# Relatório – ST9: Find the transfer function

## 1. Enunciado

## Texto O conteúdo gerado por IA pode estar incorreto.

## 2. Achando a fórmula

## 3. Código Python

import numpy as np  
import matplotlib.pyplot as plt  
from scipy.signal import TransferFunction, zpk2tf, tf2zpk  
  
  
poles = np.array([1/3, -1/6])  
zeros = np.array([1/2])  
gain = 1 # Assumindo ganho K=1  
  
print(f"Polos: {poles}")  
print(f"Zeros: {zeros}")  
print(f"Ganho K: {gain}")  
  
  
  
num\_poly\_z, den\_poly\_z = zpk2tf(zeros, poles, gain)  
  
print(f"\nCoeficientes do numerador (polinômio em z): {num\_poly\_z}")  
print(f"Coeficientes do denominador (polinômio em z): {den\_poly\_z}")  
  
  
if len(num\_poly\_z) < len(den\_poly\_z):  
 num\_poly\_z\_padded = np.concatenate(([0]\*(len(den\_poly\_z) - len(num\_poly\_z) -1), num\_poly\_z)) # Ajuste para que b0 seja para z^0  
else:  
 num\_poly\_z\_padded = num\_poly\_z  
  
  
b\_coeffs\_z\_inv = np.array([0, 1, -0.5]) # Coefs de x[n], x[n-1], x[n-2]  
a\_coeffs\_z\_inv = np.array([1, -1/6, -1/18])# Coefs de y[n], y[n-1], y[n-2]  
  
print(f"\nCoeficientes b (numerador para z^-1): {b\_coeffs\_z\_inv} (para x[n], x[n-1], x[n-2])")  
print(f"Coeficientes a (denominador para z^-1): {a\_coeffs\_z\_inv} (para y[n], y[n-1], y[n-2])")  
  
  
# y[n] = (1/6)y[n-1] + (1/18)y[n-2] + x[n-1] - (1/2)x[n-2]  
print("\n--- Equação de Diferenças ---")  
eq\_str = f"y[n] = ({a\_coeffs\_z\_inv[1]:.4f})\*y[n-1] + ({a\_coeffs\_z\_inv[2]:.4f})\*y[n-2] " \  
 f"+ ({b\_coeffs\_z\_inv[1]:.4f})\*x[n-1] + ({b\_coeffs\_z\_inv[2]:.4f})\*x[n-2]"  
# Corrigindo os sinais para a forma y[n] = ...  
# a[0]y[n] + a[1]y[n-1] + a[2]y[n-2] = b[0]x[n] + b[1]x[n-1] + b[2]x[n-2]  
# y[n] = (-a[1]/a[0])y[n-1] + (-a[2]/a[0])y[n-2] + (b[0]/a[0])x[n] + (b[1]/a[0])x[n-1] + (b[2]/a[0])x[n-2]  
term\_y1 = -a\_coeffs\_z\_inv[1]/a\_coeffs\_z\_inv[0]  
term\_y2 = -a\_coeffs\_z\_inv[2]/a\_coeffs\_z\_inv[0]  
term\_x0 = b\_coeffs\_z\_inv[0]/a\_coeffs\_z\_inv[0] # b0 é 0  
term\_x1 = b\_coeffs\_z\_inv[1]/a\_coeffs\_z\_inv[0]  
term\_x2 = b\_coeffs\_z\_inv[2]/a\_coeffs\_z\_inv[0]  
  
final\_eq\_str = f"y[n] = {term\_y1:.4f}\*y[n-1] + {term\_y2:.4f}\*y[n-2] + {term\_x1:.4f}\*x[n-1] + {term\_x2:.4f}\*x[n-2]"  
print(final\_eq\_str.replace("+ -", "- "))  
  
  
  
print("\n--- Estabilidade e Causalidade ---")  
magnitudes\_poles = np.abs(poles)  
print(f"Magnitudes dos polos: {magnitudes\_poles}")  
  
is\_stable = np.all(magnitudes\_poles < 1)  
is\_causal\_if\_stable = is\_stable  
  
if is\_stable:  
 print("O sistema é ESTÁVEL, pois todos os polos estão dentro do círculo unitário.")  
 if is\_causal\_if\_stable: # Assumindo que queremos um sistema causal  
 print("Assumindo uma Região de Convergência (ROC) causal (para fora do polo mais externo), o sistema também é CAUSAL.")  
 print("Portanto, a função de transferência é ESTÁVEL e CAUSAL.")  
else:  
 print("O sistema NÃO é estável, pois um ou mais polos estão no ou fora do círculo unitário.")  
  
print("\n--- Plotando Polos e Zeros no Plano Z ---")  
fig, ax = plt.subplots(figsize=(7, 7))  
  
unit\_circle = plt.Circle((0, 0), 1, color='grey', fill=False, linestyle='--', linewidth=1)  
ax.add\_artist(unit\_circle)  
  
if zeros.size > 0:  
 ax.plot(np.real(zeros), np.imag(zeros), 'o', markersize=10, markerfacecolor='none', markeredgecolor='blue', label='Zeros')  
  
if poles.size > 0:  
 ax.plot(np.real(poles), np.imag(poles), 'x', markersize=10, markeredgecolor='red', label='Polos')  
  
ax.set\_xlabel("Parte Real ($\mathbb{R}$)")  
ax.set\_ylabel("Parte Imaginária ($\mathbb{I}$)")  
ax.set\_title("Plano Z: Polos e Zeros de $H(z)$ para ST9")  
ax.grid(True, linestyle=':', linewidth=0.5)  
ax.axhline(0, color='black', lw=0.5)  
ax.axvline(0, color='black', lw=0.5)  
ax.axis('equal')  
  
all\_coords = np.concatenate((np.real(zeros), np.real(poles), np.imag(zeros), np.imag(poles), [-1.1, 1.1]))  
max\_abs\_val = np.max(np.abs(all\_coords)) if all\_coords.size > 0 else 1.1  
plot\_limit = np.ceil(max\_abs\_val \* 1.5) # Adiciona uma margem  
if plot\_limit < 1.2:  
 plot\_limit = 1.2  
ax.set\_xlim([-plot\_limit, plot\_limit])  
ax.set\_ylim([-plot\_limit, plot\_limit])  
  
ax.legend()  
plt.show()

## 3. Resultado

Texto

O conteúdo gerado por IA pode estar incorreto.

Gráfico

O conteúdo gerado por IA pode estar incorreto.