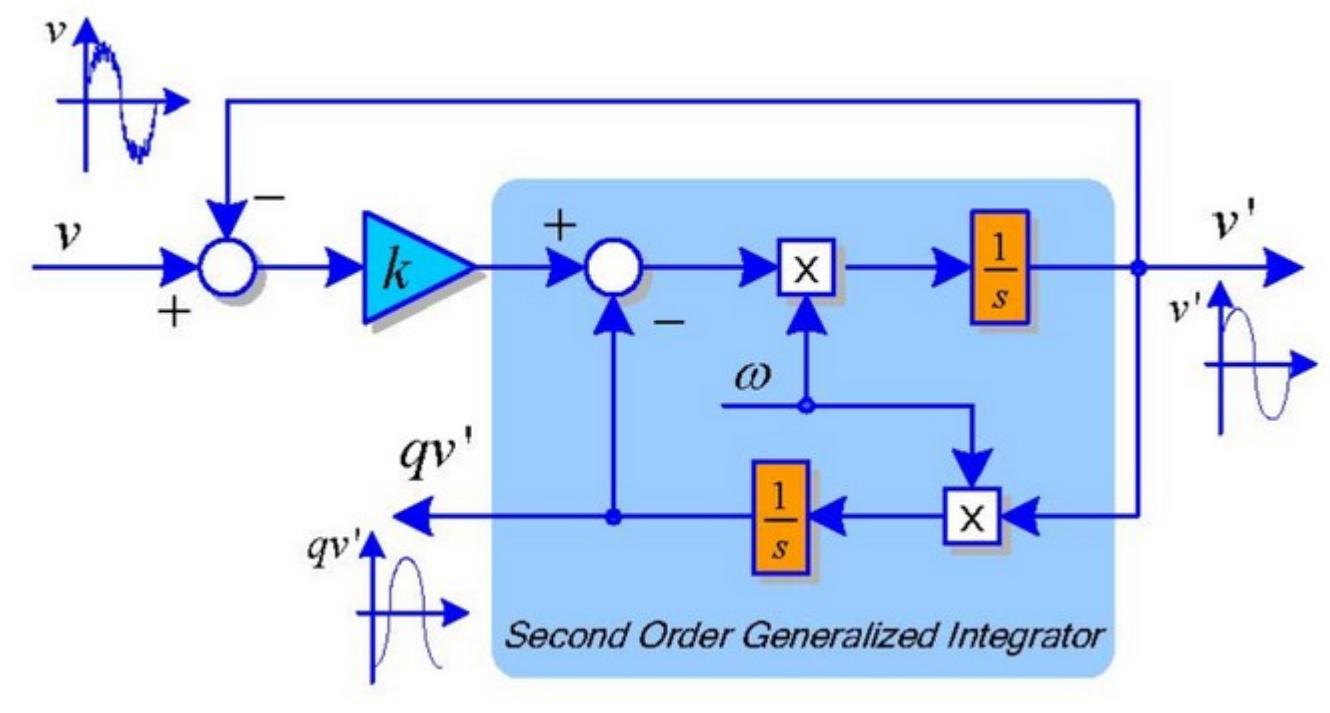
UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA ELÉTRICA E ELETRÔNICA

Disciplina: Controle Digital Aplicado em Eletrônica de Potência

SOGI-PLL

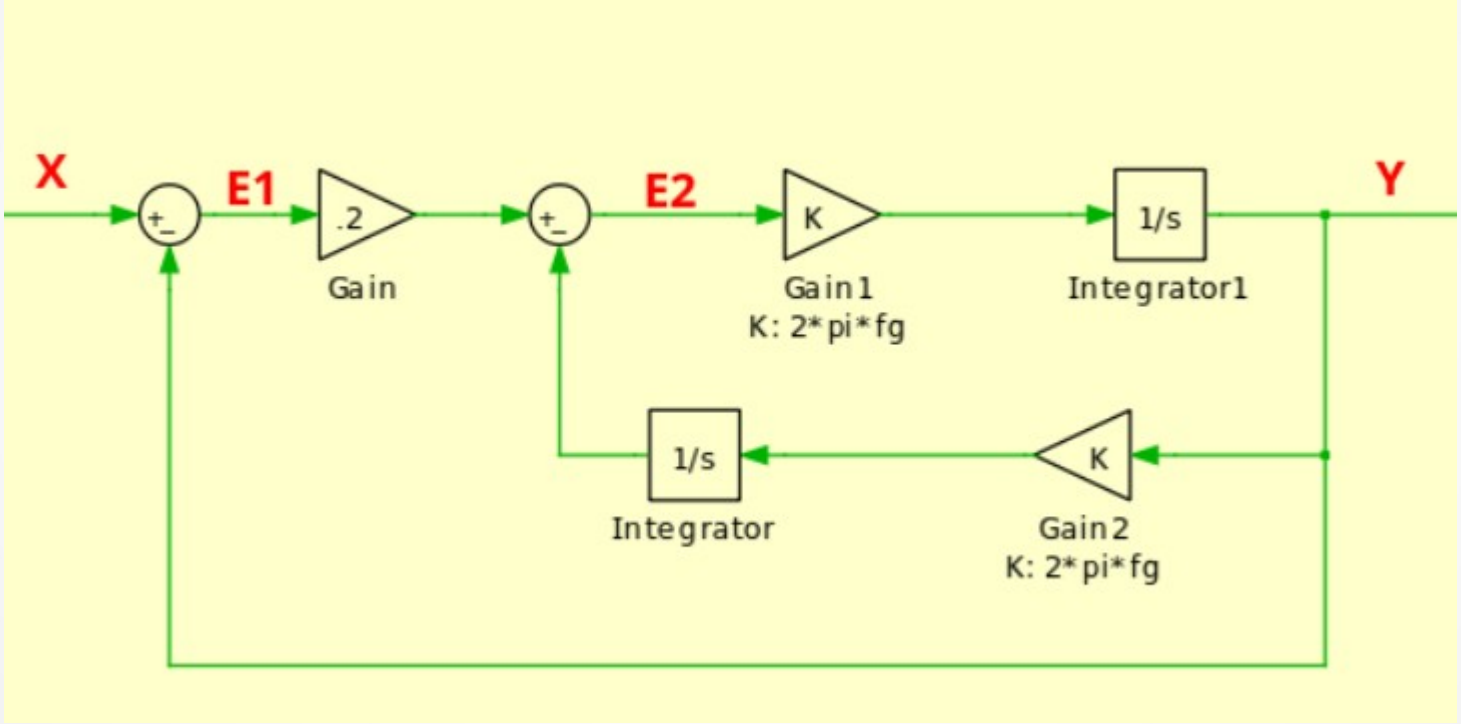
Eduardo Behr
Oscar Hernández
William Rocha

Modelo analógico não linear

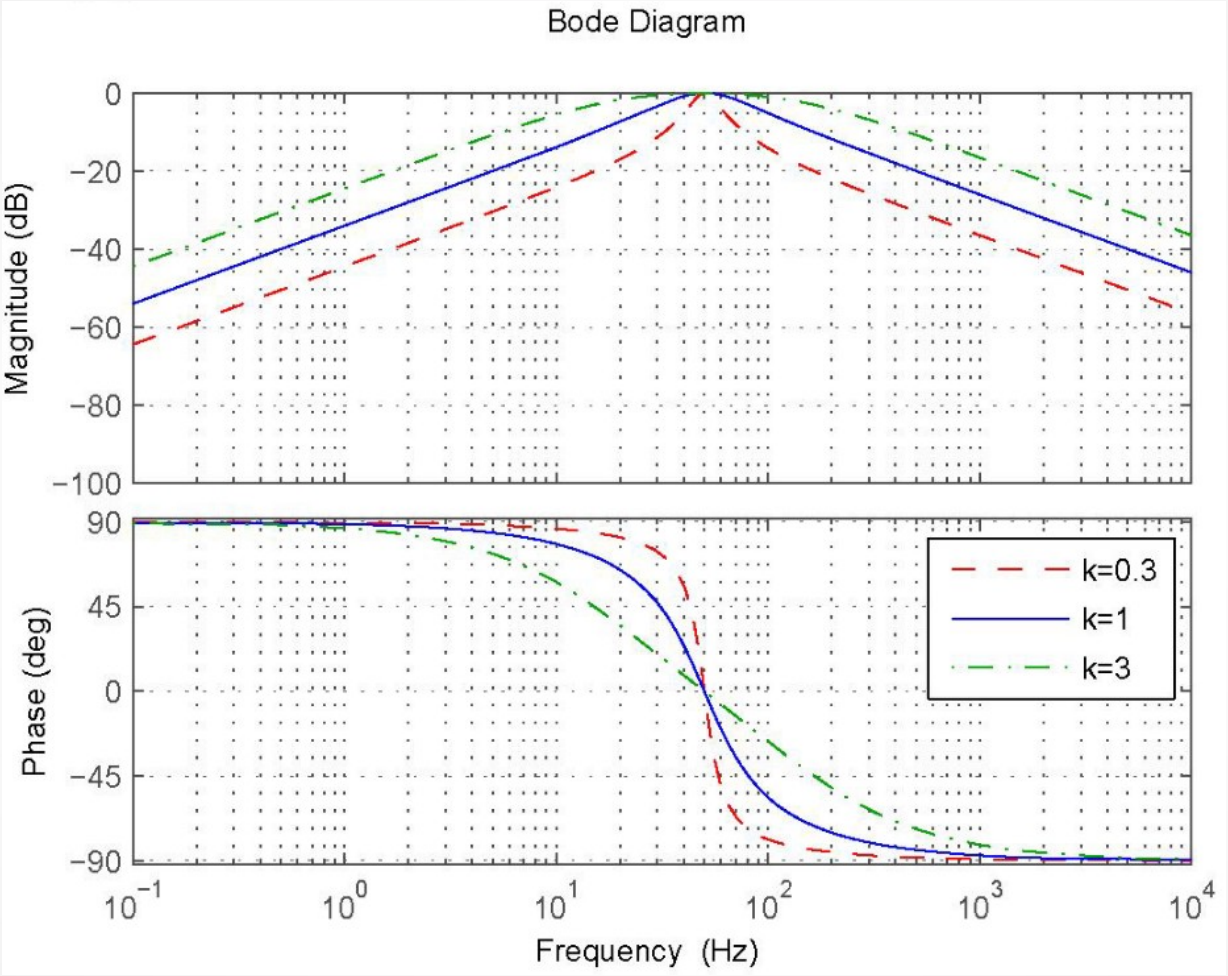


Modelo analógico linear

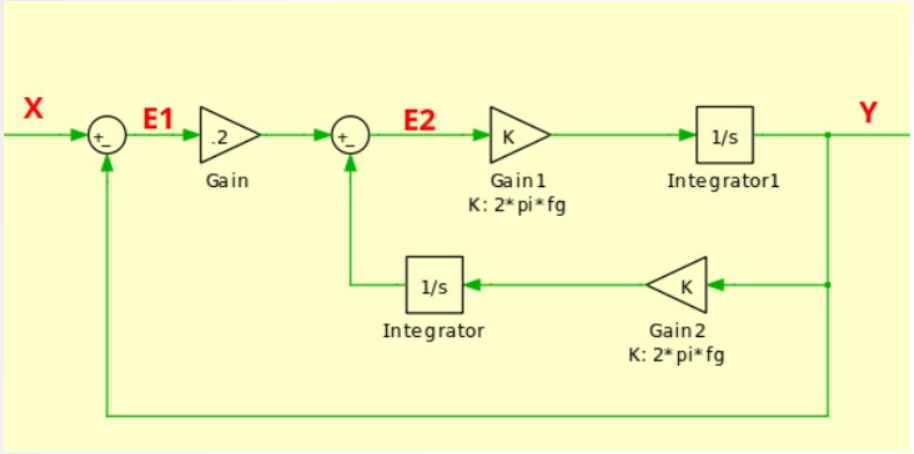
Assumindo frequência fundamental constante: $\omega = 2\pi f_g$



Resposta em frequência para f_g = 50 Hz



Dedução da função de transferência em s




```
E1 = Xs-Ys
E2 = K*E1-(2*pi*fg/s)*Ys
eq_y = sp.Eq(Ys, 2*pi*fg*E2/s); eq_y
```

✓ 0.4s Python

$$Y(s) = \frac{2\pi f_g \left(K\left(X(s) - Y(s)\right) - \frac{2\pi Y(s) f_g}{s} \right)}{s}$$

```
eq_tf_s = Eq(Ys/Xs, solve(eq_y, Ys)[0]/Xs);
```

✓ 0.1s Python

Função de transferência em s do SOGI-PLL:

$$\frac{Y(s)}{X(s)} = \frac{2\pi K f_g s}{2\pi K f_g s + 4\pi^2 f_g^2 + s^2}$$

Conversão para o domínio z

```python
def z_transform(eq: Eq):
    rhs = eq.rhs.subs({s: (2/Ts)*(z-1)/(z+1)}).simplify()
    lhs = Yz/Xz
    return Eq(lhs, rhs)
```

✓ 0.2s Python

```python
eq_tf_z = z_transform(eq_tf_s);
```

✓ 0.1s Python

Função de transferência em z do SOGI-PLL:

$$\frac{Y(z)}{X(z)} = \frac{\pi K T_s f_g (z-1)(z+1)}{\pi K T_s f_g (z-1)(z+1) + \pi^2 T_s^2 f_g^2 (z+1)^2 + (z-1)^2}$$

Equação de diferenças

$$\frac{Y(z)}{X(z)} = \frac{\pi K T_s f_g (z-1)(z+1)}{\pi K T_s f_g (z-1)(z+1) + \pi^2 T_s^2 f_g^2 (z+1)^2 + (z-1)^2}$$

$$\pi K T_s X(z) f_g \left(z^2-1\right) = Y(z)\left(\pi K T_s f_g z^2 - \pi K T_s f_g + \pi^2 T_s^2 f_g^2 z^2 + 2\pi^2 T_s^2 f_g^2 z + \pi^2 T_s^2 f_g^2 + z^2 - 2z + 1\right)$$

$$\frac{\pi K T_s X(z) f_g \left(z^2-1\right)}{z^2} = \frac{Y(z)\left(-\pi K T_s f_g + \pi^2 T_s^2 f_g^2 + z^2\left(\pi K T_s f_g + \pi^2 T_s^2 f_g^2 + 1\right) + 2z\left(\pi^2 T_s^2 f_g^2 - 1\right) + 1\right)}{z^2}$$

$$B_0 X + B_2 z^{-2} Y = A_0 Y + A_1 z^{-1} Y + A_2 z^{-2} Y$$

$$B_0 X[n] + B_2 Y[n-2] = A_0 Y[n] + A_1 Y[n-1] + A_2 Y[n-2]$$

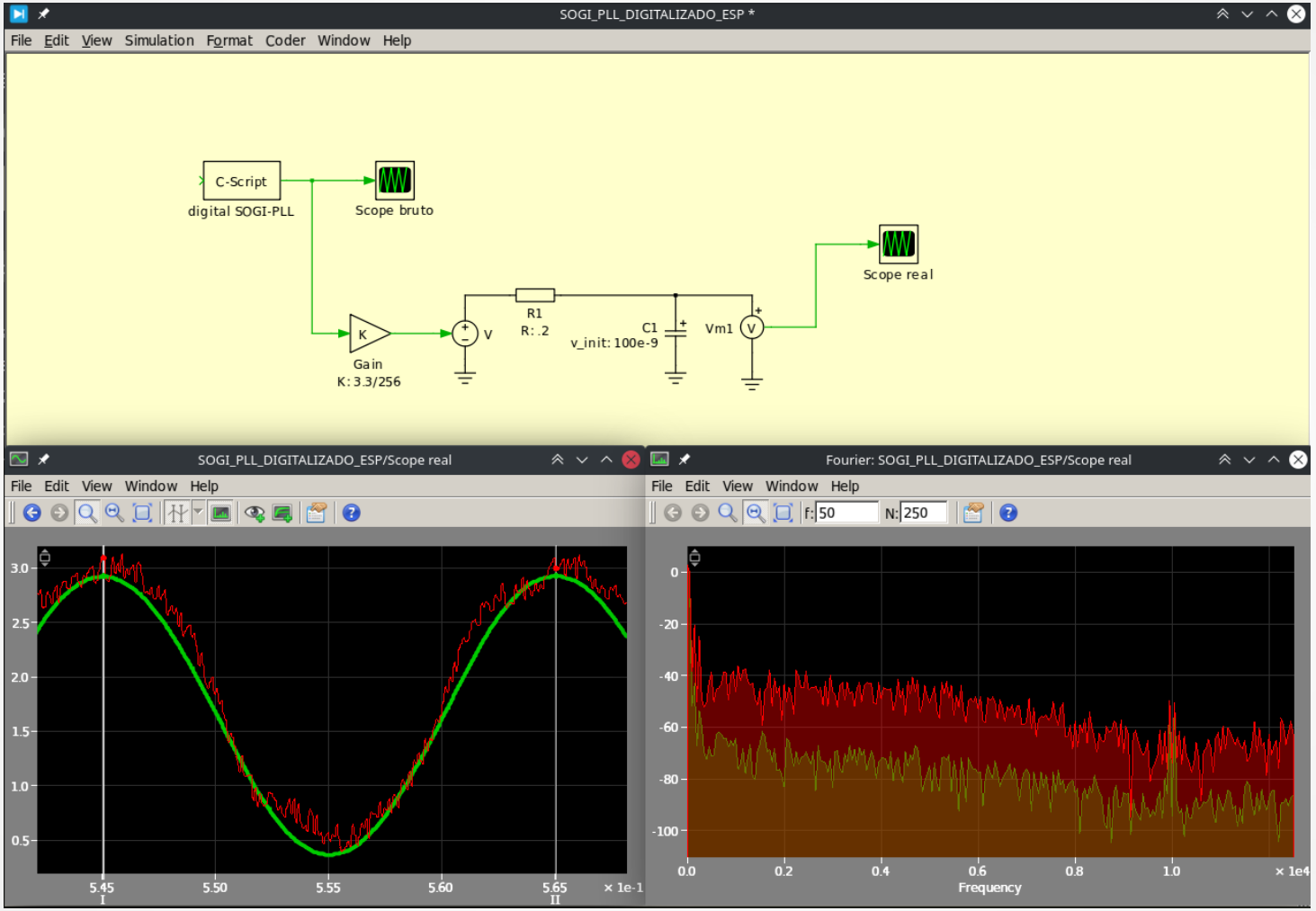
Implementação do SOGI-PLL em C

```c
double sogi_pll(const double* input, float K){
    // definir buffers de entrada e saida
    //                 Z-index:  0      -1      -2
    static double out_buff[] = {0.0,    0.0,    0.0};
    static double in_buff[] =  {0.0,    0.0,    0.0};

    // atualizar variável de entrada
    in_buff[0] =  *input;

    const double A0 = 1 + PI*K*Ts*FG + pow(PI*Ts*FG, 2);
    const double A1 = 2*(pow(PI*Ts*FG, 2)-1);
    const double A2 = 1 - PI*K*Ts*FG + pow(PI*Ts*FG, 2);
    const double B0 = PI*K*Ts*FG;
    const double B2 = -PI*K*Ts*FG;

    // calcular resultado
    out_buff[0] = (B0*in_buff[0] + B2*in_buff[2] - A2*out_buff[2] -A1*out_buff[1])/A0;

    // atualizar variáveis passadas
    in_buff[2] = in_buff[1];
    in_buff[1] = in_buff[0];

    out_buff[2] = out_buff[1];
    out_buff[1] = out_buff[0];

    return out_buff[0];
}
```

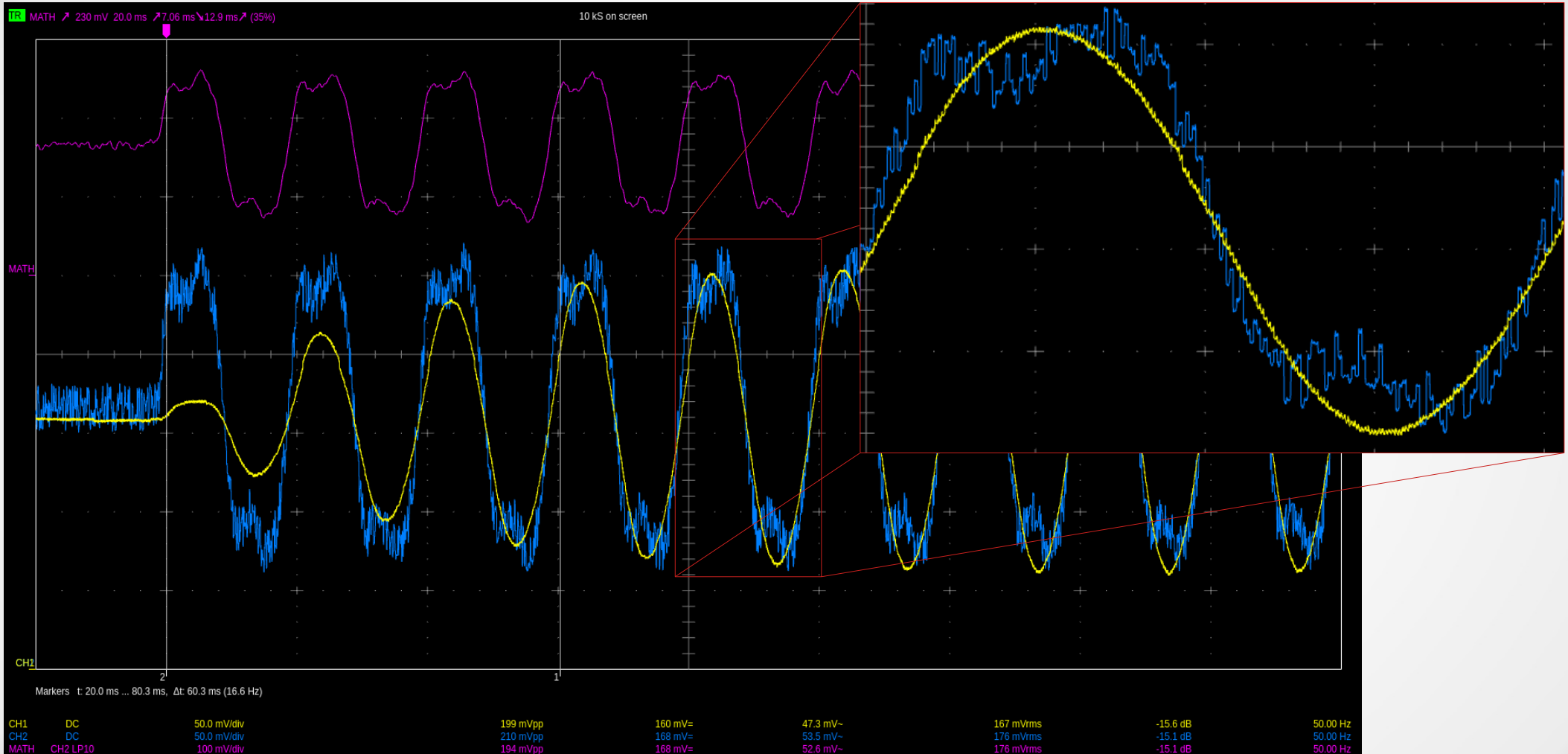
Sequência de processamento

```
void output_update_callback(void*){
    static int index = 0; // índice da tabela do sinal com harmônicas

    // sinal de entrada com degraus repetidos e ruído branco
    double input_signal = amplitude_step(20000, 0.2)*wave_table[index] + 0.3*rand()/RAND_MAX;

    // cálculo do sinal de saída
    double pll_output = sogi_pll(&input_signal, 0.2);

    // escrever sinais de entrada e saída no conversor digital para analógico
    const uint8_t offset = 100;
    const float gain = 80;
    dac_output_voltage(DAC_CHANNEL_1, clamp_8_bits(offset+gain*(input_signal))); // GPIO25
    dac_output_voltage(DAC_CHANNEL_2, clamp_8_bits(offset+gain*pll_output)); // GPIO26

    // incrementar índice da tabela do seno
    index = (index + 1) % WAVE_TABLE_LEN;
    counter++;
}
```

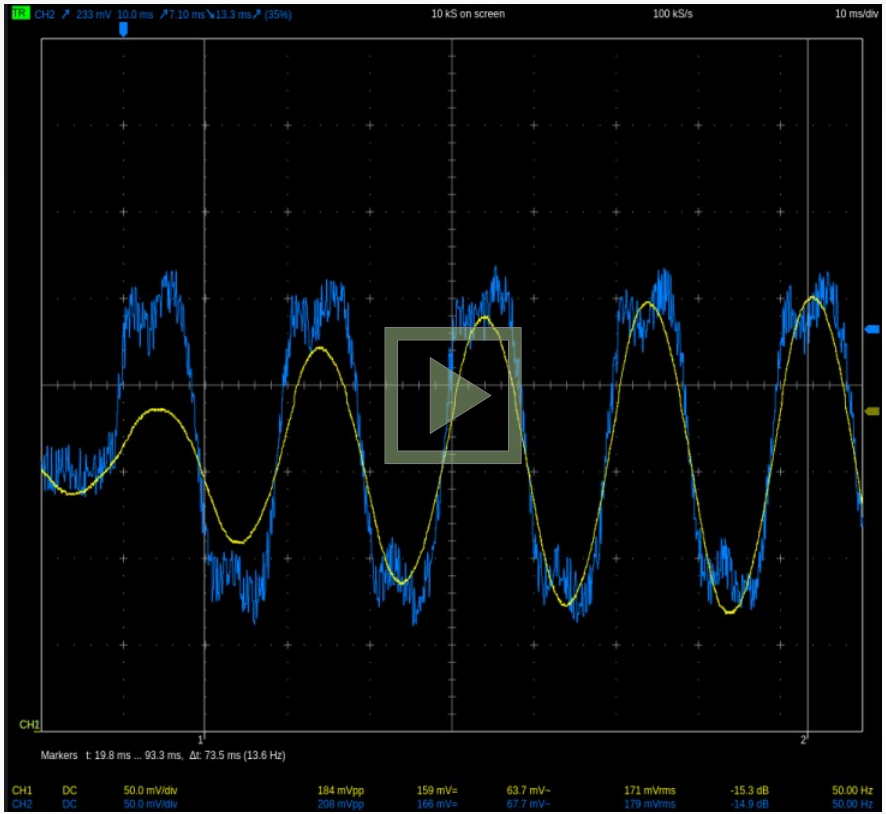
Simulação para validação



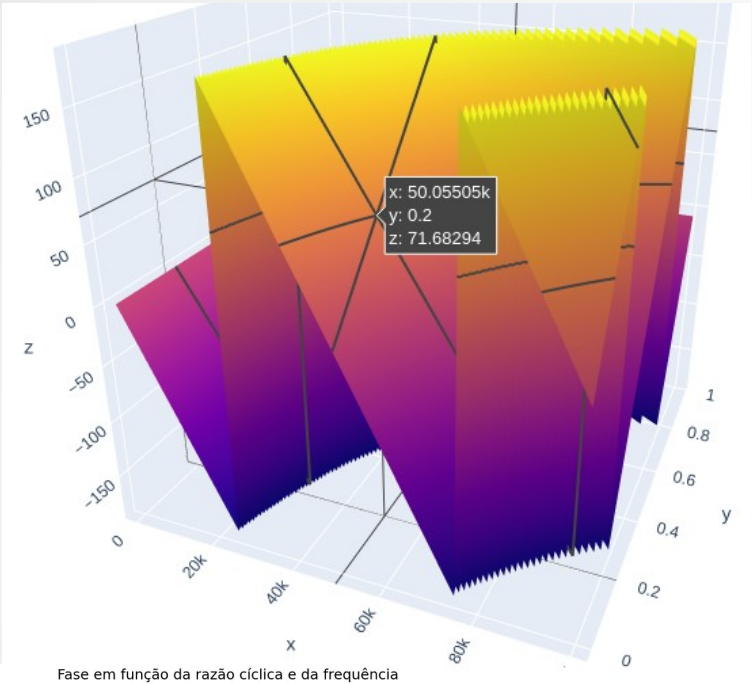
Degrau de amplitude

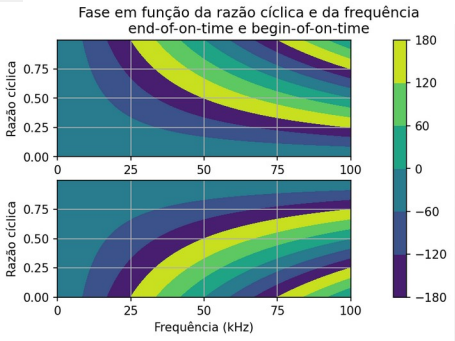


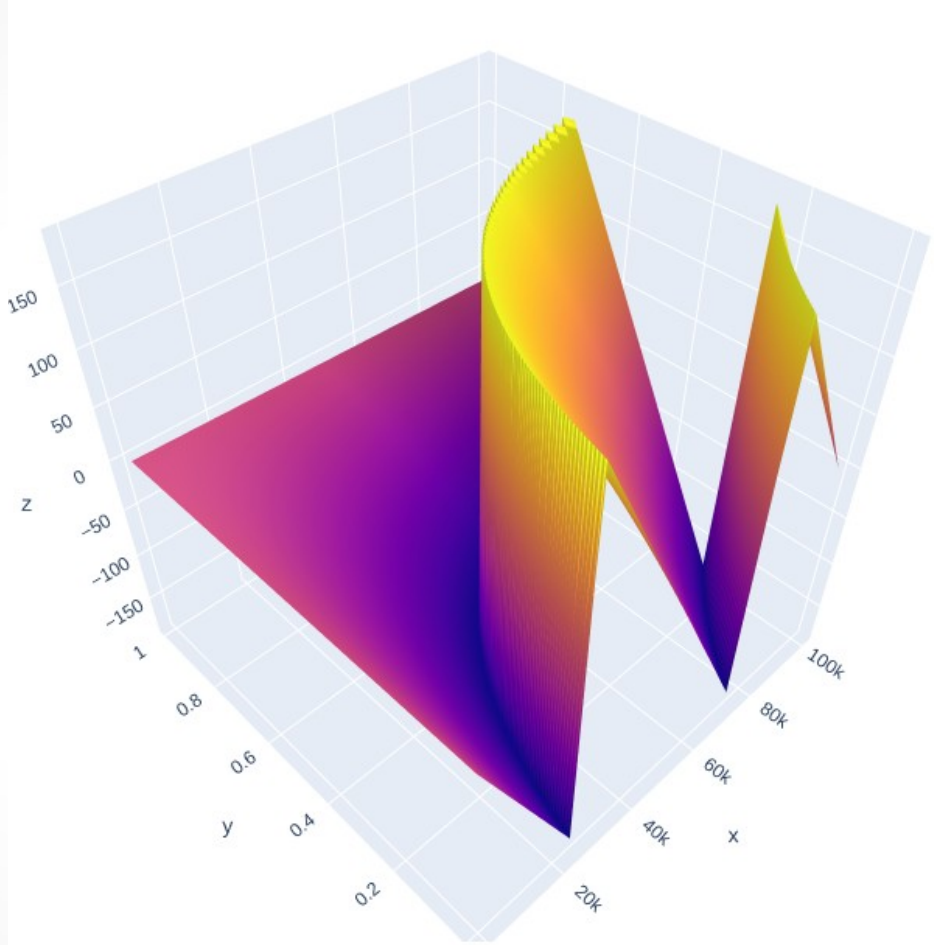
GIF


CH2 233 mV 10.0 ms 7.10 ms 13.3 ms (35%)
10 kS on screen
100 kS/s
10 ms/div
CH1
Markers t: 19.8 ms ... 93.3 ms, Δt: 73.5 ms (13.6 Hz)
CH1 DC 50.0 mV/div 184 mVpp 159 mV= 63.7 mV~ 171 mVrms -15.3 dB 50.00 Hz
CH2 DC 50.0 mV/div 208 mVpp 165 mV= 67.7 mV~ 179 mVrms -14.9 dB 50.00 Hz

Extra







https://colab.research.google.com/drive/11Cujfo8Uc-MEohfnKseUFG1_Fw_BVrYH#scrollTo=0C484HoudUth