Processamento digital de sinais

05 – Estruturas para implementação de filtros digitais

Prof. Carlos Speranza
DAELN/IFSC

Conteúdo

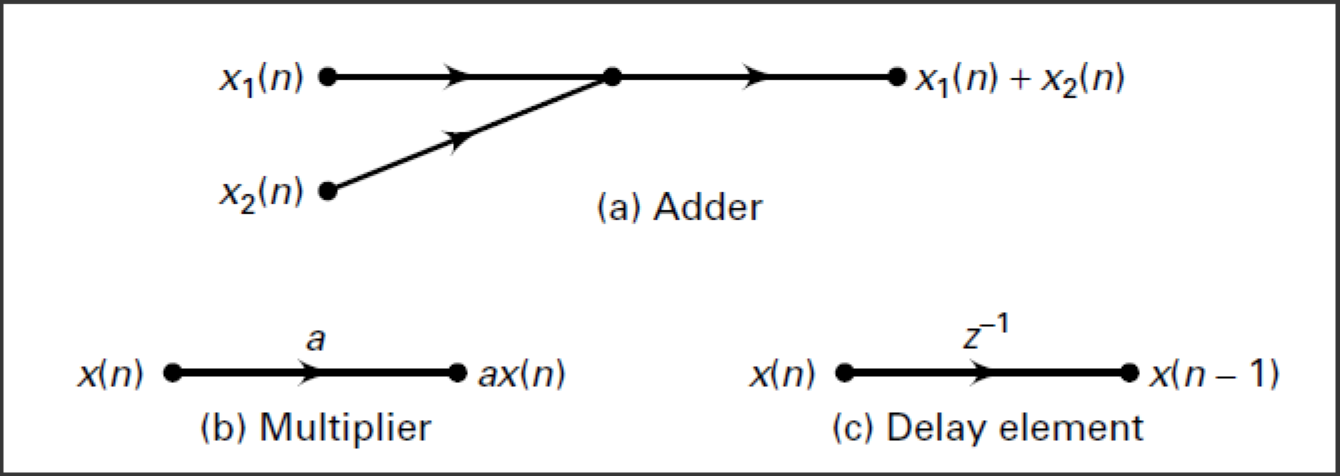
1. Formas de implementação: filtros FIR
2. Formas de implementação: filtros IIR
3. Efeitos numéricos da precisão finita: quantização dos coeficientes

Introdução

- Diferentes formas de implementação de sistemas digitais são possíveis pois existem diversas **estruturas computacionais com** mesma relação entre as sequências de entrada $x[n]$ e saída $y[n]$ - **supondo precisão infinita**.
- No entanto, se levarmos em conta efeitos de quantização de coeficientes e de operações (**precisão numérica finita**), as diferentes estruturas podem geram resultados diversos.
- Além disso, as estruturas podem apresentar diferenças no número de operações e de coeficientes presentes.

Representação gráfica dos sistemas

- Os elementos gráficos básicos utilizados aqui serão:
  - Somadores (Adder)
  - Multiplicadores (Multiplier)
  - Blocos de atraso – memória (Delay element)



Implementação de filtros FIR

- Não possuem laços de realimentação, com resposta ao impulso diferente de zero para (nº *delays* + 1) amostras (no maior caminho)
- São sempre estáveis (todos os polos na origem).
- Função de transferência e equação de diferenças:

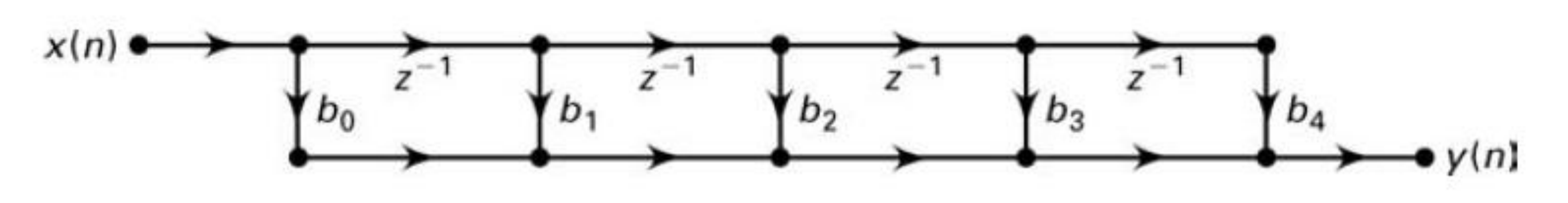
$$H(z) = b_0 + b_1 z^{-1} + \cdots + b_{M-1} z^{1-M} = \sum_{n=0}^{M-1} b_n z^{-n}$$

$$y(n) = b_0 x(n) + b_1 x(n-1) + \cdots + b_{M-1} x(n-M+1)$$

- Formas de implementação: **direta, cascata, fase-linear**, *lattice* e amostragem em frequência.

Forma Direta (FIR)

- Forma mais simples, pois é a implementação direta da equação de diferenças a coeficientes constantes:



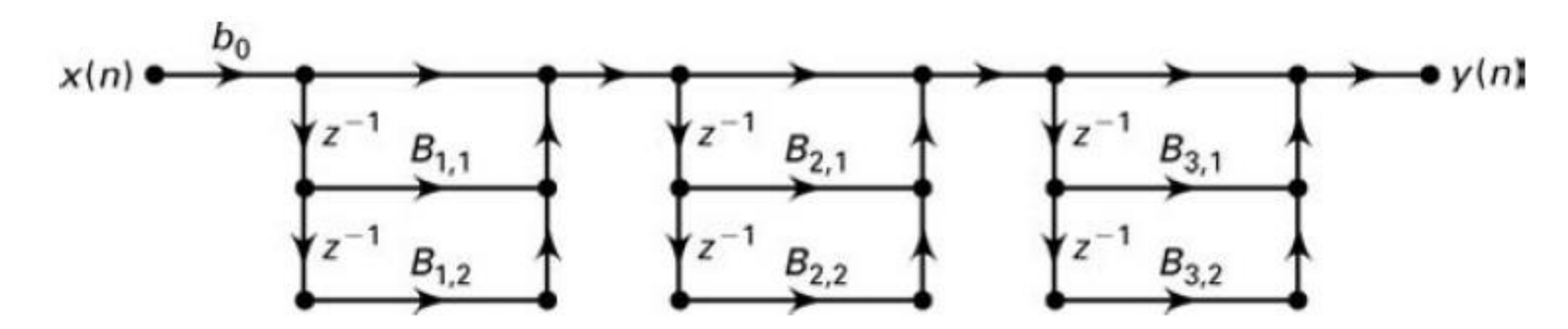
- No entanto, esta implementação possui **alta sensibilidade** aos seus coeficientes: uma pequena variação em apenas <u>um</u> coeficiente b_i afeta a posição de <u>todos</u> os zeros (e toda resposta muda).

- Python: função *lfilter* (*scipy*), com vetor de parâmetros igual a 1.

Forma cascata (FIR)

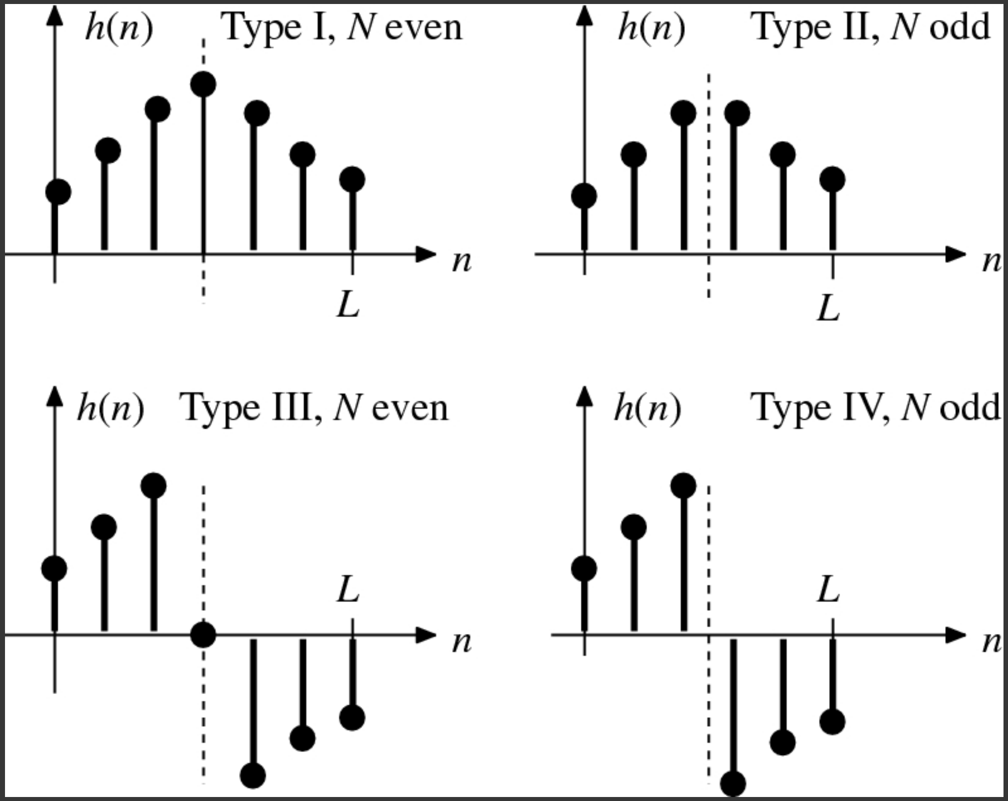
- Maneira de "isolar" os zeros conjugados de cada estágio ou seção
- $H(z)$ deve ser convertida em produtos de seções de 2ª ordem com coeficientes reais:

$$
\begin{aligned}
H(z) &= b_0 + b_1 z^{-1} + \cdots + b_{M-1} z^{-M+1} \\
&= b_0 \left(1 + \frac{b_1}{b_0} z^{-1} + \cdots + \frac{b_{M-1}}{b_0} z^{-M+1}\right) \\
&= b_0 \prod_{k=1}^{K} \left(1 + B_{k,1} z^{-1} + B_{k,2} z^{-2}\right)
\end{aligned}
$$



Forma de fase linear (FIR)

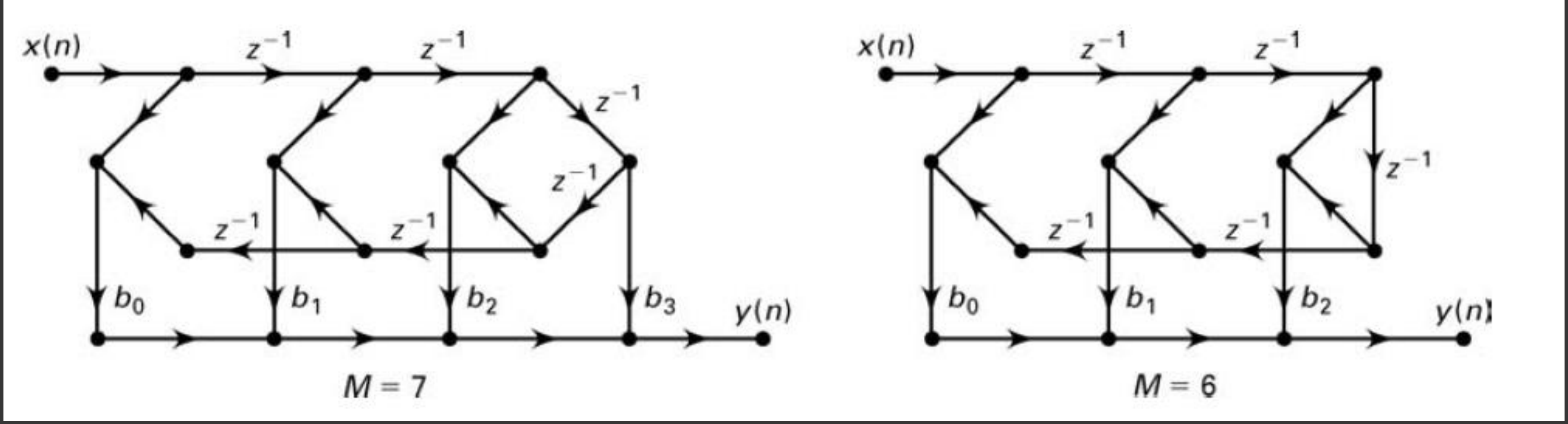
Nos casos em que fase é linear, a resposta ao impulso $h[n]$ do sistema FIR será simétrica ou antissimétrica



Forma de fase linear (FIR)

- Então a equação de diferenças no caso de resposta ao impulso simétrica (por exemplo) é dada por:

$$
\begin{aligned}
y(n) &= b_0 x(n) + b_1 x(n-1) + \cdots + b_1 x(n-M+2) + b_0 x(n-M+1) \\
&= b_0[x(n) + x(n-M+1)] + b_1[x(n-1) + x(n-M+2)] + \cdots
\end{aligned}
$$



Exemplo 6.4

- Dado o seguinte filtro FIR, determinar e desenhar as formas direta, fase linear e cascata

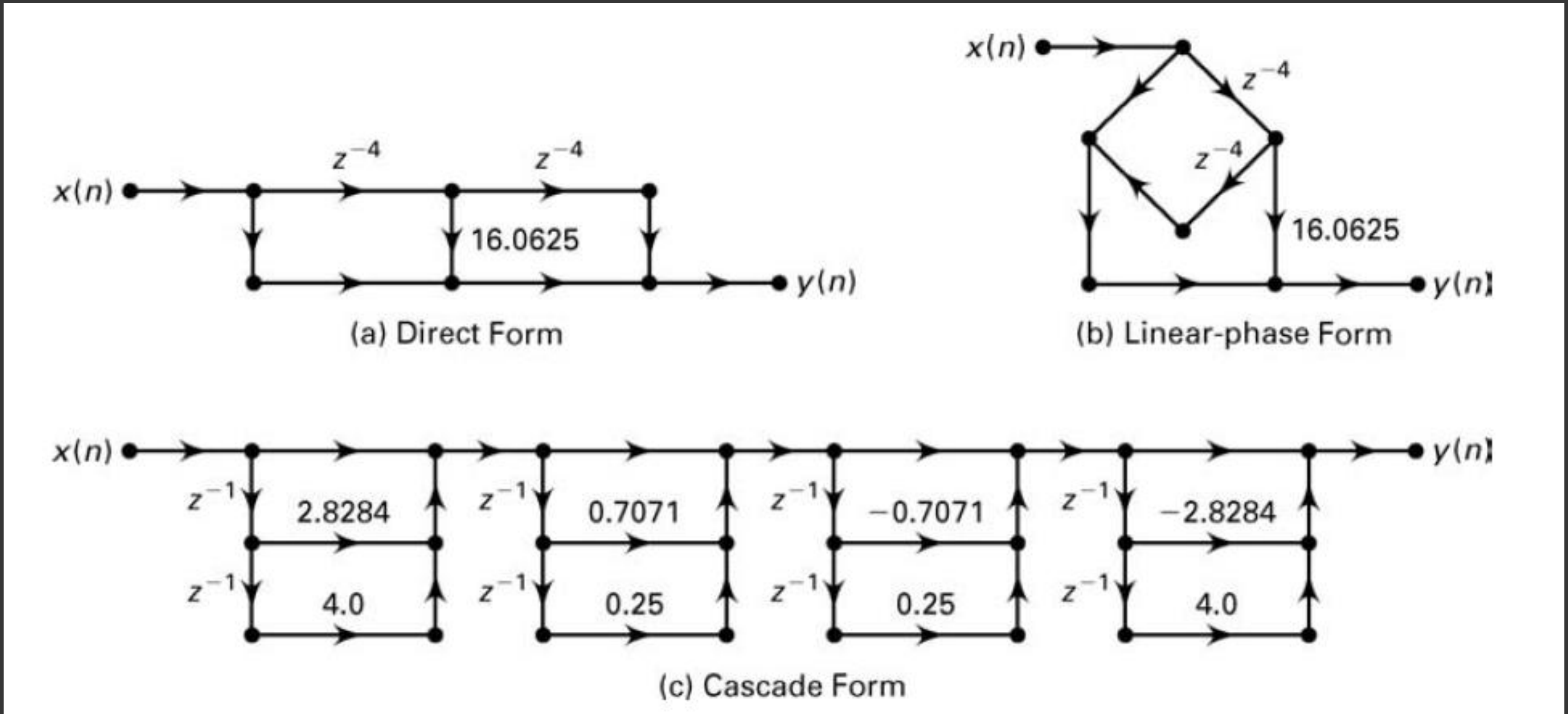
$$H(z) = 1 + \left(16 + \frac{1}{16}\right) z^{-4} + z^{-8}$$

- Forma direta: $y(n) = x(n) + 16.0625x(n-4) + x(n-8)$

- Forma fase linear: $y(n) = [x(n) + x(n-8)] + 16.0625x(n-4)$

- Forma cascata: usar *tf2sos(b,[1])*

Exemplo 6.4



x(n)
z^{-4}
z^{-4}
16.0625
y(n)
(a) Direct Form
x(n)
z^{-4}
z^{-4}
16.0625
y(n)
(b) Linear-phase Form
x(n)
y(n)
z^{-1}
2.8284
0.7071
−0.7071
−2.8284
4.0
0.25
0.25
4.0
(c) Cascade Form

Implementação de filtros IIR

- Possuem laços de realimentação, resultando em realizações recursivas e podem ter comportamento instável (polo estiver fora do circulo de raio unitário).

- Função de transferência e equação de diferenças:

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{n=0}^{M} b_n z^{-n}}{\sum_{n=0}^{N} a_n z^{-n}} = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}};\ a_0 = 1$$

$$y(n) = \sum_{m=0}^{M} b_m x(n-m) - \sum_{m=1}^{N} a_m y(n-m)$$

- Principais formas de implementação: **diretas (1 e 2), cascata**, paralela e *lattice*.

Formas Diretas 1 e 2 (IIR)

- A equação de diferenças é implementada diretamente em 2 etapas: recursiva - denominadores de $H(z)$ e não-recursiva - numeradores de $H(z)$.

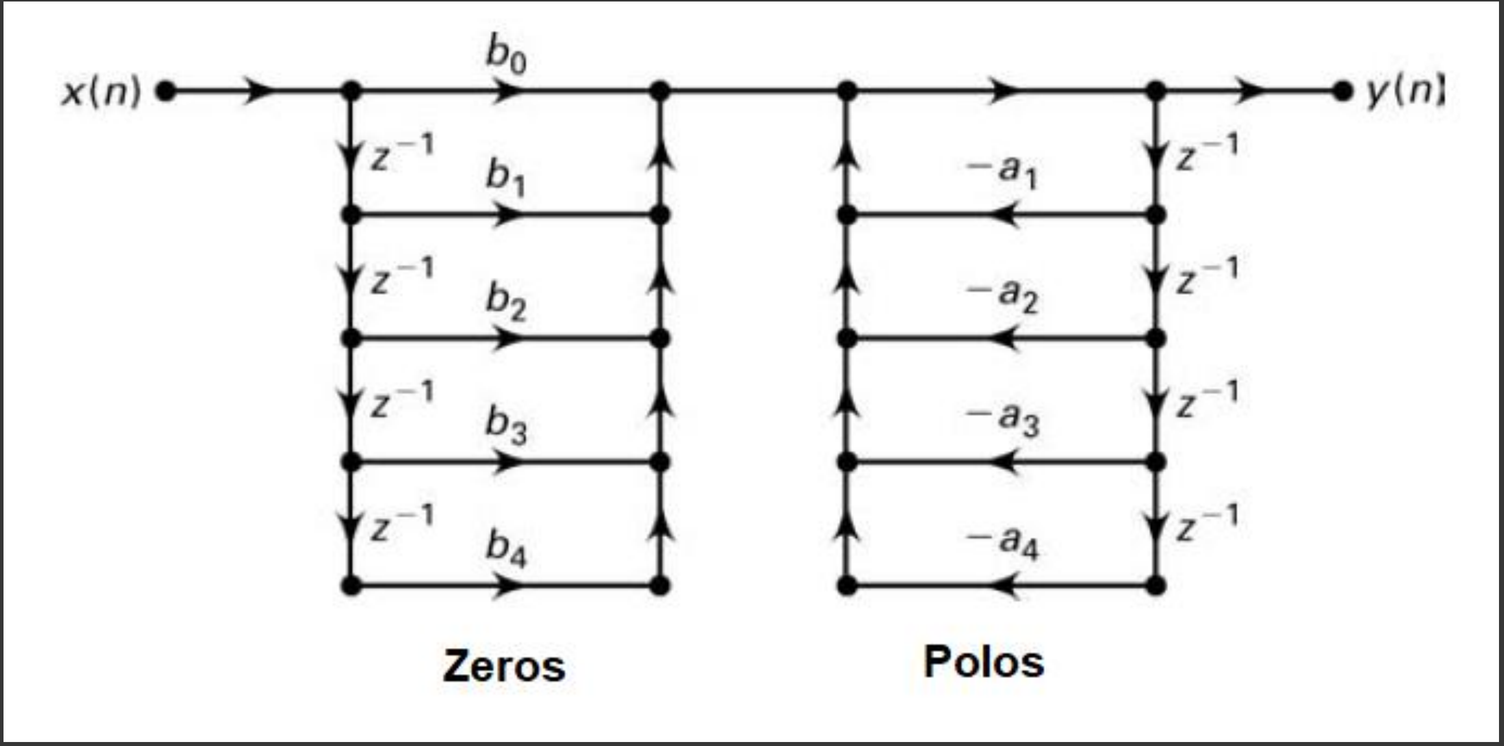
- Na **forma direta 1** os zeros são implementados primeiro, enquanto que na **forma direta 2** ocorre o inverso.

- A **forma direta 2 canônica** (mais comum) apresenta a vantagem de minimizar o número de atrasos necessários (espaço de memória num sistema microprocessado).

Exemplo de Forma Direta 1 (IIR)

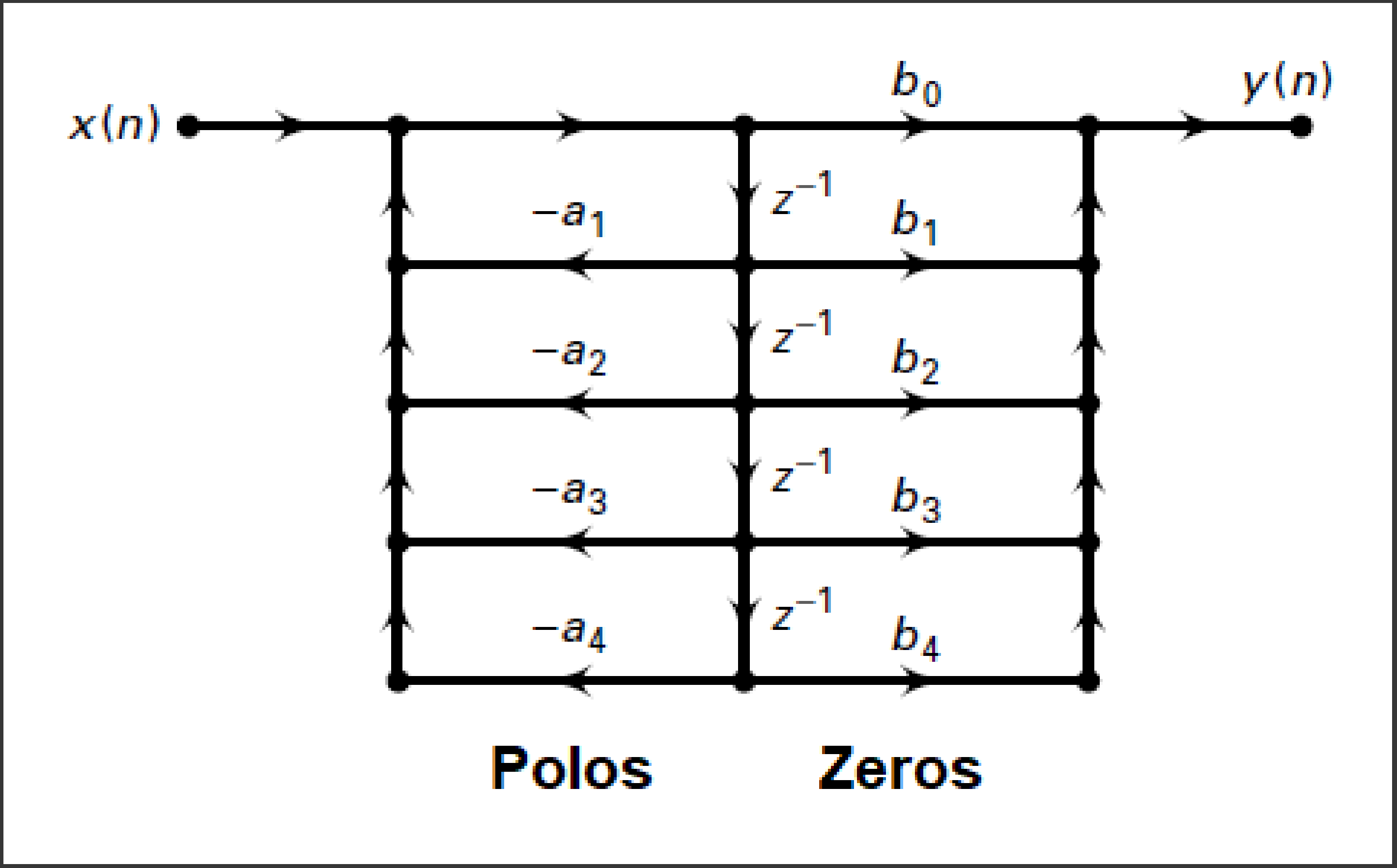
Para a seguinte equação de diferenças com M=N=4:

$$y(n) = b_0x(n) + b_1x(n-1) + b_2x(n-2) + b_3x(n-3) + b_4x(n-4)$$
$$- a_1y(n-1) - a_2y(n-2) - a_3y(n-3) - a_4y(n-4)$$



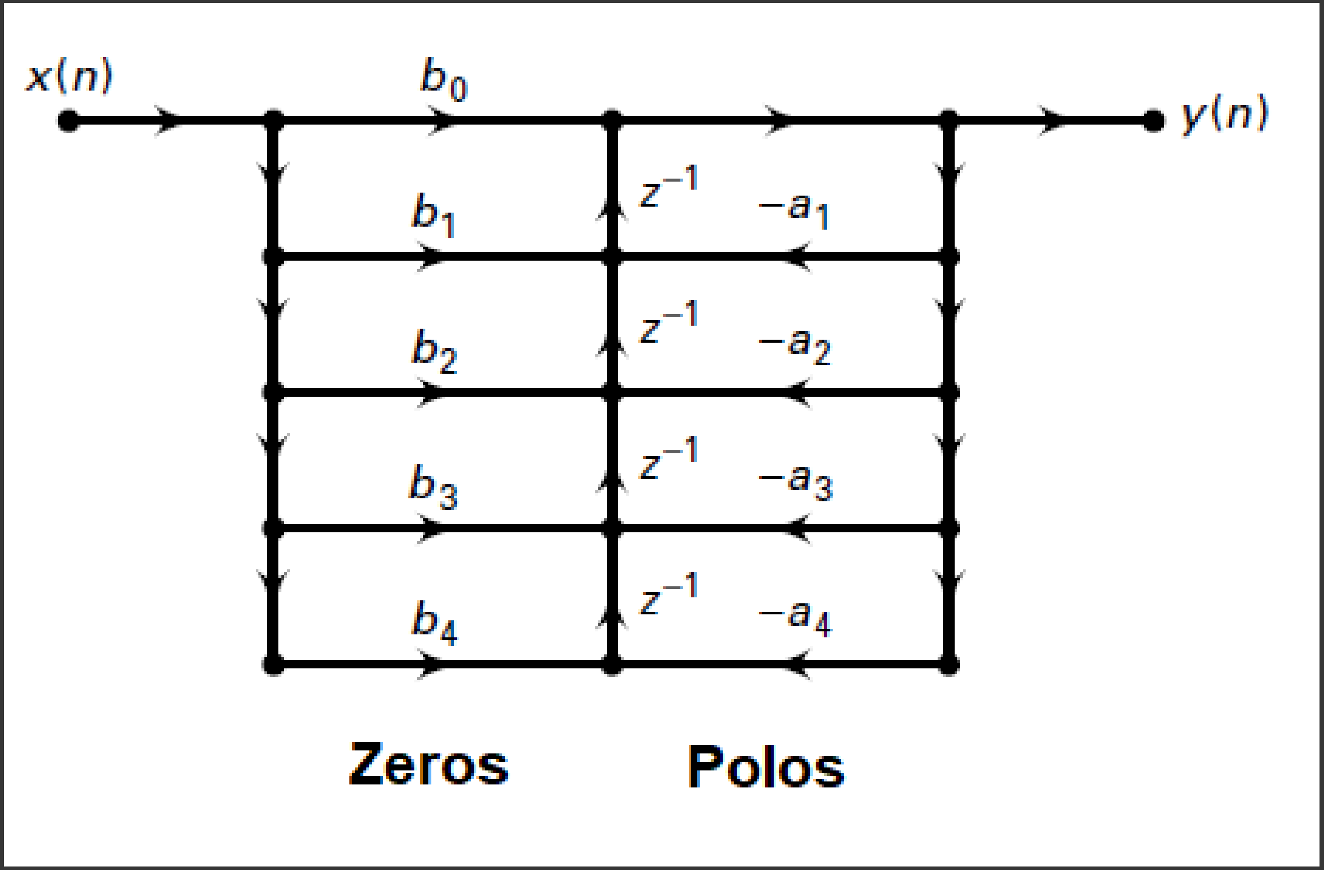
Forma Direta 2 canônica (IIR)

Economiza memória (é a forma implementada pela função *filter* do MATLAB)



Estruturas transpostas

Utilização de propriedades da implementação gráfica das estruturas LTI: reversão do fluxo de dados e troca da entrada pela saída (e vice versa) – implementação pela função *lfilter* do *scipy*.



Desvantagem das formas diretas

- Semelhante aos filtros FIR, as formas diretas 1 e 2 dos IIR possuem **alta sensibilidade** aos seus coeficientes.

- Só que agora, além dos zeros, uma variação em apenas <u>um</u> coeficiente a_i afeta <u>todos</u> os polos, **podendo gerar instabilidade num filtro IIR que era estável na etapa de projeto**.

- Novamente, isso pode ser minimizado dividindo $H(z)$ em seções cascatas de 2ª ordem (biquadradas), contendo um par de polos e de zeros "isolados" em cada uma das seções (gera coeficientes reais devido ao agrupamento de polos e zeros complexos conjugados).

Forma Cascata (IIR)

Transformação da função de transferência total $H(z)$ em uma cascata (produto) de funções parciais de 2ª ordem.

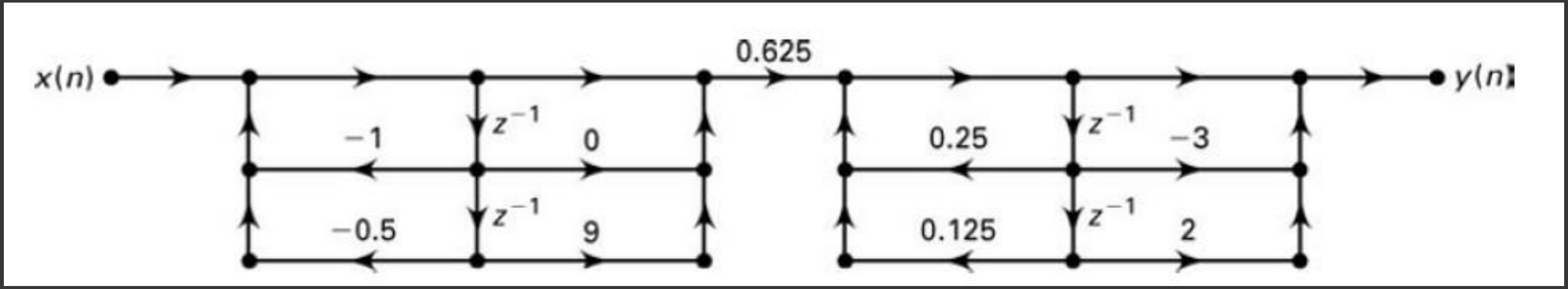
$$
\begin{aligned}
H(z) &= \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \\
&= b_0 \frac{1 + \frac{b_1}{b_0} z^{-1} + \cdots + \frac{b_N}{b_0} z^{-N}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \\
&= b_0 \prod_{k=1}^{K} \frac{1 + B_{k,1} z^{-1} + B_{k,2} z^{-2}}{1 + A_{k,1} z^{-1} + A_{k,2} z^{-2}}
\end{aligned}
$$

Exemplo 6.1

- Encontrar a forma cascata de sistemas de 2º ordem do filtro IIR:

$$\begin{aligned}16y(n) + 12y(n-1) + 2y(n-2) - 4y(n-3) - y(n-4) \\ = x(n) - 3x(n-1) + 11x(n-2) - 27x(n-3) + 18x(n-4)\end{aligned}$$

- Python: usar funções *tf2sos* (calcula coeficientes) e *sosfilt* (aplica filtro), ambos do pacote **scipy**.



Matriz sos

sos — Second-order section representation
matrix

Second-order section representation, specified as a matrix. `sos` is an L-by-6 matrix

$$\text{sos} = \begin{bmatrix} b_{01} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & 1 & a_{12} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{0L} & b_{1L} & b_{2L} & 1 & a_{1L} & a_{2L} \end{bmatrix}$$

whose rows contain the numerator and denominator coefficients b_{ik} and a_{ik} of the second-order sections of $H(z)$:

$$H(z) = g\prod_{k=1}^{L} H_k(z) = g\prod_{k=1}^{L} \frac{b_{0k} + b_{1k}z^{-1} + b_{2k}z^{-2}}{1 + a_{1k}z^{-1} + a_{2k}z^{-2}}.$$

Example: `[2 4 2 6 0 2;3 3 0 6 0 0]` specifies a third-order Butterworth filter with normalized 3 dB frequency 0.5π rad/sample.

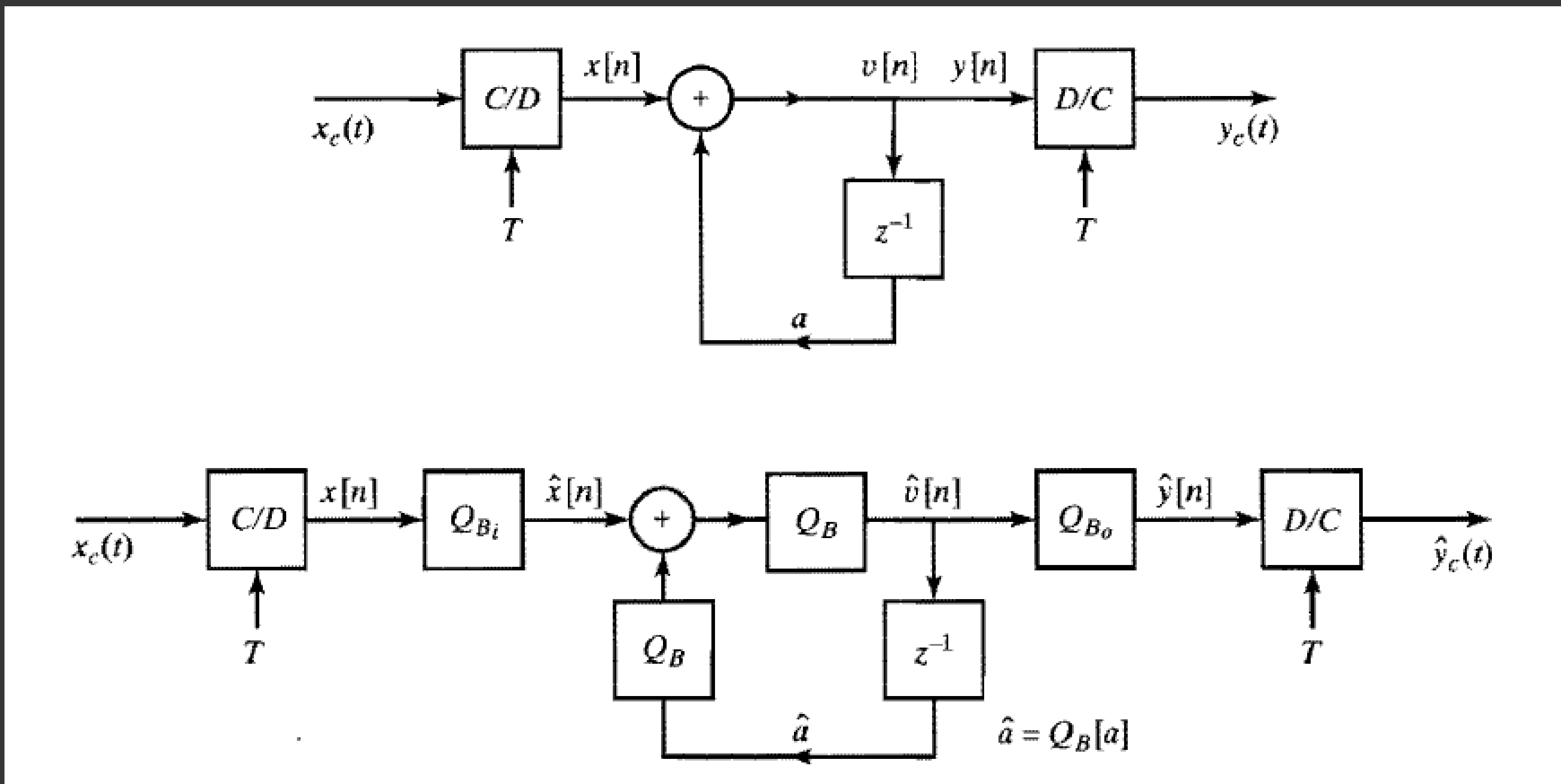
Data Types: `double`
Complex Number Support: Yes

Efeitos numéricos da precisão finita

- Quando um filtro é implementado em sistemas computacionais reais utilizando $B+1$ bits, três efeitos de quantização podem causar alteração na resposta ou comportamento do filtro digital:
  - Quantização da sequência de entrada
  - Quantização dos coeficientes
  - Quantização dos resultados das operações aritméticas

- Pela sua complexidade, usualmente se realiza essa análise por simulação (ex: matlab, python) para investigar parcialmente ou totalmente esses efeitos.

Efeitos numéricos da precisão finita



Quantização dos coeficientes (FIR)

- Para os filtros FIR, embora não gere instabilidade, a quantização dos coeficientes b's pode deteriorar seu desempenho.

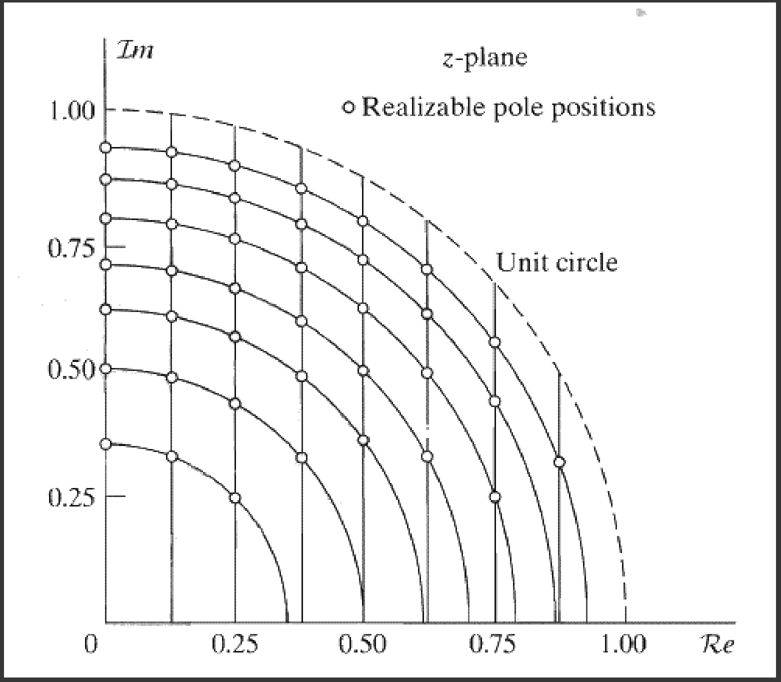
- <u>Exemplo 6.29</u>: filtro passa-baixo FIR de ordem 30 (função remez - **scipy**). Quantização com 8 bits gera filtro com menor atenuação.

- Uso da função QCoeff (fornecida pelo livro e adaptada para o Python)

Quantização dos coeficientes (IIR)

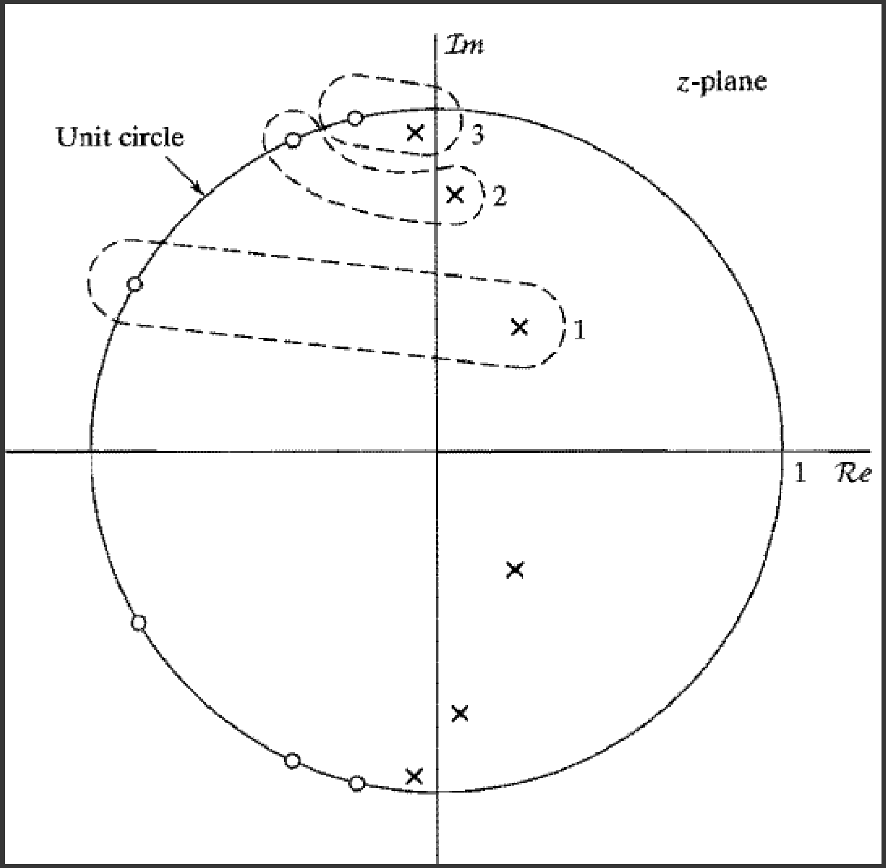
Para os filtros IIR esse efeito pode ser muito crítico, pois pode causar a instabilidade de um sistema estável para precisão infinita.

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \rightarrow \hat{H}(z) \triangleq \frac{\sum_{k=0}^{M} \hat{b}_k z^{-k}}{1 + \sum_{k=1}^{N} \hat{a}_k z^{-k}}$$



Forma cascata (IIR)

- É menos sensível do que a forma direta, principalmente no caso de polos próximos
- O casamento e ordenamento polos/zeros e a ordem dos sistemas são importantes quando são levados em conta os efeitos da quantização



Exemplo 6.26/6.27

- Filtro IIR com 10 polos próximos, num raio de r = 0.9 em torno de ângulo de ±45° (separação de 5°). Calcular **localização dos polos** e a **resposta em magnitude** com:
  1. Forma direta e precisão infinita
  2. Forma direta e 16 bits de precisão (1 + 6 + 9)*
  3. Forma cascata e 16 bits de precisão (1 + 1 + 14)*
  4. Forma cascata e 11 bits de precisão (1 + 1 + 9)*

- Uso da função QCoeff (fornecida pelo livro e adaptada para Python)

* (bit sinal + bits parte inteira + bits parte fracionária)