# Modelagem de Sistemas Dinâmicos - Trabalho Nº2

Leonardo Soares da Costa Tanaka - DRE: 121067652

Engenharia de Controle e Automação/UFRJ

Rio de Janeiro, Brasil

Junho de 2023

Considerando um motor de corrente contínua (DC) controlado por corrente de armadura com entrada de tensão $V_a(t)$ (V), saída de velocidade angular $\omega_m(t)$ (rad/s), representado pelo circuito abaixo:

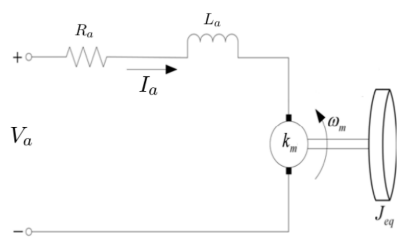



Figura 1: Circuito

O motor considerado tem as seguintes características dadas pelo fabricante:

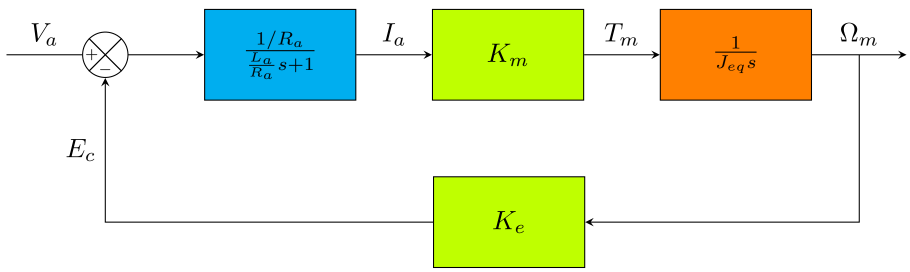
- Resistência de armadura: $R_a = 10.6\Omega$
- Indutância de armadura: $L_a = 0.82mH$
- Momento de Inércia do Rotor do Motor: $J_m = 1.16 \cdot 10^{-6} kgm^2$
- Constantes do Motor: $K_m = 0.0502Nm/A$, $K_e = 0.0502Vs$
- Tensão máxima: 15 volts
- Massa do disco de inércia: $0.068kg$
- Raio do disco de inércia: $0.0248m$

## 1 Modelagem teórica

Organizando as equações do sistema para obter o diagrama de blocos e a função de transferência. Utilizando a Lei de Kirchhoff das Tensões, Equação do Torque, Lei do Motor e Lei do Gerador. Depois, realizando a Transformada de Laplace:

$$
\begin{gathered}
R_a \cdot i_a + L_a \frac{di_a}{dt} = V_a - e_c, \quad J_{eq} \frac{d\omega_m}{dt} = T_m, \quad T_m = K_m \cdot i_a, \quad e_c = K_e \cdot \omega_m \\
I_a(s) = \frac{1/R_a}{\frac{L_a}{R_a}s + 1}(V_a(s) - e_c(s)), \quad \Omega(s) = \frac{1}{J_{eq}s}T_m(s), \quad T_m(s) = K_m \cdot I_a(s), \quad E_c(s) = K_e \Omega_m(s)
\end{gathered} \tag{1}
$$

1.1. Montando um diagrama de blocos do sistema (Desconsiderando o atrito do sistema). O bloco azul é o subsistema elétrico, o bloco laranja é o subsistema mecânico e os blocos verdes são interligações entre o sistema elétrico e mecânico:




1.2. Calculando a função de transferência do motor $G(s)$ de $V_a(s)$ para $\Omega_m(s)$:

$$
G(s) = \frac{\Omega_m(s)}{V_a(s)} = \frac{\frac{K_m}{R_a \cdot J_{eq}} \frac{1}{s(\frac{L_a}{R_a}s+1)}}{1 + \frac{K_e \cdot K_m}{R_a \cdot J_{eq}} \frac{1}{s(\frac{L_a}{R_a}s+1)}} = \frac{k_m}{R_a \cdot J_{eq} \cdot s \cdot (\frac{L_a}{R_a}s + 1) + K_e \cdot K_m} = \frac{\frac{k_m}{J_{eq} \cdot L_a}}{s^2 + \frac{R_a}{L_a}s + \frac{K_e \cdot K_m}{J_{eq} \cdot L_a}} = \frac{\frac{1}{K_e}}{\frac{L_a \cdot J_{eq}}{K_m \cdot K_e}s^2 + \frac{R_a \cdot J_{eq}}{K_m \cdot K_e}s + 1} \tag{2}
$$

1.3. Representando o sistema no espaço de estados com a realização canónica controlável:

$$
\begin{aligned}
G(s) = \frac{b_1 s + b_2}{s^2 + a_1 s + a_2}, \quad \dot{x} &= \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\
y &= \begin{bmatrix} b_2 & b_1 \end{bmatrix} x \\
\dot{x} &= \begin{bmatrix} 0 & 1 \\ -\frac{K_e \cdot K_m}{J_{eq} \cdot L_a} & -\frac{R_a}{L_a} \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \\
y &= \begin{bmatrix} \frac{k_m}{J_{eq} \cdot L_a} & 0 \end{bmatrix} x
\end{aligned} \tag{3}
$$

1.4. Calculando a inércia do disco (feito de alumínio):

$$
\begin{gathered}
J = \int r^2 dm, \quad \frac{dm}{M} = \frac{dA}{A} = \frac{2\pi r dr}{\pi R^2}, \quad J_d = \int_0^R \frac{r^2 M 2\pi r}{\pi R^2} dr = \frac{2M}{R^2} \int_0^R r^3 dr = \frac{2Mr^4}{4R^2}\bigg|_0^R = \frac{MR^2}{2} \\
J_d = \frac{0.068 \cdot 0.0248^2}{2} = 20.91136 \ 10^{-6} \ kgm^2
\end{gathered} \tag{4}
$$

Determinando momento de inércia total (rotor e disco), levando em consideração que estão conectadas no mesmo eixo de rotação e estão perfeitamente fixadas juntas. Podemos usar a seguinte fórmula e valores:

$$
J_{eq} = J_m + J_d = (1.16 \ 10^{-6} + 20.91136 \ 10^{-6}) \ kgm^2 = 22.07136 \ 10^{-6} \ kgm^2 \tag{5}
$$

1.5. Considerando que a função de transferência G(s) tenha a seguinte estrutura:

$$G(s) = \frac{K}{(\tau_e s + 1)(\tau_m s + 1)} \tag{6}$$

Utilizando a equação (2) e as características dadas pelo fabricante para calcular os pólos da função de transferência:

$$p_{1,2} = \frac{-\frac{R_a}{L_a} \pm \sqrt{\frac{R_a^2}{L_a^2} - 4\frac{K_e \cdot K_m}{J_{eq} \cdot L_a}}}{2} = \frac{-\frac{10.6}{0.82 \cdot 10^{-3}} \pm \sqrt{\frac{10.6^2}{(0.82 \cdot 10^{-3})^2} - 4\frac{0.0502 \cdot 0.0502}{22.07136\ 10^{-6} \cdot 0.82 \cdot 10^{-3}}}}{2} = \frac{-12926.82927 \pm 12905.26847}{2}$$
$$p_1 = -12916.04887, \quad p_2 = -10.7804 \tag{7}$$
$$K = \frac{k_m}{J_{eq} \cdot L_a \cdot p_1 \cdot p_2} = 19.92031563, \quad \tau_e = -\frac{1}{p_1} = 7.742305794\ 10^{-5}, \quad \tau_m = -\frac{1}{p_2} = 0.09276093651$$

A constante de tempo elétrica $\tau_e$ pode ser considerada despressível comparada com a constante de tempo mecânica $\tau_m$, porque a constante de tempo elétrica é pequena em relação aos tempos de interesse, isso significa que o sistema tem uma dinâmica inicial rápida devido ao polo rápido, mas em seguida, a resposta se estabiliza gradualmente devido ao polo lento. O efeito do polo lento será mais predominante à medida que o tempo avança.

1.6. Considerando que não existe perturbação nem atrito, é possível determinar a velocidade máxima do motor $\omega_{max}$ pelo Teorema do Valor Final:

$$V_{max} = K_e \cdot \omega_{max}, \quad \omega_{max} = \lim_{s \to 0} s \cdot \Omega_m(s) = \lim_{s \to 0} s \frac{V_{max}}{s \cdot K_e} = \frac{15}{0.0502} = 298.8047809\ rad/s \tag{8}$$

1.7. Determinando a corrente máxima $I_{max}$ e máximo de torque gerado $T_{max}$ utilizando as equações (1) e Teorema do Valor Final:

$$I_{max} = \lim_{s \to 0} s \cdot I_a(s) = \lim_{s \to 0} s \frac{\frac{V_{max}}{R_a \cdot s}}{\frac{L_a}{R_a}s + 1} - s\frac{K_e \Omega_m(s)}{\frac{L_a}{R_a}s + 1} = \frac{V_{max}}{R_a} = \frac{15}{10.6} = 1.4150943396\ A,$$
$$T_{max} = K_m I_{max} = 0.0502 \cdot 1.4150943396 = 0.0710377358\ Nm \tag{9}$$

## 2 Identificação Experimental

2.1. Determinando experimentalmente no kit QET da Quanser, de forma estatística, as constantes utilizadas como parâmetros na modelagem teórica:

(a) Aplicando voltagem constante $V_a$ e medindo a corrente $I_a$, levando em consideração que é mantido o eixo do motor parado e que o indutor de armadura se torna um curto-cicuito no momento da medição.

$$R_a \cdot i_a + L_a \frac{di_a}{dt} = V_a - e_c \rightarrow R_a \cdot i_a + 0 = V_a - 0 \rightarrow R_a = \frac{V_a}{i_a} \tag{10}$$

Foi utilizado os dados obtidos experimentalmente para realizar uma Regressão Linear em Python com Scikit-Learn para obter o valor estimado da Resistência de armadura $R_a$:

```
import numpy as np
from sklearn.linear_model import LinearRegression
# Dados de voltagem Va(V) em Volts e corrente Ia(A) que foram enviados em forma de tabela
voltagem = np.array([1.0, 2.0, 3.0, 4.0, 4.9, 5.9, 6.9, 7.9, 8.9, 9.9])
corrente = np.array([0.076, 0.150, 0.225, 0.305, 0.425, 0.474, 0.550, 0.620, 0.720, 0.800])
# Ajuste da regressão linear
x = corrente.reshape(-1, 1)  # Converte corrente para formato adequado
y = voltagem
regressor = LinearRegression()
regressor.fit(x, y)
resistencia_media = regressor.coef_[0] # coeficiente angular da regressão linear
print("Resistência média:", resistencia_media, "ohms") # Exibindo os resultados
```

Valor estimado da Resistência de armadura $R_a$: 12.222223261373172 $\Omega$

(b) Aplicando voltagem constante $V_a$ e medindo corrente e velocidade. Considerando que a resistência de armadura é conhecida. Foi calculado a potência mecânica e a potência elétrica do motor DC para achar a relação entre as constantes do motor.

$$\begin{gathered} P_m = T_m \cdot \omega_m, P_e = e_c \cdot i_a, P_m = P_e \rightarrow T_m \cdot \omega_m = e_c \cdot i_a, \\ K_m \cdot i_a \cdot \omega_m = K_e \cdot i_a \cdot \omega_m \rightarrow K_m = K_e \end{gathered} \tag{11}$$

Utilizando o resultado dos cálculos acima é possível obter o valor da constante mecânica do motor ao afirmar que o circuito está em curto-circuito nos dados experimentais calculados:

$$e_c = V_a - R_a \cdot i_a \rightarrow K_m = \frac{V_a - R_a \cdot i_a}{\omega_m} \tag{12}$$

Foi utilizado os dados obtidos experimentalmente para realizar uma Regressão Linear em Python com Scikit-Learn para obter o valor estimado da Constante de torque do motor $K_m$:

```
import numpy as np
from sklearn.linear_model import LinearRegression
Ra = 10.6 # Ra resistência de armadura
Jeq = Jeq = 22.07136 * 10**(-6)
# Dados de voltagem Va(V) em Volts, velocidade angular wm(rad/s) e corrente Ia(mA)
Va = np.array([1.0, 2.0, 3.0, 4.0, 4.9, 5.9, 6.9, 7.9, 8.9, 9.9])
wm = np.array([12, 29, 49, 70, 89, 105, 127, 148, 168, 185])
Ia = np.array([1, 1, 1, 1, 1, 1, 2, 2, 2, 3])
# Ajuste da regressão linear
x = wm.reshape(-1, 1)  # Converte corrente para formato adequado
```

```
y = Va - Ra * Ia * 10**(-3)
regressor = LinearRegression()
regressor.fit(x, y)
constante_torque = regressor.coef_[0] # coeficiente angular da regressão linear
print("Constante de torque do motor:", constante_torque, "Nm/A") # Exibindo os resultados
```

Valor estimado do Constante de Torque do motor $K_m$: 0.05048676538284488 Nm/A

2.2. Calculando a função de transferência do sistema G(s) com os parâmetros $\{R_a, K_m\}$ estimados acima. Determinando os parâmetros $\{K, \tau_e, \tau_m\}$:

$$G(s) = \frac{\frac{k_m}{J_{eq}\cdot L_a}}{s^2 + \frac{R_a}{L_a}s + \frac{K_e\cdot K_m}{J_{eq}\cdot L_a}} = \frac{\frac{1}{K_e}}{\frac{L_a\cdot J_{eq}}{K_m\cdot K_e}s^2 + \frac{R_a\cdot J_{eq}}{K_m\cdot K_e}s + 1} = \frac{19.8071710955718}{(6.713348814760633\ 10^{-5}\ s+1)(0.10576662346101155\ s+1)} \quad (13)$$

$$K = 19.8071710955718, \quad \tau_e = 6.713348814760633\ 10^{-5}, \quad \tau_m = 0.10576662346101155$$

2.3. Já que $\frac{La}{Ra} \approx 0$, foi determinado a função de transferência simplificada $\hat{G}(s)$ desprezando a dinâmica elétrica, isto é (determinando $\hat{K}$ e $\hat{\tau}$):

$$\hat{G}(S) = \frac{\hat{K}}{\hat{\tau}s+1} = \frac{\frac{K_m}{J_{eq}\cdot R_a\cdot s}}{1 + \frac{K_m\cdot K_e}{J_{eq}\cdot R_a\cdot s}} = \frac{\frac{1}{K_e}}{\frac{J_{eq}\cdot R_a}{K_m\cdot K_e}s+1} = \frac{19.80717109557183}{0.10583375694916465\cdot s+1} \quad (14)$$

$$\hat{K} = 19.80717109557183, \quad \hat{\tau} = 0.10583375694916465$$

# 3 Validação dos modelos

```
# Bibliotecas utilizadas durante essa parte do relatório
import os
from scipy.io import loadmat
import numpy as np
import control as ctrl
import matplotlib.pyplot as plt
# Coletando os dados experimentais de formato .mat
data = loadmat("data/trabalho2dados.mat")
# Resposta a onda quadrada +/- 2Volts
t2v = np.array(data["t2v"]) # t2v:  tempo (s)
u2v = np.array(data["u2v"]) # u2v:  entrada Va (v)
y2v = np.array(data["y2v"]) # y2v:  saida  wm  (rad/s)
# Resposta a onda quadrada +/- 4Volts
t4v = np.array(data["t4v"]) # t4v:  tempo (s)
u4v = np.array(data["u4v"]) # u4v:  entrada Va (v)
```

```
y4v = np.array(data["y4v"]) # y4v:  saida  wm  (rad/s)
# Resposta a onda quadrada +/- 10Volts
t10v = np.array(data["t10v"]) # t10v:  tempo (s)
u10v = np.array(data["u10v"]) # u10v:  entrada Va (v)
y10v = np.array(data["y10v"]) # y10v:  saida  wm  (rad/s)
# Resposta a Senoide de  5Volts
tsin5v = np.array(data["tsin5v"]) # tsin5v:  tempo (s)
usin5v = np.array(data["usin5v"]) # usin5v:  entrada Va (v)
ysin5v = np.array(data["ysin5v"]) # ysin5v:  saida  wm  (rad/s)
# Definindo as constantes do sistema
Ra = 10.6
La = 0.82 * 10**(-3)
Jeq = 22.07136 * 10**(-6)
Km = 0.0502
Ke = 0.0502
```

Para obter as simulações numéricas das respostas do sistema não foi possível utilizar as entradas e os intervalos de tempo experimentais. Então, foi necessário obter por meio da visualização dos dados experimentais e do que era esperado como entrada, as entradas e os intervalos de tempo aproximados:

```
# Resposta a onda quadrada +/- 2Volts
t2 = np.linspace(0, len(t2v)/100, len(t2v)) # t2:  tempo (s)
u2 = 1.98 * np.sign(np.sin(2 * np.pi * (1/2) * (t2-0.475))) # u2:  entrada Va (V)
# Resposta a onda quadrada +/- 4Volts
t4 = np.linspace(0, len(t4v)/100, len(t4v)) # t4:  tempo (s)
u4 = 3.96 * np.sign(np.sin(2 * np.pi * (1/2) * (t4-0.655))) # u4:  entrada Va (V)
# Resposta a onda quadrada +/- 10Volts
t10 = np.linspace(0, len(t10v)/100, len(t10v)) # t10:  tempo (s)
u10 = 4.95 * np.sign(np.sin(2 * np.pi * (1/2) * (t10-0.915))) + 4.95 # u10:  entrada Va (V)
# Resposta a Senoide de +/- 5Volts
tsin5 = np.linspace(0, len(tsin5v)/100, len(tsin5v)) # tsin5:  tempo (s)
usin5 = 4.965666 * np.sin(tsin5*np.pi + 3*np.pi/2 - 0.15) # usin5:  entrada Va (V)
```

Utilizando o espaço de estados dos cálculos (3) foi feita a simulação numérica do G(s). Além disso, foi obtido as respostas do sistema $G(s)$ as entradas ao degrau (2, 4 e 10 volts) e senoidal (5 volts):

```
# Definindo as matrizes A, B, C e D do sistema
A = np.array([[0, 1], [-Ke*Km/Jeq/La, -Ra/La]])
B = np.array([[0], [1]])
```

```
C = np.array([[Km/Jeq/La, 0]])
D = np.array([[0]])
sys = ctrl.StateSpace(A, B, C, D) # Criando um objeto StateSpace para representar seu sistema.
# Resposta do Sistema G(s).
t2, y2 = ctrl.forced_response(sys, T=t2, U=u2)
t4, y4 = ctrl.forced_response(sys, T=t4, U=u4)
t10, y10 = ctrl.forced_response(sys, T=t10, U=u10)
tsin5, ysin5 = ctrl.forced_response(sys, T=tsin5, U=usin5)
```

Utilizando o sistema $\hat{G}(s)$ foi feita a simulação numérica das respostas as entradas ao degrau (2, 4 e 10 volts) e senoidal (5 volts). Porém, foram feitos ajustes no $\hat{K}$ e $\hat{\tau}$ para que as respostas fossem similares as respostas experimentais (tempo de subida, tempo de assentamento, sobrepasso, etc). Portanto, foram escolhidos $\hat{K} = 18$ e o $\hat{\tau} = 0.12$. Então, o $\hat{K}$ escolhido foi o de menor valor, porque ele que representa melhor o máximo de atrito proporcional a velocidade angular $\omega_m$. Além disso, o $\hat{\tau}$ escolhido foi de maior valor, porque ele que representa melhor o máximo de atraso da resposta em relação a entrada u.

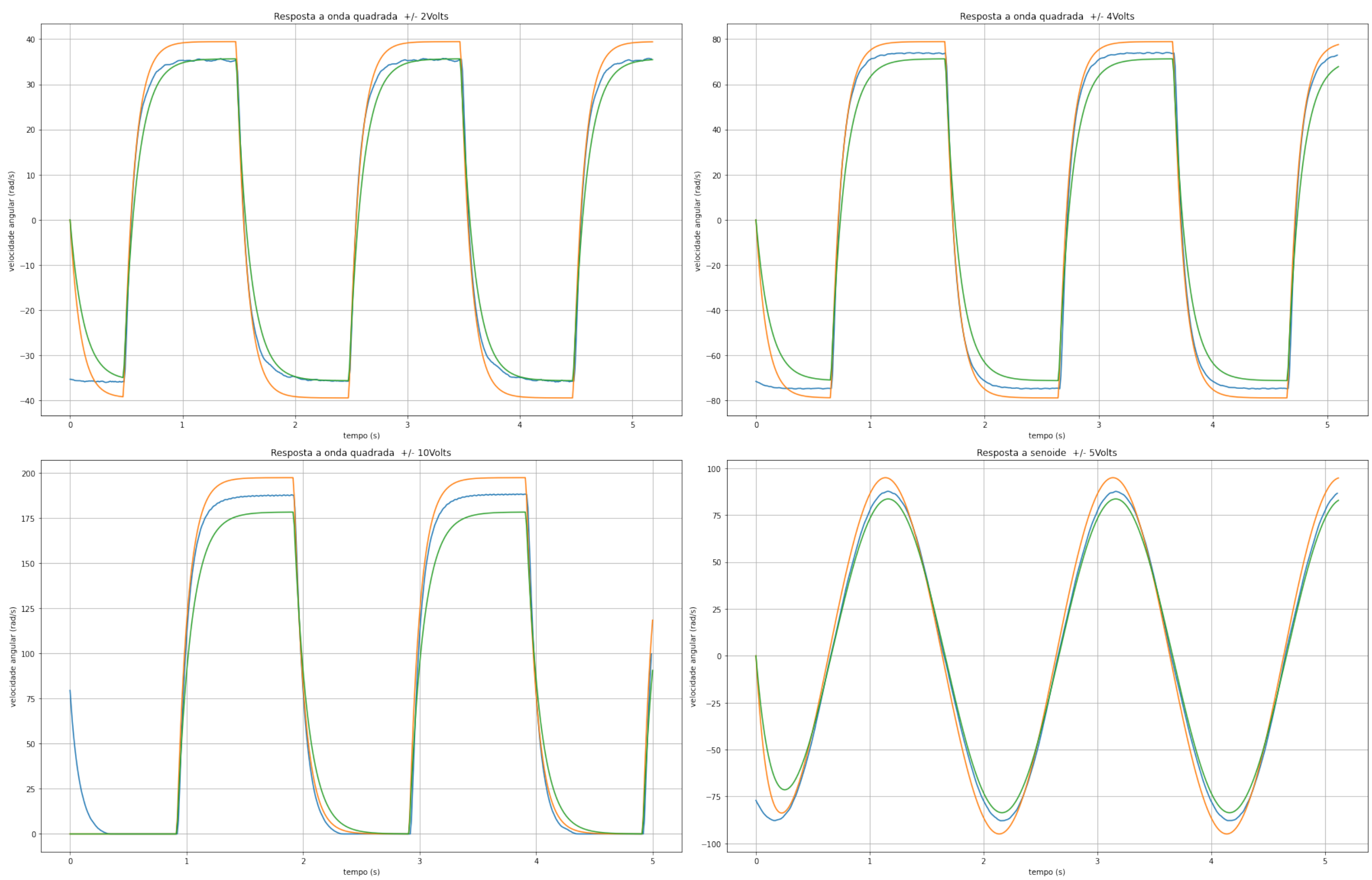



Figura 2: Respostas as entradas u

Legenda: y(t) experimental (Azul), y(t) do G(s) (Laranja), y(t) do $\hat{G}(s)$ (Verde)

```
# Definir a funcao de transferencia do sistema G_Hat(s)
num = [18]  # o K foi escolhido pegando o menor dos que se ajustaram as quatro respostas
den = [0.12, 1]  # o tau foi escolhido pegando o maior dos que se ajustaram as quatro respostas
sys_hat = ctrl.TransferFunction(num, den)  # criar o objeto que representa o sistema
# Criar a figura e os quatro subplots com tamanho ajustável
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(25, 16))
# Plotar o gráfico no subplot 1
ax1.plot(t2v, y2v)
ax1.plot(t2, y2)
t_hat_2, y_hat_2 = ctrl.forced_response(sys_hat,T=t2,U=u2)
ax1.plot(t_hat_2, y_hat_2)
ax1.set_title('Resposta a onda quadrada  +/- 2Volts')
ax1.set_xlabel('tempo (s)')
ax1.set_ylabel('velocidade angular (rad/s)')
ax1.grid(True)
# Plotar o gráfico no subplot 2
ax2.plot(t4v, y4v)
ax2.plot(t4, y4)
t_hat_4, y_hat_4 = ctrl.forced_response(sys_hat,T=t4,U=u4)
ax2.plot(t_hat_4, y_hat_4)
ax2.set_title('Resposta a onda quadrada  +/- 4Volts')
ax2.set_xlabel('tempo (s)')
ax2.set_ylabel('velocidade angular (rad/s)')
ax2.grid(True)
# Plotar o gráfico no subplot 3
ax3.plot(t10v, y10v)
ax3.plot(t10, y10)
t_hat_10, y_hat_10 = ctrl.forced_response(sys_hat,T=t10,U=u10)
ax3.plot(t_hat_10, y_hat_10)
ax3.set_title('Resposta a onda quadrada  +/- 10Volts')
ax3.set_xlabel('tempo (s)')
ax3.set_ylabel('velocidade angular (rad/s)')
ax3.grid(True)
# Plotar o gráfico no subplot 4
ax4.plot(tsin5v, ysin5v)
ax4.plot(tsin5, ysin5)
```

```
t_hat_sin5, y_hat_sin5 = ctrl.forced_response(sys_hat,T=tsin5,U=usin5)
ax4.plot(t_hat_sin5, y_hat_sin5)
ax4.set_title('Resposta a senoide  +/- 5Volts')
ax4.set_xlabel('tempo (s)')
ax4.set_ylabel('velocidade angular (rad/s)')
ax4.grid(True)
# Ajustar o espaçamento entre os subplots
plt.tight_layout()
# Exibir o gráfico
plt.show()
```

# 4 Conclusão

A discrepância entre os modelos de motor DC de armadura e os resultados experimentais ocorre devido a simplificações inerentes aos modelos teóricos, parâmetros variáveis do motor na prática e efeitos não considerados. Além disso, condições de operação realistas, como variações de carga e perturbações ambientais, afetam o desempenho do motor de maneira não capturada pelos modelos teóricos. Limitações na medição, como erros, ruído e atrasos de resposta, também contribuem para a diferença entre os resultados teóricos e experimentais. É importante reconhecer essas limitações e buscar abordagens mais abrangentes para modelar e analisar o comportamento do motor DC de armadura, levando em consideração a complexidade do mundo real.

# Referências

[1] Documentação do Python Control. Disponível em: `https://python-control.readthedocs.io/en/0.9.3.post2/`. Acesso em: 17 de junho de 2023.

[2] Documentação do SciPy. Disponível em: `https://docs.scipy.org/doc/scipy/`. Acesso em: 17 de junho de 2023.

[3] Documentação do NumPy. Disponível em: `https://numpy.org/doc/stable/`. Acesso em: 17 de junho de 2023.

[4] Documentação do Matplotlib. Disponível em: `https://matplotlib.org/stable/api/index`. Acesso em: 17 de junho de 2023.

[5] DORF, R. C.; BISHOP, R. H. *Modern Control Systems*. 11th edition. [S.l.]: PEARSON PRENTICE HALL, 2008.