Lecture 32: Series and parallel filter combinations

Mark Hasegawa-Johnson

ECE 401: Signal and Image Analysis

1. Review: FIR and IIR Filters, and Transfer Functions
2. Effects of numerical errors on poles and zeros
3. Series connection of second-order sections
4. Parallel connection of second-order sections
5. Partial fraction expansion
6. Summary

Outline

1. Review: FIR and IIR Filters, and Transfer Functions
2. Effects of numerical errors on poles and zeros
3. Series connection of second-order sections
4. Parallel connection of second-order sections
5. Partial fraction expansion
6. Summary

System Functions

A first-order autoregressive filter,

$$y[n] = x[n] - bx[n-1] + ay[n-1],$$

has the impulse response and transfer function

$$h[n] = a^n u[n] - ba^{n-1} u[n-1] \leftrightarrow H(z) = \frac{1 - bz^{-1}}{1 - az^{-1}},$$

where a is called the **pole** of the filter, and b is called its **zero**.

Impulse Response of a Second-Order All-Pole Filter

A general second-order all-pole filter has the system function

$$H(z) = \frac{1}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})} = \frac{1}{1 - (p_1 + p_2)z^{-1} + p_1 p_2 z^{-2}}$$

If the filter is underdamped, then $p_2 = p_1^*$, and we can write

$$H(z) = \frac{1}{1 - 2e^{-\sigma_1}\cos(\omega_1)z^{-1} + e^{-2\sigma_1}z^{-2}}$$

and

$$h[n] = \frac{e^{-\sigma_1(n+1)}\sin(\omega_1(n+1))}{e^{-\sigma_1}\sin(\omega_1)}u[n]$$

Outline

1. Review: FIR and IIR Filters, and Transfer Functions
2. Effects of numerical errors on poles and zeros
3. Series connection of second-order sections
4. Parallel connection of second-order sections
5. Partial fraction expansion
6. Summary

Transfer function of a general difference equation

Suppose you have a set of N poles and N zeros, including complex conjugate pairs, that you want to implement as a filter. Your desired $H(z)$ is:

$$H(z) = \frac{\prod_{i=1}^{N}(1 - z_i z^{-1})}{\prod_{i=1}^{N}(1 - p_i z^{-1})}$$

The first step is to multiply them out to find the coefficients:

$$H(z) = \frac{1 - \sum_{k=1}^{N} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}$$

Then implement it as a difference equation:

```
y[n] = x[n] - sum(b*x[n-1:n-N-1:-1])\\
+ sum(a*y[n-1:n-N-1:-1])
```

What about errors?

Suppose now we've implemented the difference equation

$$y[n] = x[n] - \sum_{k=1}^{N} b_k x[n-k] + \sum_{k=1}^{N} a_k y[n-k]$$

But now, suppose there are some errors:

- Errors in $x[n]$ might be caused by noise, or by quantization
- Errors in a_k might be caused by quantization, or by computation mistakes

How badly do those errors effect $y[n]$?

The effect of errors gets worse with higher-order filters

- Suppose $x[n]$ has some errors. If those errors are near a pole of $H(z)$, then $y[n]$ will have exponentially worse errors.
  - The number of poles gets larger as N gets larger.
- Suppose a_k has an error. Its effect on the poles, p_i, can be estimated using something called the implicit function theorem.
  - The effect of each error, and the probability of a large error, increases exponentially as N increases.
- Both of these effects are much worse for a_k than for b_k, because poles increase the signal amplitude, whereas zeros decrease it.

Reducing errors $\Leftrightarrow$ Decompose the filter into sections

- Error increases exponentially with N
- We can reduce the dependence to linear by decomposing the filter into a number of small sub-filters:

$$H(z) = H_1(z) + H_2(z) + \cdots + H_K(z)$$

Then

$$y[n] = (h_1[n] * x[n]) + \cdots + (h_K[n] * x[n])$$

So the total error in $y[n]$ is just the sum of the errors in the sub-filters, i.e., linear dependence.

- Let's talk about how to do that decomposition.

Outline

1. Review: FIR and IIR Filters, and Transfer Functions
2. Effects of numerical errors on poles and zeros
3. Series connection of second-order sections
4. Parallel connection of second-order sections
5. Partial fraction expansion
6. Summary

Why use block diagrams?

A first-order difference equation looks like

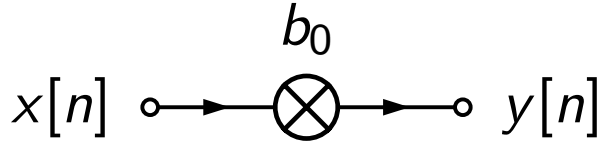
$$y[n] = b_0 x[n] - b_1 x[n-1] + a y[n-1]$$

- It's pretty easy to understand what computation is taking place in a first-order difference equation.
- As we get to higher-order systems, though, the equations for implementing them will be kind of complicated.
- In order to make the complicated equations very easy, we represent the equations using block diagrams.

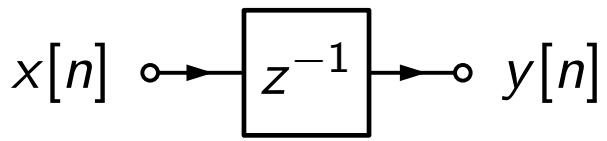
Elements of a block diagram

A block diagram has just three main element types:

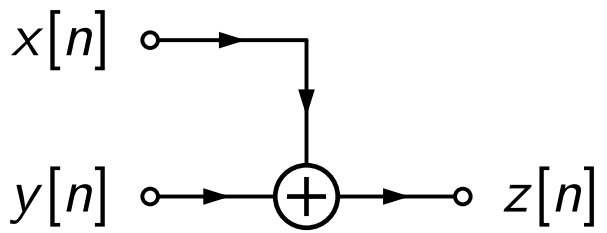
1. **Multiplier:** the following element means $y[n] = b_0 x[n]$:



2. **Unit Delay:** the following element means $y[n] = x[n-1]$ (i.e., $Y(z) = z^{-1}X(z)$):

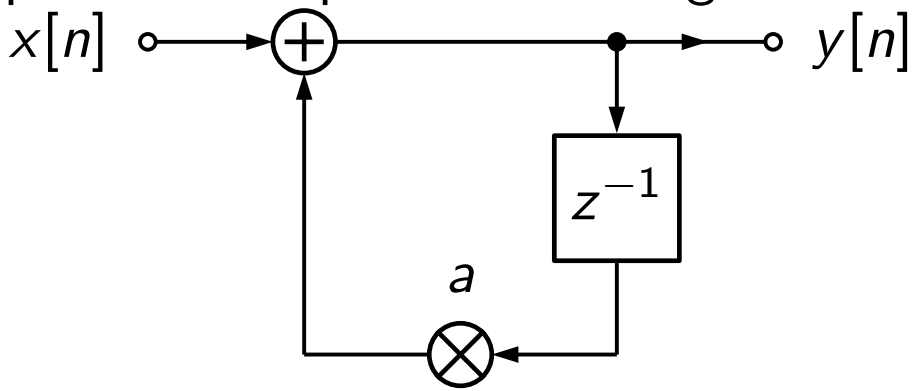


3. **Adder:** the following element means $z[n] = x[n] + y[n]$:



Example: Time Domain

Here's an example of a complete block diagram:



This block diagram is equivalent to the following equation:

$$y[n] = x[n] + ay[n-1]$$

Notice that we can read it, also, as

$$Y(z) = X(z) + az^{-1}Y(z) \quad \Rightarrow \quad H(z) = \frac{1}{1 - az^{-1}}$$

A Complete First-Order IIR Filter

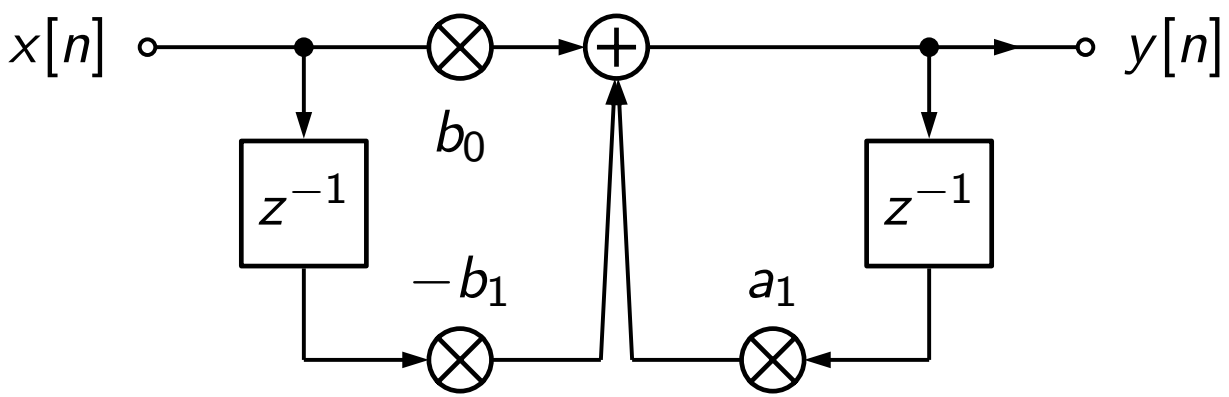
Now consider how we can represent a complete first-order IIR filter, including both the pole and the zero. Here it is in the z-domain:

$$Y(z) = b_0 X(z) - b_1 z^{-1} X(z) + a_1 z^{-1} Y(z).$$

When we implement it, we would write a line of python that does this:

$$y[n] = b_0 x[n] - b_1 x[n-1] + a_1 y[n-1],$$

which is exactly this block diagram:



Series and Parallel Combinations

Now let's talk about how to combine systems.

- **Series combination**: passing the signal through two systems **in series** is like multiplying the system functions:

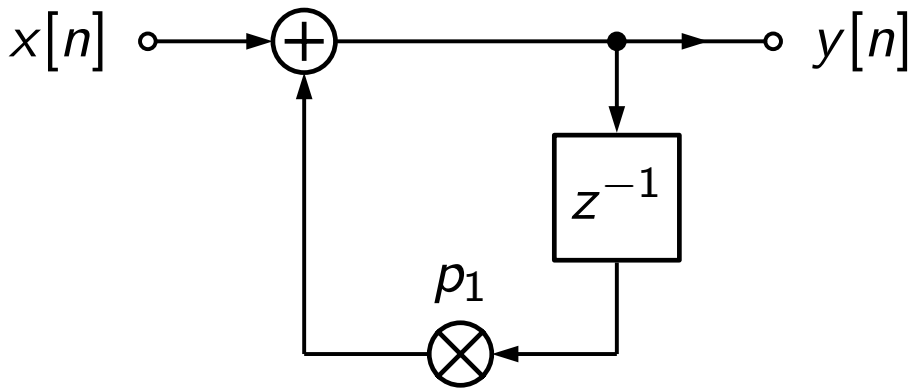
$$H(z) = H_2(z)H_1(z)$$

- **Parallel combination**: passing the signal through two systems in **parallel**, then adding the outputs, is like adding the system functions:

$$H(z) = H_1(z) + H_2(z)$$

One Block for Each System

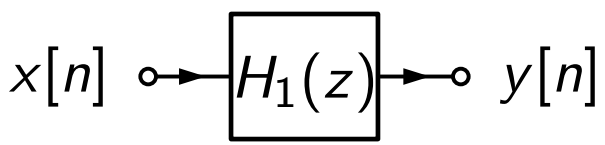
Suppose that one of the two systems, $H_1(z)$, looks like this:



and has the system function

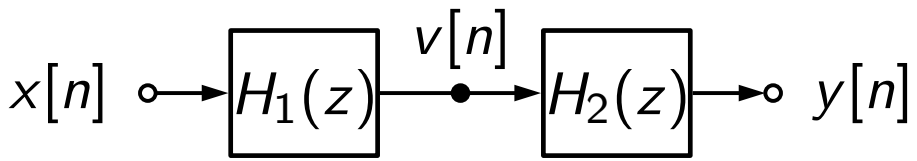
$$H_1(z) = \frac{1}{1 - p_1 z^{-1}}$$

Let's represent the whole system using a single box:



Series Combination

The series combination, then, looks like this:



This means that

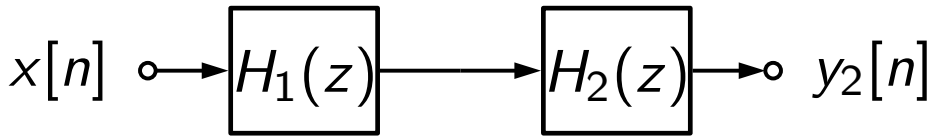
$$Y(z) = H_2(z)V(z) = H_2(z)H_1(z)X(z)$$

and therefore

$$H(z) = \frac{Y(z)}{X(z)} = H_1(z)H_2(z)$$

Series Combination

The series combination, then, looks like this:



Suppose that we know each of the systems separately:

$$H_1(z) = \frac{1}{1 - p_1 z^{-1}}, \qquad H_2(z) = \frac{1}{1 - p_2 z^{-1}}$$

Then, to get $H(z)$, we just have to multiply:

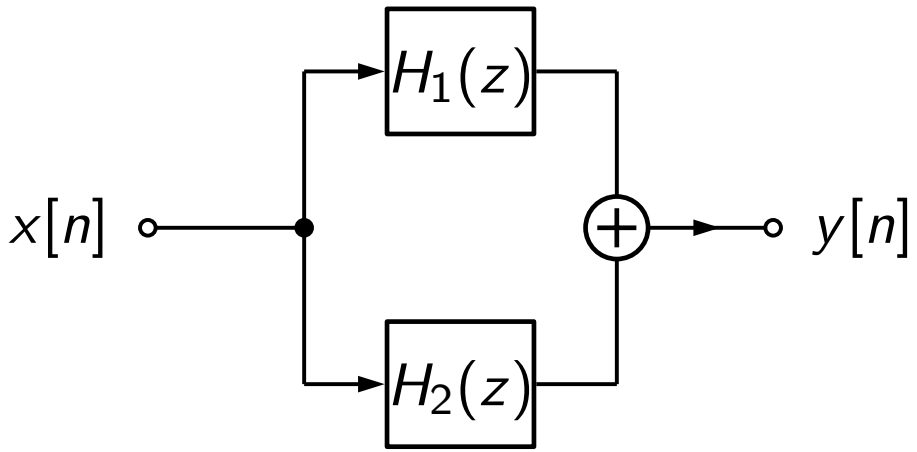
$$H(z) = \frac{1}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})} = \frac{1}{1 - (p_1 + p_2) z^{-1} + p_1 p_2 z^{-2}}$$

Outline

1. Review: FIR and IIR Filters, and Transfer Functions
2. Effects of numerical errors on poles and zeros
3. Series connection of second-order sections
4. Parallel connection of second-order sections
5. Partial fraction expansion
6. Summary

Parallel Combination

Parallel combination of two systems looks like this:



This means that

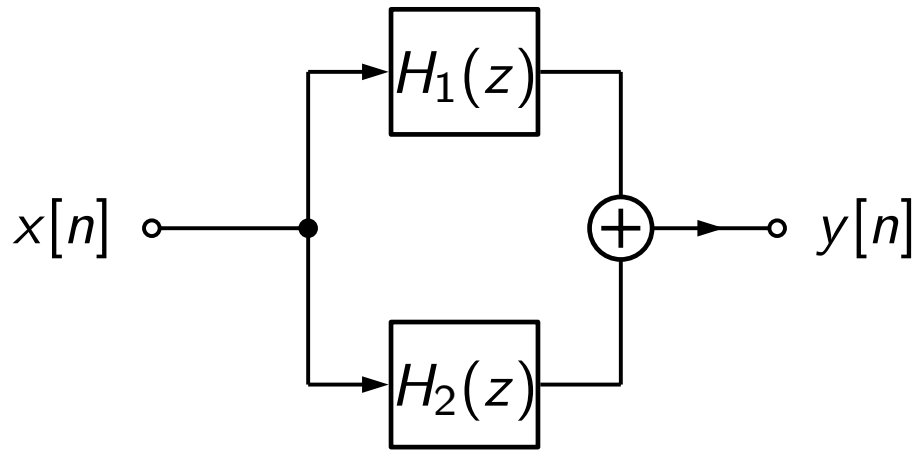
$$Y(z) = H_1(z)X(z) + H_2(z)X(z)$$

and therefore

$$H(z) = \frac{Y(z)}{X(z)} = H_1(z) + H_2(z)$$

Parallel Combination

Parallel combination of two systems looks like this:



Suppose that we know each of the systems separately:

$$H_1(z) = \frac{1}{1 - p_1 z^{-1}}, \qquad H_2(z) = \frac{1}{1 - p_2 z^{-1}}$$

Then, to get $H(z)$, we just have to add:

$$H(z) = \frac{1}{1 - p_1 z^{-1}} + \frac{1}{1 - p_2 z^{-1}} = \frac{2 - (p_1 + p_2) z^{-1}}{1 - (p_1 + p_2) z^{-1} + p_1 p_2 z^{-2}}$$

Second-order sections

- Until now, we've pretended that you want to decompose $H(z)$ into first-order sections.
- That's never true, because if p_1 is complex, then the first-order filter has complex-valued coefficients:

$$y[n] = x[n] + p_1 y[n-1]$$

- Real implementations like `scipy.signal.sosfilt` decompose an N^{th}-order filter into $\frac{N}{2}$ second-order sections. Each section has poles $p_2 = p_1^*$, so its coefficients are real-valued:

$$y[n] = x[n] - by[n-1] + cy[n-2]$$

where $b = 2\Re(p_1)$ is real, and $c = |p_1|^2$ is also real.

Outline

1. Review: FIR and IIR Filters, and Transfer Functions
2. Effects of numerical errors on poles and zeros
3. Series connection of second-order sections
4. Parallel connection of second-order sections
5. Partial fraction expansion
6. Summary

Parallel decomposition

Suppose you know $H(z)$, and you want to find $H_1(z), \ldots, H_N(z)$ so that

$$H(z) = H_1(z) + \cdots + H_N(z)$$

How can we find $H_1(z)$ through $H_N(z)$?

Step #1: Numerator Terms

Suppose you already know that

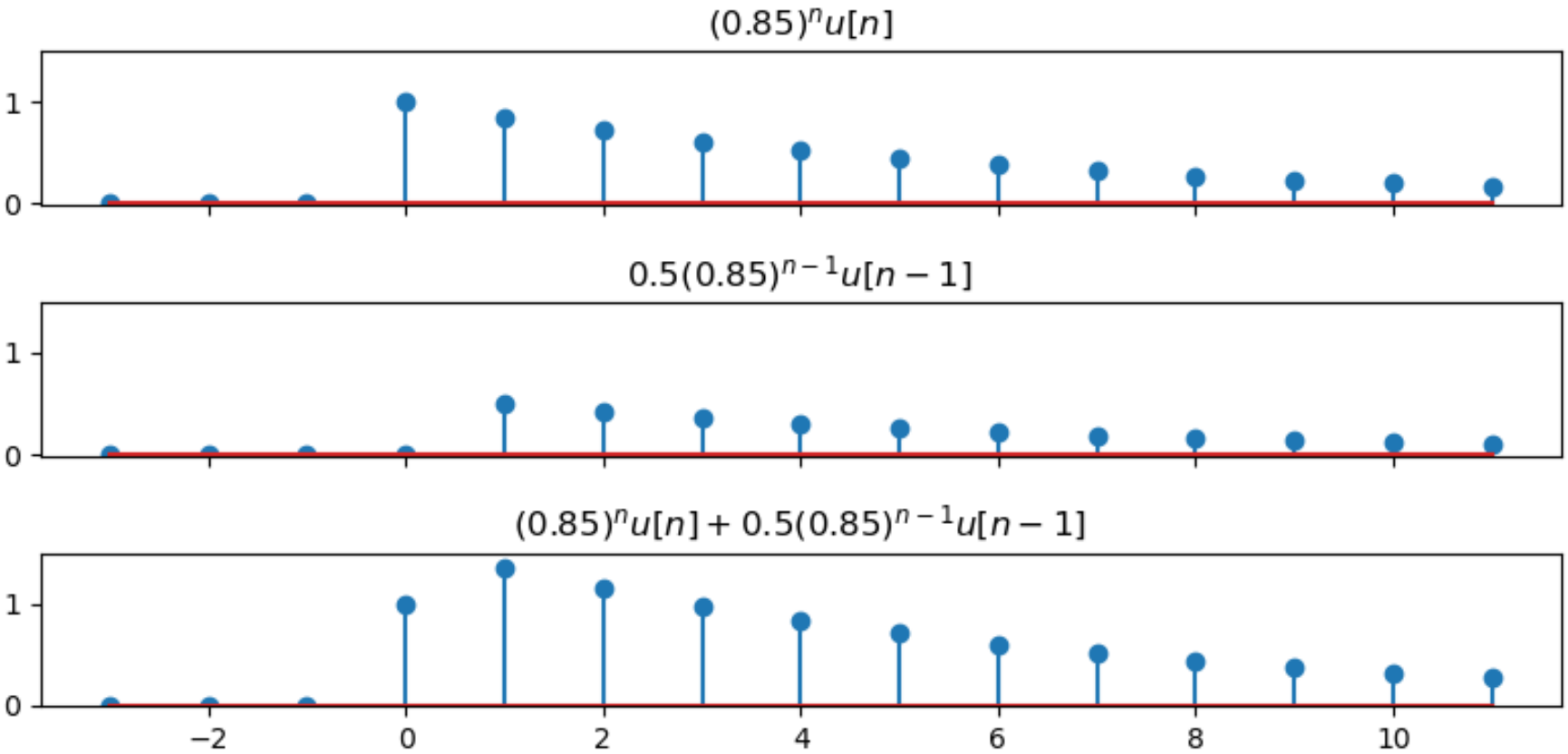
$$H(z) = \frac{1 - \sum_{k=1}^{N} b_k z^{-k}}{A(z)} = G(z) \left(1 - \sum_{k=1}^{N} b_k z^{-k} \right)$$

and suppose you already know $g[n]$, the inverse transform of $G(z) = 1/A(z)$. Then $h[n]$ is just

$$h[n] = g[n] - \sum_{k=1}^{N} b_k g[n-k]$$

Step #1: Numerator Terms

For example, here is the inverse transform of $H(z) = \frac{1+0.5z^{-1}}{1-0.85z^{-1}}$:



Step #2: Denominator Terms

Now we need to figure out how to decompose

$$G(z) = \frac{1}{A(z)}$$

We will solve this using a method called **partial fraction expansion.**

Step #2: Partial Fraction Expansion

Partial fraction expansion works like this:

1. Factor $A(z)$:
$$G(z) = \frac{1}{\prod_{\ell=1}^{N}(1 - p_\ell z^{-1})}$$
2. Assume that $G(z)$ is the result of a parallel system combination:
$$G(z) = \frac{C_1}{1 - p_1 z^{-1}} + \frac{C_2}{1 - p_2 z^{-1}} + \cdots$$
3. Find the constants, C_ℓ, that make the equation true. Such constants always exist, as long as none of the roots are repeated ($p_k \neq p_\ell$ for $k \neq \ell$).

Partial Fraction Expansion: Example

Step # 1: Factor it:

$$\frac{1}{1-1.2z^{-1}+0.72z^{-2}} = \frac{1}{(1-(0.6+j0.6)z^{-1})(1-(0.6-j0.6)z^{-1})}$$

Step #2: Express it as a sum:

$$\frac{1}{1-1.2z^{-1}+0.72z^{-2}} = \frac{C_1}{1-(0.6+j0.6)z^{-1}} + \frac{C_2}{1-(0.6-j0.6)z^{-1}}$$

Step #3: Find the constants.

Partial Fraction Expansion: Example

The system function is:

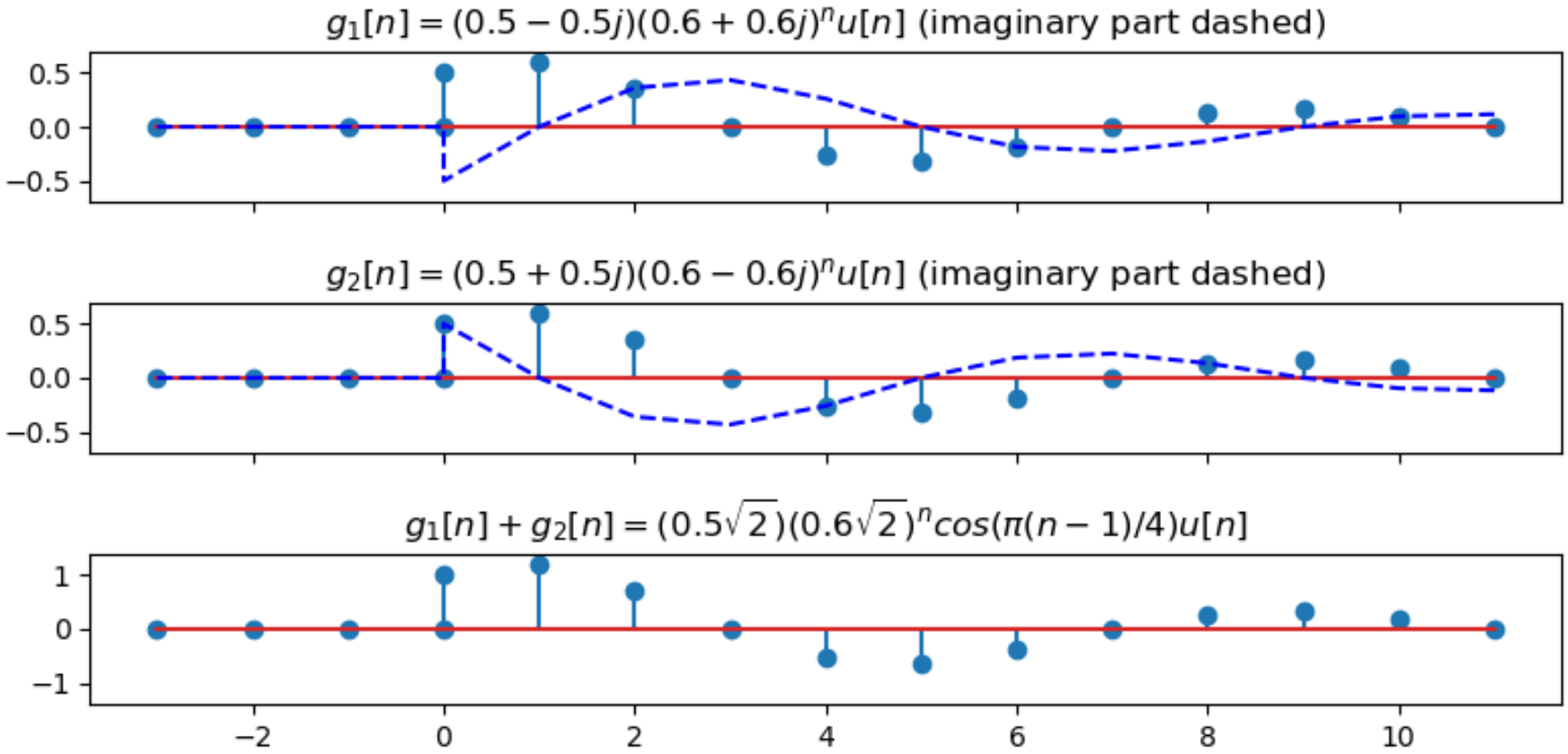
$$G(z) = \frac{1}{1 - 1.2z^{-1} + 0.72z^{-2}}$$
$$= \frac{C_1}{1 - (0.6 + j0.6)z^{-1}} + \frac{C_2}{1 - (0.6 - j0.6)z^{-1}}$$

and therefore the impulse response is:

$$g[n] = C_1(0.6 + 0.6j)^n u[n] + C_2(0.6 - j0.6)^n u[n]$$

After finding the constants and simplifying, we get

$$g[n] = \sqrt{2}(0.6\sqrt{2})^n \cos\left(\frac{\pi}{4}(n-1)\right) u[n]$$


$g_1[n] = (0.5 - 0.5j)(0.6 + 0.6j)^n u[n]$ (imaginary part dashed)
0.5
0.0
−0.5
$g_2[n] = (0.5 + 0.5j)(0.6 - 0.6j)^n u[n]$ (imaginary part dashed)
0.5
0.0
−0.5
$g_1[n] + g_2[n] = (0.5\sqrt{2})(0.6\sqrt{2})^n cos(\pi(n-1)/4)u[n]$
1
0
−1
−2
0
2
4
6
8
10

How to find the inverse Z transform

Any IIR filter $H(z)$ can be written as. . .

- a **partial fraction expansion** into a sum of **denominator** terms, each with this form:

$$G_\ell(z) = \frac{1}{1 - az^{-1}} \quad \leftrightarrow \quad g_\ell[n] = a^n u[n],$$

- each possibly multiplied by a **numerator** term, like this one:

$$D_k(z) = b_k z^{-k} \quad \leftrightarrow \quad d_k[n] = b_k \delta[n - k].$$

How to implement second-order sections

Any IIR filter $H(z)$ can be written as. . .

- a **partial fraction expansion** into a sum of **denominator** terms, each with this form:

$$G_\ell(z) = \frac{1}{1 - az^{-1}} \quad \leftrightarrow \quad g_\ell[n] = a^n u[n],$$

  . . . which you then combine in complex-conjugate pairs, to create $\frac{N}{2}$ second-order sections with real coefficients.
- each possibly multiplied by a **numerator** term, like this one:

$$D_k(z) = b_k z^{-k} \quad \leftrightarrow \quad d_k[n] = b_k \delta[n - k].$$

Outline

1. Review: FIR and IIR Filters, and Transfer Functions
2. Effects of numerical errors on poles and zeros
3. Series connection of second-order sections
4. Parallel connection of second-order sections
5. Partial fraction expansion
6. Summary

Summary: Block Diagrams

- A **block diagram** shows the delays, additions, and multiplications necessary to compute output from input.
- **Series combination**: passing the signal through two systems **in series** is like multiplying the system functions:

$$H(z) = H_2(z)H_1(z)$$

- **Parallel combination**: passing the signal through two systems in **parallel**, then adding the outputs, is like adding the system functions:

$$H(z) = H_1(z) + H_2(z)$$

Summary: Inverse Z Transform

Any IIR filter $H(z)$ can be written as. . .

- a **partial fraction expansion** into a sum of **denominator** terms, each with this form:

$$G_\ell(z) = \frac{1}{1 - az^{-1}} \quad \leftrightarrow \quad g_\ell[n] = a^n u[n],$$

  . . . which you can then combine in complex-conjugate pairs if you want difference equations with real-valued coefficients.

- each possibly multiplied by a **numerator** term, like this one:

$$D_k(z) = b_k z^{-k} \quad \leftrightarrow \quad d_k[n] = b_k \delta[n - k].$$