2]) = 3(x_1[n]+x_2[n]) + 4(x_1[n-1]+x_2[n-1])$

Nonlinear Systems: $y[n] = \sin(x[n])$; $\quad y[n] = |x[n]|$; $\quad y[n] = \frac{x[n]}{x[n-1]}$; $\quad y[n] = x[n] + 1$

That last one is tricky–it's graph is a straight line, but it isn't linear (doubling $x[n]$ does not double $y[n]$). This is called an *affine* system; "affine" means "linear+constant."

C. Time-Invariant Systems

A *time-invariant* (TI) system has the property that delaying the input by any *constant* D delays the output by the same amount:

$$x[n] \rightarrow \boxed{\text{TIME-INVARIANT}} \rightarrow y[n] \quad \text{implies} \quad x[n-D] \rightarrow \boxed{\text{TIME-INVARIANT}} \rightarrow y[n-D]$$

A time-invariant system thus has no internal clock–it does not know that the input is delayed.

How can you tell whether a system is time-invariant? Here there is a simple rule that works:

$\boxed{\text{If } n \text{ appears } \textit{only} \text{ inside brackets (like } x[n-1]\text{), then the system is TI}}$.

Note that, inside brackets, shifts are acceptable but scales are not: $x[n-2]$ is OK but $x[2n]$ and $x[-n]$ are not (note time reversal $x[-n]$ is a scaling). Also, weird dependencies like $x[n^2]$ are not acceptable.

TI, not Linear: $y[n] = x[n]^2$; $y[n] = |x[n] + x[n-1]|$; $y[n] = \sin(x[n])$; $y[n] = \frac{x[n]}{x[n-1]}$.
Linear, not TI: $y[n] = nx[n]$; $y[n] = x[2n]$; $y[n] = \sin(n)x[n]$; $y[n] - ny[n-1] = 2x[n] + 3x[n-1]$.

D. Linear Time-Invariant (LTI) Systems

An LTI system is one that is both linear and time-invariant (surprise). The LTI systems that will be of particular interest in EECS 206 are *AutoRegresive Moving-Average (ARMA) difference equations:*

$$\underbrace{y[n] + a_1 y[n-1] + a_2 y[n-2] + \ldots + a_N y[n-N]}_{\text{AUTOREGRESSIVE(AR)}} = \underbrace{b_0 x[n] + b_1 x[n-1] + \ldots + b_M x[n-M]}_{\text{MOVING AVERAGE(MA)}}$$

This ARMA difference equation of order (N,M) is a discrete-time analogue of a differential equation.

An <u>autoregression</u> is a computation of a signal $y[n]$ from its N past values $\{y[n-1], y[n-2] \ldots y[n-N]\}$. Regression analysis is an attempt to compute future values from past values, such as stock market closes.

A <u>moving average</u> is a weighted linear combination of present value $x[n]$ and M past values $\{x[n-1], x[n-2] \ldots x[n-M]\}$. It's called a moving average since the quantities being averaged move as time progresses.

An ARMA difference equation can be implemented recursively in time n by rewriting it as

$$y[n] = b_0 x[n] + \ldots + b_M x[n-M] - a_1 y[n-1] - \ldots - a_N y[n-N]$$

In Matlab, use `y=filter([b0 b1...bM],[1 a1...aN],X]`.

A *Moving Average* (MA) difference equation of order M has just the MA part:

$$y[n] = b_0 x[n] + b_1 x[n-1] + \ldots + b_M x[n-M]$$

We will focus on MA systems (described by MA difference equations) in these notes. For ARMA systems, we need the z-transform (covered in the next set of notes).

The following problem shows how useful the concept of LTI can be.

Given the following two input-output pairs: $\{\underline{1},3\} \rightarrow \boxed{\text{LTI}} \rightarrow \{\underline{1},5,6\}$ and $\{\underline{2},7\} \rightarrow \boxed{\text{LTI}} \rightarrow \{\underline{2},11,14\}$
Compute the response to $\{\underline{6},7\}$.

Solution: Using linearity, $\{\underline{2},7\}$-$2\{\underline{1},3\}=\{\underline{0},1\} \rightarrow \boxed{\text{LTI}} \rightarrow \{\underline{2},11,14\}$-$2\{\underline{1},5,6\}=\{\underline{0},1,2\}$
Using time-invariance, we can advance the output in time by 1 to get $\{0,\underline{1},0\} \rightarrow \boxed{\text{LTI}} \rightarrow \{\underline{1},2\}$
Using superposition on the responses to $\delta[n]$ and $\delta[n-1]$, we have
$\{\underline{6},7\}=6\{0,\underline{1},0\}+7\{\underline{0},1\} \rightarrow \boxed{\text{LTI}} \rightarrow 6\{\underline{1},2\}+7\{\underline{0},1,2\}=\boxed{\{\underline{6},19,14\}}$.

This example suggests that once we know the response to $\{0,\underline{1},0\}=\delta[n]$, we can find the response to any input. This is indeed the case, as we show later.

III. Impulse Response and its Computation

The *impulse response* $h[n]$ of an LTI system is just the response to an impulse: $\delta[n] \rightarrow \boxed{\text{LTI}} \rightarrow h[n]$.

The significance of $h[n]$ is that we can compute the response to *any* input once we know the response to impulse. We will derive an explicit formula in the next section. This section will focus on computing $h[n]$.

MA System: We can just read off $h[n]$ from its coefficients. Setting $x[n] = \delta[n]$ and $y[n] = h[n]$,

$$y[n] = b_0x[n]+b_1x[n-1]+\ldots+b_Mx[n-M] \rightarrow h[n] = b_0\delta[n]+b_1\delta[n-1]+\ldots+b_M\delta[n-M] = \{\underline{b_0}, b_1 \ldots b_M\}$$

Example: For $y[n] = 2x[n] + x[n-2] + 4x[n-3]$, we have $h[n] = \{\underline{2}, 0, 1, 4\}$. Don't forget the zero!

1^{st}**-order AR System:** This is the one AR system we can analyze without the z-transform (see the next set of notes for that). We wish to compute the impulse response $h[n]$ for $\underline{y[n] - ay[n-1] = bx[n]}$.

Setting $x[n] = \delta[n]$ and $y[n] = h[n]$ and $n = 0, 1, 2\ldots$ yields

$$n = 0 \rightarrow h[0] - ah[-1] = b\delta[0] = b \rightarrow h[0] = b$$
$$n = 1 \rightarrow h[1] - ah[0] = b\delta[1] = 0 \rightarrow h[1] = ba$$
$$n = 2 \rightarrow h[2] - ah[1] = b\delta[2] = 0 \rightarrow h[2] = ba^2$$
$$n = 3 \rightarrow h[3] - ah[2] = b\delta[3] = 0 \rightarrow h[3] = ba^3$$

An induction argument confirms $h[n] = ba^n u[n]$. Using LTI, the response to $b_k\delta[n-k]$ is $b_k a^{n-k} u[n-k]$.

$$y[n]-ay[n-1] = b_0x[n]+\ldots+b_Mx[n-M] \rightarrow h[n] = b_0a^nu[n]+b_1a^{n-1}u[n-1]+\ldots+b_Ma^{n-M}u[n-M]$$

ARMA difference equations with AR-part order $N \geq 2$ will have to wait for the z-transform, in the next set of notes. But see again how powerful the concept of LTI can be?

IV. Convolution and its Computation

A. Derivation of Convolution

For LTI systems, we now show how to compute the response to *any* input from the impulse response $h[n]$:

1. $\delta[n] \rightarrow \boxed{\text{LTI}} \rightarrow h[n]$ (Definition of impulse response)
2. $\delta[n-i] \rightarrow \boxed{\text{LTI}} \rightarrow h[n-i]$ (Time-invariance–works for any constant i)
3. $x[i]\delta[n-i] \rightarrow \boxed{\text{LTI}} \rightarrow x[i]h[n-i]$ (Scaling property of Linear–works for any constant $x[i]$)
4. $\sum_{i=-\infty}^{\infty} x[i]\delta[n-i] \rightarrow \boxed{\text{LTI}} \rightarrow \sum_{i=-\infty}^{\infty} x[i]h[n-i]$ (Superposition property of Linear)

When you have no idea what to make of a summation, like the one here, try writing it out explicitly. And if it goes from $-\infty$ to ∞, write out the terms for positive and negative indices separately. Here, this gives

$$\sum_{i=-\infty}^{\infty} x[i]\delta[n-i] = x[0]\delta[n] + x[1]\delta[n-1] + x[2]\delta[n-2] + \ldots + x[-1]\delta[n+1] + x[-2]\delta[n+2] + \ldots$$
$$= \{\ldots x[-2], x[-1], \underline{x[0]}, x[1], x[2]\ldots\} = x[n].$$

So $\sum_{i=-\infty}^{\infty} x[i]\delta[n-i] = x[n]$! And the response of the LTI system to any input $x[n]$ is

$x[n] \rightarrow \boxed{\text{LTI}} \rightarrow y[n] = \sum_{i=-\infty}^{\infty} x[i]h[n-i] = h[n] * x[n]$

This is called the *convolution* of the two signals $\{h[n]\}$ and $\{x[n]\}$, because it is so convoluted (scrambled). Again, writing out the summation explicitly makes things clearer. Let $h[n] = 0$ for $n < 0$. Then we have

$y[n] = h[n] * x[n] = h[0]x[n] + h[1]x[n-1] + h[2]x[n-2] + h[3]x[n-3] + \ldots$

which looks like a MA difference equation. Examples of computing convolution will be given below.

Remember, $\boxed{x[n] \rightarrow \boxed{\text{LTI}} \rightarrow y[n] = h[n] * x[n] = \sum x[i]h[n-i]}$

B. Causal and Stable Systems

Some quick definitions (if you think you understand these, you do):

- A signal $x[n]$ is **causal** if and only if $x[n] = 0$ for all $n < 0$.
- $x[n]$ is causal if and only if $x[n]u[n] = x[n]$ (think about it).
- An LTI system is **causal** if and only if its impulse response $h[n]$ is causal.
- All real-world systems are causal; otherwise they could see into the future!

- A signal $x[n]$ is **bounded** if and only if $|x[n]| \leq M$ for some constant M.
- **Example:** $3\cos(2n+1)$ is bounded (but not periodic) since $|3\cos(2n+1)| \leq 3$.
- **Example:** Causal $(\frac{1}{2})^n u[n]$ is bounded but non-causal $(\frac{1}{2})^n$ is not bounded.

A system is **BIBO stable** (Bounded-Input-Bounded-Output) if and only if any Bounded Input results in a Bounded Output (again, sensible nomenclature!). How are you supposed to know whether this is true?

Theorem: An LTI system is BIBO stable if and only if its impulse response is *absolutely summable*, i.e., $\sum_{n=-\infty}^{\infty} |h[n]| < \infty$, i.e., is a finite number.

What's the difference between summable and absolutely summable? Consider:

$\sum h[n] = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \ldots = \log_e 2 = 0.693.$

$\sum |h[n]| = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \ldots \rightarrow \infty.$

Proof: Since this is an "if-and-only-if" proof, there are two parts to it.

Sufficiency: <u>Suppose</u> that $h[n]$ is absolutely summable: $\sum |h[n]| = N$ for some N.

Now prove that the system is BIBO stable. How do we do this?

<u>Suppose further</u> that $x[n]$ is bounded: $|x[n]| \leq M$ for some M. Now prove that $y[n]$ is bounded:

$|y[n]| = |h[n] * x[n]| = |\sum h[n-i]x[i]| \leq \sum |h[n-i]| \cdot |x[i]| \leq \sum |h[n-i]|M = NM,$

using the triangle inequality. Q.E.D. (Latin for, "Thank God that's over").

Necessity: <u>Suppose</u> that $h[n]$ is not absolutely summable. Then the response to the non-causal input $x[n] = \frac{h[-n]}{|h[-n]|}$ at $n = 0$ is $y[0] = \sum |h[n]| \rightarrow \infty$, so this bounded input produces an unbounded output.

Note that $\boxed{\text{any MA system is BIBO stable}}$, since there are only a finite number of non-zero $|h[n]|$.

Example: $h[n] = a^n u[n]$ is BIBO stable if and only if $|a| < 1$ since then

$\sum_{n=-\infty}^{\infty} |h[n]| = \sum_{n=-\infty}^{\infty} |a|^n u[n] = \sum_{-\infty}^{-1} 0 + \sum_{n=0}^{\infty} |a|^n = \frac{1}{1-|a|} < \infty.$

Note how to handle the step function $u[n]$ inside the summation: break up the sum into two parts.

Example: $h[n] = \{2, 3, -4\} \to \sum |h[n]| = |2| + |3| + |-4| = 9 < \infty \to$ BIBO stable (MA system).

Example: $h[n] = (-\frac{1}{2})^n u[n] \to \sum |h[n]| = \sum |-\frac{1}{2}|^n = \frac{1}{1-0.5} < \infty \to$ BIBO stable.

Example: $h[n] = \frac{(-1)^n}{n+1} u[n] \to \sum |h[n]| = \sum \frac{1}{n+1} u[n] \to \infty \to$ NOT BIBO stable.

No, "BIBO stable" is not a hobbit from "Lord of the Rings" (although it does sound like one).

BIBO stability is a necessity if you don't want smoke coming out of your system!

C. Properties of Convolution

Basic Properties: You should be able to prove these yourself immediately:

- $(ah[n]) * x[n] = h[n] * (ax[n]) = a(h[n] * x[n])$ (scaling property)
- $x[n] * \delta[n] = x[n]$ and $x[n] * \delta[n - D] = x[n - D]$ (sifting property)
- $h[n - D] * x[n] = h[n] * x[n - D] = y[n - D]$ where $y[n] = h[n] * x[n]$ (delay property)
- This is very useful for signals that don't start at $n = 0$
- (causal signal)$*$(causal signal)=causal signal.

Commutative: $h[n] * x[n] = x[n] * h[n] = \sum x[i]h[n - i] = \sum h[i]x[n - i]$.

Significance: The order in which two signals are convolved doesn't matter. This means that we can exchange input and impulse response and get the same output:

$x[n] \to \boxed{h[n]} \to y[n]$ gives same output as $h[n] \to \boxed{x[n]} \to y[n]$.

Associative: $(x[n] * y[n]) * z[n] = x[n] * (y[n] * z[n])$

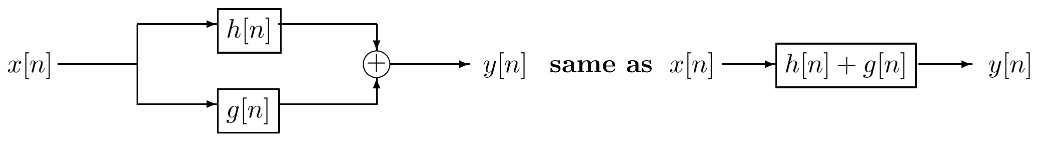
Significance: The overall impulse response of two LTI systems connected in **series** or **cascade** is the convolution of their individual impulse responses:

$x[n] \to \boxed{g[n]} \to \boxed{h[n]} \to y[n]$ **same as** $x[n] \to \boxed{h[n] * g[n]} \to y[n]$.

To see this, let $z[n]$ be the signal between the two systems. Then we have $z[n] = g[n] * x[n]$ and $y[n] = h[n] * z[n]$ so that $y[n] = h[n] * (g[n] * x[n]) = (h[n] * g[n]) * x[n]$.

Distributive: $(x[n] + y[n]) * z[n] = x[n] * z[n] + y[n] * z[n]$

Significance: The impulse response of two LTI systems in **parallel** is sum of their impulse responses:



To see this, note that $y[n] = h[n] * x[n] + g[n] * x[n] = (h[n] + g[n]) * x[n]$.

D. Computation of Convolution

How do you go about computing $h[n] * x[n] = \sum x[i]h[n-i] = \sum h[i]x[n-i]$?

Two Finite-Duration Signals:

- $h[n] = \{h[0], h[1] \ldots h[L-1]\}$ is causal, has finite duration L and support $[0, L-1]$.
- $x[n] = \{x[0], x[1] \ldots x[M-1]\}$ is causal, has finite duration M and support $[0, M-1]$.
- $y[n] = \{y[0] \ldots y[L+M-2]\}$ is causal, has finite duration $L+M-1$ and support $[0, L+M-2]$.
- Note duration($h[n] * x[n]$)=duration($h[n]$)+duration($x[n]$)-1: don't forget that "-1."
- Note causal$*$causal=causal. This makes sense–how can there be any output before any input?

Example: Compute $\{\underline{1}, 2, 3\} * \{\underline{4}, 5, 6, 7\} = \{\underline{4}, 13, 28, 34, 32, 21\}$:

- $y[0] = h[0]x[0] = (1)(4) = 04$.
- $y[1] = h[1]x[0] + h[0]x[1] = (2)(4) + (1)(5) = 13$.
- $y[2] = h[2]x[0] + h[1]x[1] + h[0]x[2] = (3)(4) + (2)(5) + (1)(6) = 28$.
- $y[3] = h[2]x[1] + h[1]x[2] + h[0]x[3] = (3)(5) + (2)(6) + (1)(7) = 34$.
- $y[4] = h[2]x[2] + h[1]x[3] = (3)(6) + (2)(7) = 32$.
- $y[5] = h[2]x[3] = (3)(7) = 21$.
- **Note:** Length $y[n]$=Length $h[n]$+Length $x[n] - 1 = 3 + 4 - 1 = 6$.

In Matlab, use `conv([1 2 3],[4 5 6 7])` to get the above answer.

If this looks like polynomial multiplication, that's because it is. The z-transform uses this fact.

A much faster way to compute convolution is to use the Discrete Fourier Transform (DFT). We have:

$$y[n] = h[n] * x[n] \rightarrow DFT\{y[n]\} = (DFT\{h[n]\})(DFT\{x[n]\})$$

- All three DFTs have the same length; multiplication is for each k;
- The DFT length$\geq$duration of $y[n]$ (otherwise this won't work);
- DFTs of length=power of 2 (e.g., 256 or 512 or 1024) can be computed quickly.

Example: Compute $\{\underline{1}, 2\} * \{\underline{3}, 4\} = \{\underline{3}, 10, 8\}$ using a 4-point DFT (omit $\frac{1}{4}$ for clarity):

SIGNAL	$\rightarrow$	$\mathbf{X_0}$	$\mathbf{X_1}$	$\mathbf{X_2}$	$\mathbf{X_3}$
$\{1,2,0,0\}$	$\rightarrow$	3	$1-2j$	-1	$1+2j$
$\{3,4,0,0\}$	$\rightarrow$	7	$3-4j$	-1	$3+4j$
$*\downarrow$		$\times\downarrow$	$\times\downarrow$	$\times\downarrow$	$\times\downarrow$
$\{3,10,8,0\}$	$\leftarrow$	21	$-5-10j$	1	$-5+10j$

Matlab: `real(ifft(fft([1,2],4).* fft([3,4],4)))` This is *much* faster for long signals.

One Finite and One Infinite-Duration Signal:

To compute $h[n] * x[n]$, where $h[n]$ has finite duration and $x[n]$ has infinite duration, proceed as follows. Assume WLOG (without loss of generality) that $h[n]$ has duration $L+1$ and is causal. Then we have

$h[n]*x[n]{=}(h[0]\delta[n]+h[1]\delta[n-1]+\ldots+h[L]\delta[n-L])*x[n]{=}h[0]x[n]+h[1]x[n-1]+\ldots+h[L]x[n-L]$

using linearity and the distributive property of convolution.

Example: Compute $\{\underline{2},-1,3\}*(\frac{1}{2})^n u[n]=2\delta[n]+3(\frac{1}{2})^{n-2}u[n-2]$:

- $\{\underline{2},-1,3\}*(\frac{1}{2})^n u[n]=(2\delta[n]-1\delta[n-1]+3\delta[n-2])*(\frac{1}{2})^n u[n]=$
- $2(\frac{1}{2})^n u[n]-1(\frac{1}{2})^{n-1}u[n-1]+3(\frac{1}{2})^{n-2}u[n-2]=2\delta[n]+3(\frac{1}{2})^{n-2}u[n-2]$.
- **Aside:** $\{\underline{2},-1\}*(\frac{1}{2})^n u[n]=2\delta[n]$. So we also have $x[n]\to\boxed{(\frac{1}{2})^n u[n]}\to\boxed{\text{\{2,-1\}}}\to 2x[n]$.

V. Frequency Response of LTI Systems

We now show how to analyze the effect of a given LTI system on the spectrum of a signal fed into it. While this may seem simple, it draws heavily on the material on LTI systems covered above.

A. Response to Complex Exponential Input

Let $x[n]=e^{j\omega n}$ be input into an LTI system with causal impulse response $h[n]$. The output is

$y[n]=h[n]*x[n]=h[n]*e^{j\omega n}=\sum_{i=0}^{\infty}h[i]e^{j\omega(n-i)}=e^{j\omega n}\sum_{i=0}^{\infty}h[i]e^{-j\omega i}=H(e^{j\omega})e^{j\omega n}$ where

$H(e^{j\omega})=\sum_{i=0}^{\infty}h[i]e^{-j\omega i}{=}$**Frequency Response**$=|H(e^{j\omega})|e^{j\arg[H(e^{j\omega})]}$

- $H(e^{j\omega})$=frequency response function (a conjugate symmetric function of ω)
- $|H(e^{j\omega})|$=gain function (an even function of ω)
- $\arg[H(e^{j\omega})]$=phase function (an odd function of ω)
- $H(e^{j\omega})$ is periodic with period=2π
- This is why I prefer to use $H(e^{j\omega})$ instead of $H(\omega)$–as a reminder!

B. Response to Sinusoidal Input

Now let $x[n]=\cos(\omega n)$ be input into an LTI system with causal impulse response $h[n]$. The output is

- $e^{j\omega n}\to\boxed{\text{LTI}}\to H(e^{j\omega})e^{j\omega n}$
- $e^{-j\omega n}\to\boxed{\text{LTI}}\to H(e^{-j\omega})e^{-j\omega n}$
- Adding and using linearity and conjugate symmetry,
- $\cos(\omega n)\to\boxed{\text{LTI}}\to A\cos(\omega n+\theta)$ where
- $A=|H(e^{j\omega})|$ and $\theta=\arg[H(e^{j\omega})]$.

To see this in more detail, note the output $y[n]$ when input $x[n]=e^{j\omega n}+e^{-j\omega n}=2\cos(\omega n)$ is

$$H(e^{j\omega})e^{j\omega n}+H(e^{-j\omega})e^{-j\omega n}=|H(e^{j\omega})|e^{j\arg[H(e^{j\omega})]}e^{j\omega n}+|H(e^{-j\omega})|e^{j\arg[H(e^{-j\omega})]}e^{-j\omega n}$$
$$=|H(e^{j\omega})|[e^{j\arg[H(e^{j\omega})]}e^{j\omega n}+e^{-j\arg[H(e^{j\omega})]}e^{-j\omega n}]=|H(e^{j\omega})|[e^{j(\omega n+\arg[H(e^{j\omega})])}+e^{-j(\omega n+\arg[H(e^{j\omega})])}]$$
$$=|H(e^{j\omega})|2\cos(\omega n+\arg[H(e^{j\omega})]).$$

In summary: $\boxed{e^{j\omega n}\to\boxed{\text{LTI}}\to H(e^{j\omega})e^{j\omega n}}$ and $\boxed{\cos(\omega n)\to\boxed{\text{LTI}}\to|H(e^{j\omega})|\cos(\omega n+\arg[H(e^{j\omega})])}$.

where $\boxed{H(e^{j\omega}) = h[0] + h[1]e^{-j\omega} + h[2]e^{-j2\omega} + h[3]e^{-j3\omega} + \ldots}$=frequency response function.

The response of an LTI system to a sinusoidal or complex exponential input is a sinusoid or complex exponential output **at the same frequency as the input**. LTI systems cannot change frequencies.

C. Examples

Example: $x[n] = \cos(\frac{\pi}{2}n) \rightarrow \boxed{h[n] = \{\underline{7}, 4, 4\}} \rightarrow y[n]$. Compute $y[n]$.

Solution: Proceed as follows:

- $H(e^{j\omega}) = 7 + 4e^{-j\omega} + 4e^{-j2\omega}$. This is a typical frequency response function.
- $H(e^{j\pi/2}) = 7 + 4e^{-j\pi/2} + 4e^{-j\pi} = 7 - 4j - 4 = 3 - 4j = 5e^{-j0.927}$
- $\cos(\frac{\pi}{2}n) \rightarrow \boxed{\text{LTI}} \rightarrow 5\cos(\frac{\pi}{2}n - 0.927) = \{\ldots 3, 4, -3, -4, 3, 4 \ldots\}$
- Confirm: $\{7, 4, 4\} * \{\ldots 1, 0, -1, 0, 1, 0 \ldots\} = \{\ldots 3, 4, 3, -4, 3, 4 \ldots\}$ omitting startup and ending.

Example: $x[n] = 5\cos(\frac{\pi}{2}n) \rightarrow \boxed{h[n] = (\frac{1}{2})^n u[n]} \rightarrow y[n]$. Compute $y[n]$.

Solution: Proceed as follows:

- $H(e^{j\omega}) = 1 + \frac{1}{2}e^{-j\omega} + \frac{1}{4}e^{-j2\omega} + \ldots = \frac{1}{1-\frac{1}{2}e^{-j\omega}}$.
- $H(e^{j\pi/2}) = \frac{1}{1-e^{-j\pi/2}/2} = \frac{1}{1+j/2} = 1/(1.12e^{j0.464}) = 0.89e^{-j0.464}$.
- $5\cos(\frac{\pi}{2}n) \rightarrow \boxed{\text{LTI}} \rightarrow 5(0.89)\cos(\frac{\pi}{2}n - 0.464) = \{\ldots 4, 2, -4, -2, 4, 2 \ldots\}$
- Try doing this by convolving two infinite-duration signals–it's difficult.

Example: Compute gain functions for $g[n] = \{\underline{1}, 1\}$ and $h[n] = \{\underline{1}, -1\}$.

Solution: Plugging into the definition of frequency response function, we have

$$g[n] = \{\underline{1}, +1\} \rightarrow G(e^{j\omega}) = 1 + e^{-j\omega} = e^{-j\omega/2}(e^{j\omega/2} + e^{-j\omega/2}) = 2\cos(\tfrac{\omega}{2})e^{-j\omega/2}.$$

$$h[n] = \{\underline{1}, -1\} \rightarrow H(e^{j\omega}) = 1 - e^{-j\omega} = e^{-j\omega/2}(e^{j\omega/2} - e^{-j\omega/2}) = 2\sin(\tfrac{\omega}{2})e^{-j\omega/2}j.$$

Now you have to be careful in taking magnitudes. The answers are low-pass and high-pass filters:
Gain functions=$|G(e^{j\omega})| = 2|\cos(\frac{\omega}{2})|$ (low-pass) and $|H(e^{j\omega})| = 2|\sin(\frac{\omega}{2})|$ (high-pass).
Try sketching these yourself to see how they complement each other.

VI. Fourier Series Response of LTI Systems

Who cares about the response to a single sinusoid or complex exponential? That is of little interest. But using LTI (again), we can compute the response to any linear combination of sinusoids or complex exponentials by taking the same linear combination of the responses to each individual sinusoid or complex exponential. So we can find the response to a Fourier series, which means we can find the response to any periodic function, and see immediately what its effect will be on its spectrum. That is of great interest.

A. Response to Discrete-Time Fourier Series

Again, we proceed in steps, using LTI along the way (see how useful and important LTI has become?):

1. $e^{j\omega n} \rightarrow \boxed{\text{LTI}} \rightarrow H(e^{j\omega})e^{j\omega n}$ from above. Now set $\omega = 2\pi\frac{k}{N}$:

2. $e^{j2\pi kn/N} \rightarrow \boxed{\text{LTI}} \rightarrow H(e^{j2\pi k/N})e^{j2\pi kn/N}$. Using the scaling property:
3. $x_k e^{j2\pi kn/N} \rightarrow \boxed{\text{LTI}} \rightarrow H(e^{j2\pi k/N})x_k e^{j2\pi kn/N}$. Using superposition:
4. $\sum_{k=0}^{N-1} x_k e^{j2\pi kn/N} \rightarrow \boxed{\text{LTI}} \rightarrow \sum_{k=0}^{N-1} H(e^{j2\pi k/N})x_k e^{j2\pi kn/N}$. Identifying Fourier series:
5. $x[n] \rightarrow \boxed{\text{LTI}} \rightarrow y[n]$ where $x[n]$ and $y[n]$ have Fourier series expansions
6. $x[n] = \sum_{k=0}^{N-1} x_k e^{j2\pi kn/N}$ where $x_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}$ for $k = 0, 1 \ldots N-1$
7. $y[n] = \sum_{k=0}^{N-1} y_k e^{j2\pi kn/N}$ where $y_k = \frac{1}{N}\sum_{n=0}^{N-1} y[n]e^{-j2\pi nk/N}$ for $k = 0, 1 \ldots N-1$
8. $\boxed{y_k = H(e^{j2\pi k/N})x_k}$: **The spectrum of x[n] has been altered by multiplication by** $H(e^{j\omega})$

Use this to compute the response of an LTI system to an input whose Fourier series expansion is known:

B. Examples

Example: $x[n] = 1 + 2\cos(\frac{\pi}{2}n) + 3\cos(\pi n) \rightarrow \boxed{h[n] = (\frac{1}{2})^n u[n]} \rightarrow y[n]$. Compute $y[n]$.

Solution: Compute the response to each input term separately, and then add the results:

$x[n]$ term:	1	$2\cos(\frac{\pi}{2}n)$	$3\cos(\pi n)$
$H(e^{j\omega})$=:	$\frac{1}{1-\frac{1}{2}1}$	$\frac{1}{1-\frac{1}{2}(-j)}$	$\frac{1}{1-\frac{1}{2}(-1)}$
Simplifies to:	$2e^{j0}$	$0.89e^{-j0.464}$	$\frac{2}{3}e^{j0}$
$y[n]$ term:	2	$1.78\cos(\frac{\pi}{2}n - 0.464)$	$2\cos(\pi n)$

$y[n] = 2 + 1.78\cos(\frac{\pi}{2}n - 0.464) + 2\cos(\pi n)$. OK, but so what? Try a different system:

Example: $x[n] = 1 + 2\cos(\frac{\pi}{2}n) + 3\cos(\pi n) \rightarrow \boxed{h[n] = \{\underline{1}, 0, 1\}} \rightarrow y[n]$. Compute $y[n]$.

Solution: Now $h[n] = \{\underline{1}, 0, 1\} \rightarrow H(e^{j\omega}) = 1 + e^{-j2\omega} = 2\cos(\omega)e^{-j\omega}$. Now we have:

$x[n]$ term:	1	$2\cos(\frac{\pi}{2}n)$	$3\cos(\pi n)$
$H(e^{j\omega})$=:	$1 + e^{-j0}$	$1 + e^{-j\pi}$	$1 + e^{-j2\pi}$
Simplifies to:	$2e^{j0}$	0 (!)	$2e^{j0}$
$y[n]$ term:	2	0 (!)	$6\cos(\pi n)$

$y[n] = 2 + 6\cos(\pi n)$. The system killed off the $\cos(\frac{\pi}{2}n)$ component and doubled the remaining two components. It *filtered* out that component and left the remaining components. It isn't hard to see how:

$\{1, 0, 1\} * \{\ldots 1, 0, -1, 0, 1, 0 \ldots\} = \{\ldots 0, 0, 0, 0, 0, 0 \ldots\}$

C. Notch Filters

Now that we can *analyze* a filter that eliminates a single frequency, let's *design* a filter that will eliminate any desired single frequency. It is easy to see that $\boxed{h[n] = \{\underline{1}, -2\cos(\omega_o), 1\}}$=**notch filter** yields

$$H(e^{j\omega}) = 1 - 2\cos(\omega_o)e^{-j\omega} + e^{-j2\omega} = 2[\cos(\omega) - \cos(\omega_o)]e^{-j\omega}$$

which eliminates any term of the form $A\cos(\omega_o n + \theta)$.

This is called a *notch filter* since its gain function looks like a notch chopped in a log with an axe.

Application: Why would you want to eliminate a single frequency anyways? Next time you are in a lab, look around: there are electrical outlets and power cords everywhere. All of them are carrying electricity at 120 volts rms (170 volts amplitude!) and 60 Hertz. This radiates into the lab and induces 60 Hertz signals in the leads to your oscilloscope, sensor, or any other measuring device. Proper grounding helps greatly, but it does not eliminate the problem (remember this for EECS 215 or 314!). A notch filter can eliminate this.

If you are using a DSP system with 1 kHz sampling rate, then to eliminate 60 Hertz we need to use

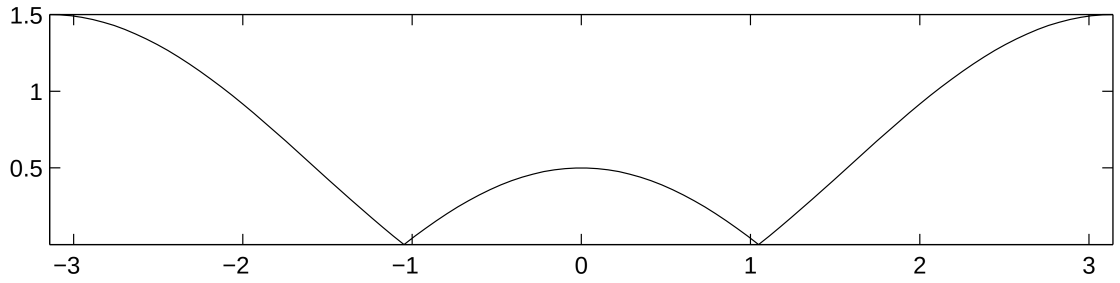
$$\omega_o = 2\pi \frac{60 \text{ Hz}}{1000 \text{ Hz}} = 0.377 \rightarrow h[n] = \{1, -1.86, 1\} \rightarrow y[n] = x[n] - 1.86x[n-1] + x[n-2]$$

To see why, recall that the spectrum of the sampled signal is periodic with period=sampling rate. Then:

Cont.-time frequency (Hz)	60	1000
Discrete-time frequency	$2\pi\frac{60}{1000}$	2π

Also, note that dimensionally this is the only formula that makes sense!

A typical notch filter has a magnitude response that looks like this (note the notches):



Matlab: `W=linspace(-pi,pi,100);E=abs(cos(W)-cos(pi/3));subplot(311),plot(W,E),axis tight`

Note that this filter also reduces all components with frequencies anywhere near ω_o. Despite its name, it is not a very selective filter. We will design much better filters for rejecting single frequencies in the next set of notes, by using an ARMA, instead of just an MA, difference equation.

VII. Application: Digital Speedometer

A. Introduction

The notch filter represented the first time we actually designed a digital filter to perform a specific task. We now design a digital speedometer (i.e., digital differentiator). Note how many different EECS 206 concepts enter into the discussion to follow. There were good reasons for studying all of that material!

A speedometer is a device that computes instantaneous speed, which is the derivative with respect to time of total distance travelled. The latter is measured using an odometer; this is our continuous-time input $x(t)$. The goal is to compute the continuous-time output speed=$y(t) = \frac{dx}{dt}$ digitally.

We will use a DSP system with a 1 kHz sampling rate. Since $x(t)$ varies very slowly (it gradually increases with time, with no sudden changes or discontinuities), it is likely bandlimited to much less than 500 Hz. So $x(t)$ is heavily oversampled. This has two advantages:

- We don't need an anti-alias filter prior to sampling (A/D);
- We can use a zero-order hold for the interpolator (D/A).

We will also omit all overall scale factors, to make the computation easier to follow.

B. Frequency Response of Differentiator

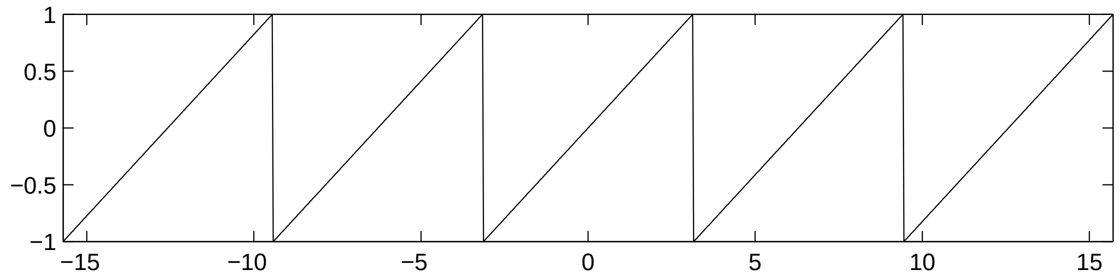
What should be the frequency response of our digital filter? This is the same as asking, what is the continuous-time frequency response of a differentiator? To find out, let's see what is the effect of differentiation on a complex exponential:

$$x(t) = e^{j\omega t} \rightarrow \frac{dx}{dt} = j\omega e^{j\omega t} \rightarrow H(\omega) = j\omega = |\omega| e^{j\text{sign}(\omega)\pi/2}$$

where $H(\omega)$ is the continuous-time frequency response (the only difference from discrete time is that $H(\omega)$ is no longer periodic with period$=2\pi$). So differentiation results in a gain proportional to the frequency and a 90-degree phase shift (-90 degrees for negative frequencies).

Hence our desired discrete-time frequency response is $H(e^{j\omega}) = j\omega$ for $|\omega| < \pi$. Since $H(e^{j\omega})$ must be periodic with period$=2\pi$, we use the periodic extension of $j\omega$, not $j\omega$ itself (which is not periodic).

What LTI system impulse response $h[n]$ results in an $H(e^{j\omega})$ that looks like this? You have seen how to compute $h[n]$ but in a completely different context. Stare at the figure for awhile before reading on.



Since $H(e^{j\omega})$ is a continuous function of ω with period$=2\pi$, it has a Fourier series expansion in ω:

$$H(e^{j\omega}) = h_0 + h_1 e^{j2\pi\omega/T} + h_2 e^{j4\pi\omega/T} + \ldots + h_{-1} e^{-j2\pi\omega/T} + h_{-2} e^{-j4\pi\omega/T} + \ldots$$

where T=period$=2\pi$ and the Fourier series coefficients h_k are computed from one period of $H(e^{j\omega})$ using

$$h_k = \frac{1}{T} \int_{-T/2}^{T/2} H(e^{j\omega}) e^{-j2\pi k\omega/T} d\omega = \frac{1}{2\pi} \int_{-\pi}^{\pi} j\omega e^{-jk\omega} d\omega = \frac{(-1)^k}{-k}$$

for $k \neq 0$ (and $h_0 = 0$). What good does this do us? Comparing the previous equation with the definition

$$H(e^{j\omega}) = h[0] + h[1]e^{-j\omega} + h[2]e^{-j2\omega} + \ldots + h[-1]e^{j\omega} + h[-2]e^{j2\omega} + \ldots$$

(extended to non-causal $h[n]$), we see that $h[-k] = h_k$! So for a digital differentiator

$$h[n] = (-1)^n/n = \{\ldots - \tfrac{1}{4}, \tfrac{1}{3}, -\tfrac{1}{2}, 1, \underline{0}, -1, \tfrac{1}{2}, -\tfrac{1}{3}, \tfrac{1}{4} \ldots\}$$

$$\boxed{y[n] = h[n] * x[n] = \ldots + \tfrac{1}{3}x[n+3] - \tfrac{1}{2}x[n+2] + x[n+1] - x[n-1] + \tfrac{1}{2}x[n-2] - \tfrac{1}{3}x[n-3] + \ldots}$$

which is the digital filter for an ideal digital differentiator. Note that:

- Since $H(e^{j\omega}) = j\omega$ (in one period) is purely imaginary and odd, $h[n]$ is also;
- $h[n]$ is non-causal and unstable, since $\sum |h[n]| = 2(1 + \frac{1}{2} + \frac{1}{3} + \ldots) \to \infty$
- Truncating $h[n]$ to $\{1, \underline{0}, -1\}$ yields $y[n] = x[n+1] - x[n-1]$, the central difference approximation

The overall system looks like this: $x(t) \to \boxed{\text{A/D}} \to x[n] \to \boxed{h[n]} \to y[n] \to \boxed{\text{D/A}} \to y(t)$

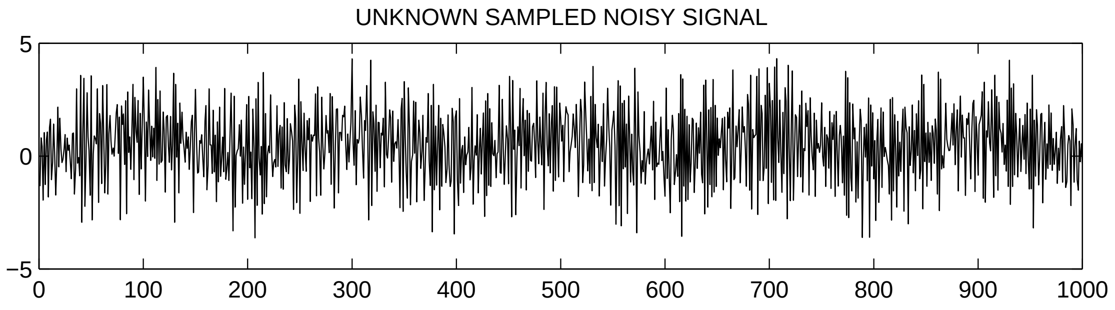
A/D: $x[n] = x(t = n/1000)$. **D/A:** $y(t) = y[n]$ for $\frac{n}{1000} < t < \frac{n+1}{1000} = \frac{n}{1000} + 0.001$ (staircase function).

VIII. Application: Digital Low-Pass Filter

Problem: We are given the noisy signal shown below. Our goal is to filter out the noise.

All we know about the signal is the following:

- It is periodic with period=0.2 seconds
- It is bandlimited to about 20 Hz.
- It is sampled at 1 kHz and we have 1000 samples



Solution: We can infer the following facts:

- Since the signal has period=0.2 seconds, the fundamental frequency is 1/0.2=5 Hz.
- The signal also has harmonics at 10,15,20 Hz; higher harmonics are negligible.
- Sampling at 1000 Hz→signal better be bandlimited to 500 Hz.
- Actually bandlimited to 20 Hz→ 25x oversampling (500/20=25).
- (1000 samples)/(1000 Hz)=1 second duration=5(0.2 seconds)=5 periods given
- Using a 1000-point DFT→following continuous-time and discrete-time frequencies:

Cont.-time frequency (Hz)	20	500	1000
Discrete-time frequency	$\frac{\pi}{25}$	π	2π
DFT frequency index k	20	500	none

This suggests the following filtering procedure:

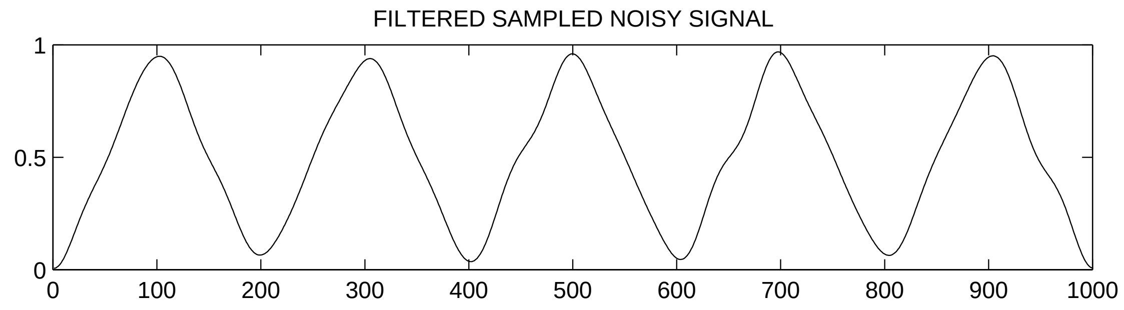
- Compute a 1000-point DFT of the noisy data;
- Set the result=0 for $21 \le k \le 979$ (Matlab indices: $22 \le k \le 980$);
- Keep: $k = 0$ (DC); $1 \le k \le 20$ (positive frequencies); $980 \le k \le 999$ (negative frequencies);
- Compute a 1000-point inverse DFT of the result.

The result is shown below.

- This is the way to do it *if we already have all of the data before we start.*
- Matlab indexing: 1=DC; (6 and 996)=$\pm$5 Hz, (11 and 991)=$\pm$10 Hz, (16 and 986)=$\pm$15 Hz, etc.
- Note that the noise was formed by taking differences between successive values of random numbers. From the section on digital speedometers (differentiators), we know that this increases high-frequency components and reduces low-frequency components. This makes the example more dramatic.
- Since the signal has only 5 significant components (at 0,5,10,15,20 Hertz), can't we just estimate 9 numbers (the DC component is real) from the data? We certainly can, but such methods lie beyond EECS 206.

Matlab code for this example:

```
for I=0:2:8;X(100*I+1:100*I+100)=linspace(0,1,100);end
for I=1:2:9;X(100*I+1:100*I+100)=1-linspace(0,1,100);end
N=4*rand(1,1001);N=N(2:1001)-N(1:1000);Y=X+N;plot(Y)
FY=fft(Y);FY(22:980)=0;Z=real(ifft(FY));plot(Z)
```



Sometimes we wish to filter the data *as it arrives*, without waiting for all of it. This is called *real-time signal processing*. In this case, we can filter the data using the impulse response (note this is non-causal, but we can delay it to make it causal; this then delays the output):

$h[n] = \text{sinc}(n/25) = 25[\sin(\frac{\pi}{25}n)]/(\pi n)$ where sinc(x)=$[\sin(\pi x)]/(\pi x)$ is even and sinc(0)=1.

$h[n]$ comes from expanding the periodic function $H(e^{j\omega}) = \begin{cases} 1 & \text{for } 0 \leq |\omega| < \pi/25; \\ 0 & \text{for } \pi/25 < |\omega| \leq \pi \end{cases}.$
in a Fourier series, as in the speedometer section. Filter:

$$\boxed{y[n] = \ldots+\text{sinc}(\tfrac{2}{25})x[n+2]+\text{sinc}(\tfrac{1}{25})x[n+1]+x[n]+\text{sinc}(\tfrac{1}{25})x[n-1]\ +\text{sinc}(\tfrac{2}{25})x[n-2]+\ldots}$$

In Matlab use `Y=filter([sinc([-20:20]*pi/25)],1,X)` to implement this (you may want to use a filter length more that 2(20)=41). In EECS 451 and EECS 452 you will learn how to design optimal MA lowpass filter using numerical optimization algorithms. If you can't wait, try `help remez` in Matlab.

You have now learned how to: (1) eliminate individual frequencies; (2) filter noisy signals; (3) differentiate signals. But how do you do other things? How do you take advantage of the AR part of ARMA difference equations? For this, we need a powerful tool–the z-transform. See the next set of notes.