Análise e Projeto de Controle Discreto para Conversor DC-DC Abaixador (Buck)

Nome: Jackson José Gomes do Valle
RA: 2376032

20 de julho de 2025

1 Modelagem Matemática do Conversor Buck

O conversor buck pode ser representado por um conjunto de equações que relacionam as variáveis de estado i_L e v_o com a entrada de controle, que é a razão cíclica $d(t)$. As equações que regem o circuito são:

$$L\frac{di_L(t)}{dt} = -v_o(t) + d(t)V_i \tag{1}$$

$$C\frac{dv_o(t)}{dt} = i_L(t) - \frac{v_o(t)}{R} \tag{2}$$

Onde V_i representa a tensão de entrada e L, C, R são os valores do indutor, capacitor e resistor de carga, respectivamente.

1.1 Obtenção das Funções de Transferência

Para analisar o sistema, aplica-se a Transformada de Laplace, considerando condições iniciais nulas. Este processo resulta em:

$$sLI_L(s) = -V_o(s) + D(s)V_i \tag{3}$$

$$sCV_o(s) = I_L(s) - \frac{V_o(s)}{R} \tag{4}$$

A partir da manipulação algébrica das equações (3) e (4), determinam-se as funções de transferência de interesse: $G_v(s) = \frac{V_o(s)}{D(s)}$, que relaciona a tensão de saída com a razão cíclica, e $G_i(s) = \frac{I_L(s)}{D(s)}$, que relaciona a corrente no indutor com a razão cíclica.

1.1.1 Desenvolvimento de $G_v(s)$

Primeiramente, isola-se o termo $I_L(s)$ na equação (4):

$$I_L(s) = CsV_o(s) + \frac{V_o(s)}{R} = V_o(s)\left(Cs + \frac{1}{R}\right) \tag{5}$$

Em seguida, a equação (5) é substituída na equação (3):

$$Ls\left[V_o(s)\left(Cs+\frac{1}{R}\right)\right] = -V_o(s) + D(s)V_i$$
$$LCs^2V_o(s) + \frac{L}{R}sV_o(s) + V_o(s) = D(s)V_i$$

Agrupando os termos que multiplicam $V_o(s)$, chega-se à expressão final da função de transferência da tensão:

$$G_v(s) = \frac{V_o(s)}{D(s)} = \frac{V_i}{LCs^2 + \frac{L}{R}s + 1} = \frac{\frac{V_i}{LC}}{s^2 + \frac{1}{RC}s + \frac{1}{LC}} \tag{6}$$

1.1.2 Desenvolvimento de $G_i(s)$

Utilizando a relação $V_o(s) = G_v(s)D(s)$ na equação (5), obtém-se diretamente a função de transferência para a corrente:

$$G_i(s) = \frac{I_L(s)}{D(s)} = G_v(s)\left(Cs+\frac{1}{R}\right) = \frac{V_i(Cs+\frac{1}{R})}{LCs^2 + \frac{L}{R}s + 1} \tag{7}$$

1.2 Parâmetros e Solução Numérica

Para a análise numérica e simulação, foram utilizados os seguintes parâmetros de circuito:

- L = 2 mH
- C = 10 μF
- R = 25 Ω
- V_i = 46 V
- T_s = 60 μs

Substituindo os valores, as funções de transferência numéricas são:

$$G_v(s) = \frac{2.3 \times 10^9}{s^2 + 4000s + 5 \times 10^7} \tag{8}$$

$$G_i(s) = \frac{0.00046s + 1.84}{2 \times 10^{-8}s^2 + 8 \times 10^{-5}s + 1} \tag{9}$$

1.3 Análise em Malha Aberta

Um modelo da planta foi implementado no ambiente Simulink para avaliar seu comportamento em malha aberta. A resposta a um degrau unitário na entrada de controle ($d(t) = 1$) foi simulada para caracterizar o regime transitório da tensão de saída.

```
clear; clc; close all;

L = 2e-3;      % Indutancia
C = 10e-6;     % Capacitancia
R = 25;        % Resistor da saida (carga)

```

```
7     Vi = 46;        % Tensao de Entrada (V)
8     Ts = 60e-6;     % Periodo de Chaveamento (s)
9
```

Listing 1: Parâmetros do circuito. Esse código deve ser executado antes do *Simulink*, a fim de simular corretamente o circuito.

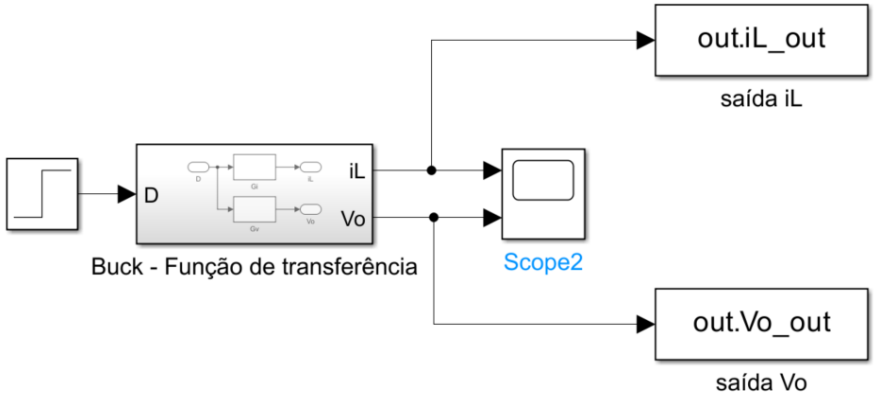


Figura 1: Modelo da planta no Simulink para ensaio em malha aberta.

A resposta da simulação foi enviada ao *workspace*, utilizando-se o comando *stepinfo* para obter os resultados. Desse modo, os parâmetros de desempenho obtidos foram:

- **Tempo de subida (0-100%):** 0,274 ms
- **Tempo de acomodação (2%):** 1,946 ms
- **Sobressinal percentual:** 39,6 %

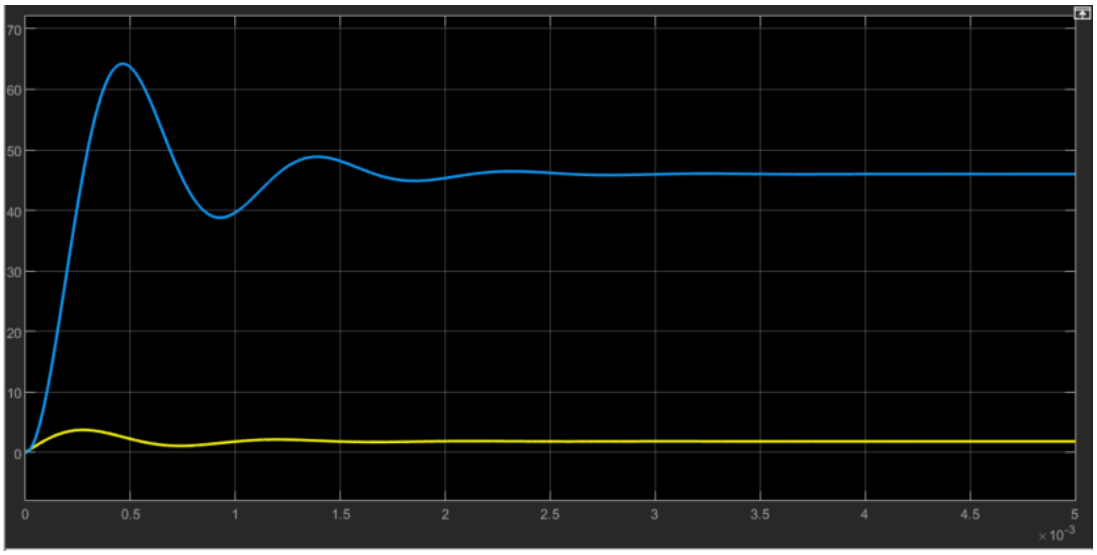


Figura 2: Resposta ao degrau para $G_v(s)$ e $G_i(s)$ com $d(t) = 1$.

1.4 Análise de Ondulação (Ripple) com PWM

Analisaram-se as ondulações de tensão e corrente com frequência de chaveamento $f_c = 50$ kHz e razão cíclica fixa de 50%, cujo modelo do *Simulink* é apresentado na figura 3. Nestas condições, o valor de regime esperado para a tensão de saída é $V_o = D \cdot V_{in} = 0.5 \cdot 46 = 23$ V e para a corrente no indutor é $I_L = \frac{V_o}{R_o} = 0,920$ A. A partir da figura 4 e da figura 5, a ondulação percentual foi medida:

- **Ondulação de tensão** (Δv_o): $\frac{28,80mV}{23V} \approx 0,12\%$
- **Ondulação de corrente** (Δi_L): $\frac{0,114A}{0,920A} \approx 12,39\%$

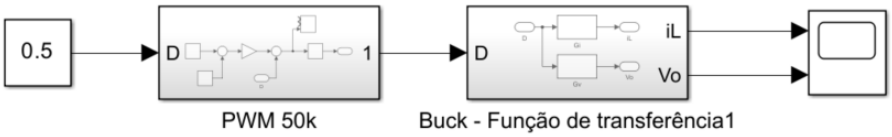


Figura 3: Modelo no Simulink para análise de ondulação com PWM.

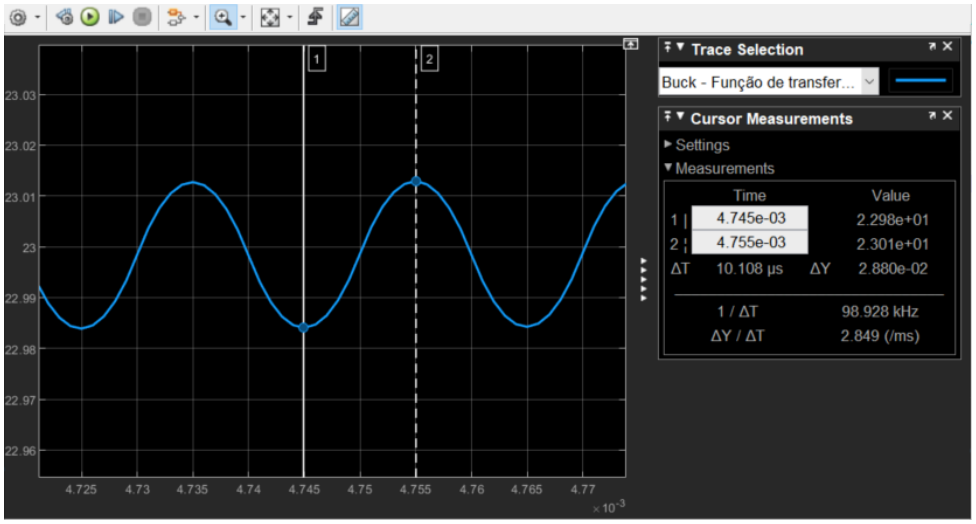


Figura 4: Ondulação do capacitor de saída C_o

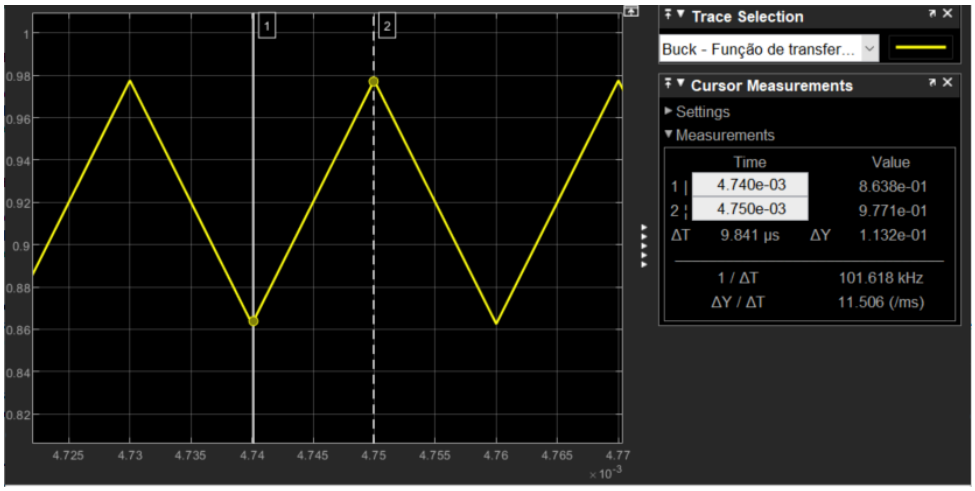


Figura 5: Ondulação do indutor L.

2 Modelagem e Controle Discreto

2.1 Obtenção da função de transferência discreta

Nesta etapa, as plantas contínuas $G_v(s)$ e $G_i(s)$ são discretizadas pelo método ZOH (Zero-Order Hold) com período de amostragem $T_s = 60\mu s$ para se obter as funções de transferência discretas $G_v(z)$ e $G_i(z)$.

```
num_Gv_s = [Vi/(L*C)];
den_Gv_s = [1 1/(R*C) 1/(L*C)];
Gv_s = tf(num_Gv_s, den_Gv_s)

Gv_z = c2d(Gv_s, Ts, 'zoh')
rltool(Gv_z)

num_Gi_s = [Vi*C Vi/R];
den_Gi_s = [L*C L/R 1];
Gi_s = tf(num_Gi_s, den_Gi_s)

Gi_z = c2d(Gi_s, Ts, 'zoh')

```

Listing 2: Código do MATLAB que executa a discretização da planta e permite o ajuste do controlador.

A partir da função *c2d*, definem-se as funções de transferência discretas e a planta discreta é comparada com a contínua. Os resultados obtidos estão na figura 6.

$$\boxed{G_v(z) = \frac{3.772z + 3.48}{z^2 - 1.629z + 0.7866}} \tag{10}$$

$$\boxed{G_i(z) = \frac{1.341z - 1.051}{z^2 - 1.629z + 0.7866}} \tag{11}$$

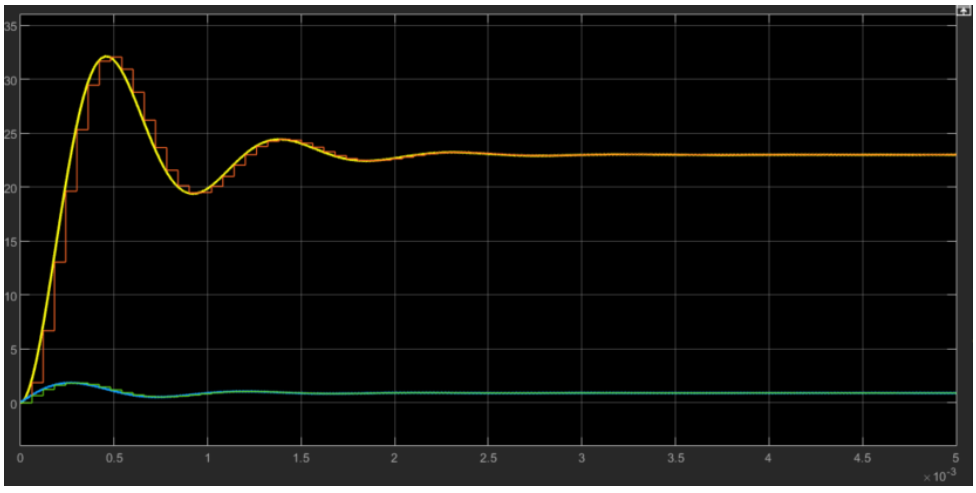


Figura 6: Comparação entre o modelo discreto e o contínuo do conversor. Percebe-se que os modelos coincidem. Desse modo, a planta discreta pode ser usada para projetar um controle adequado.

2.2 Projeto do Controlador Digital PID

O objetivo de controle é projetar um PID que atenda aos seguintes requisitos de desempenho:

- Tempo de acomodação (T_{ac}) $\leq 5 \times T_{subida} \approx 1.37$ ms
- Sobressinal (Overshoot) $\leq 5\%$
- Tensão de saída regulada em 24V.

Utilizando a função *rltool*, projetou-se um controlador PID discreto, apresentado na figura 7. Além disso, a resposta do sistema e o lugar das raízes são apresentados nas figura 8 e figura 9, respectivamente.

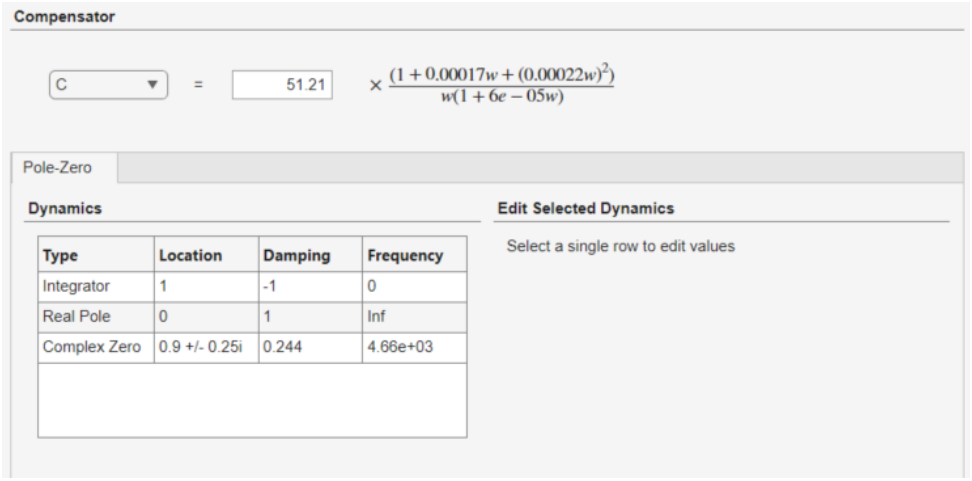


Figura 7: Compensador obtido.

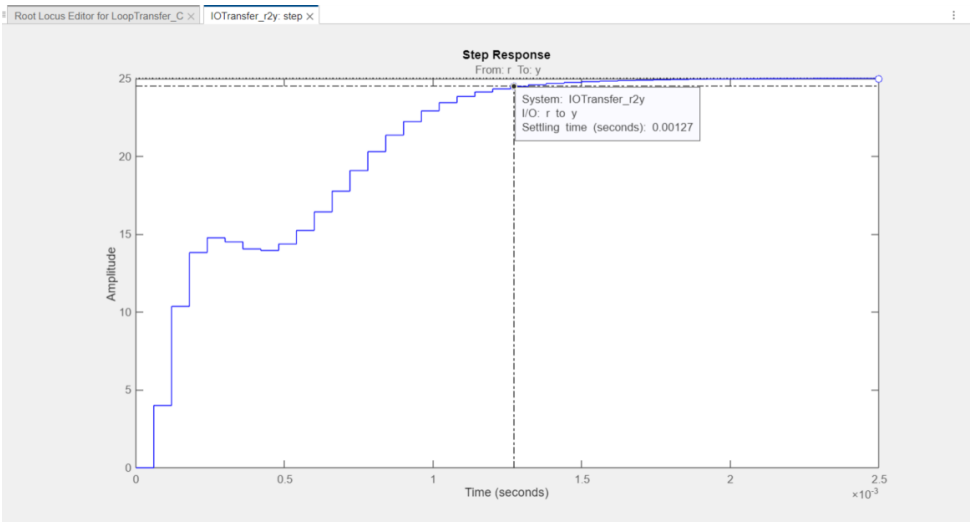


Figura 8: Resposta do sistema em malha fechada com o controle discreto.

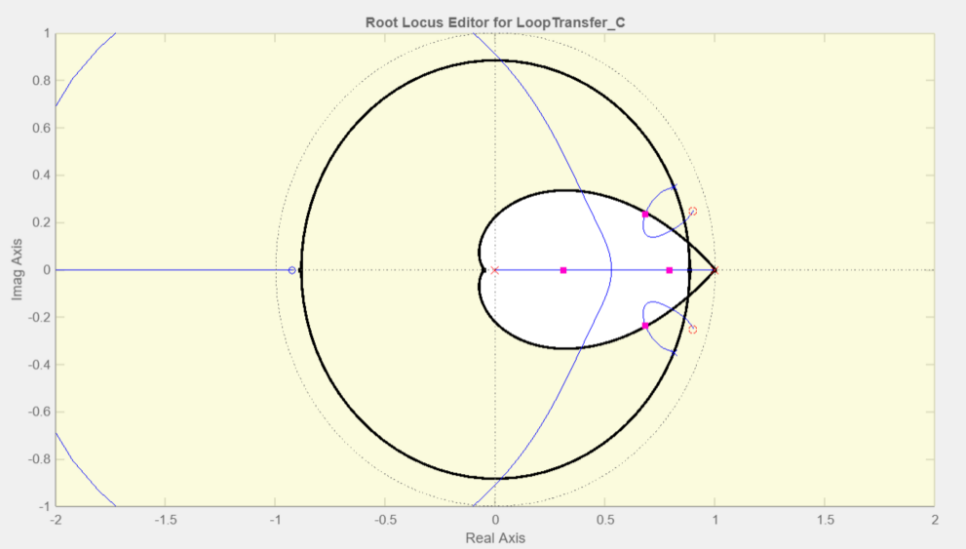


Figura 9: Lugar das raízes do compensador escolhido.

A função de transferência do controlador no domínio w é dada por:

$$C(w) = 51.21 \times \frac{1 + 0.00017w + (0.00022w)^2}{w(1 + 6 \times 10^{-5}w)} \tag{12}$$

A regra de transformação do domínio z para o domínio w é a aproximação *Forward Euler*:

$$w = \frac{z - 1}{T_s} \tag{13}$$

onde T_s é o período de amostragem.

Para obter a função de transferência no domínio z, $C(z)$, substitui-se a equação (13) na equação (12):

$$C(z) = 51.21 \times \frac{1 + 0.00017\left(\frac{z-1}{T_s}\right) + \left(0.00022\left(\frac{z-1}{T_s}\right)\right)^2}{\left(\frac{z-1}{T_s}\right)\left(1 + 6 \times 10^{-5}\left(\frac{z-1}{T_s}\right)\right)} \tag{14}$$

Com o auxílio da biblioteca *sympy*, substitui-se $T_s = 60\mu s$, obtendo-se a função de transferência do controlador no domínio z.

```
import sympy

z, Ts = sympy.symbols('z Ts')

w = (z - 1) / Ts
C_z = 51.21 * (1 + 0.00017*w + (0.00022*w)**2) / (w * (1 + 6e-5*w))

print("--- eq original C(z) ---")
sympy.pprint(C_z)
print("\n" + "="*50 + "\n")

Ts_n = 60 * 1e-6

C_z2= C_z.subs(Ts, Ts_n) # subs de Ts

```

```
    print("--- eq com Ts substituido ---")

    sympy.pprint(C_z2)
    print("\n" + "="*50 + "\n")

    print("--- eq simplificada ---")
    C_z_simpl = sympy.simplify(C_z2) # simplifica
    sympy.pprint(C_z_simpl)

```

Listing 3: Código que executa a substituição de Ts no controlador C(z).

$$\boxed{C(z) = \frac{0.0413094z^2 - 0.0739131z + 0.0356763}{z(z-1)}} \tag{15}$$

Esse controlador foi adicionado ao *Simulink*. Uma representação da montagem do sistema é exibida na figura 10

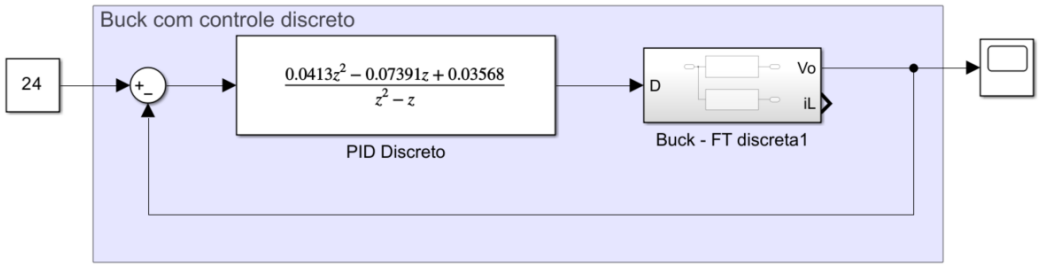
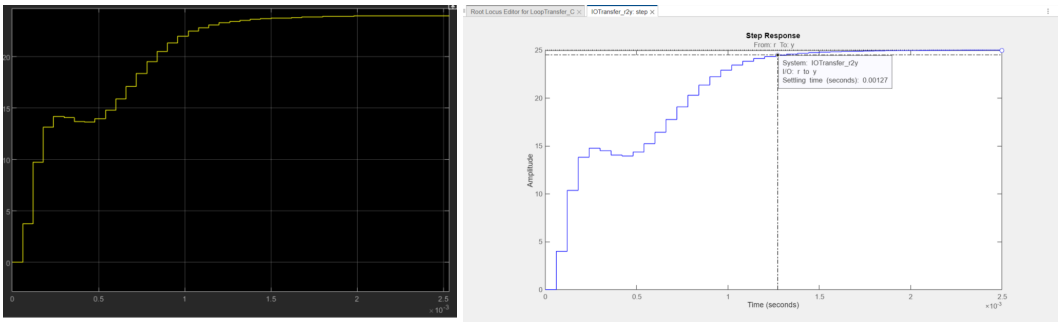


Figura 10: Simulação do controle PID no *Simulink*.

Percebe-se, a partir da figura 11, que os resultados obtidos com o SISOTOOL coincidem com o *Simulink*, validando o controle proposto.



(a) Simulink (b) SISOTOOL

Figura 11: Comparação entre o resultado obtido no SISOTOOL e a simulação do *Simulink*.

Com o controle validado, realiza-se a simulação do PID discreto com a planta contínua (figura 12).

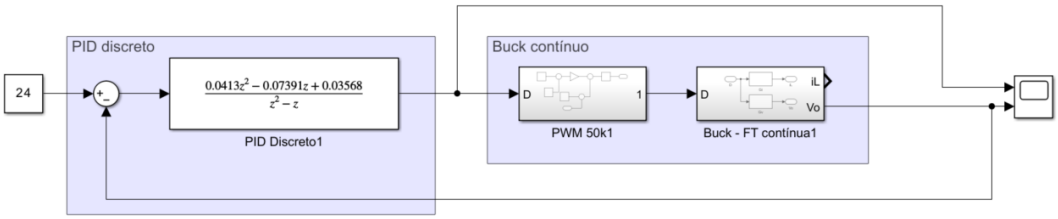


Figura 12: Simulação do controle PID no *Simulink*, com a planta contínua.

Percebe-se que o resultado obtido (figura 13) é próximo do controle discreto já modelado. Além disso, o valor de D manteve-se entre 0 e 1 em regime permanente, caracterizando um controle estável e não oscilatório.

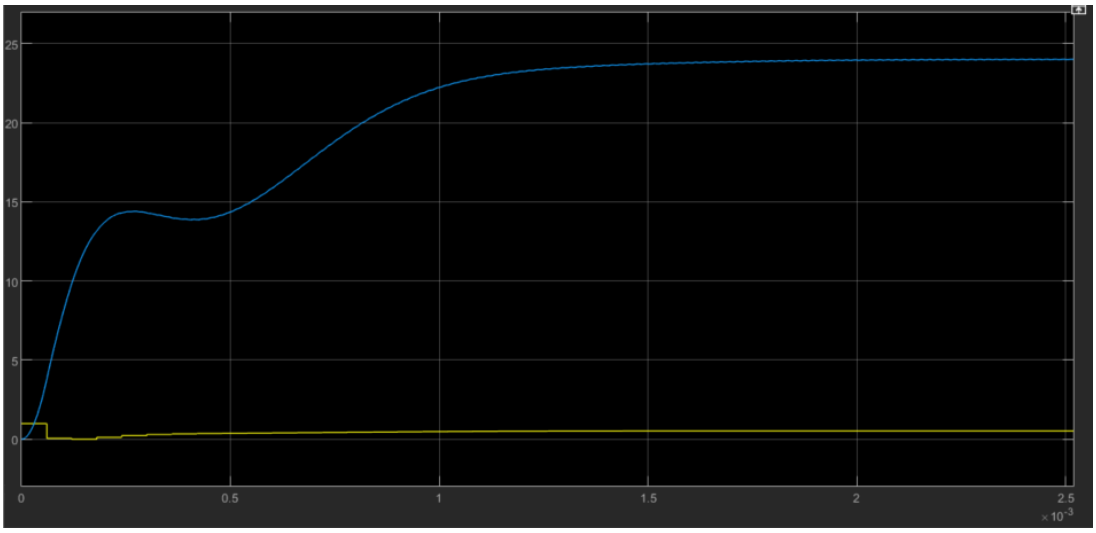


Figura 13: Resultado da tensão V_o e da razão cíclica D, no *Simulink*.

3 Simulação de uma Implementação Realística

Para uma análise mais próxima da realidade, o sistema de controle foi implementado considerando os seguintes elementos práticos:

- **Sensor de Tensão:** Um sensor de efeito Hall (LV25-P), modelado por seus ganhos e resistores associados.
- **Circuito de Condicionamento:** Um circuito com amplificadores operacionais para adequar a faixa de tensão do sensor à entrada do conversor A/D (0 a 3.3V).
- **Conversor A/D:** Modelo com 12 bits de resolução.
- **Controlador:** O algoritmo PID implementado como uma equação a diferenças em um bloco MATLAB Function.

Para encontrar os valores de resistência usados no circuito de condicionamento, utilizou-se o código a seguir:

```
# Ponto A: (entrada, saida)
v1a, v2a = 100*0.025, 0.3
# Ponto B: (entrada, saida)
v1b, v2b = 0, 3

# --- 2. Valores fixos ---
R1 = 10e3
R4 = 100e3

# Resolve para Vm, R2 e R3
Vm = (v2a * v1b - v2b * v1a) / ((v2a - v2b) + (v1b - v1a))
R2 = -R1 * (v2a - Vm) / (v1a - Vm)
R3 = (30 * R4 / (Vm + 15)) - R4

# --- 3. Exibe os resultados ---
print(f"Vm = {Vm:.4f} V")
print(f"R2 = {R2/1e3:.3f} kOhm")
print(f"R3 = {R3/1e3:.3f} kOhm")

```

Listing 4: Código usado para encontrar os valores das resistências do circuito de tratamento de sinal.

Usando aproximações comerciais, obtém-se:

$$\begin{aligned} R_1 &= 10\,\text{k}\Omega \\ R_2 &= 10.8\,\text{k}\Omega \\ R_3 &= 82\,\text{k}\Omega \\ R_4 &= 100\,\text{k}\Omega \end{aligned}$$

Dada a função de transferência do controlador $C(z) = \frac{U(z)}{E(z)}$, onde $U(z)$ é a saída e $E(z)$ é a entrada:

$$\frac{U(z)}{E(z)} = \frac{0.0413094z^2 - 0.0739131z + 0.0356763}{z^2 - z} \tag{16}$$

Para encontrar a equação a diferenças, reorganiza-se a expressão em termos de atrasos (z^{-1}) para isolar a saída atual $U(z)$. Multiplicando ambos os lados por z^{-2}:

$$\frac{U(z)(1-z^{-1})}{E(z)} = 0.0413094 - 0.0739131z^{-1} + 0.0356763z^{-2}$$

$$U(z) = U(z)z^{-1} + E(z)\left(0.0413094 - 0.0739131z^{-1} + 0.0356763z^{-2}\right)$$

Aplicando a Transformada Z Inversa, onde z^{-n} corresponde a um atraso de n amostras, obtém-se a equação a diferenças no tempo discreto k:

$$u[k] = u[k-1] + 0.0413094e[k] - 0.0739131e[k-1] + 0.0356763e[k-2] \tag{17}$$

Em notação simplificada, a equação final é:

$$u = u_1 + 0.0413094e - 0.0739131e_1 + 0.0356763e_2 \tag{18}$$

Onde u_k é o esforço de controle e e_k é o erro no instante k. Esse código foi implementado em um bloco *Matlab Function*, apresentado a seguir:

```
function Vactrl = fcn(Vad)

% Valor de Vo desejado, programado no uC
Vo_ref = 24

R_in = 10000

% Aprox. comercial
R1 = 10e3
R2 = 10800
R3 = 82000
R4 = 100e3

% Mapeamento reverso
V2 = 3.3 * Vad / 4095
Vm = 30*(R4/(R3+R4))-15
V1 = R1/R2*(Vm-V2)+Vm
Vo = V1 * R_in / 250

e = Vo_ref - Vo

persistent u_1;
persistent e_1;
persistent e_2;
if isempty(u_1)
u_1 = cast(0, 'like', Vo);
end
if isempty(e_1)
e_1 = cast(0, 'like', e);
end
if isempty(e_2)
e_2 = cast(0, 'like', e);
end

Vactrl = u_1 + 0.0413 * e - 0.0739 * e_1 + 0.0357 * e_2;

% Deve sair valor de 0 a 3.3 V para o PWM.
if (Vactrl < 0)
```

```
  Vactrl = 0
  end
  if (Vactrl > 3.3)
  Vactrl = 3.3
  end

  % store the current input
  u_1 = Vactrl;
  e_1 = e;
  e_2 = e_1;

```

Listing 5: Código do bloco *Matlab Function*, simulando o comportamento de um microcontrolador.

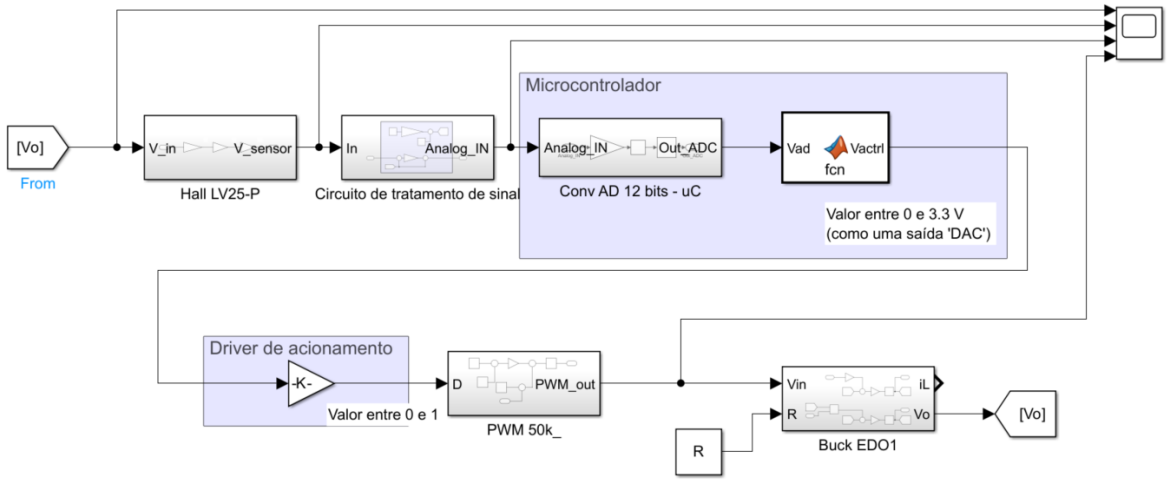


Figura 14: Modelo de simulação realístico completo no Simulink.

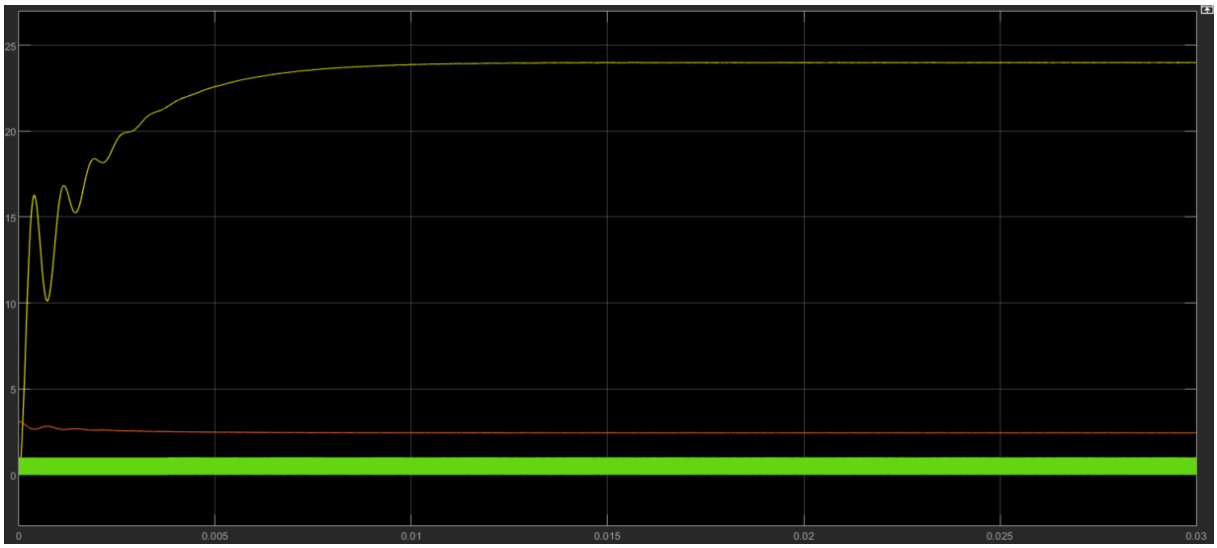


Figura 15: Resultado da simulação realística.

3.1 Teste de Robustez

A robustez do controlador foi avaliada submetendo o sistema a perturbações. Foram aplicados degraus na carga (redução de 50% na resistência) e na tensão de entrada (aumento de 30%). Os resultados demonstraram que o controlador é capaz de rejeitar tais distúrbios, mantendo a tensão de saída regulada em 24V.

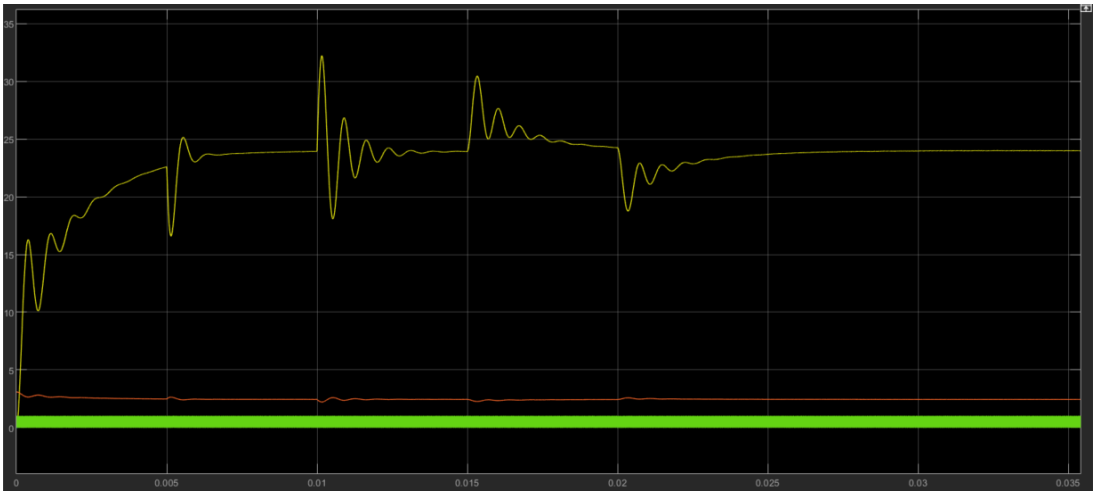


Figura 16: Resposta do sistema às perturbações de carga e tensão de entrada.

4 Abordagem Alternativa: Projeto Contínuo com Discretização

Um controlador PID foi projetado (usando a função *SISOTOOL*) no domínio contínuo para a planta $G_v(s)$, visando os mesmos requisitos de desempenho. O controlador obtido possui a seguinte função de transferência:

$$C(s) = 100 \times \frac{1 + 0.00013s + (0.00018s)^2}{s} \tag{19}$$

Expandindo esta função para a forma padrão $C(s) = K_p + \frac{K_i}{s} + K_d s$, é possível identificar os ganhos proporcionais, integrais e derivativos equivalentes:

- $K_p = 0.013$
- $K_i = 100$
- $K_d = 3.24 \times 10^{-6}$

Para a implementação digital, as ações derivativa e integral foram aproximadas numericamente:

- **Derivada (Backward-Euler):** $\frac{de(t)}{dt} \approx \frac{e_k - e_{k-1}}{T_s}$
- **Integral (Trapezoidal):** $\int e(t)dt \approx \sum \frac{e_k + e_{k-1}}{2} T_s$

O esforço de controle é então calculado como:

$$u_k = K_p e_k + K_i I_k + K_d D_k \tag{20}$$

Onde I_k e D_k são as aproximações da integral e da derivada no instante k. Essa aproximação foi inserida no microcontrolador, por meio do código abaixo:

```
function Vactrl = fcn(Vad)

% Valor de Vo desejado, programado no uC
Vo_ref = 24

R_in = 10000

% Aprox. comercial
R1 = 10e3
R2 = 10800
R3 = 82000
R4 = 100e3

% Mapeamento reverso
V2 = 3.3 * Vad / 4095
Vm = 30*(R4/(R3+R4))-15
V1 = R1/R2*(Vm-V2)+Vm
Vo = V1 * R_in / 250

e = Vo_ref - Vo

persistent u_1;
persistent e_1;
```

```
24  persistent e_2;
25  persistent Ik_1;
26
27  if isempty(u_1)
28  u_1 = cast(0, 'like', Vo);
29  end
30  if isempty(e_1)
31  e_1 = cast(0, 'like', e);
32  end
33  if isempty(e_2)
34  e_2 = cast(0, 'like', e);
35  end
36
37  if isempty(Ik_1)
38  Ik_1 = 0;
39  end
40
41  % Ganhos do controlador PID continuo discretizado
42  kp = 0.013;
43  kd = 3.24e-6;
44  ki = 100;
45
46  % Calculo das aprox. derivada e integral (controle continuo)
47  Ts = 6e-5;
48  dk = (e - e_1)/Ts;
49  Ak = (e + e_1)/2*Ts
50  Ik = Ik_1 + Ak;
51
52  %Calc. do esforco de controle
     ---------------------------------------------
53  Vactrl = kp*e + ki*Ik + kd*dk; %Controle continuo discretizado
54  %Vactrl = u_1 + 0.0413 * e - 0.0739 * e_1 + 0.0357 * e_2; %Controle
     discreto
55
56  % Deve sair valor de 0 a 3.3 V para o PWM.
57  if (Vactrl < 0)
58  Vactrl = 0
59  end
60  if (Vactrl > 3.3)
61  Vactrl = 3.3
62  end
63
64  % store the current input
65  u_1 = Vactrl;
66  e_1 = e;
67  e_2 = e_1;
68  Ik_1 = Ik;
```

Listing 6: Código do bloco *Matlab Function*, simulando o comportamento de um microcontrolador.

4.1 Comparação de Resultados

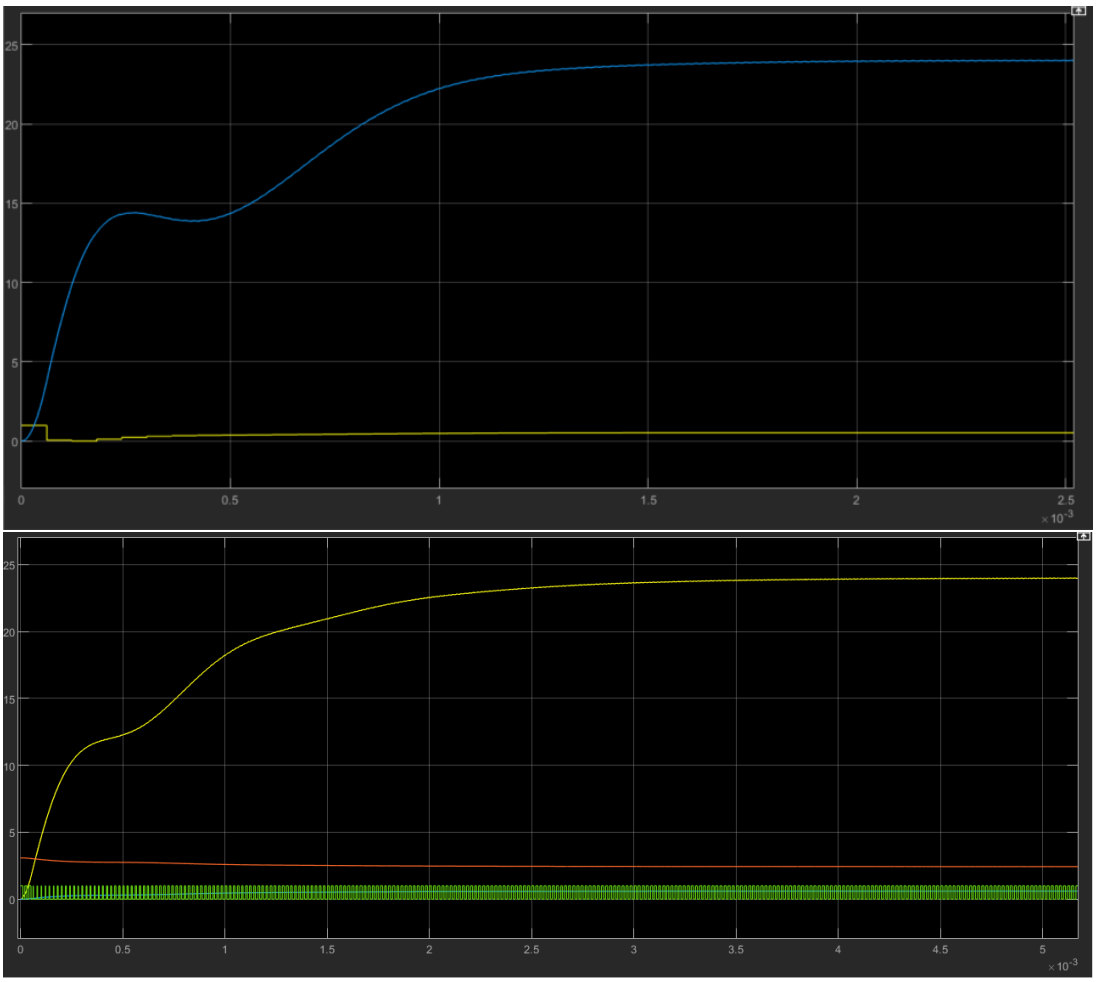


Figura 17: Comparação da resposta à perturbação entre o controlador discreto (acima) e o contínuo discretizado (abaixo).

Ambos os controladores foram simulados sob as mesmas condições. Os resultados indicam que as duas abordagens são eficazes, embora apresentem pequenas diferenças nos transitórios.

Os arquivos utilizados nesse projeto estão no seguinte link: `https://github.com/JacksonJoseG/Simulink-buck-model`