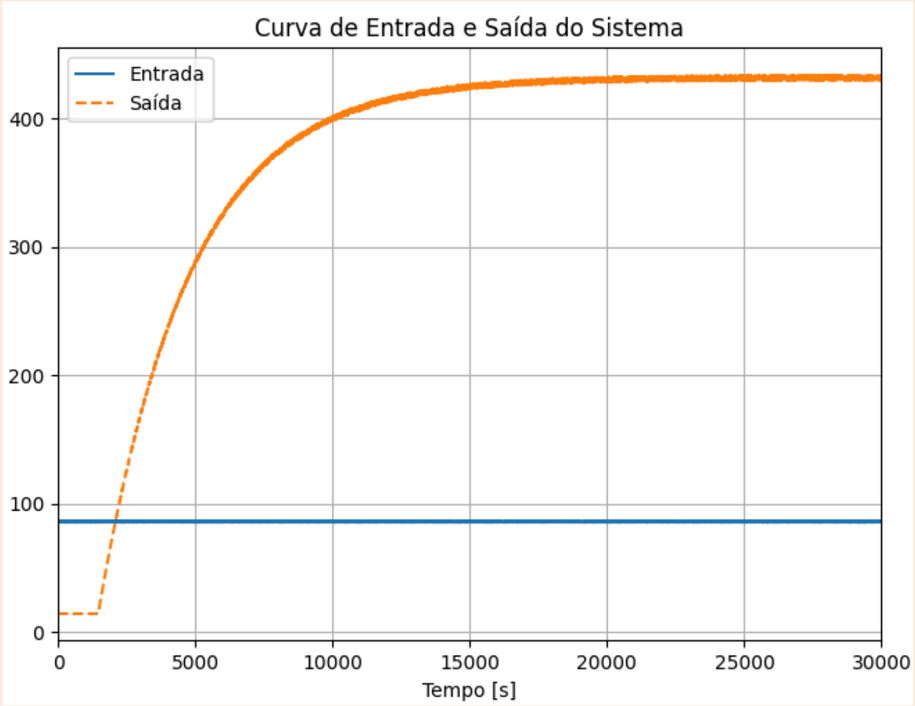
IDENTIFICAÇÃO DE PROCESSOS E SINTONIA DE CONTROLADORES PID

Tiago Augusto | Wiliane Carolina

GRÁFICOS

CARREGAMENTO E PREPAÇÃO DE DADOS

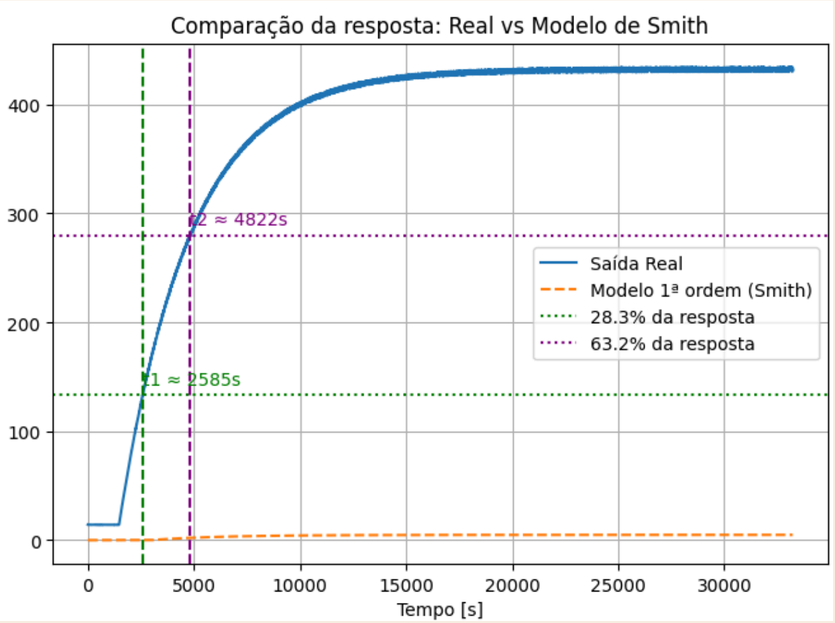
```python
def load_dados(filename):
    data = loadmat(filename)
    data = data.get(whosmat(filename)[0][0])[0][0]
    Tempo, Entrada, Saida, QuantidadeFisica, Unidades = data
    #pegando o valor medio da entrada
    return Tempo[0], Entrada[0], Saida[0], QuantidadeFisica[0], Unidades[0]

def plot_entrada_saida(tempo, entrada, saida):
    plt.figure(figsize=(10, 5))
    plt.plot(tempo, entrada, label='Entrada')
    plt.plot(tempo, saida, label='Saída', linestyle='--')
    plt.title('Curva de Entrada e Saída do Sistema')
    plt.xlabel('Tempo [s]')
    plt.ylabel('Amplitude')
    plt.grid(True)
    plt.legend()
    plt.xlim(0, 30000)
    plt.tight_layout()
    plt.show()
```



GRÁFICOS

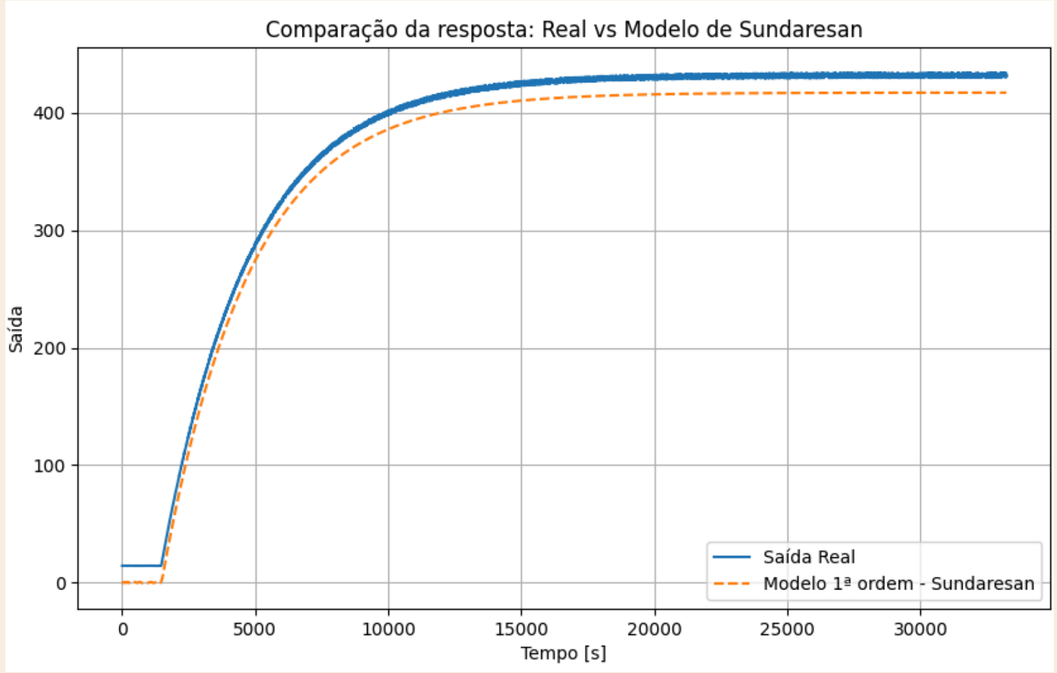
SMITH

```
if metodo.lower() == 'smith':
    y1 = saida_inicial + 0.283 * delta_saida
    y2 = saida_inicial + 0.632 * delta_saida

    if not (saida_inicial <= y1 <= saida_final) or not (saida_inicial <
        raise ValueError("Níveis de 28.3% ou 63.2% fora do intervalo da

    t1 = tempo[np.where(saida >= y1)[0][0]]
    t2 = tempo[np.where(saida >= y2)[0][0]]

    tau = 1.5 * (t2 - t1)
    theta = t2 - tau

    print(f"\nIdentificação via Método de Smith:")
```



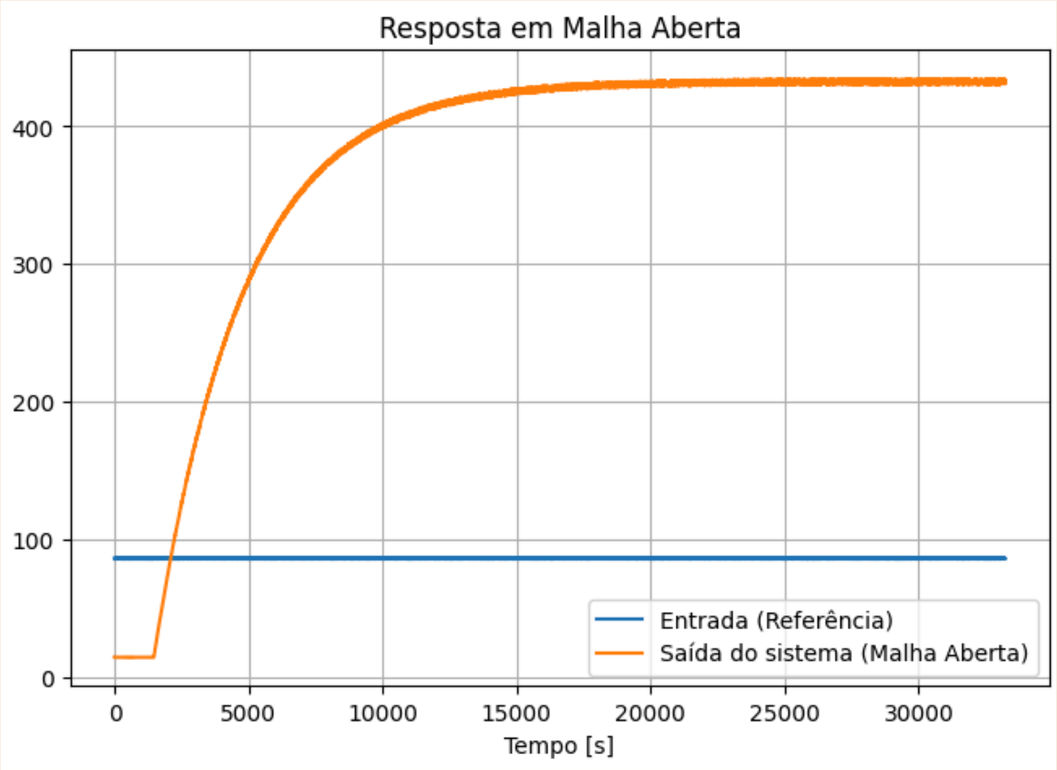
SUDARESAN

```
elif metodo.lower() == 'sundaresan':
    #saida_norm = (saida - saida_inicial) / delta_saida
     y1 = 0.353 * saida_final
     y2 = 0.853 * saida_final

     t1 = tempo[np.where(saida >= y1)[0][0]]
     t2 = tempo[np.where(saida >= y2)[0][0]]

     tau = (2/3) * (t2 - t1)
     theta = (1.3 * t1) - (0.29 * t2)
```



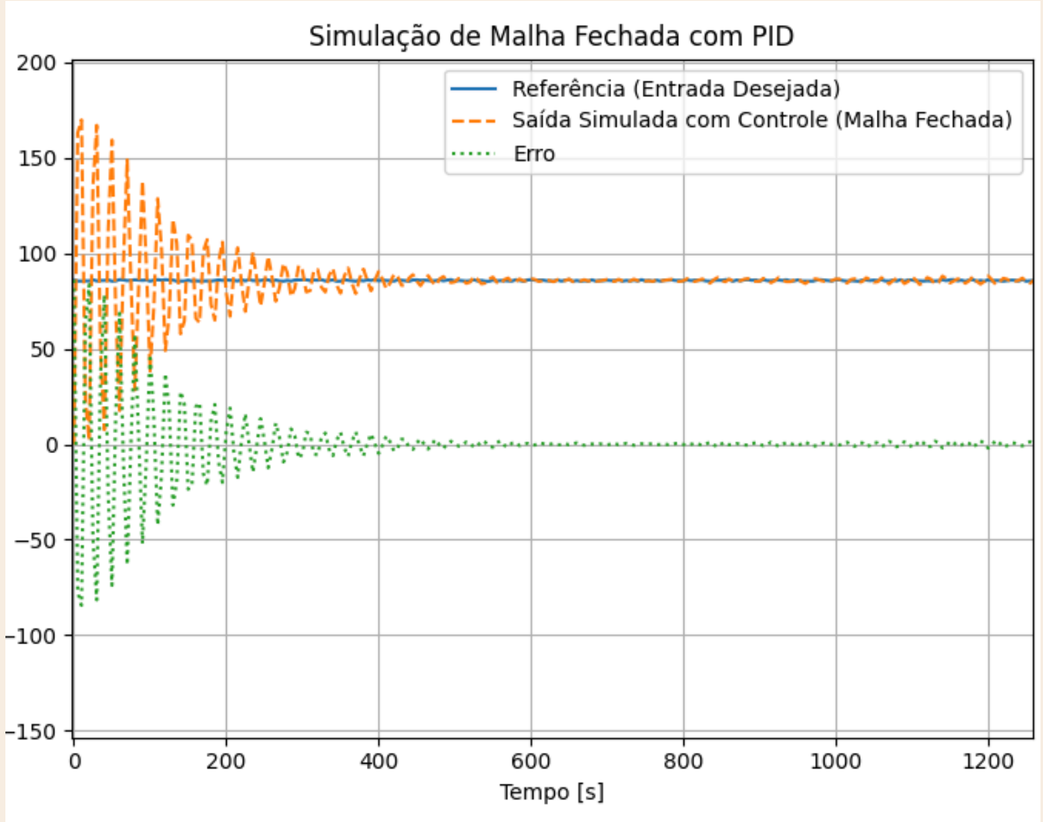
GRÁFICOS

MALHA ABERTA

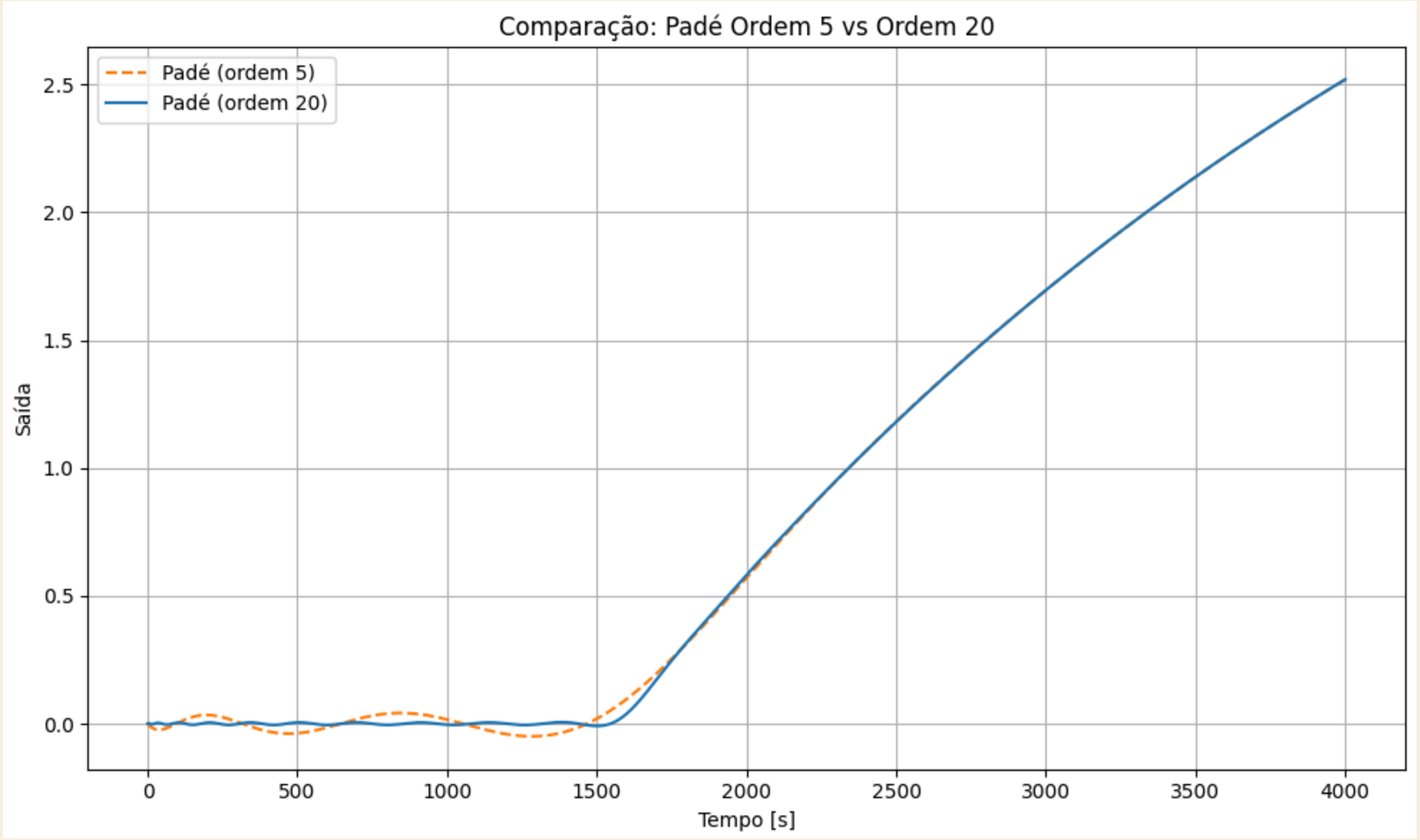


MALHA FECHADA

```python
def simular_malha_fechada(tempo, referencia, planta_saida, Kp=3.96, Ki=37.0, Kd=9.25):
    dt = tempo[1] - tempo[0]  # intervalo de amostragem
    saida_controlada = np.zeros_like(referencia)
    erro_anterior = 0
    integral = 0
    for i in range(1, len(tempo)):
        erro = referencia[i] - saida_controlada[i - 1]
        integral += erro * dt
        derivada = (erro - erro_anterior) / dt
        # PID
        controle = Kp * erro + Ki * integral + Kd * derivada
        # Planta simulada (resposta simples de 1ª ordem com inércia)
        saida_controlada[i] = saida_controlada[i - 1] + 0.01 * (controle - saida_controlada[i - 1])
        erro_anterior = erro
    return saida_controlada
```

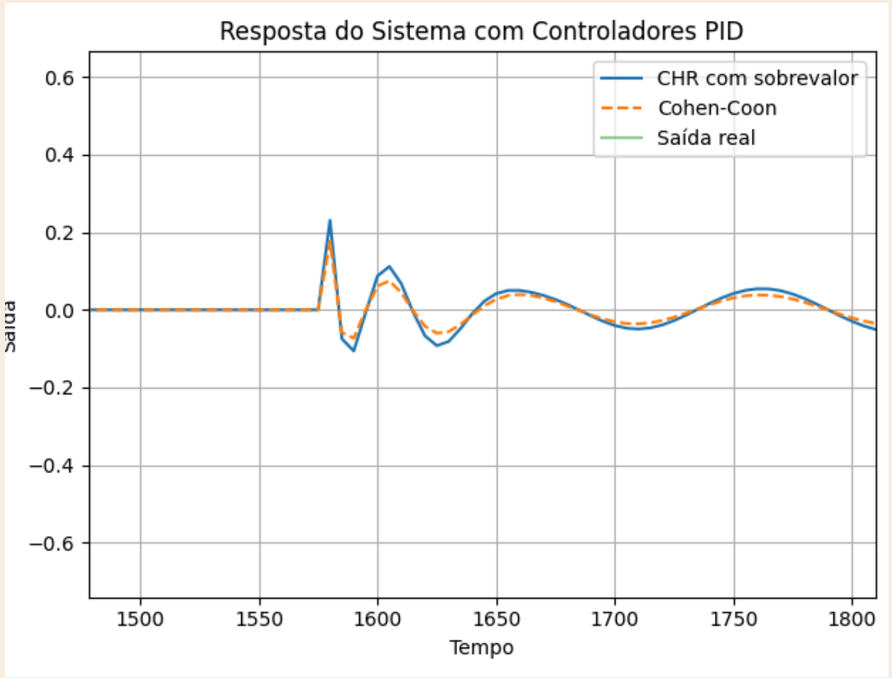


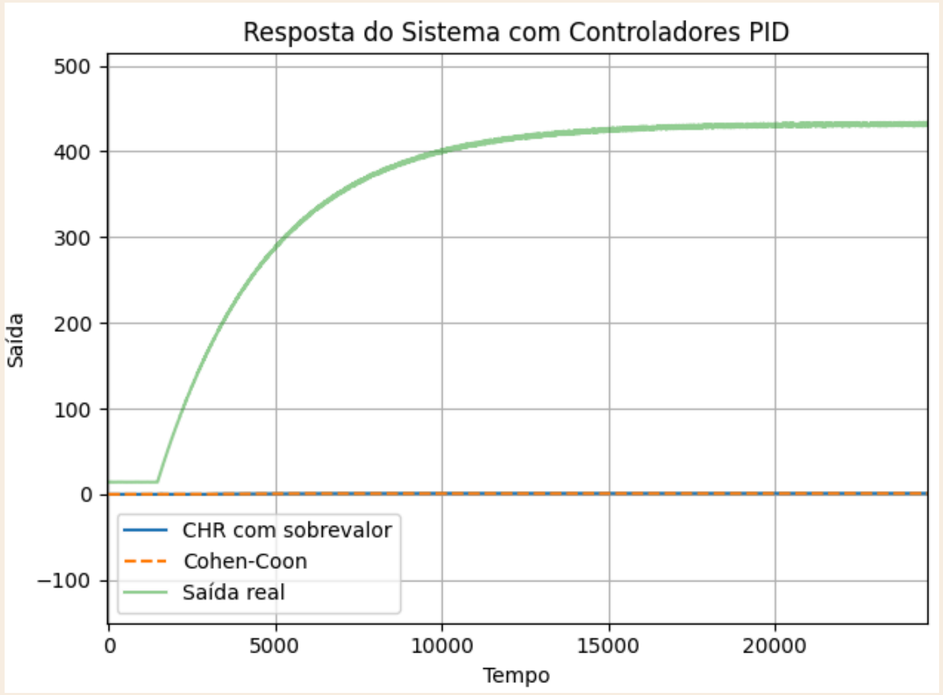
APROXIMAÇÃO DE PADÉ



*A ordem 20 é melhor porque aproxima com maior precisão o atraso puro

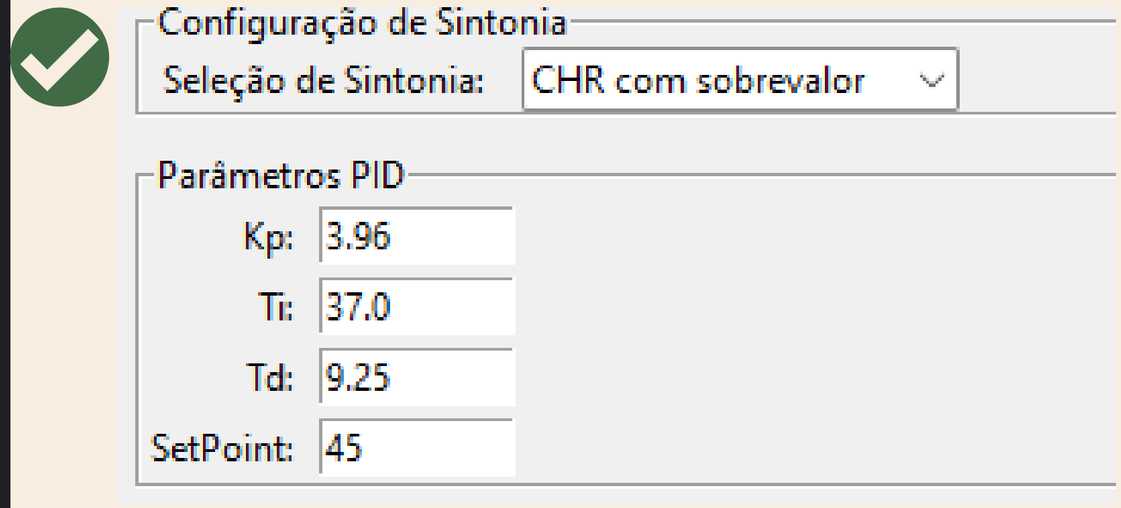
SINTONIA PID

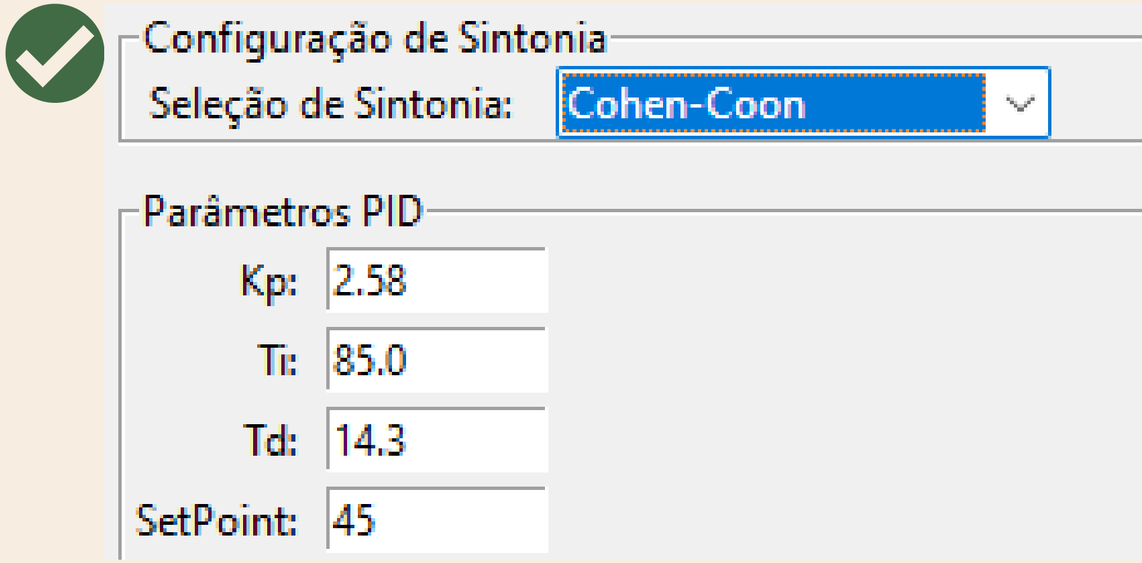




Critério	CHR com Sobrevalor	Cohen-Coon
Tempo de subida	Mais rápido	Um pouco mais lento
Pico	Mais acentuado > 60	Mais suave < 60
Oscilações	Menores, sistema estabiliza mais rápido	Mais amortecidas, mas com leve atraso
Estabilidade	Estabiliza mais cedo	Estabiliza com leve oscilação residual
Comportamento inicial	Resposta rápida e agressiva	Mais conservadora

SINTONIAS


Configuração de Sintonia
Seleção de Sintonia:
CHR com sobrevalor
Parâmetros PID
Kp:
3.96
Ti:
37.0
Td:
9.25
SetPoint:
45


Configuração de Sintonia
Seleção de Sintonia:
Cohen-Coon
Parâmetros PID
Kp:
2.58
Ti:
85.0
Td:
14.3
SetPoint:
45

OBRIGADO PELA ATENÇÃO!